# FTM: A Frame-Level Timeline Modeling Method for Temporal Graph Representation Learning

**Bowen Cao**[1*], **Qichen Ye**[1*], **Weiyuan Xu**[1], **Yuexian Zou**[1,2†]

[1]ADSPLAB, School of ECE, Peking University, Shenzhen, China
[2]Peng Cheng Laboratory, Shenzhen, China
{cbw2021,xuwy}@stu.pku.edu.cn, {yeeeqichen,zouyx}@pku.edu.cn

## Abstract

Learning representations for graph-structured data is essential for graph analytical tasks. While remarkable progress has been made on static graphs, researches on temporal graphs are still in its beginning stage. The bottleneck of the temporal graph representation learning approach is the neighborhood aggregation strategy, based on which graph attributes share and gather information explicitly. Existing neighborhood aggregation strategies fail to capture either the short-term features or the long-term features of temporal graph attributes, leading to unsatisfactory model performance and even poor robustness and domain generality of the representation learning method. To address this problem, we propose a Frame-level Timeline Modeling (**FTM**) method that helps to capture both short-term and long-term features and thus learns more informative representations on temporal graphs. In particular, we present a novel link-based framing technique to preserve the short-term features and then incorporate a timeline aggregator module to capture the intrinsic dynamics of graph evolution as long-term features. Our method can be easily assembled with most temporal GNNs. Extensive experiments on common datasets show that our method brings great improvements to the capability, robustness, and domain generality of backbone methods in downstream tasks. Our code can be found at https://github.com/yeeeqichen/FTM.

## Introduction

Graph representation learning intends to transform nodes and links on the graph into lower-dimensional vector embeddings, which can be quite challenging due to the complex graph topological structures and node/link attributes. While approaches on **static graphs** have made breakthroughs and demonstrated distinguishable applicability in various fields (Graepel et al. 2010; He et al. 2014; Li et al. 2022; Zhu et al. 2022), those on **temporal graphs** are just getting started. Modeling a temporal graph (which may evolve over time with the addition, deletion, and changing of its attributes) is a core problem in developing real-world industrial systems (*e.g.*, social network, citation network, recommendation systems) where many data are time-dependent, and is much more difficult because of the temporal factors. Figure 1 gives an example of temporal graph modeling.

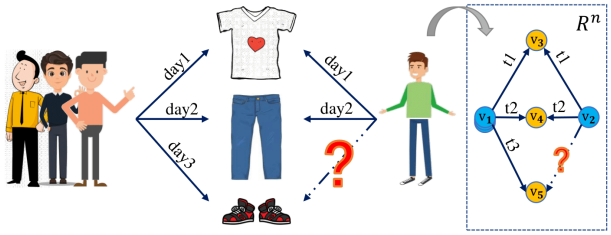



Figure 1: An example of temporal graph modeling. Given a model that has learnt the dynamics of a large number of users' shopping behaviors in high-dimensional space, what the man in green tends to buy in the future is predictable.

In learning representations on temporal graphs, a key point is the **neighborhood aggregation strategy**, which allows information passing and gathering among graph attributes, so that nodes learn their representations from their neighbors. For static graphs, directly linked nodes are neighbors to each other because they all appear in the one and only topology. In contrast, temporal graph attributes scatter sparsely across the timeline, leading to temporal-structure inconsistency. For any node in a temporal graph, a node connected to it is not necessarily a neighbor, for this node may appear a long time ago or disappear soon. Each node in a temporal graph may also have several temporal neighborhoods, posing a challenge for information aggregation. Therefore, how to design the neighborhood aggregation strategy on temporal graphs remains an open question.

Recent works introduce snapshot-based methods (Kumar, Zhang, and Leskovec 2019; Pareja et al. 2020) and temporal random walk-based methods (Nguyen et al. 2018; Xu et al. 2020) for neighborhood aggregation, but are often too simple to capture the evolution of temporal graphs over time. The comparison of the above two methods and our method is shown in Figure 2. In particular, snapshot-based methods equally slice the timeline into a sequence of snapshots, each of which contains nodes and links that occurred within its time span. This kind of method treats a snapshot as a static graph and fails to model the temporal properties within a snapshot, losing short-term features of graph attributes. On

*These authors contributed equally.
†Corresponding author.

the other hand, temporal random walk-based methods do not impose restrictions on the time range, but select temporal neighbors from the past according to a certain rule (most often randomly) and learn representations based on the neighborhood attributes and their time information. However, the problem is that the randomly constructed temporal neighborhood cannot ensure a balance between short-term features and long-term features.

To develop a representation learning method on temporal graphs that adequately captures both short-term and long-term features, we propose a simple but effective **F**rame-level **T**imeline **M**odeling method (**FTM** for short), at the heart of which is the innovation of the temporal neighborhood aggregation strategy: first, we refer to the concept of frame[1] in signal processing, and put forward a novel method called **link-based framing technique**, where we separate most recent links into several frames (*i.e.*, temporal neighborhoods) to emphasize short-term features; then, we extract frame features with a **frame aggregator**, which can be easily replaced by most GNN methods; finally, we design a **timeline aggregator** for learning the intrinsic dynamics of successive frames across the timeline to capture long-term features.

