TGOnline: Enhancing Temporal Graph Learning with Adaptive Online Meta-Learning

Ruijie Wang
University of Illinois
Urbana-Champaign, USA
ruijiew2@illinois.edu

Jingyuan Huang
Zhejiang University
Zhejiang, China
jingyuanh.20@intl.zju.edu.cn

Yutong Zhang
Stanford University
California, USA
yutongz7@stanford.edu

Jinyang Li
University of Illinois
Urbana-Champaign, USA
jinyang7@illinois.edu

Yufeng Wang
University of Illinois
Urbana-Champaign, USA
yufengw2@illinois.edu

Wanyu Zhao
University of Illinois
Urbana-Champaign, USA
wanyu2@illinois.edu

Shengzhong Liu
Shanghai Jiao Tong University
Shanghai, China
shengzhong@sjtu.edu.cn

Charith Mendis
University of Illinois
Urbana-Champaign, USA
charithm@illinois.edu

Tarek Abdelzaher
University of Illinois
Urbana-Champaign, USA
zaher@illinois.edu

ABSTRACT

Temporal graphs, depicting time-evolving node connections through temporal edges, are extensively utilized in domains where temporal connection patterns are essential, such as recommender systems, financial networks, healthcare, and sensor networks. Despite recent advancements in temporal graph representation learning, performance degradation occurs with periodic collections of new temporal edges, owing to their dynamic nature and newly emerging information. This paper investigates online representation learning on temporal graphs, aiming for efficient updates of temporal models to sustain predictive performance during deployment. Unlike costly retraining or exclusive fine-tuning susceptible to catastrophic forgetting, our approach aims to distill information from previous model parameters and adapt it to newly gathered data. To this end, we propose TGOnline, an adaptive online meta-learning framework, tackling two key challenges. First, to distill valuable knowledge from complex temporal parameters, we establish an optimization objective that determines new parameters, either by leveraging global ones or by placing greater reliance on new data, where global parameters are meta-trained across various data collection periods to enhance temporal generalization. Second, to accelerate the online distillation process, we introduce an edge reduction mechanism that skips new edges lacking additional information and a node deduplication mechanism to prevent redundant computation within training batches on new data. Extensive experiments on four real-world temporal graphs demonstrate the effectiveness and efficiency of TGOnline for online representation learning, outperforming 18 state-of-the-art baselines. Notably, TGOnline not only outperforms the commonly utilized retraining strategy but also achieves a significant speedup of 30x.

SIGIR '24, July 14–18, 2024, Washington, DC, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0431-4/24/07.
https://doi.org/10.1145/3626772.3657791

CCS CONCEPTS

• **Computing methodologies → Machine learning**; • **Information systems → Information retrieval**.

KEYWORDS

Temporal Graph Learning, Efficient Online Learning, Meta-Learning

ACM Reference Format:
Ruijie Wang, Jingyuan Huang, Yutong Zhang, Jinyang Li, Yufeng Wang, Wanyu Zhao, Shengzhong Liu, Charith Mendis, and Tarek Abdelzaher. 2024. TGOnline: Enhancing Temporal Graph Learning with Adaptive Online Meta-Learning. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), July 14–18, 2024, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3626772.3657791

1 INTRODUCTION

Temporal graphs, depicting time-varying connections among nodes, have gained extensive utility across diverse applications. They typically manifest dynamic characteristics and undergo rapid topology changes over time. Real-world instances include interaction graphs that witness the emergence of new shopping records in E-commerce [38], follower-followee graphs where new users register on social media [5, 20, 35], and new events on temporal knowledge graphs [1, 13, 21, 23, 36, 37, 39], among others. Consequently, various Temporal Graph Neural Networks (TGNNs) have been proposed to capture the temporally evolving information, outperforming the static models in many tasks on temporal graphs [6, 18, 28, 29, 43].

Nevertheless, the disparity between the commonly used training/test setting in literature and the real-world deployment scenario constrains the utility of temporal models. Concretely, most literature deterministically splits temporal edges by time into training and testing data. They assume the availability of all training data before the start of the training process. Once the training procedure is completed, the deployment involves a fixed model tested on all testing data. Such a setting, which we refer to as *offline setting*, differs from the real-world scenarios, where the deployed model

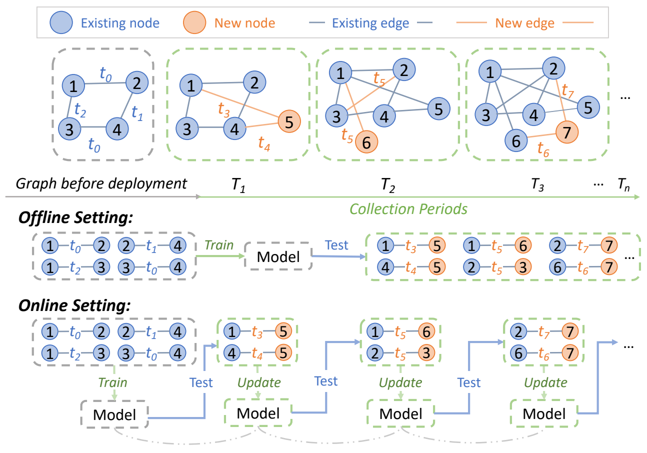


Figure 1: Task illustration. Our online setting is to update the deployed model effectively and efficiently based on periodically collected new temporal edges, ensuring sustained long-term performance.

can be periodically updated on newly acquired data, which we call *online setting* (as shown in Figure 1). Consequently, the observed superiority in an offline setting may not necessarily yield meaningful comparisons for real-world applications, which more closely align with online settings.

This paper aims to develop a temporal graph learning technique that facilitates effective and efficient online model updating strategy for sustainable performance over time, as shown in Figure 1. Different from the offline setting of temporal graph learning, a unique challenge in the online stage lies at how to incorporate the continuous appearance of new information (such as new edges and nodes over time [11]) and utilize it for model updates. Therefore, there is a need for an effective and efficient online updating strategy for temporal models to append valuable information from both pre-existing and new data. Inspired by the recent success in knowledge distillation [10], our approach revolves around a key philosophy: *can we distill enduringly valuable knowledge from old model parameter and adapt it onto freshly gathered data, instead of directly retraining on all data or solely fine-tuning on new data?* This method ensures a harmonious integration of both old and new information, preventing the excessive reuse of a substantial amount of old data and addressing the overfitting issues that may arise when simply fine-tuning advanced temporal models on new data. Consequently, it demands a novel training strategy that first extracts global knowledge for utilization across time steps from existing data and then seamlessly adapts it to newly gathered information under the guide of distillation, ensuring efficient adaptation and sustainable performance over time.

The fulfillment of the novel training strategy poses challenges in two aspects: **First**, considering the model complexity of representing temporal graphs, how to distill enduringly valuable knowledge from the sophisticated model parameters, especially for newly emerging nodes/edges, is non-trivial. **Second**, periodic online updates on new data underscore the importance of efficiency to prevent excessive time and resource consumption. Recent efforts have studied lightweight model updates on graphs in the online setting [3, 41, 55]. However, these methods primarily focus on updating shallow and static graph models, such as matrix factorization [41, 55] and static graph neural networks [3]. Their methods lack the capability of updating complex temporal models in the online setting to represent temporal information. Studies in the continual graph learning field explore a seemingly similar context, which aims to learn new patterns incrementally on evolving graphs [17, 24, 33, 45, 54]. However, their main objective is to sustain the model performance across old and emerging tasks, such as predicting a new class in node classification [22, 33] or reasoning on a new relation on knowledge graphs [17]. The goal falls outside the scope of our focus, which is centered on enhancing the same temporal task through the incorporation of newly collected data in the future, without newly introduced tasks.

