Continual-GraphLLM: Dynamic Graph Large Language Model with Invariance Regularized Adaptive Multi-Scale Experts

Tianhang Wan
Tsinghua University
Beijing, China
wanth21@mails.tsinghua.edu.cn

Xin Wang*
Tsinghua University
Beijing, China
xin_wang@tsinghua.edu.cn

Haibo Chen
Tsinghua University
Beijing, China
chb24@mails.tsinghua.edu.cn

Longtao Huang
Alibaba Group
Hangzhou, China
kaiyang.hlt@alibaba-inc.com

Wenwu Zhu*
Tsinghua University
Beijing, China
wwzhu@tsinghua.edu.cn

Abstract

Dynamic text-attributed graphs (DyTAGs) exhibit coupled textual and structural dynamics, and existing mainstream approaches for DyTAGs extend conventional large language models (LLMs) to capture both dynamics, thereby giving rise to dynamic graph LLMs. However, in DyTAGs, the continuous emergence of new nodes and edges with incoming textual content and interactions drives the joint evolution of graph structural-textual patterns, causing existing methods to struggle with evolving patterns. This motivates a largely unexplored problem of continual learning on DyTAGs, which aims to adapt to constantly evolving graph structural-textual patterns while retaining past knowledge, which imposes two challenges: 1) unlike common graphs, graph structure and textual semantics in emerging DyTAG patterns jointly evolve, requiring dynamic graph LLMs to adapt structure, text, and graph-text fusion simultaneously; and 2) updating dynamic graph LLMs to fit a new pattern may destroy the global graph-text fusion capabilities and bias the model towards recent local dynamics. To address these challenges, we propose a novel **Continual** Learning Dynamic **Graph LLM** framework (**Continual-GraphLLM**) to continually adapt to incoming patterns by routing them to experts specialized in similar past patterns, while mitigating the overwriting of previously learned patterns by assigning new experts to unseen patterns. Specifically, we propose a graph-text factor-based router to adapt to incoming structural-textual joint patterns by utilizing latent factors to adaptively activate suitable experts. Furthermore, we design invariance regularized multi-scale experts that mitigate forgetting by capturing the invariances among learned patterns assigned to the same expert, where each expert progressively integrates structural and textual information from local scale to global scale. Extensive experiments on real-world DyTAGs demonstrate the superiority of our method over competitive baselines, highlighting its effectiveness in adapting to emerging DyTAG patterns.

CCS Concepts

• **Computing methodologies** → **Artificial intelligence**; **Machine learning**.

Keywords

Continual Learning; Dynamic Text-Attributed Graph; Large Language Model

ACM Reference Format:
Tianhang Wan, Xin Wang, Haibo Chen, Longtao Huang, and Wenwu Zhu. 2026. Continual-GraphLLM: Dynamic Graph Large Language Model with Invariance Regularized Adaptive Multi-Scale Experts. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '26), August 09–13, 2026, Jeju Island, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3770855.3817924

1 Introduction

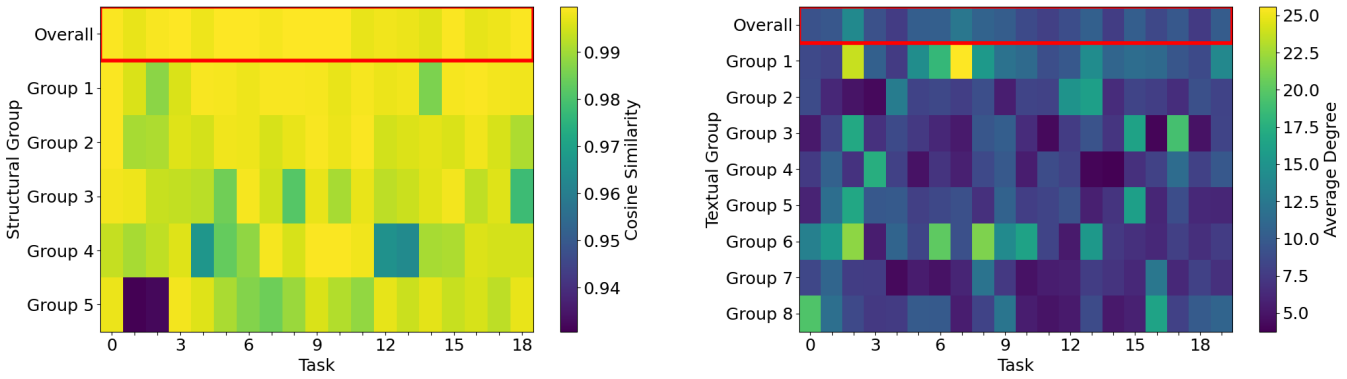
Dynamic text-attributed graphs (DyTAGs) are crucial for modeling complex, evolving real-world systems such as e-commerce platforms, social networks, and knowledge graphs [7, 17, 37–39, 56]. In these graphs, entities and interactions are not only associated with rich textual information, but also exhibit continuous temporal dynamics, where both topological structures and textual attributes change over time. Building upon the remarkable success of large language models (LLMs) in graph learning [16, 35, 44, 46, 53], numerous LLM-based methods have recently been proposed to extend the generic reasoning capabilities of LLMs to DyTAGs. By leveraging the semantic expressiveness of LLMs, these methods have demonstrated superior performance in predicting unseen interactions within closed datasets [14, 22, 36, 47, 58, 69]. In this paper, we refer to LLM-augmented methods designed for DyTAGs as **Dynamic Graph LLMs**.

However, in many real-world scenarios, new nodes and edges with incoming textual content keep accumulating, driving the joint evolution of structural-textual patterns in a continual streaming manner. For example, in communication networks, interaction patterns and textual message content drift as user behaviors and organizational routines evolve; in social networks, evolving user populations reshape interaction patterns, and the accompanying user-generated texts (e.g., posts, profiles) change concurrently [9, 43, 61].

*Corresponding authors. Xin Wang and Wenwu Zhu are affiliated with Department of Computer Science and Technology, Beijing National Research Center for Information Science and Technology, Tsinghua University.

KDD '26, Jeju Island, Republic of Korea

ACM ISBN 979-8-4007-2259-2/2026/08
https://doi.org/10.1145/3770855.3817924



(a) Textual Similarity Conditional on Structural Group

(b) Structural Average Degree Conditional on Textual Group

Figure 1: An illustration of the joint structural-textual evolution across DyTAG patterns in the Enron-DI dataset. The first row shows the smooth ungrouped textual/structural distribution separately, while the following rows show the distribution after grouping by structural/textual characteristics, which are noticeably unsmooth and differ markedly from the first row.

Figure 1 provides intuitive evidence of joint structural-textual evolution across DyTAG patterns, the details of which will be described in Appendix B. Existing dynamic graph LLMs are designed for offline training on closed datasets to fit a specific pattern, thus struggling to handle DyTAG continual learning.

In this paper, we study the problem of continual learning on DyTAGs, which remains largely unexplored in the literature. This task is particularly challenging, primarily due to two reasons. **Adaptation to Emerging Joint Structural-Textual Patterns:** In common dynamic graph continual learning, pattern evolution is primarily structural, while DyTAG patterns induce a joint evolution in graph structures and textual semantics. Since dynamic graph LLMs typically learn graph and text understanding and graph-text fusion capability from a fixed DyTAG pattern, when new DyTAG patterns arrive with joint structural-textual evolution, the previously learned graph and text understanding becomes suboptimal, and graph-text fusion capability becomes mismatched to the new regime, making it difficult to adapt effectively without re-training. **Mitigating Historical Structural-Textual Knowledge Forgetting:** Updating a dynamic graph LLM to adapt to new DyTAG patterns risks destroying the graph-text understanding and fusion learned from earlier periods, since only the current pattern is fully observable during adaptation. A trivial adaptation often biases the model toward recent local dynamics in the last observed pattern, and overwrites some global knowledge learned from long-range history, leading to catastrophic forgetting.

To address these challenges, we propose **Continual-GraphLLM**, a continual dynamic graph LLM framework that adapts to each incoming DyTAG pattern by routing it to experts specialized in similar past patterns, while allocating new experts to unseen patterns to mitigate catastrophic forgetting. Specifically, we propose a graph-text factor-based router, which adaptively activates a suitable expert to better adapt to each incoming structural-textual joint pattern based on pattern-factor matching, where factors summarize historical DyTAG patterns. Similar patterns are routed to the same expert, making adaptation easier. Then, we propose invariance regularized multi-scale experts, which use invariance regularization to capture the invariance among similar patterns routed to the same expert, thereby mitigating forgetting. Each expert progressively models the graph-text fusion from local scale to global scale, empowering the LLM to reason over the unified structural-textual context for precise final predictions. The contributions of our work are summarized as follows:

- We propose a dynamic graph LLM framework **Continual-GraphLLM** which is able to continually adapt to incoming DyTAG patterns with joint structural-textual evolution while mitigating catastrophic forgetting. To the best of our knowledge, this is the first method to study DyTAG continual learning.
- We introduce two novel modules: **graph-text factor-based router**, which adaptively activates suitable experts to adapt to the joint evolution of structure and text in DyTAGs; and **invariance regularized multi-scale experts**, which capture the invariances among similar patterns to mitigate forgetting, with each expert performing graph-text fusion from local scale to global scale.
- We conduct extensive experiments on real-world DyTAG datasets in two different challenging continual learning settings (domain-incremental and class-incremental), demonstrating that our proposed method outperforms all baselines in DyTAG continual learning.

2 Problem Formulation

2.1 Dynamic Text-Attributed Graphs

A dynamic text-attributed graph (DyTAG) is a sequence of interaction events over time, where each timestamped event involves two entities and both interactions and entities are associated with textual attributes [56]. Given a set of nodes $\mathcal{V}$, a set of edges $\mathcal{E}$, node text attributes $\mathcal{D}$, edge text attributes $\mathcal{R}$, and a label set $\mathcal{Y}$, a DyTAG can be represented as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{D}, \mathcal{R}, \mathcal{Y}\}$, where $(u, v, r, y, t) \in \mathcal{E}$ denotes an interaction between node $u \in \mathcal{V}$ and node $v \in \mathcal{V}$ with textual attribute $r \in \mathcal{R}$ and label $y \in \mathcal{Y}$ at timestamp t. Each node $v \in \mathcal{V}$ is also associated with its own textual attributes $d_v \in \mathcal{D}$.

2.2 DyTAG Continual Learning

In real-world scenarios, graphs are generated continuously over time. A DyTAG $\mathcal{G}$ can be naturally divided into a stream of time-evolving disjoint patterns $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_N\}$, where each pattern $\mathcal{G}_i = \{(u, v, r, y, t) \in \mathcal{G} : t_i^{start} \leq t < t_i^{end}\}$ contains interaction events within a specific time interval $[t_i^{start}, t_i^{end})$ and the time intervals are non-overlapping. Accompanying the DyTAG patterns is a sequence of tasks $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\}$, and their corresponding label sets $\{\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_N\}$.