We conduct experiments on several widely-used benchmarks in both transductive and inductive settings, and the results demonstrate the effectiveness of our proposed method. Moreover, the robustness and domain generality of baselines and our method are also evaluated through quantitative and qualitative experiments, which further suggest the insights of **FTM**. Our main contributions are summarized as follows:

- We propose a simple but effective frame-level timeline modeling method for temporal graph representation learning, namely **FTM**, which makes contributions to the neighborhood aggregation strategy, and can be easily assembled with most GNN methods.
- We conduct comprehensive experiments to show that models assembled with **FTM** achieve better performance on common benchmarks, and we further evaluate its effectiveness through quantitative and qualitative analyses.
- We point out the robustness and domain generality issues of several state-of-the-art GNN-based temporal graph representation learning methods, and demonstrate that FTM could greatly alleviate these issues.

## Related Work

Learning representations with GNNs has become a popular research area for graph modeling. Earlier works explore learning representations of topological structures (Kipf and Welling 2016a; Grover and Leskovec 2016), extending GNN to inductive learning (Hamilton, Ying, and Leskovec 2017), and integrating attention mechanisms (Veličković et al. 2018). In all these works, however, the time information of graph attributes are discarded.

Recent approaches take advantage of the temporal property. Certain approaches learn to access time-aware knowledge by equally slicing the timeline into a sequence of **snapshots** (Trivedi et al. 2019; Singer, Guy, and Radinsky 2019). They aggregate the topological features in a snapshot and combine time-dependent features with sequence-modeling techniques to learn temporal graph embeddings. However, they ignore the sequential nature of nodes and links within the same snapshot, losing short-term features that can guide learning. Meanwhile, the amount of nodes and links within each snapshot is inconsistent, leading to great data biases in learning topological features.

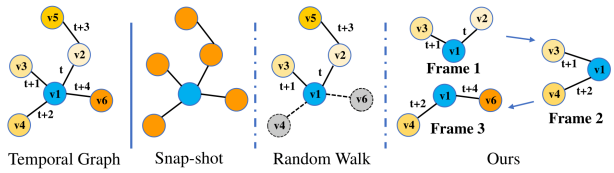



Figure 2: An example illustrating prior techniques and our link-based framing technique (where frame length is 2 and hop length is 1) for neighborhood construction.

More recently, TGAT (Xu et al. 2020) leverages a time encoding function to learn time-aware representations in continuous time. TGN (Rossi et al. 2020), as a variant of TGAT, integrates a memory module to keep track of the evolution of node-level features. These methods make progress in capturing short-term features since the time encoding makes it possible to model the temporal properties of a neighborhood. However, in most cases, they randomly sample neighbors from the past to form a temporal neighborhood for a target node, which means that they cannot ensure a balance between short-term features and long-term features.

Our work adopt the idea of time encoding, but make contributions to the way that temporal neighborhoods are constructed and information is aggregated, so that the model learn more informative representations.

## Proposed Method: FTM

### Problem Formalization

Graph representation learning aims to obtain node or link representations based on their own properties and their interactions with neighbors. Let $E^{T-} = \{e_{i,j,t} | 1 \leq i, j \leq n, 0 \leq t < T\}$ and $V^{T-} = \{v_s | s = 1 \dots n\}$ denote the set of links and the set of nodes observed before time $T$, respectively, where $n$ is the amount of nodes, $v_s$ is the $s$-th node ($s$ is only used to distinguish nodes), and $e_{i,j,t}$ is an link between $v_i$ and $v_j$ emerged at time $t \in \mathbb{R}^+$. Let $E_s^{T-} = \{e_{s,i,t} | 1 \leq i \leq n, 0 \leq t < T\} \cup \{e_{j,s,t} | 1 \leq j \leq n, 0 \leq t < T\}$ denotes the subset of $E^{T-}$ containing links that link to node $v_s$ and satisfy the time constraint (we mainly consider undirected graphs, where the two parts of $E_s^{T-}$ are equivalent). Supposing that $G^{T-} = (V^{T-}, E^{T-})$ denotes the final state of a temporal graph before time $T$, learning representations on it is mainly to obtain the node and link representations at time $t$ based on $G^{T-}$.

[1] A fundamental technique to decompose raw signal into multiple ranges according to frame length and hop length.