We introduce a novel framework for online learning on temporal graphs, namely TGOnline. This temporal meta-training framework parameterizes knowledge through attentive sampling and aggregation from temporal neighbors, facilitating the learning of node temporal patterns. During the online stage, TGOnline fine-tunes global parameters, acquired from previous time steps, on newly collected data following each collection period. To promote the inheritance of enduringly valuable knowledge from old parameters and mitigate overfitting on potentially different and unseen distributions in new data, we employ the PAC-Bayes method [25] to analyze the optimization bound for fine-tuning on new data. This method acts as an adaptation regularizer, guiding the distillation process from old parameters and enhancing generalization over time. Crucially, it allows for the utilization of old parameters, eliminating the need for expensive model retraining on old data. To ensure the encoded global parameters carry enduringly valuable knowledge, we formulate a set of meta-tasks and meta-train the global parameters across data collection periods during the offline stage. This approach eliminates the need for manual configuration during the online fine-tuning stage, determining which information to distill and which to ignore. The optimization of the fine-tuning and meta-training procedures is achieved through a nested bi-level optimization process, comprising an inner-loop adaptation of global knowledge on newly collected data and an outer-loop training to meta-learn the global knowledge. To illustrate the effectiveness of our proposed training strategy, we implement and integrate it with an temporal attention module for a real-world temporal link prediction task. Additionally, to enhance online adaptation efficiency, we introduce an edge reduction mechanism that bypasses edge samples lacking additional information for fine-tuning. We also incorporate a node deduplication mechanism proposed in [42] to eliminate duplicated computations within batches. Empirically, extensive experiments conducted on four real-world streaming temporal graphs validate the effectiveness of TGOnline. It significantly outperforms 18 baselines, encompassing both static/temporal graph learning methods and online graph learning methods, achieving up to 8.8% relative gains on average in terms of Recall and NDCG. Moreover, our approach showcases superior efficiency compared to other temporal baselines, and its efficiency is on par with that of simple static models.

Overall, our contributions can be summarized as follows:

- **Problem setting**: We study and evaluate temporal graph learning in a more practical online setting, where users can gain access to the new data and peodically update models for future tests;
- **Novel framework**: We propose a temporal meta-learning framework TGOnline with efficiency optimization. It extracts enduringly valuable knowledge across data collection periods during the offline phase and efficiently fine-tunes the model to encode newly emerging patterns during the online phase.
- **Extensive evaluation**: Empirical experiments on four real-world temporal graphs demonstrate the effectiveness and efficiency of TGOnline, compared with 18 state-of-the-art baselines.

2 PRELIMINARY

*Definition 2.1 (***Temporal Graphs***).* A temporal graph consists of a set of temporal edges $\{(u, v, t)\}$, where $u, v \in \mathcal{V}$ denotes the source and target nodes, t denotes the specific timestamp attached to each temporal edge.

Existing studies are commonly categorized into two types: temporal graphs with discrete graph snapshots (DTDGs) [6, 14, 26, 29] and those with continuous edge timestamps (CTDGs) [18, 28, 43]. In most real-world graphs, t represents a continuous timestamp that reflects the actual time of each edge occurrence. Therefore, in this paper, we default to studying continuous temporal graphs (CTDGs), although the proposed method can effortlessly be applied to discrete temporal graphs by utilizing a discrete t.

In the offline setting commonly utilized in literature, the temporal edges are split into training set and testing set, *i.e.*, $\mathcal{D}^{\text{train}} = \{(u, v, t)|0 \leq t < T\}$ and $\mathcal{D}^{\text{test}} = \{(u, v, t)|t \geq T\}$. They test a non-update model trained on $\mathcal{D}^{\text{train}}$ on all edges in $\mathcal{D}^{\text{test}}$. However, in reality, new edges in $\mathcal{D}^{\text{test}}$ are collected incrementally, meaning that we can utilize the newly collected data to update the model periodically before a newer test. Therefore, we study a more practical online setting on temporal graphs, where edges in the test set $\mathcal{D}^{\text{test}}$ can be further divided by collection periods, *i.e.*, $\mathcal{D}^{\text{test}} = \bigcup_{i=1}^{N} \mathcal{D}^{T_i}$, and $\mathcal{D}^{T_i} = \{(u, v, t)|T_{i-1} \leq t < T_i\}$. Figure 2a presents the ratio of newly introduced nodes and edges absent in the temporal graphs before this period. The online learning on temporal graphs is formalized as follows:

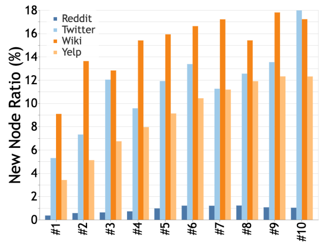
*Definition 2.2 (***Online Learning on Temporal Graphs***).* Let Θ denote the deployed temporal models. On each collection period T_i, we consider the following two steps during online learning:

$$\begin{aligned} &\text{Step-1}: \Theta^{T_{i-1}} \underset{\text{test}}{\rightarrow} \mathcal{D}^{T_i}, \\ &\text{Step-2}: \bigcup \left(\mathcal{D}^{\text{train}}(\text{optional}), \cdots, \mathcal{D}^{T_i}\right) \underset{\text{update}}{\rightarrow} \Theta^{T_i}, \end{aligned} \tag{1}$$

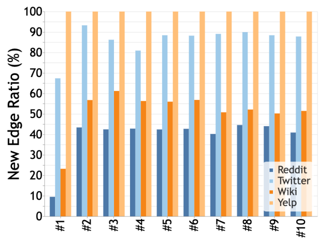
and our ultimate goal is to maximize the average performance tested in each online collection period (Step-1).

3 TGONLINE FRAMEWORK

In this section, we present a novel framework TGOnline to solve the online learning task on temporal graphs, as shown in Figure 3.



(a) New node emergence.



(b) New edge emergence.

Figure 2: Motivating examples. New nodes and edges continuously emerge on four real-world temporal graphs, leading to significant performance decay.

3.1 Framework Overview

During the online stage, in handling the new data collection $\mathcal{D}^{T_i}$, our approach is guided by a fundamental principle: TGOnline distills enduringly valuable knowledge from the global model parameter Θ across different collection periods and adapts it onto new data as Θ^{T_i} for utilization. This stands in contrast to the direct retraining on the entire dataset $\bigcup \left(\mathcal{D}^{\text{train}}, \cdots, \mathcal{D}^{T_i}\right)$ or the exclusive fine-tuning the old $\Theta^{T_{i-1}}$ on the new data $\mathcal{D}^{T_i}$. It guarantees the utilization of valuable information from both previous and current data, while simultaneously conserving computational resources.

To this end, TGOnline mainly consists of two design components:

- **Distillation-Guided Online Model Updates** aims to infer the updated model parameters Θ^{T_i} based on fine-tuning on new data $\mathcal{D}^{T_i}$ and distilling knowledge from global model parameters across collection periods Θ (Section 3.2);
- **Temporal Meta Training** involves the extraction of global model parameters Θ from the previous dataset during offline training, aiming to acquire generalizable information across time (such as how to encode time, how to aggregate neighbor's information, etc). It engages in a nested bi-level optimization process on a set of meta-tasks during offline training. Each task is simulated to predict new edges by based on existing data, and it mimics the online task's behavior after the model is deployed (Section 3.3);

we implement the strategy above on an attentive temporal module for representing temporal graphs (Section 3.4), and further introduce acceleration techniques to improve efficiency of the Distillation-Guided Online Model Updates (Section 3.5). They include *edge reduction* that bypasses new edges without novel information and *node deduplication* that avoids representation computation on duplicated temporal neighbors.

3.2 Distillation-Guided Online Model Updates

The online update of the deployed model is essentially estimating the new parameter distribution $p(\Theta^{T_i})$ given the newly collected data $\mathcal{D}^{T_i}$ based on the prior estimation of the global knowledge $p(\Theta)$. As pointed out by [54, 55], a unique challenge arises to avoid the overfitting and catastrophic forgetting issues on the new data. Otherwise, informative knowledge presented in $p(\Theta)$ might be dominated and biased by the new observations. For example, the representation of old nodes (that are not active in the new period)

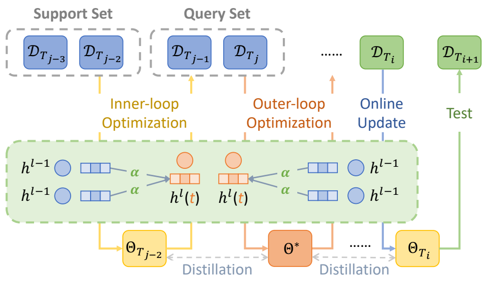


Figure 3: Framework overview. TGOnline consists of a distillation-guided online model updates and a temporal meta training for learning global temporal parameters.

should be largely preserve and instead of being misled by the new observations.

