# Temporal-Aware Evaluation and Learning for Temporal Graph Neural Networks

**Junwei Su**[2,*], **Shan Wu**[1,*]

[1]School of Resources and Environmental Engineering, Hefei University of Technology
[2]School of Computing and Data Science, University of Hong Kong
jwsu@cs.hku.hk, wus@hfut.edu.cn

## Abstract

Temporal Graph Neural Networks (TGNNs) are a family of graph neural networks designed to model and learn dynamic information from temporal graphs. Given their substantial empirical success, there is an escalating interest in TGNNs within the research community. However, the majority of these efforts have been channelled towards algorithm and system design, with the evaluation metrics receiving comparatively less attention. Effective evaluation metrics are crucial for providing detailed performance insights, particularly in the temporal domain. This paper investigates the commonly used evaluation metrics for TGNNs and illustrates the failure mechanisms of these metrics in capturing essential temporal structures in the predictive behaviour of TGNNs. We provide a mathematical formulation of existing performance metrics and utilize an instance-based study to underscore their inadequacies in identifying volatility clustering (the occurrence of emerging errors within a brief interval). This phenomenon has profound implications for both algorithm and system design in the temporal domain. To address this deficiency, we introduce a new volatility-aware evaluation metric (termed volatility cluster statistics), designed for a more refined analysis of model temporal performance. Additionally, we demonstrate how this metric can serve as a temporal-volatility-aware training objective to alleviate the clustering of temporal errors. Through comprehensive experiments on various TGNN models, we validate our analysis and the proposed approach. The empirical results offer revealing insights: 1) existing TGNNs are prone to making errors with volatility clustering, and 2) TGNNs with different mechanisms to capture temporal information exhibit distinct volatility clustering patterns. Moreover, our empirical findings demonstrate that our proposed training objective effectively reduces volatility clusters in error.

## 1 Introduction

Many real-world problems and systems are naturally modeled as *temporal graphs* (also referred to as *dynamic graphs*), characterized by continuously changing relationships, nodes, and attributes. To address this temporal dynamic nature, Temporal Graph Neural Networks (TGNNs), the temporal counterparts to GNNs, have emerged as promising deep learning models capable of modelling time-varying graph structures (Kazemi et al. 2020; Skarding, Gabrys, and Musial 2021; Zhang et al. 2023; Xu et al. 2020a). Unlike their static counterparts, TGNNs excel at capturing temporal dependencies and learning temporal representations within the context of temporal graphs. Consequently, they are widely employed in applications such as traffic prediction (Zhao et al. 2019; Guo et al. 2019; Zhang et al. 2020), financial analysis (Wang et al. 2021a; Su, Wu, and Li 2024), social network (Zhang et al. 2021b), recommender systems (Kumar, Zhang, and Leskovec 2019), and climate modeling (Khodayar and Wang 2018).

Given their substantial empirical success, there is growing interest in TGNNs within the research community. However, most efforts have been concentrated on algorithm and system design, with various classes of TGNNs emerging based on their mechanisms for capturing temporal information (e.g., RNN-based, memory-based, and attention-based; see related work for more details). Conversely, the evaluation of TGNNs has received comparatively less attention. There are only a few benchmark studies on TGNNs that predominantly investigate how various combinations of learning settings and datasets impact the performance of TGNN models. *Notably, these benchmarks typically utilize common instance-based evaluation metrics like Average Precision (AP) and Area Under the ROC Curve (AU-ROC), where each test sample is considered identically and independently*. An intriguing finding from these benchmark studies is that almost all existing TGNNs demonstrate remarkable (and similar) performance when evaluated against these instance-based metrics. *This uniformity in performance poses a significant challenge in model selection for practical applications, as distinguishing between models based on these metrics alone becomes difficult.* Therefore, there is an urgent need to develop more nuanced evaluation metrics that can better capture the unique capabilities and efficiencies of different TGNN architectures.

In addition to model selection, this paper argues that instance-based evaluation metrics are insufficient and ineffective at capturing the temporal structure of the predictive behavior of TGNNs. Data samples in temporal graphs could exhibit *temporal correlation*, impacting the predictions made by TGNNs and introducing patterns such as *volatility clusters*—periods where large fluctuations are


* corresponding author

grouped together. This aspect is crucial for the functionality of temporal algorithms and systems in TGNNs. For example, in financial trading algorithms or risk management systems, accurately measuring and predicting volatility clusters can be crucial for effective strategy deployment and risk assessment. Similarly, in fault-tolerant systems, understanding volatility clusters can aid in preemptively identifying periods of potential system stress or failure, thereby enabling proactive maintenance or system adjustments to prevent downtime. Adequate performance evaluation ensures that these systems are not only accurate but also robust and responsive under varying temporal dynamics. This, in turn, aids in optimizing operational efficiency, improving decision-making processes, and ensuring reliability in critical applications where timing and the evolution of data play a vital role. Therefore, developing and refining evaluation metrics that can effectively measure the performance of TGNNs is essential for advancing these technologies and their applications.

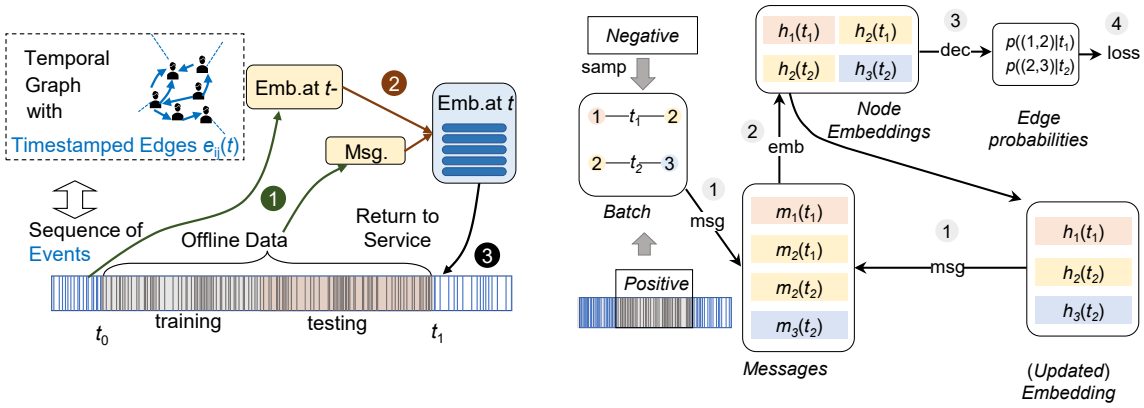
**Contribution.** This paper aims to spotlight an underexplored aspect of TGNNs—the evaluation metrics. We examine and highlight the inadequacies of current evaluation metrics in capturing the temporal structures of TGNNs and propose a novel performance metric tailored to detect nuanced temporal information such as volatility clusters. The key contributions and findings of this paper are summarized and highlighted as follows