There are three main types of learning settings in graph continual learning literature [41, 65], including task-incremental, domain-incremental, and class-incremental learning. In task-incremental learning, the task identities are revealed to models during both training and testing, making the problem relatively easier [61]. In this work, we focus on the other two more challenging settings, i.e., domain-incremental learning and class-incremental learning.

In domain-incremental learning, different patterns have different data distributions, i.e., $P(\mathcal{G}_i) \neq P(\mathcal{G}_j)$ for $i \neq j$, but all the tasks share the same closed label set, i.e., $\mathcal{Y}_i = \mathcal{Y}_j = \mathcal{Y}$. In class-incremental learning, the label sets of different tasks are disjoint, i.e., $\mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset$ for $i \neq j$. In both settings, the task identity i of task $\mathcal{T}_i$ is not accessible during both training and testing.

At the i-th time interval, the model receives the new incoming patterns $\mathcal{G}_i$ and learns new knowledge from the task $\mathcal{T}_i$ without forgetting prior tasks. While learning from the current task $\mathcal{T}_i$, previous patterns $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_{i-1}\}$ are no longer accessible.

3 Method

In this section, we introduce **Continual-GraphLLM** for continual learning on DyTAGs with LLMs. This method comprises two key components: the graph-text factor-based router and the invariance regularized multi-scale experts. The overall framework is illustrated in Figure 2.

3.1 Graph-Text Factor-Based Router

In continual DyTAG streams, each incoming pattern $\mathcal{G}_i$ exhibits a new joint structural-textual evolution, requiring dynamic graph LLMs to adapt sequentially to these evolving patterns. It's difficult for a single shared model to sequentially adapt to all patterns without interference since parameters are repeatedly pulled toward mismatched patterns. To address this, we propose a graph-text factor-based router with dynamically expandable experts to improve adaptability across diverse tasks and evolving patterns. Given an incoming pattern, the router produces an assignment over the current expert pool or allocates a new expert if necessary. The router isolates tasks with a large pattern difference to reduce negative transfer, while allowing tasks with similar patterns to reuse experts and share mutual knowledge.

Factor Learning with Distribution Matching. Numerous prior works perform distribution matching for graph condensation by minimizing the maximum mean discrepancy (MMD) [11] between the original graph and a learned compact surrogate [31–33, 68]. Inspired by these methods, we learn a factor for each incoming DyTAG pattern with distribution matching by minimizing the MMD between the original DyTAG pattern and the learned factor.

Suppose that the current task is $\mathcal{T}_i$, with a DyTAG pattern $\mathcal{G}_i$. Our goal is to learn a condensed factor $\mathbf{F}_i$ that captures the essential information of $\mathcal{G}_i$. In general, the objective can be formulated as

$$\mathbf{F}_i = \arg\min_{\mathbf{F}} \mathrm{Dist}(\mathcal{G}_i, \mathbf{F}), \tag{1}$$

where $\mathrm{Dist}(\cdot, \cdot)$ measures the distribution discrepancy.

We emphasize that the DyTAG pattern evolution couples topology and textual semantics. To explicitly capture this joint evolution, we model the condensed factor as two complementary components, $\mathbf{F}_i = (\mathbf{F}_i^{\text{struct}}, \mathbf{F}_i^{\text{text}})$. For the structural component, we use a graph neural network (GNN) to encode $\mathcal{G}_i, F_i^{\text{struct}}$ into a shared latent space and measure the distribution discrepancy in the latent space by MMD. The objective of learning $\mathbf{F}_i^{\text{struct}}$ can be formulated as:

$$\begin{aligned} \mathbf{F}_i^{\text{struct}} &= \arg\min_{\mathbf{F}} \mathrm{Dist}^{\text{struct}}(\mathcal{G}_i, \mathbf{F}) \\ &= \arg\min_{\mathbf{F}} \mathrm{MMD}(\mathrm{GNN}(\theta, \mathcal{G}_i), \mathrm{GNN}(\theta, \mathbf{F})), \end{aligned} \tag{2}$$

where $\mathrm{GNN}(\theta, \cdot)$ is a shared graph neural network encoder with randomly initialized parameters θ, and the graph structure of $\mathbf{F}_i^{\text{struct}}$ is set to be self-loops for simplicity. For the textual component, we use a pre-trained language model (PLM) and a multi-layer perceptron (MLP) to encode the text attributes of $\mathcal{G}_i, \mathbf{F}_i^{\text{text}}$ into a shared latent space, and again measure the MMD in the latent space. The objective of learning $\mathbf{F}_i^{\text{text}}$ can be formulated as:

$$\begin{aligned} \mathbf{F}_i^{\text{text}} &= \arg\min_{\mathbf{F}} \mathrm{Dist}^{\text{text}}(\mathcal{G}_i, \mathbf{F}) \\ &= \arg\min_{\mathbf{F}} \mathrm{MMD}(\mathrm{PLM}(\mathcal{G}_i), \mathrm{MLP}(\mathbf{F})), \end{aligned} \tag{3}$$

where $\mathrm{PLM}(\cdot)$ is a pre-trained language model for direct text encoding, and $\mathrm{MLP}(\cdot)$ is a multi-layer perceptron to project the learnable text factor into the PLM latent space. The overall distribution discrepancy between $\mathcal{G}_i$ and $\mathbf{F}_i$ is defined as the weighted sum of the structural and textual discrepancies:

$$\mathrm{Dist}(\mathcal{G}_i, \mathbf{F}_i) = \beta\, \mathrm{Dist}^{\text{struct}}(\mathcal{G}_i, \mathbf{F}_i^{\text{struct}}) + (1 - \beta)\, \mathrm{Dist}^{\text{text}}(\mathcal{G}_i, \mathbf{F}_i^{\text{text}}). \tag{4}$$

Route with Pattern-Factor Matching. We maintain an assignment vector $\mathbf{R}_j \in [0, 1]^K$ for each task $\mathcal{T}_j$, where K denotes the maximum number of experts constrained by the resource budget, and $\mathbf{R}_j$ denotes the probability of routing $\mathcal{T}_j$ to each of the K experts. Before training on a new task $\mathcal{T}_i$ with an incoming DyTAG pattern $\mathcal{G}_i$, we first compute the discrepancy between this pattern and each previously learned factor,

$$d_{i,j} = \mathrm{Dist}(\mathcal{G}_i, \mathbf{F}_j), \quad j = 1, 2, \ldots, i-1. \tag{5}$$

Unlike simple similarity-attraction routing that assigns a new task to experts specialized in nearby historical patterns, our routing is also designed to be avoidance-driven. Intuitively, if an expert has primarily served past tasks that are far from the incoming pattern (i.e., large $d_{i,j}$), then routing the new task to this expert is likely to cause negative transfer and should be discouraged. Besides, a simple similarity-attraction routing strategy tends to route most tasks to a single expert. To implement this routing strategy, we calculate the probability of this new task $\mathcal{T}_i$ being routed to each expert as

$$\mathbf{R}_i = \frac{1}{z_i} \left(\sum_{j<i} e^{\alpha d_{i,j}} \mathbf{R}_j \right)^{-1}, \tag{6}$$

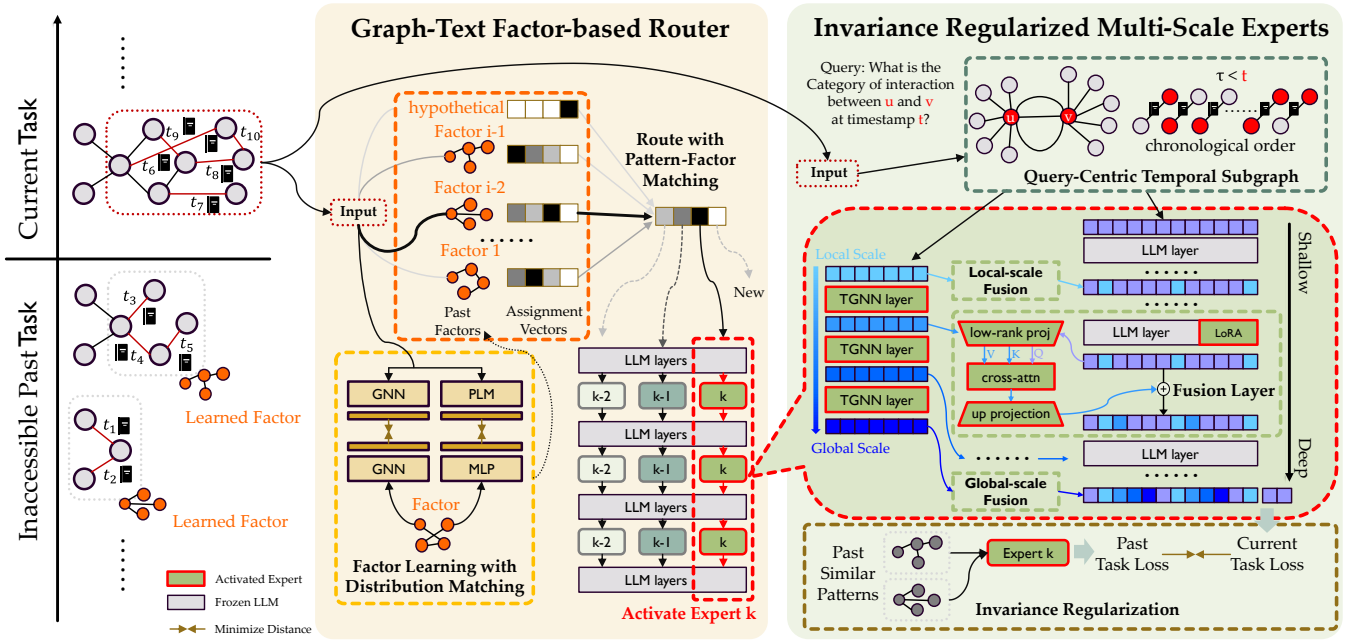


Figure 2: Overview framework of Continual-GraphLLM. Given an incoming DyTAG pattern from a new task $\mathcal{T}_i$, the graph-text factor-based router matches the pattern with previously learned factors and a hypothetical future factor to produce an assignment over the current expert pool or allocate a new expert if necessary. The invariance regularized multi-scale expert with the maximum assignment probability is activated to inject structural knowledge into the LLM. The query-centric temporal subgraph is retrieved as structural evidence for each query, and then sent to both the TGNN encoder and the LLM for modeling text-aware structural and structure-aware textual information, respectively. A low-rank cross-attention mechanism is applied to effectively integrate information from local to global levels, and the LLM with fused hidden states is used to generate the final prediction for downstream tasks. By introducing an invariance regularization, we encourage the losses of tasks assigned to the same expert to be close, so that the expert can better share mutual knowledge among these similar patterns.

where z_i is a normalization coefficient to ensure $\sum_{k=1}^{K} R_{i,k} = 1$, and α is the temperature controlling the sharpness of the discrepancy distribution. The term $e^{\alpha d_{i,j}}$ can be viewed as a negative-transfer risk score. Due to the exponential scaling, contributions from highly dissimilar patterns dominate the risk accumulation. The operator $(\cdot)^{-1}$ denotes an element-wise inverse, which assigns higher routing probability to experts with lower estimated risk.