One might view the new parameter distribution $p(\Theta^{T_i})$ as a posterior distribution conditioning on the new data. However, it is hard to compute the posterior given the prior $p(\Theta)$ and the observations $\mathcal{D}^{T_i}$, as it requires us to have already access to the likelihood of $\mathcal{D}^{T_i}$, which needs to be estimated first. To resolve this, we instead adopt the PAC-Bayes method [25, 31], which allows one to learn a parameter posterior that fits the observations without knowing the distribution of the observations. We propose the following theorem to provide the optimization objective for the online model updates:

Theorem 3.1 (PAC-Bayes Bound on New Data). *Let $p(\Theta)$ denote the prior distribution gained by the global knowledge across tasks, $p(\Theta^{T_i})$ denote the empirical estimation of new parameters on new data, $|\cdot|$ denote the set size. For any $\delta \in (0, 1)$ and learned prior $p(\Theta)$, with probability at least $1 - \delta$ over new data $\mathcal{D}^{T_i}$, the upper bound of the online optimization loss to be optimized on the new data is given by:*

$$\mathcal{L}^{\text{Online}}(\Theta^{T_i}, \mathcal{D}^{T_i}) \le \mathcal{L}(\Theta^{T_i}, \mathcal{D}^{T_i}) + \sqrt{\frac{\mathbb{KL}(p(\Theta^{T_i})\|p(\Theta)) + \log\frac{|\mathcal{D}^{T_i}|}{\delta}}{2|\mathcal{D}^{T_i}| - 1}}, \tag{2}$$

where $\mathcal{L}^{\text{Online}}(\Theta^{T_i}, \mathcal{D}^{T_i})$ denotes the online loss to update Θ^{T_i} on data $\mathcal{D}^{T_i}$, and $\mathcal{L}(\Theta^{T_i}, \mathcal{D}^{T_i})$ denotes the normal loss function determined by underlying temporal model.

Remark. Readers can refer to Appendix A.1 for proof. The Eq. (2) can be interpreted as a combination of empirical loss on the new data in the new time interval and a distillation regularizer of parameter distribution with the global parameter distribution in the form of KL-divergence. The regularizer along the time domain can guarantee that the online update is trained only on new data but without overfitting in specific time intervals. Thus, we can improve the generalization ability of our online updates procedure, which can be formalized as follows:

$$\Theta^{T_i} \leftarrow \Theta^{T_i} - \eta\frac{\partial\mathcal{L}^{\text{Online}}(\Theta^{T_i}, \mathcal{D}^{T_i})}{\partial\Theta^{T_i}}. \tag{3}$$

3.3 Temporal Meta Training

In this section, we discuss how to ensure the global parameter Θ in Eq. 2 encode generalizable knowledge across collection periods so that it provides correct distillation guideline for online model updates, as stated in Eq. (3). To achieve, we view extracting the

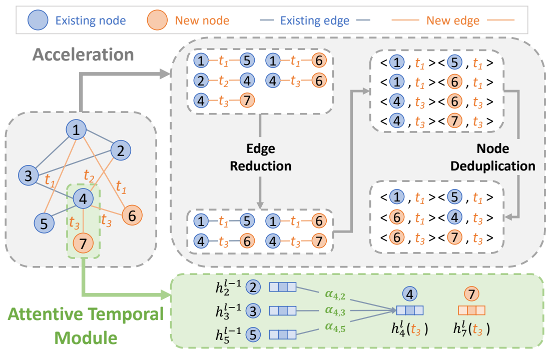


Figure 4: The attentive temporal module and the acceleration including edge reduction and node deduplication.

global parameters as a meta-task and meta-train them on a set of tasks simulated during offline stage.

Meta-Learning Setup. We follow model-agnostic meta-learning (MAML) [4] and formulate a task as adapting global knowledge on recently collected data (organized as *support set*) to update the model for future prediction ((organized as *query set*). In each task at the data collection time T_i, the global parameters Θ can be adapted on a support set to mimic the online update procedure. A query set in the future is utilized to measure the quality of global parameters Θ across time periods. Future prediction serves as an indicator of how effectively the distillation guide provided by Θ performs.

Concretely, the support set $\mathcal{S}^{T_i}$ consists of data from the new and several recent time periods: $\mathcal{S}^{T_i} = \bigcup_{T_i - T_s < T \le T_i} \mathcal{D}^T$, and the query set $\mathcal{Q}^{T_i}$ consists of data from several future time periods: $\mathcal{Q}^{T_i} = \bigcup_{T_i < T \le T_i + T_q} \mathcal{D}^T$, where T_s and T_q are hyperparameters to control the length of time periods to be considered. Each distinct task, denoted as $(\mathcal{S}^{T_i}, \mathcal{Q}^{T_i})$, is designed to facilitate model updates using the support set $\mathcal{S}^{T_i}$, with the ultimate aim of optimizing performance on the future query set $\mathcal{Q}^{T_i}$.

Bi-Level Optimization. We then introduce how to meta-train global parameters Θ for better guiding the online model updates. During the meta-training phase, we simulate a set of tasks by slicing time windows within the observed time range $(0, T)$ and construct the corresponding pairs of support and query sets, with the assumption that the support set is the freshly collected data, and the query set is the future data that awaits model deployment and evaluation. Thus, we can mimic the real online updates by adapting the global parameter Θ, on the support set via several fine-tuning steps (inner loop). And the global knowledge can be further improved by tuning the model on each query set (outer loop), as the quality of global parameters can be quantitively measured by how well the model performs on future data. Such a bi-level optimization can be detailed as follows:

$$\begin{aligned} \Theta^{\star} &\longleftarrow \arg\min_{\Theta} \mathbb{E}_{T_i}\left[\mathcal{L}(\Theta^{T_i}, \mathcal{Q}^{T_i})\right] \text{ (Outer Loop)}, \\ \text{where } \Theta^{T_i} &= \Theta^{T_i} - \eta\frac{\partial\mathcal{L}^{\text{Online}}(\Theta^{T_i}, \mathcal{S}^{T_i})}{\partial\Theta^{T_i}} \text{ (Inner Loop)}, \end{aligned} \tag{4}$$

where Θ denotes the global parameters across tasks, $\Theta^{\star}$ denotes the optimal global parameters after the meta training, and Θ^{T_i} denotes the updated model parameters at the T_i-th collection period,

Algorithm 1: Offline meta-optimization algorithm.

```
Input: Temporal graph $\mathcal{D}^{\text{train}} = \{\mathcal{D}^0, \mathcal{D}^1, \cdots, \mathcal{D}^T\}$, $T_s$, $T_q$
Output: The global parameter for distillation $\Theta^\star$
1  Construct the support/query sets based on $\mathcal{D}^{\text{train}}$, $T_s$, $T_q$;
2  while $\Theta$ not converge do
3      # Outer loop: for each task at $T_i$ do
4          # Inner loop:
5          Calculate PAC-Bayes bound on $\mathrm{S}^{T_i}$ by Eq. (2);
6          Optimize $\Theta^{T_i}$ by Eq. (3);
7          Calculate predictive loss on query set $\mathcal{L}(\Theta^{T_i}, \mathcal{Q}^{T_i})$;
8      end
9      Calculate loss across tasks $\mathbb{E}_{T_i}\left[\mathcal{L}(\Theta^{T_i}, \mathcal{Q}^{T_i})\right]$;
10     $\Theta \longleftarrow \arg\min_\Theta \mathbb{E}_{T_i}\left[\mathcal{L}(\Theta^{T_i}, \mathcal{Q}^{T_i})\right]$;
11 end
12 $\Theta^\star \longleftarrow \Theta$.
```

Algorithm 2: Online model update algorithm.

```
Input: New data $\mathcal{D}^{T_i}$, Global model parameter $\Theta^\star$
Output: Updated parameter $\Theta^{T_i}$ for new period
1  Construct $\mathcal{S}^{T_i}$ based on recent $T_s$ collections;
2  Perform edge reduction and node deduplication on $\mathcal{S}^{T_i}$;
3  Calculate PAC-Bayes bound on $\mathcal{S}^{T_i}$ by Eq. (2);
4  Optimize $\Theta^{T_i}$ by Eq. (3) untill converge.
```

$\mathcal{L}^{\text{Online}}(\Theta^{T_i}, \mathcal{S}^{T_i})$ denotes the online loss to update Θ^{T_i} on data $\mathcal{S}^{T_i}$, and $\mathcal{L}(\Theta^{T_i}, \mathcal{Q}^{T_i})$ denotes the normal loss function determined by underlying temporal model.