1. We present a mathematical formulation of existing evaluation metrics alongside a formal definition aimed at measuring the expressiveness of these metrics. This foundational framework is crucial for analyzing evaluation metrics comprehensively and formalizing the limitations inherent in current TGNN evaluation approaches. Utilizing this framework, we formally prove that instance-based evaluation metrics such as AP and AU-ROC resemble a simple counting process and fail to capture temporal structures (e.g., volatility clusters) in the predictions of TGNNs (Theorem 3.1).
2. Building on the insights from our analysis, we propose a novel evaluation metric, named **v**olatility-**c**luster **s**tatistics (VCS). Inspired by Hopkins statistics (Hopkins and Skellam 1954), VCS serves as a complementary evaluation metric designed to detect and evaluate volatility clusters in the prediction errors of TGNNs. VCS offers crucial insights into the temporal structure of the prediction errors (error pattern) of TGNNs and helps differentiate the performance of various TGNN models.
3. Beyond its use in evaluation, we demonstrate that the concept of VCS can also function effectively as a regularization technique to mitigate volatility clusters in errors with appropriate modifications. We introduce a method termed **v**olatility-**c**luster-**a**ware (VCA) learning, which is a smooth and differentiable extension of VCS. VCA helps mitigate volatility clusters in the prediction errors of TGNNs. This capability is particularly valuable in the design of systems and algorithms for critical areas such as fault-tolerant systems.
4. We validate our findings and the effectiveness of our metrics through extensive empirical studies consisting of five datasets and six SOTA methods. Our empirical results reveal several key insights: 1) existing TGNNs tend to produce volatility cluster in errors, particularly in RNN-based and memory-based models; 2) different types of TGNNs manifest varying error patterns—for instance, memory-based TGNNs generally exhibit clustered errors towards the end of the testing period, whereas RNN-based TGNNs tend to show them at the beginning. These observations indicate fundamental differences in how these models process temporal information and provide directions for model-specific improvements; 3) our proposed VCA learning objective serves as an effective regularization tool, making existing TGNNs less susceptible to volatility clustering in errors.

## 2 Related Works

**Temporal Graph Neural Network.** Temporal graph representation learning has garnered substantial attention in recent years, driven by the imperative to model and analyze evolving relationships and temporal dependencies within temporal graphs (we refer the reader to (Skarding, Gabrys, and Musial 2021; Kazemi et al. 2020) for more comprehensive surveys). TGNNs, as temporal counterparts to GNNs, have emerged as promising neural models for temporal graph representation learning(Sankar et al. 2020; Poursafaei et al. 2022; Xu et al. 2020a; Su, Zou, and Wu 2024b; Wang et al. 2021c; Kumar, Zhang, and Leskovec 2019; Trivedi et al. 2019; Zhang et al. 2023; Pareja et al. 2020; Trivedi et al. 2017; Xu et al. 2020b; Luo and Li 2022) and have shown SOTA performance in many temporal-related tasks. Roughly speaking, existing TGNNs can be categorized into three types based on the mechanism used for capturing temporal information: RNN-based (Trivedi et al. 2019), attention-based (Wang et al. 2021b), and memory-based TGNNs (Rossi et al. 2021). Due to its potential and practical significance, there has been a recent surge in both theoretical exploration (Souza et al. 2022) and architectural innovation (Rossi et al. 2021; Wang et al. 2021c; Kumar, Zhang, and Leskovec 2019; Trivedi et al. 2019; Zhang et al. 2023) related to TGNNs. In addition, there are works dedicated to optimizing both the inference and training efficiency of TGNNs, employing techniques such as incremental learning (Su et al. 2023; Su, Zou, and Wu 2024a), computation duplication (Wang and Mendis 2023), CPU-GPU communication optimization (Zhou et al. 2022), staleness (Sheng et al. 2024), and caching (Wang et al. 2021c). Despite all these efforts, the evaluation metrics of TGNNs remain underexplored. In this paper, we address this gap and focus on studying the evaluation metrics of TGNNs.

**Evaluation of TGNNs.** Evaluation is core to machine learning research (Zhang et al. 2021a). Because of this, evaluation and benchmarking have been extensively studied in static graph representation learning (Dwivedi et al. 2023; Errica et al. 2019; Hu et al. 2020; Lv et al. 2021). Due to the dynamic nature of temporal graphs, properly evaluating temporal link prediction problems has been challenging and complicated with different issues as documented in (Junuthula, Xu, and Devabhaktuni 2018; Haghani and




(a) TGNN Learning Pipeline

(b) TGNN Training and Computation

Figure 1: The Learning Procedure of TGNNs. Fig. 1(a) depicts the learning procedure of TGNN. Data/events are split based on chronological order into training and testing/validation. During the training, data/events are further divided into temporal batches. The incoming batch serves as training samples for updating the model and embedding for the subsequent batch. Fig. 1(b) visualizes the training procedure and computation of TGNNs. Incoming events are served as positive samples and negative events are sampled from the rest of the graphs.

Keyvanpour 2019; Junuthula, Xu, and Devabhaktuni 2016; Poursafaei et al. 2022; Huang et al. 2024; Yu et al. 2023). In particular, (Poursafaei et al. 2022; Huang et al. 2024; Yu et al. 2023) are recent benchmark studies focusing on TGNN evaluation on temporal link prediction. Their studies have revealed that learning settings, such as transductive vs. inductive and negative sampling strategies, play a critical role in properly evaluating TGNNs. In addition, these benchmarks reveal that almost all existing TGNN exhibit remarkable (and similar) performance with respect to the commonly used instance-based evaluation metric, rendering model selection challenging in practice. This has inspired and motivated the central research of this paper.

## 3 Preliminary and Background

In this section, we provide a concise introduction to TGNNs. Due to space limitations, a more detailed description is available in the supplementary material for completeness. We use lowercase letters to denote scalars and graph-related objects, and lower and uppercase boldface letters to denote vectors and matrices, respectively.