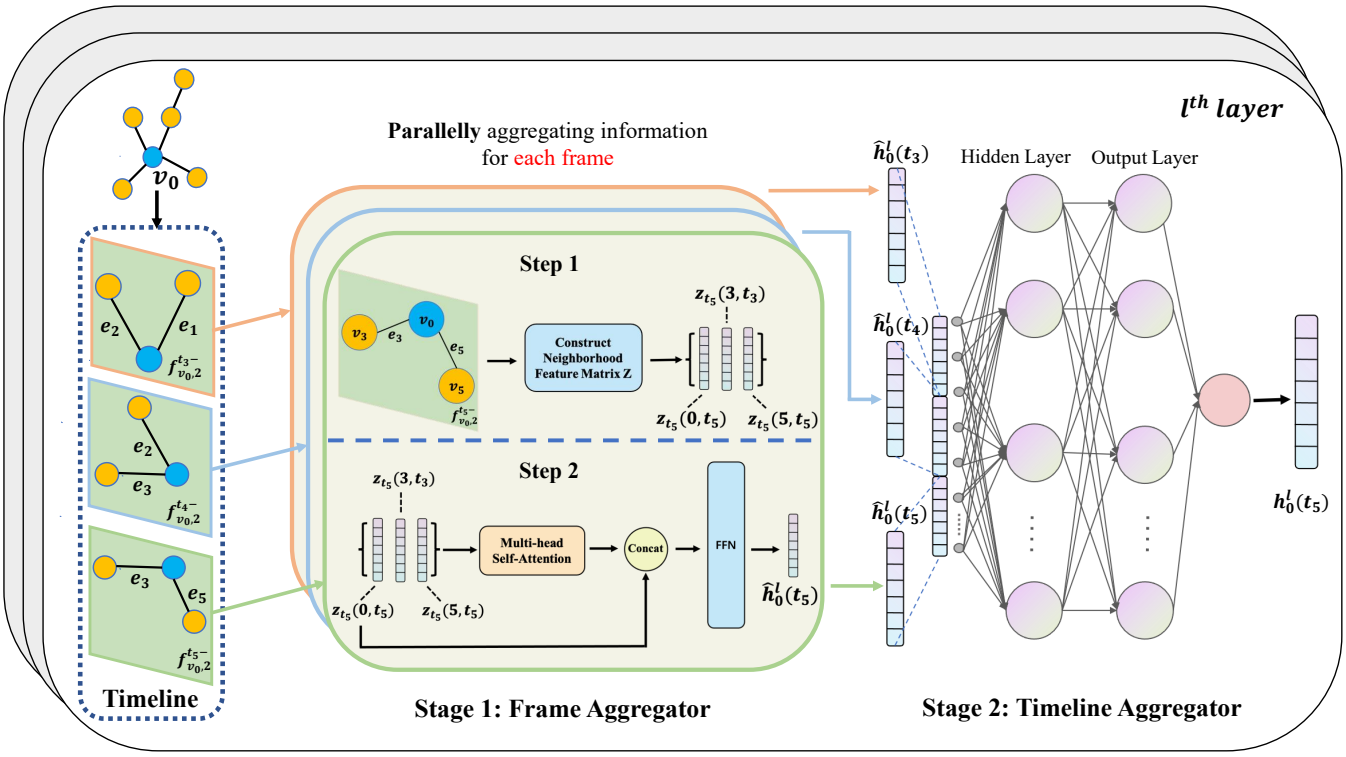



Figure 3: The architecture of the model assembling FTM with a backbone network. Assuming that our goal is to compute node $v_0$'s representation at timestamp $t_5$, we first construct a timeline consists of 3 frames $\{f^{t_3-}_{v_0,2}, f^{t_4-}_{v_0,2}, f^{t_5-}_{v_0,2}\}$ as each layer's input. At each layer - Stage 1, we compute each frame's representation $\hat{h}^l_0(t_j)$ in parallel through the backbone network (works as the frame aggregator). Stage 2, we aggregate all frames' representations to get the node representation via the timeline aggregator.

## Input Representation

Graph attributes can be recorded in various ways. For instance, online reviews are in text format, and citations are in triplet format. We encode text with BERT-base (Devlin et al. 2019), and other records with TransE (Bordes et al. 2013), to initialize node and link features. Then, we split links into frames, and feed the features of successive frames into FTM.

**Link-based Framing Technique.** The process of splitting links into temporal frames is controlled by two parameters:

- **Frame length** defines how many links are included in a frame. For example, at timestamp $t$, to construct a frame of length $k$ for node $v_s$, we take the most recent $k$ links from $E^{t-}_s$ to form this frame and denote it as $f^{t-}_{s,k}$.
- **Hop length** defines how many links to skip when taking the next frame. In practise, we set it to be $\frac{frame\ length}{2}$ (which is empirically the best and is also a convention in signal processing) to stabilize the training process. An example is provided in Figure 2.

## Frame-level Timeline Modeling

The main idea of FTM is to preserve both the short-term and long-term features of graph attributes through a **frame aggregator** and a **timeline aggregator**. The role of the frame aggregator is to model each neighborhood that generated by the link-based framing technique, so **it can be replaced by most GNN methods**. For example, the overall framework of the model assembling FTM with TGAT (Xu et al. 2020), *i.e.*, taking TGAT as the frame aggregator, is shown in Figure 3. Since TGAT is composed of a stack of identical layers (with shared parameters), the calculation process of each layer is similar. Assuming that we want node $v_i$'s embedding at timestamp $t$, the calculation process in layer $l$ can be described as the following two parts:

**Temporally Attentive Frame Aggregator.** While TGAT randomly samples links from the past to form temporal neighborhoods, we integrate $k$ **most recent** links to construct a frame in order to preserve short-term features. Meanwhile, the reason we add links by number rather than by time (as snapshot-based methods) is to guide the model to learn the common evolution of links, instead of time-interval-related knowledge. Given a frame $f^{t-}_{i,k}$ of $v_i$ that contains links $\{e_{i,j_1,t_1}, \ldots, e_{i,j_k,t_k}\}$, we obtain a temporal neighborhood feature matrix $\mathbf{Z}(t)$ as:

$$\mathbf{Z}(t) = [\mathbf{z}_t(i,t), \mathbf{z}_t(j_1,t_1), \ldots, \mathbf{z}_t(j_k,t_k)], \quad (1)$$

$$\mathbf{z}_t(j_k,t_k) = \left[\mathbf{h}^{(l-1)}_{j_k}(t_k) \,||\, \varphi(t-t_k) \,||\, \mathbf{e}_{j_k}\right], \quad (2)$$

where $\mathbf{h}^{(l-1)}_{j_k}(t_k)$ is the previous layer's output for $v_{j_k}$, $\varphi(\cdot)$ is a time encoding function, $\mathbf{e}_{j_k}$ is the feature vector of $e_{i,j_k,t_k}$, and $\mathbf{z}_t(j_k,t_k)$ maps the information of $v_{j_k}$ into a

time-aware representation. Then, we attentively aggregate $Z_t$ with the multi-head self-attention mechanism:

$$\mathbf{q}^r(t) = [\mathbf{Z}(t)]_0 \mathbf{W}_Q^r, \tag{3}$$

$$\mathbf{K}^r(t) = [\mathbf{Z}(t)]_{1:N} \mathbf{W}_K^r,\ \mathbf{V}^r(t) = [\mathbf{Z}(t)]_{1:N} \mathbf{W}_V^r \tag{4}$$

$$\alpha_j^r = \frac{\exp\left(\mathbf{q}^{r\top} \mathbf{K}_j^r\right)}{\Sigma_q \exp\left(\mathbf{q}^{r\top} \mathbf{K}_q^r\right)},\ \tilde{\mathbf{h}}_i^{l,r}(t) = \sum\nolimits_j \alpha_j^r \mathbf{v}_j^r, \tag{5}$$

where $\mathbf{W}_Q^r, \mathbf{W}_K^r, \mathbf{W}_V^r$ are query, key and value matrix, respectively, $\alpha_j^r$ denotes the attention weight, and $\tilde{\mathbf{h}}_i^{l,r}(t)$ is the output of the $r$-th attention head. Assuming that we have $n_h$ attention heads, the frame representation $\hat{\mathbf{h}}_i^l(t)$ will be:

$$\hat{\mathbf{h}}_i^l(t) = \mathrm{ReLU}(\mathbf{y}\mathbf{W}_0 + \mathbf{b}_0)\mathbf{W}_1 + \mathbf{b}_1, \tag{6}$$

$$\mathbf{y} = \left[\mathbf{z}_t(i,t) || \tilde{\mathbf{h}}_i^{l,1}(t) || \ldots || \tilde{\mathbf{h}}_i^{l,n_h}(t)\right], \tag{7}$$

where $\mathbf{W}_0, \mathbf{W}_1$ are weights and $\mathbf{b}_0, \mathbf{b}_1$ are biases.

**Attentively Frame-level Timeline Aggregator.** In the prior part, we get the representation $\hat{\mathbf{h}}_i^l(t)$ of frame $f_{i,k}^{t-}$. Now, we consider how to aggregate the information of multiple frames. Empirically, we set the hop length to half of the frame length to retain redundant information between frames. By doing so, *(i)* short-term features are further highlighted; and *(ii)* framing serves as a **scrubbing technique** because irregular links (with abnormal time interval/content) will not play a leading role and the commonalities in the evolution of links will be emphasized.

Let $F_{i,k}^{t-} = \left\{f_{i,k}^{t_j-} | 1 \le j \le n, t_n = t\right\}$ denotes a set of frames of node $v_i$, in which the timestamps satisfy:

$$t_{j-1} = \mathbb{T}_{\frac{k}{2}}(E_i^{t_j-}), 2 \le j \le n \tag{8}$$

where $n$ is the size of this set, $\mathbb{T}_{\frac{k}{2}}$ maps a set of links to its $\frac{k}{2}$-th (*i.e.*, half of the frame length) recent element's timestamp. We call this set a $n$-length **timeline** of node $v_i$ at timestamp $t$, and we get the final node representation $\mathbf{h}_i^l(t)$ as:

$$\mathbf{h}_i^l(t) = \left[\hat{\mathbf{h}}_i^l(t_1) || \ldots || \hat{\mathbf{h}}_i^l(t_n)\right]^T \mathbf{W}_2 + \mathbf{b}_2, \tag{9}$$

where $\mathbf{W}_2$ and $\mathbf{b}$ are weights and bias. Here we take 1-layer MLP as an example for simplicity, but it could be effortlessly extended to RNN-based or attention-based methods, *etc.*

$\mathbf{h}_i^l(t)$ generated by the last layer is just what we want - node $v_i$'s embedding at timestamp $t$, $\mathbf{h}_i(t)$.