The bi-level optimization procedure outlined in Eq. (4) emulates the iterative refinement process of the deployed model in the online setting. The inner loop optimization mirrors the strategy for swiftly adapting the model to recently acquired data by distilling knowledge from the fixed Θ. Meanwhile, the outer loop optimization is geared towards tuning the global parameter Θ shared across various tasks to the optimal one $\Theta^\star$, enhancing its capacity to encapsulate generalization information from the existing data. During the offline training phase, both the global parameter Θ and the online updating procedure can be learned via the nested meta-optimization procedure, which is iterated over time. During the online updating phase, the optimal knowledge $\Theta^\star$ can be further utilized to provide distillation guide for fine-tuning on the new data by optimizing the inner loop. The procedure of offline/online training is summarized in Algorithm 1 and Algorithm 2 respectively.

3.4 The Attentive Temporal Module

Towards the learning objective above, we introduce the detailed model parameterized by Θ and the specific loss function for temporal tasks. Figure 4 shows an one-layer attentive module for illustration. The attentive temporal model parameterized by Θ is designed to measure the plausibility of each temporal edge, which represents each node u into a low-dimensional latent space at each time t: $\mathbf{h}_u(t) \in \mathbb{R}^d$. On a temporal graph, nodes $u \in \mathcal{V}$ evolve as they interact with different neighbors over time. Such temporally interacted nodes are defined as temporal neighbors. Therefore, we aim to model the temporal pattern of each node u by encoding the changes of temporal neighbors.

Table 1: The dataset statistics.

Dataset	#Interaction	#User	#Item	Density	Split
Wiki	157,474	8,227	1,000	0.019141	30/2/8
Reddit	672,447	10,000	984	0.068338	30/2/8
Twitter	134,291	8,780	1,333	0.011474	30/2/8
Yelp	1,266,728	63,228	59,375	0.000337	30/2/8

Towards this goal, this module first samples temporal neighbors $\mathcal{N}_u(t) = (v_i, t_i)$ from the existing temporal graph for each node $u \in \mathcal{V}$. $\mathcal{N}_u(t)$ consists of a set of the most recently interacted nodes no later than time t. Then it attentively aggregates information from the temporal neighbors. Specifically, given the temporal neighbor $\mathcal{N}_u(t)$, we represent the entity u as $\mathbf{h}_u(t)$ at time t:

$$\mathbf{h}_u^l(t) = \text{ReLU}\left(\sum_{(v_i,t_i)\in\mathcal{N}_u(t)} \alpha_{u,v_i}^l \left(\mathbf{h}_{v_i}^{l-1}(t_i)\mathbf{W}\right)\right), \tag{5}$$

$$\begin{aligned} \alpha_{u,v_i}^l &= \frac{\exp(q_{u,v_i}^l)}{\sum_{(v_k,t_k)\in\mathcal{N}_u(t)} \exp(q_{u,v_k}^l)}, \\ q_{u,v_k}^l &= \mathbf{a}\left(\mathbf{h}_u^{l-1} \| \mathbf{h}_{v_k}^{l-1} \| \kappa(t - t_k)\right), \end{aligned} \tag{6}$$

where q_{u,v_k}^l measures pairwise importance by considering the node features of u and each v_k, and time feature, $\mathbf{a} \in \mathbb{R}^{3d}$ is the shared parameter in the attention mechanism. Following [43], we adopt random Fourier features as time encoding $\kappa(\Delta t)$ to reflect the time difference. Similarly, the edge feature, if available, can be concatenated together in the pairwise importance measurement q_{u,v_k}^l.

To measure the probability of each possible temporal edge, we utilize inner product [15] as the score function $p = \sigma((\mathbf{h}_u^t, \mathbf{h}_v^t))$, where $\sigma(\cdot)$ denotes the *Sigmoid* activation function. To optimize the parameter Θ for a task on either the support or query set, we minimize the loss for each temporal edge to train the model Θ. Taking the support set $\mathcal{S}^{T_i}$ as an example:

$$\mathcal{L}(\Theta, \mathcal{S}^{T_i}) = \mathbb{E}\left[-y_i \log(p_i) - (1 - y_i)\log(1 - p_i)\right], \tag{7}$$

where $y_i = 1$ if the edge $(u_i, v_i, t_i) \in \mathcal{S}^{T_i}$, and $y_i = 0$ otherwise.

3.5 Acceleration for Online Update

Edge Reduction. We analyze the time complexity of the inner loop optimization, which is used for online model updating. Let b denote the number of temporal neighbors for each node, l denote the number of attention layers, N denote the negative sampling factor, and the overall complexity is $O\left(lb^2N|\mathcal{S}^{T_i}|\right)$, which grows linearly with $|\mathcal{S}^{T_i}|$. To further expedite the online update process, we explore methods to diminish the size of $|\mathcal{S}^{T_i}|$ using the edge reduction technique, all while maintaining its efficacy. We propose a straightforward approach that involves bypassing edges where the two nodes are already 1- or 2-hop neighbors in recent time steps. This strategy is based on the premise that such edges might not introduce any supplementary information, as the existing historical data and PAC-Bayes bound have already covered their contribution. For example, we skip edge $(2, 4, t_2)$ in Figure 4.

Node Deduplication. The new edges are grouped into batches for the distillation-guided online model updates. In the underlying attentive temporal module, each edge is separated into source and

Table 2: The overall performance of baseline models and TGOnline. Average results of 5 independent runs are reported. For baselines tailored for offline training scenarios, we utilize both retraining and fine-tuning techniques, and retaining the approach with better performance.

Dataset	Wiki		Reddit		Twitter		Yelp	
Performance	**Recall@20**	**NDCG@20**	**Recall@20**	**NDCG@20**	**Recall@20**	**NDCG@20**	**Recall@20**	**NDCG@20**
MF	0.662 ± 0.013	0.319 ± 0.005	0.750 ± 0.017	0.522 ± 0.004	0.172 ± 0.007	0.099 ± 0.002	0.183 ± 0.002	0.125 ± 0.002
GAE	0.481± 0.018	0.221 ± 0.012	0.708 ± 0.013	0.481 ± 0.017	0.600 ± 0.000	0.307 ± 0.001	0.213 ± 0.001	0.091 ± 0.000
GAT	0.540 ± 0.007	0.328 ± 0.012	0.706 ± 0.007	0.490 ± 0.007	0.340 ± 0.009	0.154 ± 0.023	0.091 ± 0.002	0.038 ± 0.001
GIN	0.409 ± 0.016	0.193 ± 0.006	0.568 ± 0.055	0.345 ± 0.050	0.519 ± 0.009	0.287 ± 0.008	0.220 ± 0.001	0.093 ± 0.001
LightGCN	0.698 ± 0.003	0.503 ± 0.003	0.741 ± 0.001	0.561 ± 0.003	0.568 ± 0.001	0.291 ± 0.001	0.224 ± 0.003	0.092 ± 0.001
GRU4Rec	0.080 ± 0.001	0.046 ± 0.001	0.048 ± 0.001	0.037 ± 0.000	0.056 ± 0.000	0.030 ± 0.000	0.026 ± 0.000	0.010 ± 0.000
JODIE	0.239 ± 0.015	0.198 ± 0.024	0.211 ± 0.011	0.183 ± 0.025	0.139 ± 0.015	0.098 ± 0.007	OOM	OOM
EGCN-H/O	0.089 ± 0.002	0.039 ± 0.001	0.471 ± 0.007	0.285 ± 0.004	0.250 ± 0.010	0.124 ± 0.007	OOM	OOM
VGRNN	0.048 ± 0.030	0.025 ± 0.016	0.389 ± 0.073	0.193± 0.036	0.389 ± 0.073	0.193 ± 0.036	0.165 ± 0.014	0.071 ± 0.006
Euler	0.040 ± 0.010	0.018 ± 0.005	0.484 ± 0.032	0.242 ± 0.017	0.600 ± 0.003	0.334 ± 0.002	0.070 ± 0.021	0.028 ± 0.010
DySAT	0.442 ± 0.010	0.224 ± 0.002	0.668 ± 0.002	0.426 ± 0.007	0.410 ± 0.011	0.176 ± 0.011	0.020 ± 0.000	0.007 ± 0.000
DIDA	0.601 ± 0.007	0.510 ± 0.009	0.617 ± 0.015	0.392 ± 0.025	0.551 ± 0.016	0.391 ± 0.012	0.242 ± 0.002	0.136 ± 0.002
TGAT	0.664 ± 0.010	0.529 ± 0.008	0.744 ± 0.011	0.618 ± 0.017	0.604 ± 0.010	0.231 ± 0.006	0.215 ± 0.012	0.121 ± 0.013
SPMF	0.585 ± 0.007	0.358 ± 0.006	0.741 ± 0.001	0.507 ± 0.002	0.022 ± 0.003	0.007 ± 0.001	0.166 ± 0.001	0.100 ± 0.001
SML	0.374 ± 0.023	0.190 ± 0.017	0.704 ± 0.013	0.455 ± 0.016	0.500 ± 0.071	0.250 ± 0.049	0.177 ± 0.010	0.111 ± 0.003
IGC	0.685 ± 0.014	0.526 ± 0.011	0.754 ± 0.011	0.589 ± 0.010	0.577 ± 0.002	0.335 ± 0.005	0.248 ± 0.008	0.132 ± 0.006
ROLAND	0.681 ± 0.015	0.536 ± 0.028	0.757 ± 0.009	0.592 ± 0.019	0.561 ± 0.015	0.324 ± 0.021	0.231 ± 0.012	0.129 ± 0.021
MetaDyGNN	0.667 ± 0.030	0.510 ± 0.018	0.752 ± 0.023	0.610 ± 0.031	0.581 ± 0.030	0.350 ± 0.019	0.251 ± 0.024	0.129 ± 0.011
TGOnline	**0.716 ± 0.017**	**0.575 ± 0.005**	**0.807 ± 0.006**	**0.645 ± 0.010**	**0.644 ± 0.002**	**0.475 ± 0.002**	**0.272 ± 0.005**	**0.157 ± 0.002**
Gains (%)	*2.6*	*8.7*	*7.0*	*4.4*	*5.9*	*8.0*	*9.8*	*15.4*