**Event-based Representation of Temporal Graphs.** In this paper, we adopt the event-based representation of temporal graphs, as described in previous works (Skarding, Gabrys, and Musial 2021; Zhang et al. 2023). A temporal graph $\mathcal{G}$ in this representation consists of a node set $\mathcal{V} = \{1, ..., N\}$ and an event set $\mathcal{E} = \{e_{ij}(t)\}$, where $i, j \in \mathcal{V}$. The node set $\mathcal{V}$ represents the entities in the graphs. The event set $\mathcal{E}$ represents a stream of events, with each edge $e_{ij}(t)$ corresponding to an interaction event between node $i$ and node $j$ at timestamp $t \geq 0$. Node features and edge features for $v_i$ and $e_{ij}$ are denoted by $\mathbf{f}_i(t)$ and $\mathbf{f}_{ij}(t)$, respectively. In the case of non-attributed graphs, we assume $\mathbf{f}_i(t) = \mathbf{0}$ and $\mathbf{f}_{ij}(t) = \mathbf{0}$, representing zero vectors.

**Temporal Graph Neural Networks (TGNNs).** TGNNs, extended from the standard GNN to the temporal graph, can be viewed as an embedding function (encoder) for finding the temporal representation of vertices in temporal graphs (Su, Zou, and Wu 2024b; Rossi et al. 2021). The learned embedding can then be used as input for different downstream tasks. A canonical formulation of the TGNN encoder is to extend the message-passing scheme from GNNs to include time information. The formulation of TGNNs for learning the representation of vertex $i$ is given by:

$$\mathbf{h}_i(t) = \text{emb}(\{\mathbf{m}_{ij}, j \in \mathcal{N}_i(t)\}),$$
$$\mathbf{m}_{ij}(t) = \text{msg}(\mathbf{h}_i(t^-), \mathbf{h}_j(t^-), \mathbf{f}_{ij}(t), \mathbf{f}_i(t), \mathbf{f}_j(t), \Delta t),$$

where $\mathbf{h}_i(t^-)$ and $\mathbf{h}_j(t^-)$ are the embedding of nodes $i$ and $j$ before time $t$ (i.e., at the time of the previous interaction involving node $i$ or $j$), $\mathbf{m}_{ij}(t)$ is the message from vertex $j$ to $i$ at time $t$ generated from the event $e_{ij}(t)$, $\mathcal{N}_i(t)$ is the temporal neighbours of nodes $i$ up to time $t$, $h_i(t)$ is temporal embedding/representation of nodes $i$ at time $t$, and $\text{msg}(.)$ (e.g., MLP), and $\text{emb}(.)$ (e.g., GCN) are learnable functions. After obtaining the embeddings $h_i(t)$ and $h_j(t)$ in the prescribed manner, an extra simple MLP layer (or decoder in other forms) can be used for the down-stream tasks.

**TGNNs Training and Evaluation** TGNNs are frequently trained in a self-supervised manner using link prediction tasks (Poursafaei et al. 2022; Huang et al. 2024), which are commonly conceptualized as a binary classification problem aimed at predicting whether a link will form between two nodes. Consequently, the performance of TGNNs is often evaluated with respect to their success in link prediction tasks. Therefore, in this paper, we concentrate our discussion on link prediction, though the analysis and arguments can be naturally extended to other downstream tasks such as

node classification. More formally, we can assign labels for events $e_{ij}(t)$, such that:

$$y_{ij}(t) = \begin{cases} 1 & \text{if } e_{ij}(t) \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases}$$

For simplicity, we omit the specific node pair $i, j$ when referring to the event $e_{ij}(t)$ and index the event by its order of appearance in the corresponding set. Let $\mathcal{E}_{\text{test}} = \{e_k(t_k)\}_{k=1,...,M}$, be a chronologically ordered sequence of $M$ test samples from the test period, $T_{\text{test}} = [t_1, t_2], i.e., t_1 \leq t_k \leq t_{k+1} \leq t_2$. Let $\mathbf{Y} = \{y_1, ..., y_m\}$, be the ground-truth labels of the given samples, and let $\widehat{\mathbf{Y}} = \{\widehat{y}_1, ..., \widehat{y}_m\}$, be the predicted labels of the given samples by the TGNN. Then, we can define the performance evaluation metric as a function $\mu(.)$ of the form:

$$\mu : \mathbf{Y} \times \widehat{\mathbf{Y}} \times \mathcal{E} \mapsto \mathbb{R}^+.$$

In other words, $\mu$ takes in the prediction and the ground truth and maps them to a positive real value.

### Limitation of Current Evaluation Metrics

To explore the limitation of the evaluation metric, we first define a measure of its capability. In this paper, we propose extending the idea of the expressive power of GNNs to characterize the ability of an evaluation metric by its expressiveness—the capacity to differentiate between different predictions. More formally, we introduce the following definition.

**Definition 1** (Expressiveness of Evaluation Metric)**.** *For two distinct predictions $\widehat{\mathbf{Y}}_1$ and $\widehat{\mathbf{Y}}_2$, we say an evaluation metric $\mu$ can differentiate $\widehat{\mathbf{Y}}_1$ and $\widehat{\mathbf{Y}}_2$ if $\mu(\mathbf{Y}, \widehat{\mathbf{Y}}_1, \mathcal{E}) \neq \mu(\mathbf{Y}, \widehat{\mathbf{Y}}_2, \mathcal{E})$.*

As noted, the most commonly used evaluation metrics for TGNNs are instance-based, such as AP and AU-ROC, where each test sample is considered identically and independently. More formally, this family of evaluation metrics is defined as follows:

**Definition 2** (Instance-based Evaluation)**.** *For a given evaluation $\mu(\mathbf{Y}, \widehat{\mathbf{Y}}, \mathcal{E})$, we say $\mu(.)$ is an instance-based evaluation metric if it can be expressed as,*

$$\mu(\mathbf{Y}, \widehat{\mathbf{Y}}, \mathcal{E}) = g\left(\left\{f(y_i, \widehat{y}_i) | y_i, \widehat{y}_i \in \mathbf{Y}, \widehat{\mathbf{Y}}\right\}\right),$$

*where $g$ is some set function and $f : \mathbf{Y} \times \widehat{\mathbf{Y}} \mapsto \mathbb{R}^+$.*

The following result shows the limitation of instance-based evaluation metrics:

**Theorem 3.1** (Failure of Instance-Based Evaluation)**.** *Let $\widehat{\mathbf{Y}}_1$ and $\widehat{\mathbf{Y}}_2$ be two distinct predictions for the set $\mathcal{E}$ with ground-truth $\mathbf{Y}$, and $\mu(.)$ is an instance-based evaluation metric. Then, we have that,*

$$\mu(\widehat{\mathbf{Y}}_1, \mathbf{Y}, \mathcal{E}) = \mu(\widehat{\mathbf{Y}}_2, \mathbf{Y}, \mathcal{E}),$$

*so long as,*

$$\mathrm{H}(\mathbf{Y}, \widehat{\mathbf{Y}}_1) = \mathrm{H}(\mathbf{Y}, \widehat{\mathbf{Y}}_2),$$

*where* $\mathrm{H}(\mathbf{Y}, \widehat{\mathbf{Y}}) = \sum_{k=1}^{|\mathcal{E}|} \mathbb{1}[y_k \neq \widehat{y}_k]$.