Our expert pool is dynamically expandable. In particular, when the current task $\mathcal{T}_i$ is far from all previously observed patterns, the existing experts may be unable to handle this new DyTAG pattern well. In this case, rather than forcing $\mathcal{T}_i$ to share an ill-suited expert, we should allocate a new expert to specialize in this new pattern and possible future similar patterns. To realize this mechanism within the same routing formulation, we introduce a hypothetical future DyTAG pattern $\mathcal{G}_H$. Its discrepancy to the current pattern $\mathcal{G}_i$ is a constant $d_{i,H} = d_H$, and its assignment vector R_H is a one-hot vector indicating a new expert. Augmenting Equation 6 with this hypothetical task leads to the extended routing rule in Equation 7:

$$\mathbf{R}_i = \frac{1}{z_i}\left(\sum_{j<i} e^{\alpha d_{i,j}}\mathbf{R}_j + e^{\alpha d_H}\,\mathrm{OneHot}(k_{\text{new}})\right)^{-1}, \tag{7}$$

where k_{new} is the index of a new expert. Equation 7 itself does not require a predefined K and supports dynamic expansion, but in practice we can use K to control the resource budget. We route the current task $\mathcal{T}_i$ to the expert with maximum probability in $\mathbf{R}_i$ for training. By jointly considering pattern-factor discrepancy and historical routing behaviors, we dynamically and continually route each incoming new task to experts to avoid negative transfer while sharing mutual knowledge.

During testing on a task $\mathcal{T}_{\text{test}}$ without a task identity, there is no need to update any parameters of experts. Therefore, we use similarity-attraction routing to assign the test task as $\mathbf{R}_{\text{test}} = \mathrm{SoftMax}(-\alpha d_{\text{test},\cdot})\mathbf{R}$, where $d_{\text{test},\cdot}$ denotes the discrepancy between the test pattern and all learned factors, and $\mathbf{R}$ is the matrix of all learned assignment vectors.

3.2 Invariance Regularized Multi-Scale Experts

While the router mitigates interference in each expert, it does not by itself guarantee that an expert maintains a stable graph-text fusion capability over time. The same expert can be reused by multiple similar patterns, and sequential adaptation on newly assigned tasks can still bias the expert toward recent dynamics. Besides, parameters are separated across multiple experts; each expert should remain lightweight to keep the overall parameter budget manageable. To address this, we propose invariance regularized multi-scale experts

to mitigate forgetting by capturing invariance among similar patterns assigned to the same expert, where each multi-scale expert progressively fuses structural and textual information by injecting lightweight expert-specific adaptation at multiple layers and scales.

Query-Centric Temporal Subgraph Retrieval. Several prior works on dynamic graph learning suggest that, compared with multi-hop subgraph sampling, leveraging first-hop historical interactions is often sufficient to capture informative signals in real-world dynamic graphs [5, 54]. In the LLM setting, multi-hop expansion quickly inflates the subgraph and its associated texts, which incurs a high computational cost due to the quadratic attention complexity with respect to input length, degrades LLM performance as longer inputs dilute relevant evidence [23, 45], and may even exceed the context window of LLMs. Accordingly, we retrieve a query-centric 1-hop temporal subgraph as the structural evidence for each query. Specifically, given a query $q = (u, v, t)$ for edge-level prediction at time t (e.g., edge classification), where $u, v \in \mathcal{V}$ are the source and target nodes respectively, we retrieve all historical interactions of u and v before time t as $\mathcal{N}(u, t)$ and $\mathcal{N}(v, t)$, where

$$\mathcal{N}(w, t) = \{(w, x, r, y, \tau) \in \mathcal{G} \colon \tau < t\} \cup \{(x, w, r, y, \tau) \in \mathcal{G} \colon \tau < t\}. \tag{8}$$

Additionally, to capture finer-grained interaction patterns between u and v, we also retrieve their mutual historical interactions as

$$\mathcal{S}(u, v, t) = \{(u, v, r, y, \tau) \in \mathcal{G} \colon \tau < t\} \cup \{(v, u, r, y, \tau) \in \mathcal{G} \colon \tau < t\}. \tag{9}$$

We obtain $\mathcal{N}^*(u, t)$, $\mathcal{N}^*(v, t)$ and $\mathcal{S}^*(u, v, t)$ by truncating $\mathcal{N}(u, t)$, $\mathcal{N}(v, t)$ and $\mathcal{S}(u, v, t)$ to the most recent M interactions if necessary. We then merge them to form the query-centric temporal subgraph

$$\mathcal{G}_q = \mathcal{N}^*(u, t) \cup \mathcal{N}^*(v, t) \cup \mathcal{S}^*(u, v, t). \tag{10}$$

This compact subgraph offers concentrated temporal structural evidence, which is easy to serialize into a bounded-length context for LLM consumption.

Text-aware Structural Encoding. We employ DyGFormer [54] as our backbone graph encoder, which is a widely used temporal graph neural network (TGNN) for continuous-time dynamic graph representation learning. DyGFormer is explicitly designed to learn dynamic graph representations from historical 1-hop interaction contexts, which aligns well with our query-centric 1-hop temporal subgraph retrieval strategy. It encodes the temporal subgraph into a sequence of embeddings based on a transformer architecture. Denote the expert-specific L_{TGNN}-layer graph encoder as $g_k = \{g_k^{(l)}(\cdot)\}_{l=1}^{L_{\text{TGNN}}}$, where k is the index of the expert assigned to the current task by the router in Section 3.1. The structural encoding process can be formulated as:

$$\mathbf{H}_{\text{TGNN}}^{(0)} = \text{Init}\left(\mathcal{G}_q, \text{PLM}\right), \tag{11}$$

$$\mathbf{H}_{\text{TGNN}}^{(l)} = g_k^{(l)}\left(\mathbf{H}_{\text{TGNN}}^{(l-1)}\right), \quad l = 1, 2, \ldots, L_{\text{TGNN}}, \tag{12}$$

where $\mathbf{H}_{\text{TGNN}} \in \mathbb{R}^{m \times d_{\text{TGNN}}}$ is the TGNN hidden states of length m and dimension d_{TGNN}. $\text{Init}(\cdot, \text{PLM})$ maps interactions in $\mathcal{G}_q$ to a sequence of initial features $H_{\text{TGNN}}^{(0)}$ following the DyGFormer design. Specifically, to make the structural encoding text-aware, we use a pre-trained language model to get the representations of textual attributes associated with nodes and edges in $\mathcal{G}_q$, and then use the PLM embeddings to initialize the node and edge features in the process of $\text{Init}(\cdot, \text{PLM})$.

To obtain a meaningful structural encoding, we apply mean pooling over the TGNN hidden states and then attach an MLP classifier on top of this pooled representation to train the TGNN on the current task. With mean pooling, deeper hidden states tend to encode more global information, since they are jointly optimized to form a discriminative pooled summary.

Structure-aware Textual Modeling. To capture the rich semantics of textual attributes in DyTAGs, we incorporate an LLM to perform textual modeling. Instead of using LLM for modeling each node/edge attribute in isolation, we perform structure-aware textual modeling, modeling textual attributes conditioned on the query-centric temporal subgraph $\mathcal{G}_q$. We use a pre-defined prompt template $\text{Prompt}(\cdot)$ to serialize the $\mathcal{G}_q$ into a text sequence, which explicitly lists the historical interactions and their associated text attributes in the order of time. Denote the pre-trained L_{LLM} LLM layers as $f_{\text{pretrain}} = \{f^{(l)}(\cdot; \Theta^{(l)})\}_{l=1}^{L_{\text{LLM}}}$, where $\{\Theta^{(l)}\}$ are the pre-trained parameters of LLM. The textual modeling process can be formulated as:

$$\mathbf{H}_{\text{LLM}}^{(0)} = \text{Tokenizer}\left(\text{Prompt}(\mathcal{G}_q)\right), \tag{13}$$

$$\mathbf{H}_{\text{LLM}}^{(l)} = f^{(l)}\left(\mathbf{H}_{\text{LLM}}^{(l-1)}; \Theta^{(l)}\right), \quad l = 1, 2, \ldots, L_{\text{LLM}}, \tag{14}$$

where $\mathbf{H}_{\text{LLM}} \in \mathbb{R}^{n \times d_{\text{LLM}}}$ is the LLM hidden states of length n and dimension d_{LLM}.

Low-Rank Multi-Scale Fusion. In contrast to prior DyTAG approaches that treat the LLM as a frozen feature extractor for text encoding [36, 47, 58], we use the LLM not only for textual modeling but also as the final predictor for downstream tasks. Additionally, to effectively integrate structural and textual information, we propose a low-rank multi-scale fusion mechanism that injects structural knowledge into the LLM during its internal process. Specifically, we insert a set of low-rank fusion layers into the LLM at selected depths $S_L \subseteq \{1, 2, \ldots, L_{\text{LLM}}\}$. For each fusion layer at depth $l \in S_L$, we fuse the LLM hidden states $\mathbf{H}_{\text{fused}}^{(l-1)}$ from the previous layer with the TGNN hidden states $\mathbf{H}_{\text{TGNN}}^{(\text{LayerMap}(l))}$ at a corresponding depth $\text{LayerMap}(l)$ via a layer mapping function. Here, $\text{LayerMap}(\cdot)$ specifies a depth-aligned correspondence between the LLM and the TGNN, which can be formulated as

$$\text{LayerMap}(l) = \left\lceil \frac{l}{L_{\text{LLM}}} \times (L_{\text{TGNN}} + 1) - 1 \right\rceil. \tag{15}$$