target nodes using the same timestamp, creating node-timestamp pairs. Since new edges can connect the same nodes at the same time, leading to significant duplication of such pairs, such as $(1, t_1)$ and $(4, t_3)$ in Figure 4, we employ a node deduplication technique introduced in [42] to eliminate redundant computations.

4 EXPERIMENTS

4.1 Datasets

We collect four real-world temporal graphs as follows:

- **Wiki** [19]: The dataset is a public collection of edits made by users who contributed at least 5 edits to the 1,000 most edited Wikipedia pages within one month. The temporal edges represent timestamped historical edits.
- **Reddit** [19]: The public Reddit post dataset comprises one month of user posts from the 1,000 most active subreddits and the 10,000 most active users. The temporal edges are the timestamped posting requests.
- **Twitter**: The dataset collected from Twitter constitutes user-hashtag interactions, whose nodes include users and hashtags, and whose links represent who-post-what interaction records. To maintain data quality, we excluded the top 10 users and top 20 hashtags with abnormal activation levels and users/hashtags with fewer than 15 occurrences, considered invalid for analysis.
- **Yelp** [1]: The dataset is collected from the Yelp Challenge 2018 and consists of user-business interactions with ratings of 4 and 5 points after 2010. Inactive users with fewer than 8 interactions are excluded.

For each dataset, we sort all interactions in chronological order and split them into 40 timesteps, each containing an equal number of interactions. We further split 40 periods into 30/2/8 for *offline training/online validating/online testing* phases. To be concrete:

- **Offline training phase** is utilized to train both baselines as well as TGOnline before deployment. The data in this phase are assumed to be given at once;
- **Online phase** is the phase where we utilize the online data to first test the deployed models and then to update the models. The average performance in the first 2 periods is selected as an indicator to choose important hyperparameters for the online testing phase, and the average testing performance on the remaining 8 periods is reported.

4.2 Experimental Setup

4.2.1 Baselines. We compare 18 state-of-the-art baselines from three related areas. Due to page limitations, we refrain from providing a detailed description for each baseline. We report baseline setup details in Appendix A.2.

- Static graph learning methods: **Matrix Factorization (MF)** [27], **GAE** [15], **GAT** [32], **GIN** [44], **LightGCN** [8];
- Temporal graph learning methods: **GRU4Rec** [9], **JODIE** [18], **EGCN-H/O** [26], **VGRNN** [6], **Euler** [14], **DySAT** [29], **DIDA** [56], **TGAT** [43];
- Online graph learning methods: **SPMF** [41], **SML** [55], **IGC** [3], **ROLAND** [53], **MetaDyGNN** [51].

4.2.2 Evaluation Protocol and Metrics. We evaluate online link prediction tasks in a retrieval setting. For each link collected during the online phase, given the observed user nodes, we compare the predicted top-K ranking list of missing items with the ground-truth item in each testing time step. We adopt two widely-used evaluation protocols: *Recall@K* and *NDCG@K*, where $K = \{5, 10, 20\}$.

4.2.3 Setup and Implementation. All baselines and TGOnline are first evaluated and then updated on the new interactions in each period during the online validating/online testing phases. For baselines tailored for offline training scenarios, we utilize both retraining

[1]https://www.yelp.com/dataset/

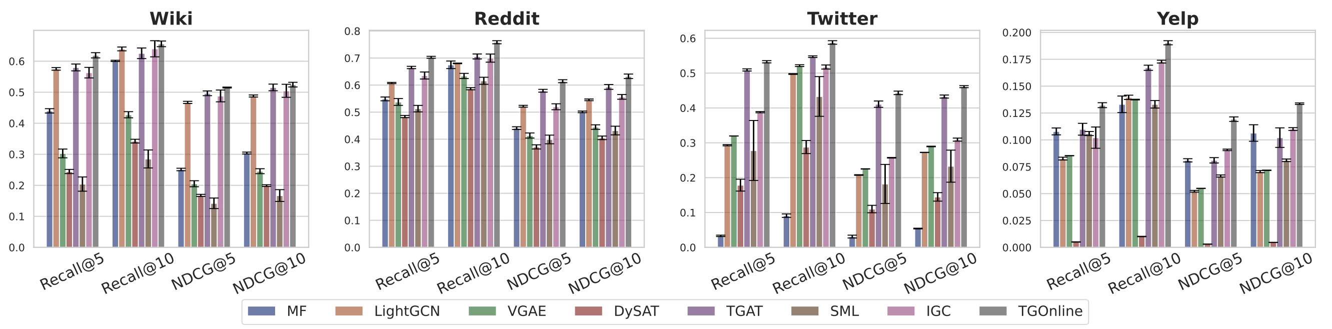


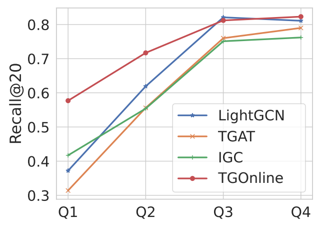
Figure 5: Detailed Recall@K and NDCG@K when $K = \{5, 10\}$. The average results of 5 independent runs are reported. TGOnline achieves the best performance, with 8.8% relative gains over the second-best results on average.

and fine-tuning techniques and retain the approach with better performance. During the evaluation, we tune hyperparameters based on *Recall@20* on the online validating phase, and report the average performance on the remaining periods of the online testing phase. We train all baseline models and TGOnline on the same GPUs (GeForce RTX 3090) and CPUs (AMD Ryzen Threadripper 3970X 32-Core Processor).

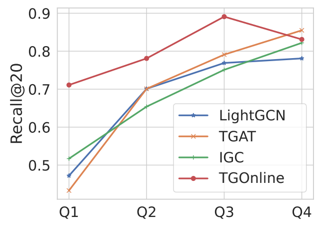
To construct the task set for the meta-learning formulation, we set temporal edges in $K_s = K_q = 2$, meaning we consider 2 consecutive historical periods as support and query se. Such a choice enable us to i) consider sufficient historical information to update the model online; ii) utilize relatively long-term performance as instant feedback for learning the global parameters in the outer-loop optimization; iii) maintain acceptable efficiency. Next, we report the choices of hyperparameters. For model training, we set the maximum number of epochs as 200 for offline training. We keep the dimension of all embeddings as 128. For the sake of efficiency, we set the neighbor budget b of temporal neighbor sampler as 16, and employ 2 neighborhood aggregation layers in temporal encoder. We perform 20 steps of gradient descent for inner loop optimization. We mainly tune inner/outer loop learning rate η and β in range $\{0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005, 0.00001\}$. For Wiki, Reddit and Twitter, we set $\eta = \beta = 0.0001$. For Yelp, we set $\eta = 0.0005$, and $\beta = 0.0001$. We report other baseline setup details in Appendix A.2.