Theorem 3.1 demonstrates that instance-based evaluation metrics cannot differentiate predictions if the number of disagreements with the ground truth is the same. Essentially, such metrics reduce all diverse information (e.g., temporal information) of predictions to a mere disagreement count. This severely limits the expressiveness of these metrics, making them inadequate for capturing insightful information about predictions within the temporal process.

**Visualization Example.** To further illustrate this, consider the examples in Fig. 2, which have identical numbers of errors and correct predictions. It is evident that the instance-based evaluation metric fails to differentiate these examples, as they exhibit the same predictive performance (i.e., the same amount of disagreement/errors). However, the patterns of errors in these examples are markedly different. Such variances in error distribution provide crucial insights into both the TGNN models and the systems they represent. For example, as previously discussed, the presence of a volatility cluster in errors is critical information for model selection in real-time fault-tolerant systems, where functionality is ensured if errors are evenly distributed. Thus, the inability to detect such error patterns can lead to catastrophic failures in many real-world algorithm and system designs. To address this issue, in the subsequent section, we introduce a novel evaluation metric and learning objective designed to detect and mitigate this type of volatility cluster in errors.

## 4 Methodology

Building on the previous discussion regarding the limitations of existing evaluation metrics, this section introduces a novel temporal-aware evaluation metric derived from the concept of Hopkins statistics (Banerjee and Dave 2004). Specifically, we focus on detecting volatility clusters within predictions, which have significant implications for algorithms and systems, as discussed earlier. Additionally, based on this proposed evaluation metric, we introduce a novel temporal-aware learning objective for TGNNs.

### Volatility-Cluster Statistics (VCS)

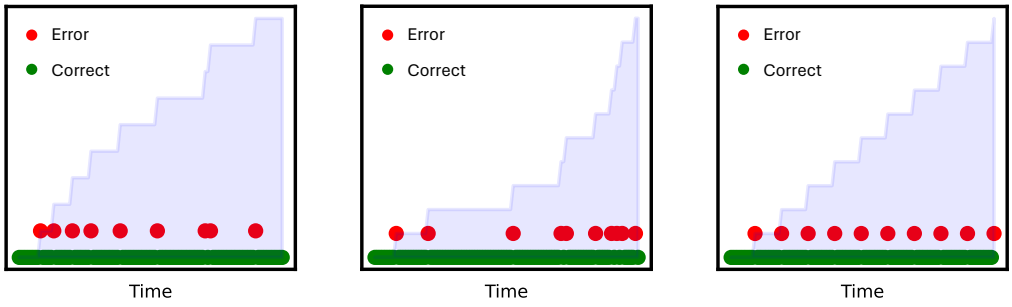
Given a test period $T_{\text{test}}$, let $\mathbf{Y}$ and $\widehat{\mathbf{Y}}$ represent the ground truth and the predictions of the model on the test set, respectively. Let $\mathcal{E}_{\text{disg}}$ denote the set of disagreement events with cardinality $K$ and let $\widehat{\mathcal{E}}_{\text{disg}}$ denote $k < K$ samples from $\mathcal{E}_{\text{disg}}$ . We first compute the sum of distances from the sampled disagreement set to the disagreement as:

$$\mathrm{D}_{\text{disg}} = \sum_{e \in \widehat{\mathcal{E}}_{\text{disg}}} \mathrm{d}(e, \mathcal{E}_{\text{disg}}), \tag{4.1}$$

$$\mathrm{d}(e, \mathcal{E}_{\text{disg}}) = \min\left\{ |t_e - t_{e'}| \middle| e' \in \mathcal{E}_{\text{disg}}, e' \neq e \right\}. \tag{4.2}$$

$\mathrm{d}(e, \mathcal{E}_{\text{disg}})$ calculates the time difference between event $e$ and the closest event in the given set. Then, $\mathrm{D}_{\text{disg}}$ is a sum of such distances for the disagreement set. Next, we generate a

Extra technical details such as proof, pseudo-code and experimental setting can be found in extended arxiv version (Su and Wu 2024).




(a) Random Error (b) Cluster Error (c) Regular Error

Figure 2: An illustration of different error patterns. Fig. 2(a) is the pattern for random error pattern where wrong predictions are randomly distributed across the time interval. Fig. 2(b) is the pattern for volatility cluster error where wrong predictions are clustered at a small time interval (the end of the temporal horizon in the example). Fig. 2(b) is the pattern for regular error where wrong predictions are evenly spaced. The shaded area in the plots indicates the accumulated count of errors.

set $\mathcal{E}_{\mathrm{r}}$ of $k$ events by uniformly randomly sampling from the test period $T_{\mathrm{test}}$. Similarly, we compute its distance to the disagreement as:

$$\mathrm{D_r} = \sum_{e \in \mathcal{E}_{\mathrm{r}}} \mathrm{d}(e, \mathcal{E}_{\mathrm{disg}}). \tag{4.3}$$