Here, we assume $L_{\text{LLM}} > L_{\text{TGNN}}$ so the layer mapping is well-defined, which is usually the practical case, as LLMs tend to be deeper than TGNNs. Earlier TGNN layers preserve more fine-grained interaction-level evidence, while later layers aggregate and abstract over longer temporal contexts. LayerMap aligns these TGNN scales to LLM depths so that shallow LLM layers receive local cues and deeper layers receive more global summaries, resulting in a hierarchical and multi-scale fusion. The fusion process at layer $l \in S_L$ can be formulated as the following two-step procedure:

$$\mathbf{H}_{\text{fused}}^{(l)\prime} = f^{(l)}\left(\mathbf{H}_{\text{fused}}^{(l-1)}; \Theta^{(l)} + \Delta\Theta_k^{(l)}\right), \qquad l \in S_L, \tag{16}$$

$$\mathbf{H}_{\text{fused}}^{(l)} = \text{Fusion}_k^{(l)}\left(\mathbf{H}_{\text{fused}}^{(l)\prime}, \mathbf{H}_{\text{TGNN}}^{(\text{LayerMap}(l))}\right), \qquad l \in S_L, \tag{17}$$

where $\Delta\Theta_k^{(l)}$ are the expert-specific LoRA adaptation weights [13] at layer l for efficient LLM fine-tuning, and $\text{Fusion}_k^{(l)}(\cdot, \cdot)$ is the expert-specific fusion mechanism at layer l, both parameterized

by the k-th expert. We implement the fusion mechanism with a low-rank cross-attention as follows:

$$\mathbf{Q} = \mathbf{H}_{\text{fused}}^{(l)'}\mathbf{W}_Q, \tag{18}$$

$$\mathbf{K} = \mathbf{H}_{\text{TGNN}}^{(\text{LayerMap}(l))}\mathbf{W}_K, \tag{19}$$

$$\mathbf{V} = \mathbf{H}_{\text{TGNN}}^{(\text{LayerMap}(l))}\mathbf{W}_V, \tag{20}$$

$$\mathbf{H}_{\text{fused}}^{(l)} = \mathbf{H}_{\text{fused}}^{(l)'} + \gamma \cdot \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{r}}\right)\mathbf{V}\mathbf{W}_O, \tag{21}$$

where $\mathbf{W}_Q \in \mathbb{R}^{d_{\text{LLM}}\times r}$, $\mathbf{W}_K \in \mathbb{R}^{d_{\text{TGNN}}\times r}$, $\mathbf{W}_V \in \mathbb{R}^{d_{\text{TGNN}}\times r}$, and $\mathbf{W}_O \in \mathbb{R}^{r\times d_{\text{LLM}}}$ are learnable linear projections with a small rank r, and γ is a learnable gating scalar that balances the contributions of injected structural information. For layers without fusion, we simply use the pre-trained LLM layers to process the hidden states:

$$\mathbf{H}_{\text{fused}}^{(0)} = \mathbf{H}_{\text{LLM}}^{(0)}, \quad \mathbf{H}_{\text{fused}}^{(l)} = f^{(l)}\left(\mathbf{H}_{\text{fused}}^{(l-1)}; \Theta^{(l)}\right) \forall l \notin S_L. \tag{22}$$

With this multi-scale fusion mechanism, the LLM can perform downstream inference with its internal hidden states explicitly conditioned on injected structural information. We use the LLM as the predictor in a generative manner. Given the fused hidden states, the model produces the task prediction by generating the answer text directly, and the generated text is mapped to a label via exact string matching. Additionally, by predicting via generation, we avoid introducing an extra learnable classifier in the end. This is desirable in class-incremental learning, where the last classifier is known to suffer from a strong bias toward new classes [34, 48, 72].

Invariance Regularization. By routing the current task $\mathcal{T}_i$ to expert k, the pattern $\mathcal{G}_i$ shares mutual knowledge and the same parameter sets with historical patterns $\{\mathcal{G}_j : \arg\max \mathbf{R}_j = k, j < i\}$. To further enhance mutual knowledge sharing among these similar patterns, we introduce an invariance regularization that encourages the model to produce consistent predictions among them by optimizing the variance of their task loss. Since the historical patterns are no longer accessible due to the continual learning setting, we use a small memory buffer to store a few representative query-centric temporal subgraphs $\bar{\mathcal{G}}_j$ for each historical pattern $\mathcal{G}_j$. The buffer is used only to compute invariance regularization, rather than as an independent replay module. For simplicity, we use the newest few queries from each historical task to construct $\bar{\mathcal{G}}_j$. Therefore, the invariance regularization can be formulated as:

$$\mathcal{L}_{\text{inv}} = \text{Var}\left(l(\Theta_k, \bar{\mathcal{G}}_j) \,|\, \arg\max \mathbf{R}_j = k\right), \tag{23}$$

where $l(\Theta_k, \mathcal{G})$ is the task loss. The final loss is a combination of the task loss on the current pattern and the invariance regularization:

$$\mathcal{L} = l(\Theta_k, \mathcal{G}_i) + \lambda \mathcal{L}_{\text{inv}}, \tag{24}$$

where λ is a hyper-parameter controlling the strength of the invariance regularization. The overall training pipeline of **ContinualGraphLLM** is summarized in Algorithm 1.

4 Experiments

In this section, we conduct experiments on real-world continuous-time DyTAG datasets to verify the design of our method.

4.1 Experimental Setup

Datasets. We conduct experiments to evaluate our method on three real-world DyTAG datasets from DTGB [56], including Enron, GDELT, and ICEWS1819. DTGB targets the standard closed DyTAG learning protocol, rather than continual learning, so we transform the datasets into a sequence of tasks. For Enron, the label set remains fixed across tasks, and the differences between tasks primarily arise from implicit evolution in DyTAG patterns over time (domain incremental learning). We refer to this variant as **Enron-DI**. For GDELT and ICEWS1819, we select the most frequent 20 labels as the overall label set and add two new labels per task (class incremental learning). We refer to these variants as **GDELT-CI** and **ICEWS1819-CI**. Following DTGB, all datasets are chronologically split into 70%/15%/15% for training, validation, and testing.

Baselines. We compare our method with three temporal graph neural networks **GraphMixer** [5], **TGAT** [50], **DyGFormer** [54], and a dynamic graph LLM method **CROSS** [58]. Since these methods are not designed for continual learning, we adapt them with several widely-used continual learning techniques: **EWC** [19], **TWP** [30], and **ER-GNN** [74]. We also evaluate the performance of a pure LLM **Qwen3-8B** [52] with our query-centric temporal subgraph prompting as another baseline, including zero-shot prompting, direct fine-tuning, and continual learning with the general-purpose approach EWC. For a fair comparison with LLM baselines, our method uses the same LLM backbone, i.e., Qwen3-8B, unless otherwise specified. More details about the baselines are provided in Appendix C.2.

Evaluation. After learning on task $\mathcal{T}_i$, the model is expected to perform well on all seen tasks $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_i\}$. Denote $P_{i,j}$ as the performance (any downstream metric) on the previous task $\mathcal{T}_j$ after learning on task $\mathcal{T}_i$, where $i \geq j$. We evaluate the model performance in continual learning from the following aspects: 1) **Average Performance (AP)**, measuring the final performance of models on all tasks: $\text{AP} = \frac{1}{N}\sum_{j=1}^{N} P_{N,j}$, where N is the total number of tasks. 2) **Average Forgetting (AF)**, measuring the extent of forgetting on previous tasks: $\text{AF} = \frac{1}{N-1}\sum_{j=1}^{N-1}(P_{N,j} - P_{j,j})$. Typically, AF is less than zero, indicating the occurrence of forgetting (i.e., performance drops on previous tasks after learning new ones). A larger AF indicates better knowledge retention.

We adopt the edge classification for the downstream task, and use weighted recall (equivalent to accuracy) and weighted F1-score as downstream evaluation metrics.

4.2 Main Results

The results of weighted recall and weighted F1-score are summarized in Table 1. From the results, we have the following observations:

- Compared to the methods using TGNNs as the final predictors, LLMs show a clear advantage for DyTAG continual learning. Notably, even directly fine-tuning LLMs on each incoming task yields competitive performance, which highlights the benefit of using LLMs as the final predictor without a classifier head, together with our query-centric temporal subgraph formulation.

Table 1: The results (%) of all the methods on the real-world datasets in terms of weighted recall and weighted F1-score. Numbers after the ± signs represent standard deviations. The best results are in bold, and the second-best results are underlined.