**Learning & Inference.** Since the temporal information is mostly reflected in the time-sensitive interactions among nodes, we choose to use the future link prediction setup for training. The goal of future link prediction is to predict the probability that an link will exist between a target node $v_i$ and another node $v_j$ at a specific future time, *i.e.*, given the set of previous links of $v_i$, we compute the probability of a future link $e_{i,j,t_{i,j}}$ between $v_i$ and $v_j$. To train the model, we sample a set of negative links ($\neq e_{i,j,t_{i,j}}$) and optimize the per-node objective:

$$L = \sum_{v_i, v_j, t_{i,j}} \mathrm{Pos}\left(i, j, t_{i,j}\right) + Q \cdot E_{v_q \sim P}\, \mathrm{Neg}\left(i, q, t_{i,j}\right) \tag{10}$$

| Dataset | Node | Link |
|---|---|---|
| Reddit (2019) | 11,000 | 672,000 |
| Wikipedia (2019) | 9,000 | 157,000 |
| Icews14 (2018) | 7,000 | 91,000 |
| Icews05-15 (2018) | 10,000 | 461,000 |
| Bitcoin-otc (2016) | 6,000 | 36,000 |
| Bitcoin-alpha (2016) | 4,000 | 24,000 |
| Mooc (2019) | 7,000 | 412,000 |

Table 1: The node and link statics for each dataset.

where $P$ is the negative link sampling distribution, $Q$ denotes the negative sampling size, $\mathrm{Pos}(\cdot,\cdot,\cdot)$ and $\mathrm{Neg}(\cdot,\cdot,\cdot)$ denote the positive and negative scoring functions:

$$\mathrm{Pos}\left(i, j, t_{i,j}\right) = -\log\left(\sigma\left(-\mathbf{h}_i(t_{i,j})^\top \mathbf{h}_j(t_{i,j})\right)\right) \tag{11}$$

$$\mathrm{Neg}\left(i, q, t_{i,j}\right) = -\log\left(\sigma\left(\mathbf{h}_i(t_{i,j})^\top \mathbf{h}_q(t_{i,j})\right)\right) \tag{12}$$

where $\sigma(\cdot)$ is an activation function, $\mathbf{h}_i(t)$ is the representation of node $v_i$ at timestamp $t$. For inference, the output of $\mathrm{Pos}(i, j, t_{i,j})$ is used as the logits.

## Experimental Setups

We evaluate our method against strong baselines (adapted to temporal settings when possible). Note that **assembling FTM with a baseline method** means that we take the baseline method as the frame aggregator of FTM.

### Tasks and Metrics

We perform future link prediction to evaluate the quality of the generated graph representations. We use average precision (AP) as the evaluation metric and consider this task in two settings: *(i)* **Transductive Task.** We predict future links among nodes that have been observed during training. *(ii)* **Inductive Task.** We perform future link prediction among nodes that have not been observed in the training phase.

### Datasets

We choose seven datasets that contain time-sensitive node interactions: **Reddit**[2] is created from posts between active users and subreddits, where users and subreddits are nodes, and posts are links. **Wikipedia**[3] is created by taking top edited pages in Wikipedia and active users as nodes, and the corresponding edits as links. **Icews14**[4], **Icews05-15**[5] contain political events and the corresponding timestamps. All nodes are real-world entities (e.g. countries) and links are event types. **Bitcoin-otc**[6], **Bitcoin-alpha**[7] are who-trusts-whom networks of people who trade with Bitcoin, where nodes are people and links are the credit evaluation. **Mooc**[8]

[2] http://snap.stanford.edu/jodie/reddit.csv
[3] http://snap.stanford.edu/jodie/wikipedia.csv
[4] https://github.com/nle-ml/mmkb
[5] https://github.com/nle-ml/mmkb
[6] https://snap.stanford.edu/data/soc-sign-bitcoinotc.csv.gz
[7] https://snap.stanford.edu/data/soc-sign-bitcoinalpha.csv.gz
[8] https://snap.stanford.edu/data/act-mooc.tar.gz

| Model | Reddit | | Wikipedia | |
|---|---|---|---|---|
| | Transductive | Inductive | Transductive | Inductive |
| GAE (2016b) | 93.23 | - | 91.44 | - |
| VAGE (2016b) | 92.92 | - | 91.34 | - |
| DeepWalk (2014) | 83.10 | - | 90.71 | - |
| Node2vec (2016) | 84.56 | - | 91.48 | - |
| CTDNE (2018) | 91.41 | - | 92.17 | - |
| DyRep (2019) | 98.25 | 96.11 | 94.76 | 92.11 |
| Jodie (2019) | 97.02 | 94.46 | 92.75 | 93.13 |
| GraphSAGE (2017) | 97.20 | 94.68 | 91.09 | 86.08 |
| **w/ FTM** | 98.01↑ | 96.28↑ | 92.91↑ | 91.93↑ |
| GAT (2018) | 97.33 | 95.37 | 94.73 | 91.27 |
| **w/ FTM** | 98.21↑ | 96.75↑ | 95.03↑ | 93.54↑ |
| TGAT (2020) | 98.27 | 96.73 | 95.13 | 93.97 |
| **w/ FTM** | 98.41↑ | 96.82↑ | 97.82↑ | 97.14↑ |
| TGN (2020) | 98.78 | 97.77 | 98.28 | 97.69 |
| **w/ FTM** | **98.88**↑ | **97.96**↑ | **98.82**↑ | **98.33**↑ |
| Average Gain | 0.48 | 0.82 | 1.34 | 2.98 |