$\mathrm{D_r}$ serves as a reference for the distance to the disagreement if the samples are randomly drawn. Then, we can compute relative statistics between the set $\mathcal{E}_{\mathrm{disg}}$ and $\mathcal{E}_{\mathrm{r}}$ as:

$$\mathcal{T}(\mathcal{E}_{\mathrm{disg}}, \mathcal{E}_{\mathrm{r}}) = \frac{\mathrm{D_r}}{\mathrm{D_r} + \mathrm{D_{disg}}},$$

where $\mathrm{D_r}$ and $\mathrm{D_{disg}}$ are described above. The formulation shows that $\mathcal{T}(\mathcal{E}_{\mathrm{disg}}, \mathcal{E}_{\mathrm{r}})$ compares the temporal distance between predictions relative to random sampling. The ratio format confines the value within the range of 0 to 1. The $\mathcal{T}(\mathcal{E}_{\mathrm{disg}}, \mathcal{E}_{\mathrm{r}})$ statistic provides insights into the distribution of data points. If $\mathcal{T}(\mathcal{E}_{\mathrm{disg}}, \mathcal{E}_{\mathrm{test}})$ is close to 1, it indicates that the data points are clustered, with the sum of distances from randomly generated points to their nearest neighbors being significantly larger than that from the sampled data points. Conversely, if $\mathcal{T}(\mathcal{E}_{\mathrm{disg}}, \mathcal{E}_{\mathrm{test}})$ is close to 0, it could suggest that the data points are regularly-spaced, resulting in smaller distances for randomly generated points compared to those from sampled data points. When $\mathcal{T}(\mathcal{E}_{\mathrm{disg}}, \mathcal{E}_{\mathrm{test}})$ approximates 0.5, it indicates a random distribution with no significant clustering or regular pattern, as both randomly generated points and sampled data points exhibit similar nearest neighbour distances.

To enhance interoperability and robustness against variance from sampling, we repeat the sampling steps multiple times and adjust based on the random sampling. The final VCS is computed as follows:

$$\begin{aligned} \mathrm{VCS} &= |1/2 - \mathcal{T}(\mathcal{E}_{\mathrm{disg}}, \mathcal{E}_{\mathrm{r}}, \tau)|, \\ &= \left| \frac{1}{2} - \frac{1}{\tau} \sum_{i=1}^{\tau} \frac{\mathrm{D_r^{(i)}}}{\mathrm{D_r^{(i)}} + \mathrm{D_{disg}^{(i)}}} \right|. \end{aligned}$$

where $\tau$ is the number of repeated samples. Our empirical study suggests that $\tau = 5$ provides a stable estimate in most cases.

## Volatility-Cluster-Aware (VCA) Learning

In the previous section, we introduced a new statistical measure for detecting volatility clusters in the temporal dimension. We discussed how the error pattern of the system can have significant implications in real-time systems, especially concerning fault-tolerant aspects of development. Typically, real-time systems prefer more uniform error distributions. Thus, an important question arises: can we use the proposed measure to help TGNNs learn a model (weight) from the hypothesis space that exhibits a more uniform error pattern?

A straightforward idea is to incorporate $\mathcal{T}(\mathcal{E}_{\mathrm{disg}}, \mathcal{E}_{\mathrm{test}}, \tau)$ as a regularization term in the learning objective. However, a technical challenge arises due to the non-differentiability of the distance function $\mathrm{d}(e, \mathcal{E})$, which is due to the $\min$ operator. To address this, we propose the following modification with a smooth and differentiable version that mimics the min function:

$$\mathrm{d_{soft}}(e, \mathcal{E}) = -\log \left( \sum_{e' \in \mathcal{E}, e' \neq e} \exp(-\beta |t_e - t_{e'}|) \right) / \beta,$$

where $\beta$ is a positive parameter that controls the sharpness of the approximation. As $\beta$ increases, the approximation becomes closer to the minimum function. This approach turns the non-differentiable minimum function into a differentiable function by summing over exponentially scaled, inverted distances,

$$\mathcal{T}_{\mathrm{soft}}(\mathcal{E}_{\mathrm{disg}}, \mathcal{E}_{\mathrm{r}}) = \frac{\widehat{\mathrm{D}}_{\mathrm{r}}}{\widehat{\mathrm{D}}_{\mathrm{r}} + \widehat{\mathrm{D}}_{\mathrm{disg}}},$$

where $\widehat{\mathrm{D}}_{\mathrm{r}}$ and $\widehat{\mathrm{D}}_{\mathrm{disg}}$ are defined similarly as before with the distance function replaced with $\mathrm{d_{soft}}(.)$. We can then incorporate this into the learning process and term the modified objective VCA.

$$\widehat{\mathcal{L}}(\widehat{\mathbf{Y}}, \mathbf{Y}) = \mathcal{L}(\widehat{\mathbf{Y}}, \mathbf{Y}) + \gamma \left\| \frac{1}{2} - \mathcal{T}_{\mathrm{soft}}(\mathcal{E}_{\mathrm{disg}}, \mathcal{E}_{\mathrm{r}}) \right\|^2, \tag{4.4}$$

where $\mathcal{L}(\widehat{\mathbf{Y}}, \mathbf{Y})$ is the standard loss function for training TGNNs (e.g., cross-entropy), and $\gamma$ is a hyper-parameter

controlling the regularization effect. If the error pattern deviates from a uniform distribution, then VCA will incur a larger value, and consequently, the training objective will reflect a larger loss. Achieving a lower value with this new training objective is expected to improve the uniformity of the error distribution within the model.

# 5 Empirical Study

In this section, we present an empirical study to further illustrate the problem addressed in this paper. The study aims to answer the following key questions:

1. Do existing TGNNs exhibit volatility clusters in errors?
2. Do existing TGNNs exhibit different error distributions?
3. Is VCS effective in detecting volatility clusters in errors?
4. Can VCA mitigate volatility clusters in errors?

## Experimental Settings

**Datasets and Baselines.** We use five public dynamic graph benchmark datasets: Reddit, Wikipedia, MOOC, LastFM, and GDELT (Poursafaei et al. 2022). We evaluate six state-of-the-art TGNN models, with two models from each of the three categories of TGNNs mentioned: TGN (Rossi et al. 2021) & Tiger (Zhang et al. 2023) (memory-based TGNNs), TCL (Wang et al. 2021b) & TGAT (Xu et al. 2020a) (attention-based TGNNs), and JOIDE (Kumar, Zhang, and Leskovec 2019) & DyRep (Trivedi et al. 2019) (RNN-based TGNNs). We adopt the implementation of these baselines from (Zhou et al. 2022; Poursafaei et al. 2022; Huang et al. 2024).