Method	Enron-DI				GDELT-CI				ICEWS1819-CI			
	Recall (Accuracy)		F1-Score		Recall (Accuracy)		F1-Score		Recall (Accuracy)		F1-Score	
	AP	AF	AP	AF	AP	AF	AP	AF	AP	AF	AP	AF
GraphMixer	$42.93_{\pm 2.39}$	$-9.42_{\pm 2.49}$	$34.61_{\pm 6.07}$	$-14.07_{\pm 6.66}$	$7.58_{\pm 1.69}$	$-60.86_{\pm 1.85}$	$7.76_{\pm 1.83}$	$-56.44_{\pm 2.69}$	$5.98_{\pm 0.08}$	$-77.64_{\pm 1.53}$	$5.85_{\pm 0.15}$	$-76.48_{\pm 2.86}$
GraphMixer + EWC	$45.46_{\pm 3.91}$	$-5.94_{\pm 4.89}$	$41.61_{\pm 3.69}$	$-5.65_{\pm 4.79}$	$11.62_{\pm 2.48}$	$-54.63_{\pm 2.70}$	$13.89_{\pm 3.15}$	$-45.65_{\pm 3.74}$	$7.30_{\pm 1.37}$	$-76.46_{\pm 1.39}$	$7.71_{\pm 2.10}$	$-75.08_{\pm 2.37}$
GraphMixer + TWP	$45.74_{\pm 2.67}$	$-5.78_{\pm 2.79}$	$42.52_{\pm 3.47}$	$-5.01_{\pm 3.66}$	$7.90_{\pm 1.78}$	$-59.51_{\pm 2.49}$	$8.80_{\pm 2.46}$	$-53.12_{\pm 4.34}$	$6.96_{\pm 0.68}$	$-77.27_{\pm 1.06}$	$7.32_{\pm 1.02}$	$-75.85_{\pm 1.64}$
GraphMixer + ER	$51.26_{\pm 2.05}$	$-2.90_{\pm 2.24}$	$47.07_{\pm 3.78}$	$-3.56_{\pm 3.80}$	$56.03_{\pm 2.41}$	$-6.59_{\pm 2.93}$	$51.02_{\pm 2.43}$	$-7.19_{\pm 3.44}$	$\underline{67.80_{\pm 3.03}}$	$\underline{-7.80_{\pm 2.03}}$	$\underline{67.13_{\pm 2.69}}$	$\underline{-7.92_{\pm 3.48}}$
TGAT	$38.59_{\pm 1.69}$	$-12.68_{\pm 1.54}$	$26.32_{\pm 1.00}$	$-20.63_{\pm 1.84}$	$6.41_{\pm 0.59}$	$-58.44_{\pm 0.83}$	$5.55_{\pm 0.85}$	$-50.06_{\pm 2.28}$	$5.45_{\pm 0.20}$	$-75.68_{\pm 0.99}$	$4.52_{\pm 0.69}$	$-74.84_{\pm 1.80}$
TGAT + EWC	$41.09_{\pm 2.09}$	$-10.68_{\pm 2.42}$	$35.82_{\pm 5.05}$	$-11.46_{\pm 5.48}$	$10.05_{\pm 4.31}$	$-56.19_{\pm 4.26}$	$10.58_{\pm 5.42}$	$-48.79_{\pm 5.28}$	$5.26_{\pm 0.16}$	$-76.13_{\pm 0.40}$	$3.93_{\pm 0.30}$	$-75.39_{\pm 0.78}$
TGAT + TWP	$42.62_{\pm 3.33}$	$-9.10_{\pm 3.55}$	$34.86_{\pm 5.52}$	$-12.09_{\pm 5.33}$	$7.73_{\pm 2.64}$	$-57.14_{\pm 1.93}$	$7.22_{\pm 3.46}$	$-50.12_{\pm 1.91}$	$5.49_{\pm 0.44}$	$-75.57_{\pm 0.61}$	$4.73_{\pm 1.05}$	$-73.83_{\pm 1.15}$
TGAT + ER	$50.86_{\pm 3.31}$	$-2.04_{\pm 2.64}$	$45.76_{\pm 4.16}$	$-3.51_{\pm 3.48}$	$53.78_{\pm 3.85}$	$-8.95_{\pm 5.04}$	$47.14_{\pm 3.28}$	$-11.00_{\pm 4.81}$	$63.88_{\pm 6.90}$	$-12.79_{\pm 7.00}$	$64.01_{\pm 5.98}$	$-11.96_{\pm 5.68}$
DyGFormer	$41.58_{\pm 2.82}$	$-11.90_{\pm 2.78}$	$31.55_{\pm 3.27}$	$-18.21_{\pm 3.63}$	$5.71_{\pm 0.45}$	$-60.07_{\pm 0.18}$	$5.23_{\pm 0.35}$	$-53.70_{\pm 1.59}$	$6.11_{\pm 1.19}$	$-74.19_{\pm 1.81}$	$5.77_{\pm 1.30}$	$-72.74_{\pm 2.77}$
DyGFormer + EWC	$44.29_{\pm 1.42}$	$-9.17_{\pm 1.21}$	$37.65_{\pm 1.31}$	$-11.64_{\pm 1.16}$	$6.51_{\pm 1.49}$	$-58.53_{\pm 0.80}$	$6.48_{\pm 1.62}$	$-51.64_{\pm 0.72}$	$6.38_{\pm 1.24}$	$-72.13_{\pm 1.67}$	$5.71_{\pm 1.92}$	$-70.30_{\pm 2.05}$
DyGFormer + TWP	$42.31_{\pm 2.28}$	$-10.94_{\pm 1.99}$	$34.16_{\pm 2.65}$	$-15.11_{\pm 2.87}$	$7.89_{\pm 1.05}$	$-57.74_{\pm 1.71}$	$8.81_{\pm 1.55}$	$-50.69_{\pm 2.07}$	$6.46_{\pm 0.67}$	$-74.67_{\pm 1.38}$	$6.10_{\pm 0.96}$	$-73.41_{\pm 2.06}$
DyGFormer + ER	$48.51_{\pm 1.48}$	$-5.27_{\pm 1.92}$	$42.52_{\pm 2.30}$	$-7.97_{\pm 2.58}$	$51.35_{\pm 5.26}$	$-10.04_{\pm 6.12}$	$47.35_{\pm 3.90}$	$-8.43_{\pm 4.85}$	$64.86_{\pm 6.85}$	$-10.40_{\pm 6.67}$	$64.82_{\pm 5.95}$	$-9.52_{\pm 5.13}$
CROSS	$42.02_{\pm 2.06}$	$-11.59_{\pm 2.30}$	$32.78_{\pm 2.10}$	$-17.11_{\pm 2.65}$	$5.75_{\pm 0.81}$	$-60.05_{\pm 0.75}$	$5.33_{\pm 0.72}$	$-53.37_{\pm 2.12}$	$5.30_{\pm 0.25}$	$-75.39_{\pm 0.79}$	$4.13_{\pm 0.85}$	$-73.35_{\pm 1.44}$
CROSS + EWC	$47.21_{\pm 2.55}$	$-5.73_{\pm 2.99}$	$43.31_{\pm 5.16}$	$-5.42_{\pm 5.63}$	$5.87_{\pm 1.10}$	$-58.44_{\pm 2.61}$	$5.13_{\pm 1.58}$	$-52.18_{\pm 2.28}$	$5.83_{\pm 1.36}$	$-76.69_{\pm 1.92}$	$4.81_{\pm 1.74}$	$-75.18_{\pm 3.11}$
CROSS + TWP	$45.25_{\pm 2.69}$	$-7.54_{\pm 3.06}$	$40.75_{\pm 3.73}$	$-7.89_{\pm 4.22}$	$5.91_{\pm 0.72}$	$-59.02_{\pm 1.23}$	$5.84_{\pm 0.78}$	$-51.59_{\pm 2.96}$	$6.89_{\pm 0.78}$	$-74.40_{\pm 1.73}$	$7.11_{\pm 1.40}$	$-71.20_{\pm 3.64}$
CROSS + ER	$48.86_{\pm 2.07}$	$-4.67_{\pm 1.94}$	$45.34_{\pm 3.15}$	$-5.45_{\pm 3.10}$	$48.23_{\pm 6.20}$	$-12.19_{\pm 5.40}$	$43.61_{\pm 6.71}$	$-11.49_{\pm 5.87}$	$53.81_{\pm 3.10}$	$-22.53_{\pm 3.49}$	$53.26_{\pm 3.17}$	$-21.78_{\pm 4.15}$
Qwen3-8B + $\mathcal{G}_q$ (frozen)	49.50	-	49.62	-	53.60	-	55.71	-	55.26	-	59.67	-
Qwen3-8B + $\mathcal{G}_q$	$49.09_{\pm 0.16}$	$-5.60_{\pm 0.08}$	$42.36_{\pm 0.07}$	$-8.35_{\pm 0.35}$	$61.91_{\pm 0.98}$	$-3.89_{\pm 0.75}$	$58.29_{\pm 1.65}$	$-6.90_{\pm 1.46}$	$63.67_{\pm 3.29}$	$-13.54_{\pm 3.67}$	$62.77_{\pm 2.98}$	$-14.08_{\pm 3.38}$
Qwen3-8B + $\mathcal{G}_q$ + EWC	$\underline{53.96_{\pm 0.44}}$	$\underline{-0.15_{\pm 0.64}}$	$\underline{49.66_{\pm 0.44}}$	$\underline{-0.33_{\pm 0.77}}$	$\underline{62.48_{\pm 0.08}}$	$\underline{-3.58_{\pm 0.12}}$	$\underline{59.86_{\pm 0.15}}$	$\underline{-5.49_{\pm 0.15}}$	$56.79_{\pm 2.62}$	$-22.30_{\pm 3.15}$	$62.11_{\pm 2.51}$	$-15.88_{\pm 3.20}$
Ours (Qwen3-8B)	$\mathbf{56.27_{\pm 0.45}}$	$\mathbf{-0.10_{\pm 0.43}}$	$\mathbf{53.74_{\pm 0.51}}$	$\mathbf{-0.03_{\pm 0.45}}$	$\mathbf{65.60_{\pm 0.51}}$	$\mathbf{-0.46_{\pm 0.28}}$	$\mathbf{65.11_{\pm 0.57}}$	$\mathbf{-0.45_{\pm 0.20}}$	$\mathbf{75.61_{\pm 0.12}}$	$\mathbf{-2.67_{\pm 0.13}}$	$\mathbf{76.00_{\pm 0.10}}$	$\mathbf{-2.08_{\pm 0.19}}$

- Our method achieves significant improvements over all baselines across datasets. In particular, our method improves 8% in terms of average performance and 5% in terms of average forgetting over the best baseline on the ICEWS1819-CI dataset. The results demonstrate the effectiveness of our method.

4.3 Ablation Study

To verify the effectiveness of the key components in our method, we compare different ablated variants of our method on each dataset: 1) **w/o TGNN** removes the TGNN model in text-aware structural encoding; 2) **w/o LLM** removes the LLM model in structure-aware textual modeling and directly uses the structural embeddings for the final prediction; 3) **w/o Inv** removes the invariance regularization for each expert; 4) **w/o Router** removes the factor-based router and uses a single expert for all tasks; 5) **Cyclic Route** replaces the router with a cyclic router that routes task $\mathcal{T}_i$ to expert i mod K; 6) **Random Route** replaces the router with a random router that uniformly randomly routes each task to an expert.

Note that for cyclic and random routing, the task identities are leaked to the router so that the routing behavior can be the same during training and testing. The results of GDELT-CI and ICEWS1819-CI are shown in Figure 3. We observe that the full model achieves the best performance across all datasets, demonstrating the effectiveness of each component in our method. We further observe the following: 1) The TGNN provides compact structural representations and helps the LLM better understand DyTAG dynamics and broadens its effective receptive field beyond the raw textual context. Removing the TGNN impairs the ability to capture and interpret

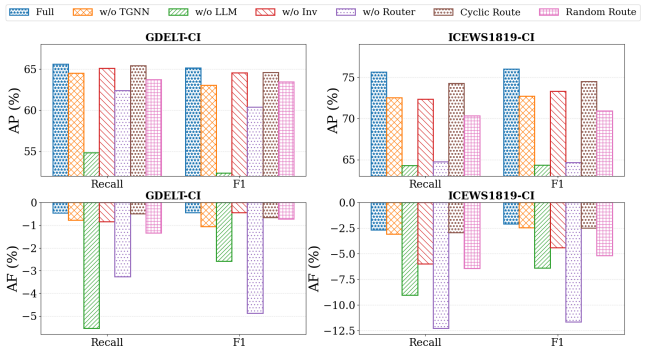


Figure 3: Comparisons of different ablated variants in terms of average performance (AP) and average forgetting (AF). "Full" denotes the full version of our method.

DyTAG structural patterns conditional on textual attributes. 2) The LLM models the rich textual attributes in DyTAGs and performs prediction over an open label set, playing a critical role in continual learning on DyTAGs. Removing the LLM causes the largest performance drop on all datasets. 3) Invariance regularization stabilizes each expert to focus on the mutual knowledge across tasks, thus enhancing knowledge retention. Removing the invariance regularization makes experts more prone to being biased toward the most recently assigned patterns. 4) The router dynamically activates new experts when necessary and encourages each expert to specialize in similar DyTAG patterns. Removing the router leads to a substantial performance drop, suggesting that individual experts have

limited capacity and that separating patterns is crucial. 5) Moreover, the two task-ID-based routing variants (cyclic and random) are suboptimal even though they exploit task identity as a shortcut, further demonstrating the effectiveness of our factor-based routing strategy.

4.4 LLM Backbone Generality

We evaluate the influence of different LLM backbones on our method, including Qwen3-8B [52] (default), Vicuna-7B [4], and Mistral-7B [15], compared with the best-performing baseline. The results are shown in Figure 4. Our method consistently outperforms the baseline across different LLM backbones.

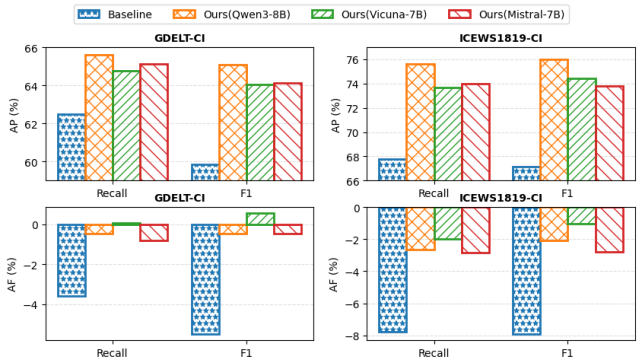


Figure 4: Comparisons of different LLM backbones in terms of average performance (AP) and average forgetting (AF).