Table 2: AP(%) for future link prediction tasks. ↑ means that FTM brings an improvement to the baseline method. The best results in each column are highlighted in bold font. '-' denotes incapability.

| Model | Attack Intensity(%) | | | | | |
|---|---|---|---|---|---|---|
| | **0** | **10** | **20** | **30** | **40** | **50** |
| GraphSAGE | 85.46(+1.58) | 34.11(+28.14) | 51.78(+3.56) | 45.89(+5.85) | 40.81(+3.04) | 48.84(+9.15) |
| GAT | 83.75(+4.31) | 49.56(+17.66) | 40.47(+19.32) | 41.89(+16.11) | 36.70(+9.39) | 41.38(+15.06) |
| TGAT | 87.36(+1.37) | 59.99(+26.83) | 56.59(+29.85) | 47.39(+38.39) | 34.61(+51.56) | 38.57(+47.29) |
| TGN | 88.19(+1.82) | 80.80(+3.18) | 81.93(+2.63) | 81.81(+4.96) | 83.39(+2.09) | 83.17(+2.40) |
| Average Gain | 2.27 | 18.95 | 13.84 | 16.33 | 16.52 | 18.48 |

Table 3: AUC (%) for node classification tasks on Wikipedia. Attack intensity controls the ratio of (the norm of) the added noise to (the maximum norm of) the link features in the dataset. x(+y) indicates that the baseline method achieves x% in AUC, and FTM brings an improvement of y% to it, *i.e.*, the model assembling FTM with this method achieves x+y%.

dataset contains user actions on a popular MOOC platform, where nodes represent users and course activities, and links represent user actions. Dataset scales are listed in Table 1.

## Baselines

**GAE**, **VAGE** (Kipf and Welling 2016b), **DeepWalk** (Perozzi, Al-Rfou, and Skiena 2014) and **Node2vec** (Grover and Leskovec 2016) are models for static graphs.

**CTDNE**, **DyRep**, **Jodie**, **GraphSAGE**, **GAT**, **TGAT** and **TGN** are baselines for temporal graphs. We do not ensemble FTM with CTDNE, DyRep and Jodie due to the conflicting schemes[9]. For other methods, we test the original version and the FTM-assembled version. There may be slight differences between our implementation and others, but it is fair for comparison.

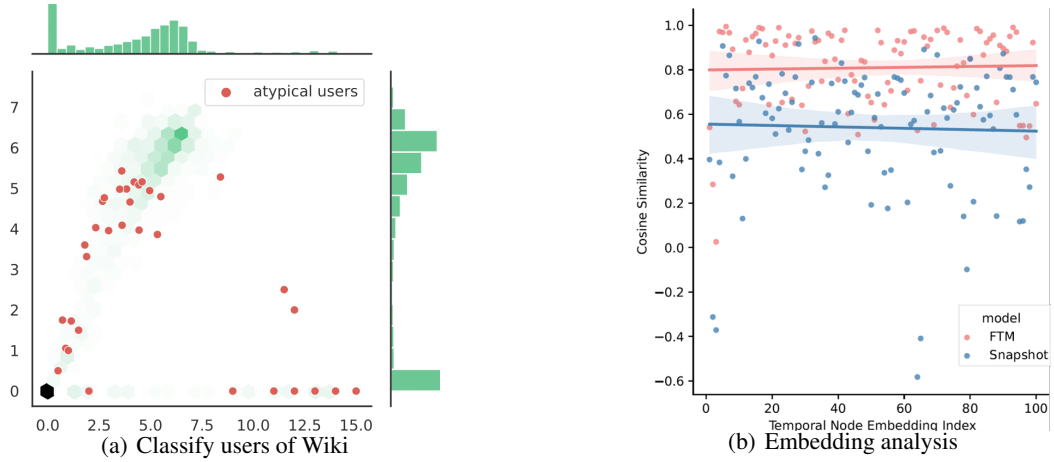
[9]These methods have their own custom temporal neighborhood construction strategies. If we apply our action-based framing technique to these methods, we are only assembling FTM with their feature extraction modules.

## Results and Analysis

**Transductive & Inductive Future Link Prediction.** As shown in Table 2, (1) temporal methods surpass static ones, suggesting the importance of temporal properties in modeling temporal graphs; (2) models assembled with FTM consistently outperform the originals on all benchmarks, demonstrating the effectiveness of FTM. For instance, on Wikipedia, FTM brings an average gain of **2.98** in AP under inductive setting. Meanwhile, TGN+FTM achieves new state-of-the-art performance on both Wikipedia and Reddit. The overall performance on this task indicates that FTM guides the learning of the evolution of temporal graphs and helps to generate more informative representations.