**Evaluation Task and Metrics.** Following the approaches outlined in (Poursafaei et al. 2022; Huang et al. 2024; Yu et al. 2023), we evaluate models for temporal link prediction, which involves predicting the probability of a link forming between two nodes at a specific time. We use a multi-layer perceptron (MLP) that takes the concatenated representations of two nodes as input and outputs the probability of a link. For evaluation metrics, we focus on AP and the proposed VCS. We train each model with and without VCA to observe the effect of our proposed learning objective. For all experiments, we follow the standard procedure and split datasets chronologically with a ratio of 70%/15%/15% for training, validation, and testing, respectively. Each experiment is conducted with five independent trials, and the average results are reported

## Experimental Results

**Temporal Error Pattern.** Our first experiment aims to demonstrate the temporal error patterns of various models and how our proposed metrics can effectively differentiate and reveal insightful information regarding these patterns. Fig. 4 illustrates that different types of TGNNs exhibit distinct error pattern behaviours. Specifically, memory-based TGNNs tend to produce volatility clusters in errors toward the end of the test period, RNN-based TGNNs are more prone to errors at the beginning of the test period and attention-based TGNNs exhibit a more uniform distribution in errors. This temporal structure in the prediction errors of memory-based and RNN-based TGNNs is reflected by a larger VCS value in Table 1 . This confirms that existing TGNNs indeed generate volatility clusters in errors, and different TGNN mechanisms induce varying volatility patterns. Furthermore, this demonstrates that VCS is an effective measure for detecting volatility clusters in errors.

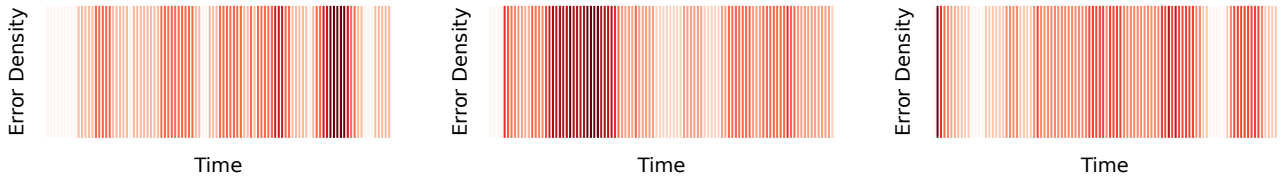
**Effectiveness of VCA.** Our next experiment aims to demonstrate the effectiveness of our proposed learning objective, VCA, as defined in Eq. 4.4, in regulating the behavior of TGNNs. As shown in Table 1, TGNN models trained with our proposed objective significantly reduce volatility clusters in errors, as evidenced by decreased VCS values. The improvement in attention-based TGNNs (e.g., TCL & TGAT) is relatively small because these models already exhibit a fairly uniform error distribution. This confirms that VCA is indeed effective in mitigating volatility clusters in errors. Such a property can be particularly beneficial for critical real-time systems where fault tolerance is important, and a more uniformly distributed error is preferred.

**Ablation Study.** The final part of the empirical study focuses on the hyper-parameters of VCS and VCA. The key hyper-parameter in VCS is $\tau$, which represents the number of independent trials conducted to compute the reference distance for random errors. As shown in Fig. 4(a), we found that increasing $\tau$ leads to a smaller variance in value but incurs a higher computational cost. However, we find that $\tau = 5$ already provides a sufficiently robust estimation. The main hyper-parameter in VCA is $\gamma$ in Eq.4.4, which controls the regularization effect of the proposed learning objective. Our experiment shows that increasing $\gamma$ results in a more uniform error pattern but worsens predictive performance (smaller AP). Thus, there is a trade-off between achieving this uniform error distribution and maintaining predictive performance. This trade-off does not undermine the effectiveness of our proposed learning objective, as the primary goal is to make the error distribution more uniform. Whether this trade-off is favourable depends on the application scenario. However, as indicated in Table 1, $\gamma = 0.1$ provides a significant improvement in VCS without significantly affecting the model's accuracy.

# 6 Discussion

**Conclusion.** We investigate the evaluation metrics for TGNNs. Specifically, we have identified the pitfalls and limitations of currently used instance-based measures, such as AP and AU-ROC, in capturing temporal structures in prediction errors, such as volatility clusters. To address this issue, we propose VCS, a metric that effectively captures volatility clusters in errors for TGNNs. Furthermore, we extend this proposed evaluation metric as a regularizer, introducing VCA to mitigate volatility clusters in errors.

**Limitation and Future Works.** In this paper, we focus on volatility clusters in errors. Other important temporal structures, such as the time arrival of errors, are not captured by the current metric. This presents an interesting avenue for future exploration. Additionally, our study primarily concentrates on the temporal aspect of error distribution. A natural

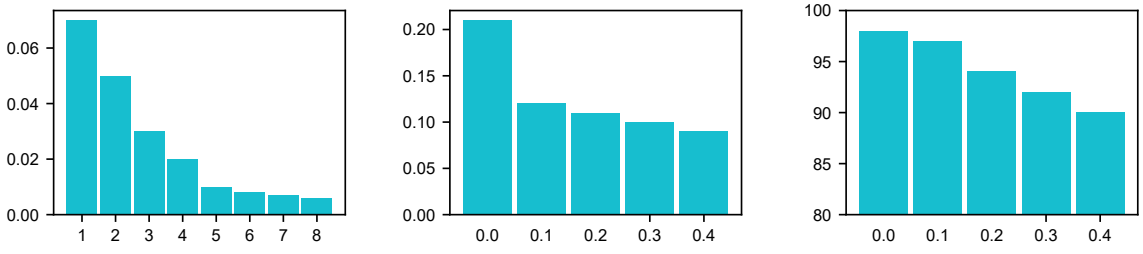



(a) Memory-based TGNN (b) RNN-based TGNN (c) Attention-based TGNN

Figure 3: An illustration of the error patterns across different types of TGNNs. The x-axis represents the time during the test period, and the color density indicates the error density (number of errors per time unit). A higher density (redder) indicates more errors. As shown in the figures, memory-based TGNNs exhibit a higher error density toward the end of the testing period, while RNN-based TGNNs display a higher error density at the beginning of the testing period. Attention-based TGNNs, on the other hand, demonstrate a more uniform error distribution.