4.3 Main Results

Overall Performance. We first discuss the main results on four datasets. Table 2 shows the overall evaluation in terms of Recall@20 and NDCG@20, and Figure 5 shows the comparison with several strong baselines with $K = \{5, 10\}$. TGOnline consistently outperforms all baseline models, exhibiting an average relative improvement of 8.8%. Intriguingly, certain prominent dynamic models, like GRU4Rec, EGCN, VGRNN, and Euler, demonstrate unexpectedly suboptimal results, even underperforming the static baseline models. We hypothesize that it is caused by too long a graph sequence, where these dynamic models struggle to effectively extract valuable knowledge from the historical data that are still useful on new data patterns. While TGAT and IGC (an online algorithm based on LightGCN) generally outperform other baselines, their results still fall short of our approach. This deficiency could be attributed



(a) Wiki.



(b) Reddit.

Figure 6: Performance across node quartiles on degree level. (Q1: lowest, Q4: highest). TGOnline has higher gains for nodes with sparse connections (quartiles Q1-Q3).

to the absence of a tailored online updating strategy specifically for the temporal module. The superior performance of TGOnline demonstrates the efficacy of the proposed temporal meta-training strategy for online link prediction.

4.4 Performance of Diverse Node Groups

The key indicator to evaluate the online algorithm is how well the model performs on newly emerging nodes with a few edges, as new nodes continuously join the graph in the real world. We divide nodes into quartiles by their degree levels, which is the number of participating edges per node. Figure 6 shows the performance distribution for each node group, from Q1 (lowest degree level) to Q4 (highest degree level), compared with three strong baselines. The major improvement of TGOnline comes from the nodes with few links (Q1, Q2), as it can borrow useful information from the high-resource nodes to the low-resource nodes via the extraction of global knowledge.

4.5 Ablation Study

We conduct the following ablation studies to evaluate performance improvements brought by the temporal meta-training strategy 1) **Retraining** directly retrains our attentive temporal model on all data; 2) **Fine-tuning** fine-tunes our attentive temporal model only on new data; 3) **TGOnline w/o PAC-Bayes Bound** removes the PAC-Bayes bound in the inner loop optimization; 4) **TGOnline w/o Edge Reduction** update the global model on all edges in the new data without skipping any sample; 5) **TGOnline w/ Random**

Table 3: Ablation studies measured by Recall@20 and NDCG@20. Online update time is also reported.

Dataset	Wiki		Reddit	
Metrics	**Recall**	**NDCG**	**Recall**	**NDCG**
Retraining	0.669	0.540	0.744	0.628
Fine-tuning	0.657	0.531	0.730	0.611
TGOnline	0.716	0.575	0.807	0.645
w/o PAC-Bayes	0.701	0.569	0.793	0.623
w/o Edge Reduction	**0.722**	**0.589**	**0.820**	**0.660**
w/ Random Reduction	0.690	0.557	0.770	0.629

Reduction reduce the same amount of training samples during the online phase by random selection.

Table 3 shows the evaluation results. Both retraining and fine-tuning approaches for the attentive temporal model fail to outperform TGOnline. The retraining strategy might be dominated by the existing data, leading to an excessive focus on historical patterns and potentially overlooking new patterns. Conversely, the fine-tuning strategy can easily result in overfitting on the new data. Upon excluding the PAC-Bayes bound from the inner loop objective, we observe a decline in performance, underscoring the bound's role in fortifying the stability and generalizability of the online updating procedure. We also delve into the impact of the simple edge reduction heuristic. While TGOnline without this heuristic yields a marginal performance boost, it prolongs online training time by up to 2× times. Substituting the heuristic with a random strategy significantly compromises performance. These outcomes underscore the value of our straightforward yet effective edge reduction algorithm, which accelerates the online update process without inflicting significant performance degradation.

4.6 Efficiency Analysis

Table 3 also reports the online updating times for each variant, quantifying the time taken per online epoch in seconds. Notably, TGOnline accomplishes swifter online updates compared to simple fine-tuning. This is due to the edge reduction algorithm employed by TGOnline, which effectively reduces the volume of training samples during the online phase in contrast to the standard fine-tuning approach. The efficacy of the edge reduction heuristic is further highlighted when contrasted with the variant that omits it. This heuristic effectively halves the online running time without introducing a notable drop in performance. This is because the omitted edges fail to contribute substantially new information for model training, and their contribution is already covered in the global knowledge regulated by the PAC-Bayes bound.

We report the online updating efficiency comparison of representative baselines in Table 4. Compared with dynamic models (DySAT, TGAT) and online updating models (IGC, MetaDyGNN, ROLAND), TGOnline achieves the best efficiecy. Notably, TGOnline achieves compatible efficiency with simple static models (VGAE and LightGCN) and achives better performance. This is because our meta-training algorithm updates the complex model parameters efficiently without expensive retraining on full dataset. And the proposed edge reduction and node deduplication further accerlate the online updating process.

Table 4: Efficiency comparison on the Reddit dataset.

Baselines	VGAE	LightGCN	DySAT	TGAT	IGC	MetaDyGNN	ROLAND	Ours
Recall	.717	.741	.668	.744	.754	.752	.757	**.807**
Time (s)	8.1	3.6	73.1	1490.3	18.1	1627.8	70.3	10.1

5 RELATED WORK

Temporal Graph Learning. Temporal graph learning has been attracting numerous research interests in the community [9, 18, 26, 28, 29, 43]. Recently, temporal graph neural networks (TGNNs) that based on Graph Neural Networks (GNNs) [7, 16, 30, 34, 46–50] have achieved the state-of-the-art performance, which is divided into two categories: *discrete methods* that organize the temporal graphs as discrete snapshot sequences (CTDGs) and utilize sequential modeling including RNNs (*e.g.*, EGCN [26], Euler [14], VGRNN [6]) and transformers (*e.g.*, DySAT [29]) to learn the evolution of node representations, *continuous methods* that operates on temporal graphs with continuous edge timestamps (CTDGs) (*e.g.*, TGAT [43], TGN [28]). Ours is designed to handle CTDGs. However, despite the numerous existing efforts, how to effectively and efficiently update temporal graph models on the online setting still remains relatively unexplored. Recent efforts have studied lightweight model updates on static graphs in the online setting [3, 41, 55]. SPMF [41] skillfully sampled for retraining to represent long-term preference and rank the items by an optimization framework. SML [55] introduces a meta-learning approach that captures the long and short-term embeddings, saving time and memory. IGC [3] re-activates the previous nodes and updates only the new-neighbor-related parameters to speed up the retrain speed. However, these methods primarily focus on updating shallow and static models, and their applicability to addressing the challenges of modeling temporal graphs remains limited. Studies in the continual graph learning field explore a seemingly similar context, which aims to learn new patterns incrementally on evolving graphs [17, 24, 33, 45, 54]. However, their main objective is to sustain the model performance across old and emerging tasks, such as predicting a new class in node classification [22, 33] or reasoning on a new relation on knowledge graphs [17]. It falls outside the scope of our focus which is centered on enhancing the same temporal task through the incorporation of newly collected data.

Meta-learning on Graphs. Given a set of tasks, meta-learning aims to learn general knowledge that is shared across all tasks and can be efficiently adapted to new tasks [52]. In this paper, we formulate the online learning task as a meta-learning problem and utilize model-agnostic meta-learning (MAML) [4] to address the challenges. Recently, meta-learning was integrated with graph neural network models for few-shot predictions on graphs, *e.g.*, Meta-Graph [2], G-Meta [12], MetaDyGNN [51], and MetaTKGR [40]. However, most works are designed for few-shot learning task on static/temproal graphs. The adaptation of the meta-learning approach for temporal graph learning in the online setting remains underexplored.