## Quantitative Analysis

Given these overall performance improvements, we investigate how FTM's improvements are reflected in the learnt node representations. Because we have the gold label of node type in Wikipedia, we conduct a downstream task of future link prediction, **node classification**, in two settings: *(i)* **Fine-tuning.** We fine-tune a MLP layer to classify nodes based on the learnt node embeddings. As the result in the




(a) Classify users of Wiki (b) Embedding analysis

Figure 4: (a) x-axis/y-axis represents the average/standard deviation of the time intervals of a user's actions. The green parts denotes user distribution. The darker the color, the greater the number of users. Red points denote atypical users that have misled TGAT but are correctly classified by TGAT+FTM. (b) The cosine similarity of successive temporal node embeddings generated by TGAT+FTM and TGAT+Snapshot, respectively. The consistency of the embeddings generated by TGAT+FTM proves that FTM helps to learn stable temporal representations.

| Training Dataset | Model | Test Dataset | | | | |
|---|---|---|---|---|---|---|
| | | **Icews14** | **Icews05-15** | **Bitcoin-otc** | **Bitcoin-alpha** | **Mooc** |
| **Reddit** | GraphSAGE | 46.89(+35.32) | 61.48(+23.08) | 70.36(+7.59) | 54.44(+16.09) | 49.86(+3.38) |
| | GAT | 63.45(+24.32) | 64.44(+20.81) | 70.66(+6.30) | 61.35(+9.49) | 47.28(+7.25) |
| | TGAT | 76.29(+9.82) | 72.80(+15.47) | 70.19(+10.81) | 65.46(+8.11) | 57.01(+16.98) |
| | TGN | 68.63(+12.20) | 70.57(+15.72) | 72.86(+6.48) | 64.55(+6.04) | 67.23(+2.48) |
| | Average Gain | 20.42 | 18.77 | 7.80 | 9.93 | 7.52 |
| **Wikipedia** | GraphSAGE | 71.88(+7.59) | 77.49(+3.46) | 58.88(+12.44) | 53.81(+18.16) | 49.11(+4.85) |
| | GAT | 67.19(+12.29) | 69.32(+15.20) | 67.20(+0.34) | 61.48(+6.71) | 49.42(+7.19) |
| | TGAT | 80.27(+6.94) | 82.03(+10.82) | 71.38(+12.16) | 71.01(+2.18) | 53.98(+22.54) |
| | TGN | 66.40(+15.73) | 67.77(+16.36) | 83.76(+0.41) | 64.69(+7.29) | 73.20(+1.86) |
| | Average Gain | 10.64 | 11.46 | 6.34 | 8.59 | 9.11 |

Table 4: AP (%) of future link prediction tasks. x(+y) indicates that the baseline method achieves x% in AP, and FTM brings an improvement of y% to it, *i.e.*, the model assembling FTM with this method achieves x+y%.

second column of Table 3 (attack intensity is 0) shows, FTM brings about 1%˜4% absolute gain in AUC to backbone methods, which reveals that models assembled with FTM generate more reasonable node embeddings. It also demonstrates the insights of our method in temporal graph representation learning; *(ii)* **Adversarial Attack.** The ability to resist Gaussian noise-perturbated examples is important because noisy data is inevitable under most circumstances (Cheng et al. 2023). We add random Gaussian noise to the original data to generate adversarial examples for five times, and record the average performance of each model. The results are reported in the last six columns of Table 3 (with attack intensity from 10% to 50%). The average gains that FTM brings to the baseline methods demonstrate that FTM can handle data noise (and maybe data biases) better, which is an important capability that guarantees the applicability of the proposed method.

## Qualitative Analysis

In this section, we examine our model's ability to generate more informative representations on the wikipedia dataset qualitatively. As Figure 4(a) shows, FTM helps to distinguish atypical users, whereas baselines are often misled; it reflects the potential of FTM in addressing data biases, since the data bias issues in data collected from platforms like Wikipedia are mainly caused by atypical users who often perform irregular/abnormal actions. Moreover, we hypothesize that the evolution of user actions has short-term stationary features, because people's personality will not change rapidly. We take the most popular snapshot-based modeling method as the opponent to demonstrate that FTM makes it possible to capture short-term stationary features over time. First, we modify the neighborhood sampling strategy of the original TGAT to be snapshot-based, namely TGAT+Snapshot. Specifically, for each node we take its neighbors within an hour to form a temporal neighbor-

| Model | Neighborhood Scale | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Inductive | | | | Generalization | | | |
| | **S** | **M** | **L** | **XL** | **S** | **M** | **L** | **XL** |
| GraphSAGE | 86.31 | 88.96 | 94.19 | 94.68 | 70.87 | 70.83 | 78.74 | 83.59 |
| **w/ FTM** | 92.24↑ | 92.31↑ | 95.53↑ | 96.28↑ | 79.37↑ | 77.26↑ | 86.30↑ | 86.53↑ |
| GAT | 91.11 | 93.15 | 95.56 | 95.37 | 69.88 | 74.96 | 83.76 | 85.84 |
| **w/ FTM** | 91.85↑ | 93.40↑ | 95.84↑ | 96.75↑ | 82.38↑ | 81.75↑ | 86.20↑ | 88.97↑ |
| TGAT | 91.12 | 92.63 | 95.95 | 96.73 | 69.22 | 71.76 | 85.64 | 87.34 |
| **w/ FTM** | 94.08↑ | 94.32↑ | 97.26↑ | 96.82↑ | 91.08↑ | 89.52↑ | 95.82↑ | 91.06↑ |
| Average Gain | 3.21 | 1.76 | 0.98 | 1.02 | 14.29 | 10.33 | 6.73 | 3.26 |