| Dataset | Reddit | | Wikipedia | | MOOC | | LastFM | | GDELT | |
|---|---|---|---|---|---|---|---|---|---|---|
| Model/Metric | VCS ↓ | AP(%) ↑ | VCS ↓ | AP(%) ↑ | VCS ↓ | AP(%) ↑ | VCS ↓ | AP(%) ↑ | VCS ↓ | AP(%) ↑ |
| TGN | 0.18±0.02 | 98.5±0.04 | 0.21±0.04 | 96.4±0.03 | 0.25±0.03 | 97.6±0.03 | 0.22±0.04 | 75.4±0.06 | 0.24±0.03 | 95.6±0.05 |
| TGN-VCA | **0.08±0.01** | 98.2±0.03 | **0.12±0.02** | 96.3±0.04 | **0.13±0.03** | 97.3±0.02 | **0.09±0.03** | 73.3±0.05 | **0.12±0.02** | 96.8±0.03 |
| Tiger | 0.23±0.01 | 97.5±0.08 | 0.23±0.03 | 94.8±0.06 | 0.30±0.02 | 95.1±0.04 | 0.23±0.03 | 77.7±0.05 | 0.23±0.03 | 97.5±0.03 |
| Tiger-VCA | **0.10±0.01** | 98.0±0.06 | **0.11±0.02** | 94.0±0.06 | **0.11±0.01** | 95.6±0.03 | **0.12±0.02** | 78.0±0.04 | **0.11± 0.01** | 97.0±0.05 |
| JOIDE | 0.19±0.03 | 96.5±0.05 | 0.25±0.04 | 95.3±0.04 | 0.21±0.03 | 97.5±0.08 | 0.20±0.03 | 72.5±0.06 | 0.27±0.04 | 96.8±0.05 |
| JOIDE-VCA | **0.09±0.02** | 96.8±0.03 | **0.11±0.03** | 94.8±0.05 | **0.11±0.02** | 97.8±0.06 | **0.10±0.02** | 72.8±0.07 | **0.13±0.03** | 97.0±0.04 |
| DyRep | 0.25±0.03 | 96.7±0.06 | 0.22±0.04 | 94.8±0.03 | 0.23±0.03 | 96.8±0.06 | 0.27±0.03 | 69.5±0.05 | 0.24± 0.04 | 97.8±0.03 |
| DyRep-VCA | **0.11±0.02** | 97.0±0.05 | **0.10±0.03** | 95.0±0.04 | **0.12±0.02** | 97.0±0.05 | **0.12±0.01** | 70.0±0.06 | **0.14±0.03** | 97.5±0.04 |
| TCL | 0.12±0.02 | 95.5±0.02 | 0.11±0.02 | 91.6±0.06 | 0.14±0.03 | 93.5±0.07 | 0.14±0.03 | 68.5±0.07 | 0.14±0.03 | 94.6±0.06 |
| TCL-VCA | **0.09±0.02** | 95.2±0.02 | **0.06±0.01** | 92.2±0.06 | **0.10±0.02** | 92.8±0.05 | **0.09±0.01** | 67.5±0.03 | **0.10±0.0** | 95.2±0.06 |
| TGAT | 0.13±0.02 | 95.8±0.03 | 0.10±0.01 | 92.3±0.03 | 0.14±0.03 | 94.3±0.03 | 0.13±0.03 | 70.1±0.05 | 0.12±0.03 | 93.3±0.03 |
| TGAT-VCA | **0.10±0.02** | 96.0±0.02 | **0.08±0.01** | 93.0±0.04 | **0.07±0.02** | 95.0±0.05 | **0.06±0.01** | 71.3±0.06 | **0.09±0.02** | 93.0±0.04 |
| ΔVCS | 0.09 | | 0.09 | | 0.1 | | 0.1 | | 0.09 | |

Table 1: The VCS of TGNNs with and without the VCA learning objective. The experiment follows the standard setting. Models labelled with .-VCA are trained using our proposed learning objective as defined in Eq. 4.4, with $\tau = 5$ and $\gamma = 0.1$. ↓ indicates that smaller values are better, while ↑ indicates that larger values are better. The bolded entry indicate improvement with VCA. The last row $\Delta$ shows the average improvement with VCA for each dataset. The results in this table collectively demonstrate that VCS can successfully detect volatility clusters in errors, and VCA is effective in mitigating them.




(a) $\tau$ vs Variance (b) $\gamma$ vs VCS (c) $\gamma$ vs AP

Figure 4: An illustration of the effects of the hyper-parameters $\tau$ and $\gamma$ on VCS and VCA. Fig. 4(b) and .4(c) demonstrate that as $\gamma$ increases, VCS performance improves while AP decreases. Hence, $\gamma$ serves as a control variable that manages the trade-off between VCS and AP. Fig. 4(a) shows that increasing $\tau$ reduces the variance in the measure, but the marginal gain diminishes after $\tau = 5$.

application of TGNNs is in spatio-temporal networks, where vertices represent physical locations, incorporating a spatial dimension. It would be intriguing to explore whether similar concepts can be extended to examine the spatial aspects of TGNNs in spatio-temporal graph networks. This represents another promising area for future research.

## Acknowledgements

We thank our anonymous reviewers for the valuable feedbacks. This research is supported by the Natural Science Foundation of China (No. 42302326), the Anhui Province Key Research and Development Plan project (No.2022107020029)

## References

Banerjee, A.; and Dave, R. N. 2004. Validating clusters using the Hopkins statistic. In *2004 IEEE International conference on fuzzy systems (IEEE Cat. No. 04CH37542)*, volume 1, 149–153. IEEE.

Dwivedi, V. P.; Joshi, C. K.; Luu, A. T.; Laurent, T.; Bengio, Y.; and Bresson, X. 2023. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43): 1–48.

Errica, F.; Podda, M.; Bacciu, D.; and Micheli, A. 2019. A fair comparison of graph neural networks for graph classification. *arXiv preprint arXiv:1912.09893*.

Guo, S.; Lin, Y.; Feng, N.; Song, C.; and Wan, H. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 922–929.

Haghani, S.; and Keyvanpour, M. R. 2019. A systemic analysis of link prediction in social network. *Artificial Intelligence Review*, 52: 1961–1995.

Hopkins, B.; and Skellam, J. G. 1954. A new method for determining the type of distribution of plant individuals. *Annals of Botany*, 18(2): 213–227.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33: 22118–22133.

Huang, S.; Poursafaei, F.; Danovitch, J.; Fey, M.; Hu, W.; Rossi, E.; Leskovec, J.; Bronstein, M.; Rabusseau, G.; and Rabbany, R. 2024. Temporal graph benchmark for machine learning on temporal graphs. *Advances in Neural Information Processing Systems*, 36.

Junuthula, R.; Xu, K.; and Devabhaktuni, V. 2018. Leveraging friendship networks for dynamic link prediction in social interaction networks. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 12.