6 CONCLUSION

We studied a realistic online learning problem on temporal graphs, which aims to effectively and efficiently update the deployed model on the newly collected graph data. To this end, we proposed a novel temporal meta-training framework TGOnline. It meta-learns the global parameters of sampling and aggregating temporal neighbors, which can be adapted quickly to new data for future prediction via distillation-guided fine-tuning steps. Such bi-level optimization is nested and alternated during the offline training to mimic the online scenario. During the online stage, we further theoretically analyzed and utilized a PAC-Bayes bound to enable distillation from global parameters, which are further accelerated by the proposed edge reduction and node deduplication techniques. We empirically validated the effectiveness of TGOnline on four real-world temporal knowledge graphs, on which the proposed framework significantly outperforms an extensive set of SOTA baselines.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. Research reported in this paper was sponsored in part by DARPA award HR001121C0165, DARPA award HR00112290105, DoD Basic Research Office award HQ00342110002, the Army Research Laboratory under Cooperative Agreement W911NF-17-20196, and NSF award CCF-2316233. It was also supported in part by ACE, one of the seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA. Shengzhong Liu is supported by the National Natural Science Foundation of China (Grant No. 62332014, 62332013).

A APPENDIX

A.1 Proof of Theorem 3.1

Proof. First, for convenience of discussion, we define the difference between the real online updating loss and the predictive loss on the new data $\mathcal{D}^{T_i}$ as follows:

$$\Delta\mathcal{L} = \mathcal{L}^{\text{Online}}(\Theta^{T_i}, \mathcal{D}^{T_i}) - \mathcal{L}(\Theta^{T_i}, \mathcal{D}^{T_i}). \tag{8}$$

We are interested in the relation of $\Delta\mathcal{L}$ and the distribution discrenpency between the global parameter $p(\Theta)$ and the desired updated $p(\Theta^{T_i})$. Towards this goal, following [25, 31, 40], we construct the following function:

$$f(\mathcal{D}^{T_i}) = 2(|\mathcal{D}^{T_i}| - 1)\mathbb{E}_{\Theta^{T_i}\sim p(\Theta^{T_i})}\left[(\Delta\mathcal{L})^2\right] - \mathbb{KL}(p(\Theta^{T_i})\|p(\Theta)). \tag{9}$$

Next, using Markov's inequality, we have:

$$p(f(\mathcal{D}^{T_i}) > \epsilon) = p(e^{f(\mathcal{D}^{T_i})} > e^{\epsilon}) \leq \frac{\mathbb{E}_t\left[e^{f(\mathcal{D}^{T_i})}\right]}{e^{\epsilon}}, \tag{10}$$

where $\mathbb{E}_t\left[e^{f(\mathcal{D}^{T_i})}\right]$ denotes the expectation of $e^{f(\mathcal{D}^{T_i})}$ w.r.t. new data collection period. To upper bound the expectation, we have the following inequality:

$$\begin{aligned} f(\mathcal{D}^{T_i}) &= 2(|\mathcal{D}^{T_i}| - 1)\mathbb{E}_{\Theta^{T_i}\sim p(\Theta^{T_i})}\left[(\Delta\mathcal{L})^2\right] - \mathbb{KL}(p(\Theta^{T_i})\|p(\Theta)) \\ &= \mathbb{E}_{\Theta^{T_i}\sim p(\Theta^{T_i})}\left[\log\left(e^{2(|\mathcal{D}^{T_i}|-1)(\Delta\mathcal{L})^2}\frac{p(\Theta)}{p(\Theta^{T_i})}\right)\right] \\ &\leq \log\left(\mathbb{E}_{\Theta^{T_i}\sim p(\Theta^{T_i})}\left[e^{2(|\mathcal{D}^{T_i}|-1)(\Delta\mathcal{L})^2}\frac{p(\Theta)}{p(\Theta^{T_i})}\right]\right) \\ &= \log\left(\mathbb{E}_{\Theta^{T_I}\sim p(\Theta)}\left[e^{2(|\mathcal{D}^{T_i}|-1)(\Delta\mathcal{L})^2}\right]\right), \end{aligned} \tag{11}$$

where Jensen's inequality is utilized to derive the inequality. Therefore, we have

$$\mathbb{E}_t\left[e^{f(\mathcal{D}^{T_i})}\right] \leq \mathbb{E}_{\Theta^{T_i}\sim p(\Theta)}\mathbb{E}_t\left[e^{2(|\mathcal{D}^{T_i}|-1)(\Delta\mathcal{L})^2}\right], \tag{12}$$

the order of expectations is swapped as $p(\Theta)$ is independent to $\mathcal{D}^{T_i}$. Next, based on Hoeffding's inequality, we have:

$$p(\Delta\mathcal{L} > \epsilon) \leq e^{-2|\mathcal{D}^{T_i}|\epsilon^2}, \tag{13}$$

and we can further derive the following inequality:

$$\mathbb{E}_t\left[e^{f(\mathcal{D}^{T_i})}\right] \leq \mathbb{E}_{\Theta^{T_i}\sim p(\Theta)}\mathbb{E}_t\left[e^{2(|\mathcal{D}^{T_i}|-1)(\Delta\mathcal{L})^2}\right] \leq |\mathcal{D}^{T_i}|. \tag{14}$$

Combining Eq. 14 and Eq. 10, we get:

$$p(f(\mathcal{D}^{T_i}) > \epsilon) \leq \frac{|\mathcal{D}^{T_i}|}{e^{\epsilon}} = \delta, \tag{15}$$

where $\delta = |\mathcal{D}^{T_i}|/e^{\epsilon}$. Therefore, with probability of at least $1 - \delta$, we have that for all Θ^t:

$$\begin{aligned} f(\mathcal{D}^{T_i}) &= 2(|\mathcal{D}^{T_i}| - 1)\mathbb{E}_{\Theta^{T_i}\sim p(\Theta^{T_i})}\left[(\Delta\mathcal{L})^2\right] - \mathbb{KL}(p(\Theta^{T_i})\|p(\Theta)) \\ &\leq \frac{\log|\mathcal{D}^{T_i}|}{\delta}. \end{aligned} \tag{16}$$

Further, by utilizing Jensen's inequality again, we have:

$$\begin{aligned} \left(\mathbb{E}_{\Theta^{T_i}\sim p(\Theta^{T_i})}\left[(\Delta\mathcal{L})\right]\right)^2 &\leq \mathbb{E}_{\Theta^{T_i}\sim p(\Theta^{T_i})}\left[(\Delta\mathcal{L})^2\right] \\ &\leq \frac{\mathbb{KL}(p(\Theta^{T_i})\|p(\Theta)) + \frac{\log|\mathcal{D}^{T_i}|}{\delta}}{2(|\mathcal{D}^{T_i}| - 1}. \end{aligned} \tag{17}$$

Substituting the definition of $\Delta\mathcal{L}$ in Eq. 17, we prove the PAC-Bayes Bound on the newly collected data with unknown distribution. □

A.2 Baseline Setup

We compare retraining all the previous data and fine-tuning the new snapshot for MF method when testing online in the last ten periods, and preserve the strategy with better performance. As for SPMF, SML, and IGC, we perform online update on the last 10 time steps. For GRU4Rec and JODIE, we follow the similar procedure. To implement full retraining strategy, we concatenate all action sequences from all time periods together for model training. And we only consider the sequence happened within the last time period to train model, as the fine-tuning implementation. We tuned the learning rate in $\{1, 0.5, 0.1, 0.05, 0.01, 5e-3, 1e-3, ..., 1e-8\}$. Particularly for SML, we tuned transfer learning rate in $\{1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6\}$.

For other baselines, we use the first 30 periods for offline training. When evaluating each online period, all baselines are trained using both fine-tuning strategy on the last snapshot and full-retrain strategy on all the previous ones. Specifically for static baselines, we merge all historical time steps as one graph via *or* operation to implement the full retraining strategy. Notably, several baselines easily encounter out-of-memory (OOM) issue on large graphs, *e.g.*, DySAT, EGCN, Euler, JODIE, etc. We use the following ways to try to deal with the OOM issue: 1. we adopt edge sampling technique instead of full graph training to reduce the training samples and graph size; 2. we devide the time steps into groups and only consider the temporal dependency within each group to save GPU memory. Each group contains $K = \{3, 10\}$ consecutive time steps.