4.5 Hyperparameter Sensitivity Analysis

We analyze the sensitivity of our method over the essential hyperparameters on ICEWS1819-CI and GDELT-CI, including the weight of the invariance loss λ and the maximum number of experts K.

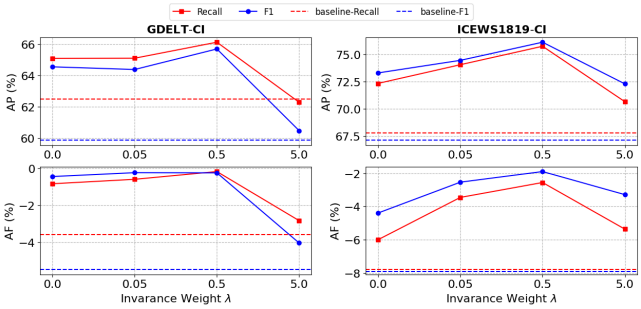


Figure 5: Sensitivity analysis on the weight of invariance regularization λ.

Weight of Invariance Regularization. We vary the weight λ from $\{0, 0.05, 0.5, 5\}$ while keeping all other hyperparameters fixed. As shown in Figure 5, increasing λ from 0 to a moderate value improves AP and especially AF, demonstrating the effectiveness of invariance regularization in enhancing knowledge retention. However, a large λ leads to significant performance degradation, as the model overly focuses on invariance and neglects task adaptations.

Number of Experts. We vary K from $\{1, 3, 4, 5, 6, 7, N\}$ while keeping all other hyperparameters fixed. As shown in Figure 6, increasing the maximum number of experts K can usually improve model performance, as it allows better separation of different DyTAG patterns. However, a larger K leads to a proportional increase in model parameters, and is not always beneficial since it may reduce the mutual knowledge sharing. In practice, if the K cannot be determined a priori, the expert expansion can be controlled solely by Equation 7, where a larger d_H leads to more conservative expansion. We further analyze the number of experts via Monte Carlo simulation: under a Gaussian assumption on $d_{ij} \sim \mathcal{N}(\mu, \sigma^2)$, setting d_H to $\mu - \sigma$, μ, and $\mu + \sigma$ corresponds to expected final number of experts K being 6.7, 4.4, and 2.7 for 10 tasks, respectively.

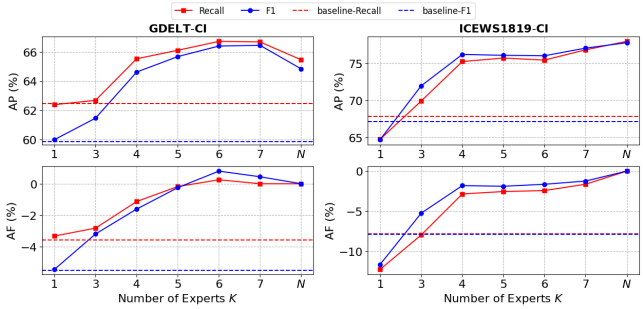


Figure 6: Sensitivity analysis on the number of experts K.

4.6 Routing Behavior Showcase and Analysis

The contribution of the router has been quantitatively verified through the ablation study in Section 4.3. To further improve the interpretability of the routing mechanism, we provide a representative showcase on GDELT-CI. After continual training, we evaluate each expert on every task. The results are reported in Table 2, where the underlined entries denote the experts selected by the router.

The average performance over all expert-task pairs is 61.8, while the average performance of the routed experts reaches 65.1. Moreover, the average rank of the routed expert is 1.6. These results indicate that the router assigns each task to the best or near-best expert during testing without a task identity.

Table 2: Expert-task weighted-recall performances (%) on GDELT-CI. Underlined entries denote the routed experts.

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
Expert 1	63.4	55.3	59.5	60.1	67.8	56.3	60.4	50.7	48.1	61.6
Expert 2	62.3	65.3	54.1	65.9	70.7	64.4	49.7	55.7	57.4	61.2
Expert 3	64.3	62.8	57.5	71.9	71.4	61.1	60.6	56.2	56.8	71.1
Expert 4	64.7	64.1	59.4	73.2	72.5	66.8	68.3	53.4	56.4	68.3
Expert 5	62.9	62.6	59.6	65.1	64.8	68.0	68.8	51.9	49.2	66.2
Rank of routed expert	3	1	2	2	1	1	1	3	1	1

Figure 7 visualizes the routing scores of all experts on each task during training, using ICEWS1819-CI with $K = 4$ as an example. In the early stage, the router tends to allocate new experts. After the budget is reached, the router builds a more global understanding

of the pattern space and reuses the existing experts for subsequent tasks. The router produces a relatively balanced assignment instead of assigning most tasks to few experts, since Equation 7 has a self-balancing property: an expert that has already been assigned to more tasks becomes less likely to receive subsequent tasks. The result also suggests stable expert specialization over time. In particular, task 4 has a larger discrepancy than the others, so expert 3 is specialized for this task and never reused by subsequent tasks. The router avoids mixing it with potentially incompatible tasks.

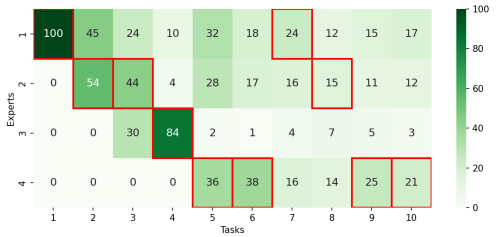


Figure 7: Routing scores (%) on ICEWS1819-CI. The red boxes denote the routed experts with the highest scores.

4.7 Efficiency Analysis

We evaluate the empirical efficiency under the fixed physical batch size of 2. Table 3 reports the total training time, total inference time and peak GPU memory of LLM methods, where runtime is measured as the sum of one-epoch runtime over all tasks. Compared with directly applying EWC to LLM, our method achieves lower runtime on Enron-DI and ICEWS1819-CI, and remains comparable on GDELT-CI, showing a favorable efficiency while improving continual learning performance.

Moreover, since exactly one expert is activated, computation in forward/backward pass remains constant as the number of experts increases. Although our method requires higher GPU memory, the additional memory is mainly from the fixed activations instead of expert parameters. In practice, when varying the number of experts, the peak GPU memory changes by less than 1GB.

Table 3: Efficiency comparison. The format of each cell is "training time / inference time / peak GPU memory".

Statistic	Enron-DI	GDELT-CI	ICEWS1819-CI
Qwen3-8B + $\mathcal{G}_q$ (frozen)	0 / 3.0h / 9.8GB	0 / 0.9h / 9.9GB	0 / 1.2h / 9.9GB
Qwen3-8B + $\mathcal{G}_q$	7.9h / 3.1h / 20.8GB	2.1h / 0.8h / 22.9GB	2.6h / 1.2h / 23.7GB
Qwen3-8B + $\mathcal{G}_q$ + EWC	14.7h / 4.1h / 20.8GB	2.6h / 0.9h / 23.0GB	4.2h / 1.8h / 23.9GB
Ours (Qwen3-8B)	13.2h / 3.2h / 31.0GB	3.1h / 1.0h / 32.9GB	3.5h / 1.2h / 36.2GB

5 Related Works

Graph Continual Learning. Different from graph OOD generalization [3, 24–28, 40, 67, 70, 71], graph continual learning aims to learn a sequence of graph tasks while retaining performance on previously learned tasks, typically under constraints such as restricted access to historical data [41, 55, 65]. Existing methods can be categorized into three main approaches: replay-based methods [32, 33, 60, 63, 64, 74] store a subset of historical data and replay it during training; regularization-based methods [1, 6, 30, 51, 59] introduce regularization terms to preserve important parameters; architecture-based methods [20, 57, 62] dynamically expand the model architecture to accommodate new tasks. Most existing graph continual learning methods focus on node classification over static graph streams or discrete-time dynamic graphs. Some works extend continual learning to continuous-time dynamic graphs. For instance, OTGNet [10] highlights the heterophily propagation issue in continuous-time dynamic graph continual learning, and handles class-incremental node classification via a triad-structure replay buffer; LTF [29] studies the scenario where old classes keep evolving while new classes emerge in continuous-time dynamic graphs, and updates TGNNs via a theory-guided subgraph selection. These graph continual learning methods operate on structure-only graphs, facing challenges in modeling textual information dynamics and handling the joint evolution of structural-textual patterns in DyTAGs.

Dynamic Graph LLM. Building upon the remarkable success of LLMs in graph learning, recent research has explored the integration of LLMs into dynamic graph modeling [2, 14, 66, 69, 73], temporal knowledge graph modeling [2, 21, 49], and DyTAG modeling by leveraging the semantic expressiveness and reasoning capabilities of LLMs. For DyTAGs, LKD4DyTAG [36] distills LLM-derived textual edge representations into a TGNN with temporal edge encoding for DyTAG tasks; CROSS [58] leverages LLM to extract the temporal semantics and a co-encoder to fuse both semantics and structures in DyTAGs into a unified representation; DyGRASP [47] leverages both the implicit and the explicit reasoning capabilities of LLMs to capture recent-global temporal semantics in DyTAGs; GAD [22] utilizes a multi-agent system where collaborative LLM agents decompose the DyTAG prediction problem into specialized sub-tasks. However, these methods assume an offline, closed-world setting and optimize for a specific distribution of patterns. This assumption breaks down in DyTAG continual learning, where the model must continually adapt to emerging patterns with joint structural-textual evolution.

6 Conclusion

We propose **Continual-GraphLLM**, a continual dynamic graph LLM framework that adapts to each incoming DyTAG pattern by routing it to experts specialized in similar past patterns, and mitigates forgetting by sharing mutual knowledge among similar patterns routed to the same expert. Specifically, **Continual-GraphLLM** routes each incoming DyTAG pattern to a suitable expert adaptively based on pattern-factor distribution matching. Similar patterns are routed to the same expert, making adaptation easier. Then each expert uses invariance regularization to capture the invariances and share mutual knowledge among similar patterns routed to the same expert, thereby mitigating forgetting. Structural information and textual information are fused progressively to enhance the model's ability to capture complex relationships between graph structure and text.

Acknowledgments

This work is supported by the National Key Research and Development Program of China under Grant No.2022ZD0115903.

References

[1] Jie Cai, Xin Wang, Chaoyu Guan, Yateng Tang, Jin Xu, Bin Zhong, and Wenwu Zhu. 2022. Multimodal continual graph learning with neural architecture search. In *Proceedings of the ACM Web Conference 2022*. 1292–1300.

[2] He Chang, Jie Wu, Zhulin Tao, Yunshan Ma, Xianglin Huang, and Tat-Seng Chua. 2025. Integrate Temporal Graph Learning into LLM-based Temporal Knowledge Graph Model. *arXiv preprint arXiv:2501.11911* (2025).