Junuthula, R. R.; Xu, K. S.; and Devabhaktuni, V. K. 2016. Evaluating link prediction accuracy in dynamic networks with added and removed edges. In *2016 IEEE international conferences on big data and cloud computing (BDCloud), social computing and networking (SocialCom), sustainable computing and communications (SustainCom)(BDCloud-SocialCom-SustainCom)*, 377–384. IEEE.

Kazemi, S. M.; Goel, R.; Jain, K.; Kobyzev, I.; Sethi, A.; Forsyth, P.; and Poupart, P. 2020. Representation learning for dynamic graphs: A survey. *The Journal of Machine Learning Research*, 21(1): 2648–2720.

Khodayar, M.; and Wang, J. 2018. Spatio-temporal graph deep neural network for short-term wind speed forecasting. *IEEE Transactions on Sustainable Energy*, 10(2): 670–681.

Kumar, S.; Zhang, X.; and Leskovec, J. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 1269–1278.

Luo, Y.; and Li, P. 2022. Neighborhood-aware scalable temporal network representation learning. In *Learning on Graphs Conference*, 1–1. PMLR.

Lv, Q.; Ding, M.; Liu, Q.; Chen, Y.; Feng, W.; He, S.; Zhou, C.; Jiang, J.; Dong, Y.; and Tang, J. 2021. Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 1150–1160.

Pareja, A.; Domeniconi, G.; Chen, J.; Ma, T.; Suzumura, T.; Kanezashi, H.; Kaler, T.; Schardl, T.; and Leiserson, C. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 5363–5370.

Poursafaei, F.; Huang, S.; Pelrine, K.; ; and Rabbany, R. 2022. Towards Better Evaluation for Dynamic Link Prediction. In *Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks*.

Rossi, E.; Chamberlain, B.; Frasca, F.; Eynard, D.; Monti, F.; and Bronstein, M. 2021. Temporal Graph Networks for Deep Learning on Dynamic Graphs. In *Proceedings of International Conference on Learning Representations*.

Sankar, A.; Wu, Y.; Gou, L.; Zhang, W.; and Yang, H. 2020. 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*, 519–527.

Sheng, G.; Su, J.; Huang, C.; and Wu, C. 2024. Mspipe: Efficient temporal gnn training via staleness-aware pipeline. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2651–2662.

Skarding, J.; Gabrys, B.; and Musial, K. 2021. Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. *IEEE Access*, 9: 79143–79168.

Souza, A.; Mesquita, D.; Kaski, S.; and Garg, V. 2022. Provably expressive temporal graph networks. *Advances in Neural Information Processing Systems*, 35: 32257–32269.

Su, J.; and Wu, S. 2024. Temporal-Aware Evaluation and Learning for Temporal Graph Neural Networks. arXiv:2412.07273.

Su, J.; Wu, S.; and Li, J. 2024. MTRGL:Effective Temporal Correlation Discerning through Multi-modal Temporal Relational Graph Learning. arXiv:2401.14199.

Su, J.; Zou, D.; and Wu, C. 2024a. On the Limitation and Experience Replay for GNNs in Continual Learning. arXiv:2302.03534.

Su, J.; Zou, D.; and Wu, C. 2024b. PRES: Toward Scalable Memory-Based Dynamic Graph Neural Networks. *arXiv preprint arXiv:2402.04284*.

Su, J.; Zou, D.; Zhang, Z.; and Wu, C. 2023. Towards robust graph incremental learning on evolving graphs. In *International Conference on Machine Learning*, 32728–32748. PMLR.

Trivedi, R.; Dai, H.; Wang, Y.; and Song, L. 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *international conference on machine learning*, 3462–3471. PMLR.

Trivedi, R.; Farajtabar, M.; Biswal, P.; and Zha, H. 2019. Dyrep: Learning representations over dynamic graphs. In *International conference on learning representations*.

Wang, D.; Zhang, Z.; Zhou, J.; Cui, P.; Fang, J.; Jia, Q.; Fang, Y.; and Qi, Y. 2021a. Temporal-aware graph neural network for credit risk prediction. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, 702–710. SIAM.

Wang, L.; Chang, X.; Li, S.; Chu, Y.; Li, H.; Zhang, W.; He, X.; Song, L.; Zhou, J.; and Yang, H. 2021b. Tcl: Transformer-based dynamic graph modelling via contrastive learning. *arXiv preprint arXiv:2105.07944*.

Wang, X.; Lyu, D.; Li, M.; Xia, Y.; Yang, Q.; Wang, X.; Wang, X.; Cui, P.; Yang, Y.; Sun, B.; et al. 2021c. Apan: Asynchronous propagation attention network for real-time temporal graph embedding. In *Proceedings of the 2021 international conference on management of data*, 2628–2638.

Wang, Y.; and Mendis, C. 2023. 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*, 354–368.

Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; and Achan, K. 2020a. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*.

Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; and Achan, K. 2020b. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*.

Yu, L.; Sun, L.; Du, B.; and Lv, W. 2023. Towards better dynamic graph learning: New architecture and unified library. *Advances in Neural Information Processing Systems*, 36: 67686–67700.

Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; and Vinyals, O. 2021a. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3): 107–115.

Zhang, Q.; Chang, J.; Meng, G.; Xiang, S.; and Pan, C. 2020. Spatio-temporal graph structure learning for traffic forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 1177–1185.

Zhang, Y.; Xiong, Y.; Li, D.; Shan, C.; Ren, K.; and Zhu, Y. 2021b. CoPE: modeling continuous propagation and evolution on interaction graph. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2627–2636.

Zhang, Y.; Xiong, Y.; Liao, Y.; Sun, Y.; Jin, Y.; Zheng, X.; and Zhu, Y. 2023. TIGER: Temporal Interaction Graph Embedding with Restarts. arXiv:2302.06057.

Zhao, L.; Song, Y.; Zhang, C.; Liu, Y.; Wang, P.; Lin, T.; Deng, M.; and Li, H. 2019. T-GCN: A temporal graph convolutional network for traffic prediction. *IEEE transactions on intelligent transportation systems*, 21(9): 3848–3858.

Zhou, H.; Zheng, D.; Nisa, I.; Ioannidis, V.; Song, X.; and Karypis, G. 2022. Tgl: A general framework for temporal gnn training on billion-scale graphs. *arXiv preprint arXiv:2203.14883*.