REFERENCES

[1] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, 2013.

[2] Avishek Joey Bose, Ankit Jain, Piero Molino, and William L. Hamilton. Meta-graph: Few shot link prediction via meta learning. *CoRR*, 2019.

[3] Sihao Ding, Fuli Feng, Xiangnan He, Yong Liao, Jun Shi, and Yongdong Zhang. Causal incremental graph convolution for recommender system retraining. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[5] Derek Greene, Donal Doyle, and Padraig Cunningham. Tracking the evolution of communities in dynamic social networks. In *2010 international conference on advances in social networks analysis and mining*, pages 176–183. IEEE, 2010.

[6] Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. Variational graph recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 10700–10710, 2019.

[7] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035, 2017.

[8] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648, 2020.

[9] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[11] Petter Holme and Jari Saramäki. Temporal networks. *Physics Reports*, 519(3):97–125, 2012.

[12] Kexin Huang and Marinka Zitnik. Graph meta learning via local subgraphs. *NeurIPS*, 2020.

[13] Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. In *EMNLP*, 2020.

[14] Isaiah J King and H Howie Huang. Euler: Detecting network lateral movement via scalable temporal link prediction. *ACM Transactions on Privacy and Security*, 2022.

[15] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[16] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR '17*, 2017.

[17] Xiaoyu Kou, Yankai Lin, Shaobo Liu, Peng Li, Jie Zhou, and Yan Zhang. Disentangle-based continual graph representation learning. *arXiv preprint arXiv:2010.02565*, 2020.

[18] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *KDD '19*, 2019.

[19] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2019.

[20] Jinning Li, Huajie Shao, Dachun Sun, Ruijie Wang, Yuchen Yan, Jinyang Li, Shengzhong Liu, Hanghang Tong, and Tarek F. Abdelzaher. Unsupervised belief representation learning in polarized networks with information-theoretic variational graph auto-encoders. *CoRR*, 2021.

[21] Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. Temporal knowledge graph reasoning based on evolutional representation learning. In *SIGIR*, 2021.

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

[23] Lihui Liu, Boxin Du, Jiejun Xu, Yinglong Xia, and Hanghang Tong. Joint knowledge graph completion and question answering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 1098–1108, New York, NY, USA, 2022. Association for Computing Machinery.

[24] H. Lium, Y. Yang, and X Wang. Overcoming catastrophic forgetting in graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8653–8661, 2021.

[25] David A. McAllester. Pac-bayesian model averaging. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, COLT '99, pages 164–170. ACM, 1999.

[26] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5363–5370, 2020.

[27] Steffen Rendle and Lars Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 251–258, 2008.

[28] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. In *ICML 2020 Workshop on Graph Representation Learning*, 2020.

[29] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 519–527, 2020.

[30] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *ESWC*, pages 593–607, 2017.

[31] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014.

[32] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[33] Chen Wang, Yuheng Qiu, Dasong Gao, and Sebastian Scherer. Lifelong graph learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13719–13728, 2022.

[34] Haiwen Wang, Ruijie Wang, Chuan Wen, Shuhao Li, Yuting Jia, Weinan Zhang, and Xinbing Wang. Author name disambiguation on heterogeneous information network with adversarial representation learning. In *AAAI '20*, 2020.

[35] Ruijie Wang, Zijie Huang, Shengzhong Liu, Huajie Shao, Dongxin Liu, Jinyang Li, Tianshi Wang, Dachun Sun, Shuochao Yao, and Tarek Abdelzaher. Dydiff-vae: A dynamic variational framework for information diffusion prediction. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, 2021.

[36] Ruijie Wang, Baoyu Li, Yichen Lu, Dachun Sun, Jinning Li, Yuchen Yan, Shengzhong Liu, Hanghang Tong, and Tarek Abdelzaher. Noisy positive-unlabeled learning with self-training for speculative knowledge graph reasoning. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2440–2457, Toronto, Canada, July 2023. Association for Computational Linguistics.

[37] Ruijie Wang, Zheng Li, Jingfeng Yang, Tianyu Cao, Chao Zhang, Bing Yin, and Tarek Abdelzaher. Mutually-paced knowledge distillation for cross-lingual temporal knowledge graph reasoning. In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 2621–2632. Association for Computing Machinery, 2023.

[38] Ruijie Wang, Zheng Li, Danqing Zhang, Qingyu Yin, Tong Zhao, Bing Yin, and Tarek Abdelzaher. Rete: Retrieval-enhanced temporal event forecasting on unified query product evolutionary graph. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 462–472, 2022.

[39] Ruijie Wang, Yuchen Yan, Jialu Wang, Yuting Jia, Ye Zhang, Weinan Zhang, and Xinbing Wang. Acekg: A large-scale knowledge graph for academic data mining. In *CIKM '18*, 2018.

[40] Ruijie Wang, zheng li, Dachun Sun, Shengzhong Liu, Jinning Li, Bing Yin, and Tarek Abdelzaher. Learning to sample and aggregate: Few-shot reasoning over temporal knowledge graphs. In *Advances in Neural Information Processing Systems*, 2022.

[41] Weiqing Wang, Hongzhi Yin, Zi Huang, Qinyong Wang, Xingzhong Du, and Quoc Viet Hung Nguyen. Streaming ranking based recommender systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 525–534, 2018.

[42] Yufeng Wang and Charith Mendis. Tgopt: Redundancy-aware optimizations for temporal graph attention networks. In *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, PPoPP '23, page 354–368, 2023.

[43] Da Xu, chuanwei ruan, evren korpeoglu, sushant kumar, and kannan achan. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations (ICLR)*, 2020.

[44] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[45] Yishi Xu, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, and Mark Coates. Graphsail: Graph structure aware incremental learning for recommender systems. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, CIKM '20, page 2861–2868, 2020.

[46] Yuchen Yan, Baoyu Jing, Lihui Liu, Ruijie Wang, Jinning Li, Tarek Abdelzaher, and Hanghang Tong. Reconciling competing sampling strategies of network embedding. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[47] Yuchen Yan, Qinghai Zhou, Jinning Li, Tarek Abdelzaher, and Hanghang Tong. Dissecting cross-layer dependency inference on multi-layered inter-dependent networks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 2341–2351, New York, NY, USA, 2022. Association for Computing Machinery.

[48] Chaoqi Yang, Jinyang Li, Ruijie Wang, Shuochao Yao, Huajie Shao, Dongxin Liu, Shengzhong Liu, Tianshi Wang, and Tarek F. Abdelzaher. Hierarchical overlapping belief estimation by structured matrix factorization. In *ASONAM'20*,

2020.
[49] Chaoqi Yang, Ruijie Wang, Shuochao Yao, and Tarek F. Abdelzaher. Hypergraph learning with line expansion. *CoRR*, abs/2005.04843, 2020.
[50] Chaoqi Yang, Ruijie Wang, Shuochao Yao, Shengzhong Liu, and Tarek F. Abdelzaher. Revisiting "over-smoothing" in deep gcns. *CoRR*, abs/2003.13663, 2020.
[51] Cheng Yang, Chunchen Wang, Yuanfu Lu, Xumeng Gong, Chuan Shi, Wei Wang, and Xu Zhang. Few-shot link prediction in dynamic networks. In *WSDM '22*, 2022.
[52] Huaxiu Yao, Ying Wei, Junzhou Huang, and Zhenhui Li. Hierarchically structured meta-learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 7045–7054, 2019.
[53] Jiaxuan You, Tianyu Du, and Jure Leskovec. Roland: Graph learning framework for dynamic graphs, 2022.
[54] Qiao Yuan, Sheng-Uei Guan, Pin Ni, Tianlun Luo, Ka Lok Man, Prudence Wong, and Victor Chang. Continual graph learning: A survey. *arXiv preprint arXiv:2301.12230*, 2023.
[55] Yang Zhang, Fuli Feng, Chenxu Wang, Xiangnan He, Meng Wang, Yan Li, and Yongdong Zhang. How to retrain recommender system? a sequential meta-learning method. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1479–1488, 2020.
[56] Zeyang Zhang, Xin Wang, Ziwei Zhang, Haoyang Li, Zhou Qin, and Wenwu Zhu. Dynamic graph neural networks under spatio-temporal distribution shift. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.