[3] Haibo Chen, Xin Wang, Guanheng Chen, Yuan Meng, Haoyang Li, Yang Yao, Zeyang Zhang, Zhiqiang Zhang, Jun Zhou, Ling Feng, et al. [n. d.]. Adaptive Mixture of Disentangled Experts for Dynamic Graph Out-of-Distribution Generalization. In *The Fourteenth International Conference on Learning Representations*.

[4] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. https://lmsys.org/blog/2023-03-30-vicuna/

[5] Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. 2023. Do we really need complicated model architectures for temporal networks? *arXiv preprint arXiv:2302.11636* (2023).

[6] Yuanning Cui, Yuxin Wang, Zequn Sun, Wenqiang Liu, Yiqiao Jiang, Kexin Han, and Wei Hu. 2023. Lifelong embedding learning and transfer for growing knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 4217–4224.

[7] Songgaojun Deng, Huzefa Rangwala, and Yue Ning. 2019. Learning dynamic context graphs for predicting social events. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1007–1016.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[9] Giovanni Donghi, Luca Pasa, Daniele Zambon, Cesare Alippi, and Nicolò Navarin. 2025. Online Continual Graph Learning. *arXiv preprint arXiv:2508.03283* (2025).

[10] Kaituo Feng, Changsheng Li, Xiaolu Zhang, and Jun Zhou. 2023. Towards open temporal graph neural networks. *arXiv preprint arXiv:2303.15015* (2023).

[11] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *The journal of machine learning research* 13, 1 (2012), 723–773.

[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[13] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR* 1, 2 (2022), 3.

[14] Shenyang Huang, Ali Parviz, Emma Kondrup, Zachary Yang, Zifeng Ding, Michael Bronstein, Reihaneh Rabbany, and Guillaume Rabusseau. 2025. Are Large Language Models Good Temporal Graph Learners? *arXiv preprint arXiv:2506.05393* (2025).

[15] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. arXiv:2310.06825 [cs.CL] https://arxiv.org/abs/2310.06825

[16] Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2024. Large language models on graphs: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering* (2024).

[17] Alexy Khrabrov and George Cybenko. 2010. Discovering influence in communication networks using dynamic graph analysis. In *2010 IEEE Second International Conference on Social Computing*. IEEE, 288–294.

[18] Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[19] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.

[20] Lecheng Kong, Theodore Vasiloudis, Seongjun Yun, Han Xie, and Xiang Song. 2025. Dynamic Mixture-of-Experts for Incremental Graph Learning. *arXiv preprint arXiv:2508.09974* (2025).

[21] Dong-Ho Lee, Kian Ahrabian, Woojeong Jin, Fred Morstatter, and Jay Pujara. 2023. Temporal knowledge graph forecasting without knowledge using in-context learning. *arXiv preprint arXiv:2305.10613* (2023).

[22] Runlin Lei, Jiarui Ji, Haipeng Ding, Lu Yi, Zhewei Wei, Yongchao Liu, and Chuntao Hong. 2025. Exploring the Potential of Large Language Models as Predictors in Dynamic Text-Attributed Graphs. *arXiv preprint arXiv:2503.03258* (2025).

[23] Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same task, more tokens: the impact of input length on the reasoning performance of large language models. *arXiv preprint arXiv:2402.14848* (2024).

[24] Haoyang Li, Xin Wang, Zeyang Zhang, Haibo Chen, Ziwei Zhang, and Wenwu Zhu. 2024. Disentangled graph self-supervised learning for out-of-distribution generalization. In *Forty-first International Conference on Machine Learning*.

[25] Haoyang Li, Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2022. Ood-gnn: Out-of-distribution generalized graph neural network. *IEEE Transactions on Knowledge and Data Engineering* 35, 7 (2022), 7328–7340.

[26] Haoyang Li, Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2025. Out-of-distribution generalization on graphs: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2025).

[27] Haoyang Li, Xin Wang, Xueling Zhu, Weigao Wen, and Wenwu Zhu. 2025. Disentangling invariant subgraph via variance contrastive estimation under distribution shifts. In *Forty-second International Conference on Machine Learning*.

[28] Haoyang Li, Ziwei Zhang, Xin Wang, and Wenwu Zhu. 2022. Learning invariant graph representations for out-of-distribution generalization. *Advances in Neural Information Processing Systems* 35 (2022), 11828–11841.

[29] Hanmo Liu, Shimin Di, Haoyang Li, Xun Jian, Yue Wang, and Lei Chen. 2025. A Selective Learning Method for Temporal Graph Continual Learning. *arXiv preprint arXiv:2503.01580* (2025).

[30] Huihui Liu, Yiding Yang, and Xinchao Wang. 2021. Overcoming catastrophic forgetting in graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 8653–8661.

[31] Mengyang Liu, Shanchuan Li, Xinshi Chen, and Le Song. 2022. Graph condensation via receptive field distribution matching. *arXiv preprint arXiv:2206.13697* (2022).

[32] Yilun Liu, Ruihong Qiu, and Zi Huang. 2023. Cat: Balanced continual graph learning with graph condensation. In *2023 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1157–1162.

[33] Yilun Liu, Ruihong Qiu, Yanran Tang, Hongzhi Yin, and Zi Huang. 2024. PUMA: Efficient continual graph learning for node classification with graph condensation. *IEEE Transactions on Knowledge and Data Engineering* (2024).

[34] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2001–2010.

[35] Xubin Ren, Jiabin Tang, Dawei Yin, Nitesh Chawla, and Chao Huang. 2024. A survey of large language models for graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6616–6626.

[36] Amit Roy, Ning Yan, and Masood Mortazavi. 2025. Llm-driven knowledge distillation for dynamic text-attributed graphs. *arXiv preprint arXiv:2502.10914* (2025).

[37] Mikołaj Sacha, Hammad Jafri, Mattie Terzolo, Ayan Sinha, and Andrew Rabinovich. 2025. GraphMatch: Fusing Language and Graph Representations in a Dynamic Two-Sided Work Marketplace. *arXiv preprint arXiv:2512.02849* (2025).

[38] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based social recommendation via dynamic graph attention networks. In *Proceedings of the Twelfth ACM international conference on web search and data mining*. 555–563.

[39] Haoran Tang, Shiqing Wu, Guandong Xu, and Qing Li. 2023. Dynamic graph evolution learning for recommendation. In *Proceedings of the 46th international acm sigir conference on research and development in information retrieval*. 1589–1598.

[40] Sibo Tian, Xin Wang, Zeyang Zhang, Haibo Chen, and Wenwu Zhu. 2026. Out-of-distribution generalized graph anomaly detection with homophily-aware environment mixup. *Advances in Neural Information Processing Systems* 38 (2026), 65681–65705.

[41] Zonggui Tian, Du Zhang, and Hong-Ning Dai. 2024. Continual learning on graphs: A survey. *arXiv preprint arXiv:2402.06330* (2024).

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

[43] Junshan Wang, Guojie Song, Yi Wu, and Liang Wang. 2020. Streaming graph neural networks via continual learning. In *Proceedings of the 29th ACM international conference on information & knowledge management*. 1515–1524.

[44] Xin Wang, Hong Chen, Zirui Pan, Yuwei Zhou, Chaoyu Guan, Lifeng Sun, and Wenwu Zhu. 2025. Automated disentangled sequential recommendation with large language models. *ACM Transactions on Information Systems* 43, 2 (2025), 1–29.

[45] Xindi Wang, Mahsa Salmani, Parsa Omidi, Xiangyu Ren, Mehdi Rezagholizadeh, and Armaghan Eshaghi. 2024. Beyond the limits: A survey of techniques to extend the context length in large language models. *arXiv preprint arXiv:2402.02244* (2024).

[46] Xin Wang, Zeyang Zhang, Linxin Xiao, Haibo Chen, Chendi Ge, and Wenwu Zhu. 2025. Towards multimodal graph large language model. *Science China Information Sciences* 68, 11 (2025), 1–20.

[47] Yunan Wang, Jianxin Li, and Ziwei Zhang. 2025. Global-Recent Semantic Reasoning on Dynamic Text-Attributed Graphs with Large Language Models. *arXiv preprint arXiv:2509.18742* (2025).

[48] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. 2019. Large scale incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 374–382.

[49] Wang Xing, Wei Song, Siyu Lin, Chen Wu, Zhesi Li, and Man Wang. 2026. Knowledge Distillation for Temporal Knowledge Graph Reasoning with Large Language Models. *arXiv preprint arXiv:2601.00202* (2026).
[50] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962* (2020).
[51] Yishi Xu, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, and Mark Coates. 2020. Graphsail: Graph structure aware incremental learning for recommender systems. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2861–2868.
[52] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388* (2025).
[53] Yang Yao, Xin Wang, Yuan Meng, Zeyang Zhang, Hong Mei, and Wenwu Zhu. 2026. Disentangled Graph LLM for Molecule Graph Editing under Distribution Shifts. In *Proceedings of the ACM Web Conference 2026*. 1149–1159.
[54] Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. 2023. Towards better dynamic graph learning: New architecture and unified library. *Advances in Neural Information Processing Systems* 36 (2023), 67686–67700.
[55] Qiao Yuan, Sheng-Uei Guan, Pin Ni, Tianlun Luo, Ka Lok Man, Prudence Wong, and Victor Chang. 2023. Continual graph learning: A survey. *arXiv preprint arXiv:2301.12230* (2023).
[56] Jiasheng Zhang, Jialin Chen, Menglin Yang, Aosong Feng, Shuang Liang, Jie Shao, and Rex Ying. 2024. DTGB: A comprehensive benchmark for dynamic text-attributed graphs. *Advances in Neural Information Processing Systems* 37 (2024), 91405–91429.
[57] Peiyan Zhang, Yuchen Yan, Chaozhuo Li, Senzhang Wang, Xing Xie, Guojie Song, and Sunghun Kim. 2023. Continual learning on dynamic graphs via parameter isolation. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*. 601–611.
[58] Siwei Zhang, Yun Xiong, Yateng Tang, Jiarong Xu, Xi Chen, Zehao Gu, Xuezheng Hao, Zian Jia, and Jiawei Zhang. 2025. Unifying text semantics and graph structures for temporal text-attributed graphs with large language models. *arXiv preprint arXiv:2503.14411* (2025).
[59] Xuling Zhang, Jindong Li, Yifei Zhang, and Menglin Yang. 2025. AL-GNN: Privacy-Preserving and Replay-Free Continual Graph Learning via Analytic Learning. *arXiv preprint arXiv:2512.18295* (2025).
[60] Xikun Zhang, Dongjin Song, Yixin Chen, and Dacheng Tao. 2024. Topology-aware embedding memory for continual learning on expanding networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4326–4337.
[61] Xikun Zhang, Dongjin Song, and Dacheng Tao. 2022. Cglb: Benchmark tasks for continual graph learning. *Advances in Neural Information Processing Systems* 35 (2022), 13006–13021.
[62] Xikun Zhang, Dongjin Song, and Dacheng Tao. 2022. Hierarchical prototype networks for continual graph representation learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 4 (2022), 4622–4636.
[63] Xikun Zhang, Dongjin Song, and Dacheng Tao. 2022. Sparsified subgraph memory for continual graph representation learning. In *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1335–1340.
[64] Xikun Zhang, Dongjin Song, and Dacheng Tao. 2023. Ricci curvature-based graph sparsification for continual graph representation learning. *IEEE Transactions on Neural Networks and Learning Systems* (2023).
[65] Xikun Zhang, Dongjin Song, and Dacheng Tao. 2024. Continual learning on graphs: Challenges, solutions, and opportunities. *arXiv preprint arXiv:2402.11565* (2024).
[66] Xin Zhang, Qiyu Wei, Yingjie Zhu, Linhai Zhang, Deyu Zhou, and Sophia Ananiadou. 2025. SynGraph: A Dynamic Graph-LLM Synthesis Framework for Sparse Streaming User Sentiment Modeling. *arXiv preprint arXiv:2503.04619* (2025).
[67] Zeyang Zhang, Xin Wang, Haibo Chen, Haoyang Li, and Wenwu Zhu. 2024. Disentangled dynamic graph attention network for out-of-distribution sequential recommendation. *ACM Transactions on Information Systems* 43, 1 (2024), 1–42.
[68] Zeyang Zhang, Xin Wang, Yijian Qin, Hong Chen, Ziwei Zhang, Xu Chu, and Wenwu Zhu. 2024. Disentangled continual graph neural architecture search with invariant modular supernet. In *Forty-first International Conference on Machine Learning*.
[69] Zeyang Zhang, Xin Wang, Ziwei Zhang, Haoyang Li, Yijian Qin, and Wenwu Zhu. 2024. LLM4DyG: can large language models solve spatial-temporal problems on dynamic graphs?. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4350–4361.
[70] Zeyang Zhang, Xin Wang, Ziwei Zhang, Haoyang Li, Zhou Qin, and Wenwu Zhu. 2022. Dynamic graph neural networks under spatio-temporal distribution shift. *Advances in neural information processing systems* 35 (2022), 6074–6089.
[71] Zeyang Zhang, Xin Wang, Ziwei Zhang, Zhou Qin, Weigao Wen, Hui Xue, Haoyang Li, and Wenwu Zhu. 2023. Spectral invariant learning for dynamic graphs under distribution shifts. *Advances in Neural Information Processing Systems* 36 (2023), 6619–6633.
[72] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. 2020. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 13208–13217.
[73] Ziwei Zhao, Fake Lin, Xi Zhu, Zhi Zheng, Tong Xu, Shitian Shen, Xueying Li, Zikai Yin, and Enhong Chen. 2024. DynLLM: when large language models meet dynamic graph recommendation. *arXiv preprint arXiv:2405.07580* (2024).
[74] Fan Zhou and Chengtai Cao. 2021. Overcoming catastrophic forgetting in graph neural networks with experience replay. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 4714–4722.

A Algorithm Details

The training procedure of our method is shown in Algorithm 1.

Algorithm 1 Training pipeline of **Continual-GraphLLM**

Require: an incoming training pattern $\mathcal{G}_i$.

```
Learn a factor F_i using Equation 1 and add into storage
Route to and activate expert k using Equation 7
for interaction (u, v, r, y, t) ∈ G_i, query q = (u, v, t) do
    Retrieve subgraph G_q using Equation 10
    Get TGNN states {H^(l)_TGNN} using Equation 12
end for
Update TGNN with the loss L using Equation 24
for interaction (u, v, r, y, t) ∈ G_i, query q = (u, v, t) do
    Retrieve subgraph G_q using Equation 10
    Get TGNN states {H^(l)_TGNN} using Equation 12
    Get fused states {H^(l)_fused} using Equation 17, 22
end for
Update other weights with the loss L using Equation 24
```

B Joint Structural-Textual Evolution Across Patterns

Figure 1 provides intuitive evidence of joint structural-textual shifts across DyTAG patterns. In Figure 1a, we divide interactions in each pattern $\mathcal{G}_i$ into multiple structural groups $\{\mathcal{G}_{i,g}\}_g$ based on the degree of nodes, and then we compute the similarity of the mean textual features between two consecutive patterns $\mathcal{G}_{i,g}$ and $\mathcal{G}_{i+1,g}$ for each group g. Specifically, the first row shows the similarity of mean textual features between $\mathcal{G}_i$ and $\mathcal{G}_{i+1}$ without grouping. The results show that if we ignore structural information, the textual information looks stable across patterns. However, the textual information conditioned on structural information shows significant shifts across patterns, which can be formulated as

$$P(\text{text}(\mathcal{G}_i)) \approx P(\text{text}(\mathcal{G}_j)), \tag{25}$$

$$P(\text{text}(\mathcal{G}_i) \mid \text{struct}(\mathcal{G}_i) = g) \neq P(\text{text}(\mathcal{G}_j) \mid \text{struct}(\mathcal{G}_j) = g). \tag{26}$$

Symmetrically, in Figure 1b, we divide nodes in each pattern $\mathcal{G}_i$ into multiple textual groups $\{\mathcal{G}_{i,g}\}_g$ based on the mean textual features of related interactions, and then we compute the average degree of each pattern and group $\mathcal{G}_{i,g}$. Specifically, the first row shows the average degree of $\mathcal{G}_i$ without grouping. The results show that if we ignore textual information, the structural information looks stable across patterns. However, the structural information conditioned on textual information shows significant shifts across patterns, which can be formulated as

$$P(\text{struct}(\mathcal{G}_i)) \approx P(\text{struct}(\mathcal{G}_j)), \tag{27}$$

$$P(\text{struct}(\mathcal{G}_i) \mid \text{text}(\mathcal{G}_i) = g) \neq P(\text{struct}(\mathcal{G}_j) \mid \text{text}(\mathcal{G}_j) = g). \tag{28}$$

These results provide intuitive evidence of joint structural-textual shifts across patterns, which motivates us to design our method to

capture such joint shifts instead of modeling structural and textual information separately.

C Experiment Details

C.1 Dataset Details

Table 4: Dataset statistics for the experiments.

Statistic	Enron-DI	GDELT-CI	ICEWS1819-CI
# Nodes	22,185	3,333	12,554
# Edges	340,910	73,231	91,276
# Timestamps	235	2,000	730
# Labels	10	20	20
# Tasks	20	10	10
# Labels per Task	10	2	2
Setting	domain-incremental	class-incremental	class-incremental

We summarize the dataset statistics in Table 4 and provide descriptions of these datasets as follows.

Enron-DI[1] is an email communication dataset collected from the Enron Corporation. The nodes represent employees, and the edges represent email communications between employees. The text attribute of each node is the information from the department and position of employees, if available, and the text attribute of each edge is the email content. This dataset contains 10 labels of edges in total, indicating the category of email. The most frequent 3 labels are "notes_inbox", "personal", and "deal_communication". We partition the dataset into 20 tasks based on the timestamp. Each task corresponds to a contiguous time interval and contains at least 15,000 interactions.

GDELT-CI[2] is a global event dataset collected from the Global Database of Events, Language, and Tone project. The nodes represent political entities such as countries, and the edges represent relationships between political entities such as "Make Statement". The text attribute of each node is the name of the entity, and the text attribute of each edge is the description of the relationship. We select the most frequent 20 labels of edges, and partition the dataset into 10 tasks based on the timestamp, where each task corresponds to a contiguous time interval and introduces 2 new labels.

ICEWS1819-CI[3] is also an event dataset collected from the Integrated Crisis Early Warning System project. The nodes represent political entities, and the edges represent relationships between them. The text attribute of each node includes the name, sector, and nationality. The text attribute of each edge is the description of the relationship. We select the most frequent 20 labels of edges, and partition the dataset into 10 tasks based on the timestamp, where each task corresponds to a contiguous time interval and introduces 2 new labels.

C.2 Baseline Details

We adopt several TGNN methods and dynamic graph LLM methods, with different continual learning approaches as baselines for comparisons.

- TGNNs and dynamic graph LLMs: **GraphMixer** [5] uses a conceptually and technically simple architecture with only MLPs and pooling to encode the temporal information of interactions. It avoids RNN [12] or attention mechanism [42] and still achieves competitive performance on dynamic graph learning tasks. **TGAT** [50] uses a self-attention mechanism as a building block and a functional time encoding technique, which efficiently aggregates neighborhood information to generate time-aware embeddings. **DyGFormer** [54] learns from nodes' historical first-hop interactions. It utilizes a neighbor co-occurrence encoding scheme and a patching technique, and then uses a transformer architecture to encode the temporal information of interactions. **CROSS** [58] is a dynamic graph LLM method that uses an LLM to dynamically extract the temporal semantics and generate cohesive representations unifying both semantics and structures. It then uses a co-encoder for synthesizing illuminating representations.
- Continual learning approaches: **EWC** [19] is a regularization-based method that selectively slows down the learning of parameters important to previous tasks, where the importance is estimated by the Fisher information matrix. **TWP** [30] is a regularization-based method that explicitly explores the local structures of the input graph and attempts to stabilize the parameters playing pivotal roles in the topological aggregation. **ER-GNN** [74] is a replay-based method that stores knowledge from previous tasks as experiences and replays them when learning new tasks to mitigate the catastrophic forgetting issue.

C.3 Implementation Details

We use Qwen3-8B as the default LLM backbone. We load the LLM in 4-bit quantization for memory efficiency and perform training with bfloat16 mixed precision. We inject a fusion mechanism every 4 layers in the LLM, and LoRA is applied to the attention weights W^{qkv} on the layers with the fusion mechanism. For our method's hyperparameters, we set $K = 5$ for the maximum number of experts on GDELT-CI and ICEWS1819-CI and $K = 10$ on Enron-DI, $M = 8$ for the size of the query-centric temporal subgraph, $\lambda = 0.5$ for the weight of invariance regularization, and $r = 64$ for the rank of the fusion mechanism. The factor size is set to 300, and the replay buffer size is set to at most 5% of the training data. We set $\alpha = 2, d_H = 0.6$ for routing, and set $\beta = 0.9$ for pattern-factor discrepancy calculation. Following DTGB [56], we adopt Bert [8] as the PLM for textual feature extraction.

For every method and every dataset, the learning rate is set to 10^{-4}, the batch size is set to 64, and the Adam optimizer [18] is used for optimization. The training is stopped after the validation performance does not improve for P consecutive epochs. P is set to 1 for LLM methods and set to 5 for non-LLM methods.

We run all experiments with 3 random seeds for LLM-based methods and 5 random seeds for TGNN-based methods. Other learning hyperparameters and model hyperparameters of TGNN-based methods are set following DTGB [56].

[1] https://www.cs.cmu.edu/~enron/
[2] https://www.gdeltproject.org/
[3] https://dataverse.harvard.edu/dataverse/icews