Table 5: Case studies on neighborhood scale, where neighborhood scale expands from S to XL. We do not take TGN into consideration, because the way TGN updates node-wise memory has little to do with the neighborhood scale and the percentage of training data. We report AP(%) of future link prediction on Reddit (inductive; generalize from Wiki).

| Model | Percentage of Training Data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Inductive | | | | Generalization | | | |
| | **1%** | **5%** | **10%** | **50%** | **1%** | **5%** | **10%** | **50%** |
| GraphSAGE | 65.31 | 85.39 | 91.17 | 95.64 | 57.99 | 62.79 | 73.04 | 80.15 |
| **w/ FTM** | 70.40↑ | 87.58↑ | 91.95↑ | 96.65↑ | 61.98↑ | 74.92↑ | 82.71↑ | 85.34↑ |
| GAT | 68.99 | 90.81 | 93.13 | 95.10 | 59.53 | 76.44 | 79.70 | 85.80 |
| **w/ FTM** | 73.13↑ | 91.02↑ | 93.70↑ | 96.68↑ | 68.45↑ | 81.99↑ | 85.91↑ | 90.30↑ |
| TGAT | 65.65 | 88.92 | 92.67 | 96.25 | 74.51 | 77.16 | 81.27 | 86.38 |
| **w/ FTM** | 80.76↑ | 92.32↑ | 93.45↑ | 96.25 | 81.84↑ | 87.88↑ | 87.22↑ | 88.53↑ |
| Average Gain | 8.10 | 1.67 | 0.64 | 0.69 | 5.00 | 8.25 | 5.69 | 2.87 |

Table 6: Case studies on the percentage of training data, where models are trained on limited training data of Reddit, *e.g.*, 1% means models are trained/validated on one-percent of the original training/validation data. We report AP(%) of future link prediction on Reddit (inductive; generalize from Wiki).

hood. Then, we compute the cosine similarity of successive temporal node embeddings for TGAT+Snapshot and our TGAT+FTM respectively. As shown in Figure 4(b), the temporal node embeddings generated by TGAT+FTM show higher consistency. It demonstrates that TGAT+FTM learns more stable representations of users and we believe that the main reason lies in capturing short-term stationary features. Intuitively, this ability helps to stabilize the training process and capture the dynamics of user actions.

## Domain Generality

Our reported results thus far demonstrate the effectiveness of FTM in improving the capability and robustness of temporal GNNs. In this section, we explore whether FTM could help improve the domain generality of baseline methods. From the results shown in Table 4, we can observe that (1) these baseline methods suffer from severe domain generality issues, *e.g.*, GraphSAGE trained on Reddit only get **46.89** in AP on Icews14; and (2) assembling FTM with these baseline methods greatly improves their domain generality, *e.g.*, when applying models trained on Reddit to Icews14, FTM brings an average gain of **20.42** in AP to them. It illustrates the efficacy of FTM in deriving generalizable knowledge of graph evolution. Furthermore, we test the capability of our method in handling domain gaps from a new perspective - we subsample user-action data from the wikipedia dataset with different time interval distribution and evaluate our method on it. The result shows that assembling FTM with baseline methods improves their AP by **1.5** in average, but is not listed here for space-saving issues.

## Case Studies

In normal experiments, we set the number of model layers to be 2 and the length of frames to be 20 to form a node's temporal neighborhood. In this section, we record the performance of aforementioned methods under different neighborhood scales and data sizes. Note that the test data is the same as aforementioned experiments.

In studying the influence of neighborhood scale, we separately let (the number of model layers, the length of frames) be $(1, 10)$, $(1, 20)$, $(2, 10)$, $(2, 20)$ to form a S-scale, M-scale, L-scale, XL-scale neighborhood respectively. The results are provided in Table 5. In all cases, models assembled with FTM outperform the originals. It illustrates that, even under low-resource settings, assembling FTM with backbone methods can enhance the capability, the robustness, and the domain generality of these models.

In studying the influence of data size, we sample $x$-percent of the training/validation set to form new training/-validation sets. As the results in Table 6 illustrate, models assembled with FTM outperform the originals in most cases. It indicates that FTM is not totally data-driven, but superior in understanding the evolution of the temporal graph. This ability is of practical importance.

| Aggregation Function | AP on Wikipedia | | Convergence Time | Parameter Size |
|---|---|---|---|---|
| | Transductive | Inductive | | |
| 1-layer MLP | 97.68 | 97.19 | **$8.5 \times 10^3$ s** | **100%** |
| 2-layer MLP | 97.26 | 96.79 | $1.3 \times 10^4$ s | 105% |
| LSTM | 97.64 | 96.99 | $2.7 \times 10^4$ s | 112% |
| Self-attention | **97.93** | **97.44** | $1.1 \times 10^4$ s | 110% |

Table 7: Comparison of different aggregate functions in Timeline Aggregator module.

## Implementation & Training Details

**Hyper-parameters.** We do the chronological train-validation-test split with 70%-15%-15% according to the timestamps of links. In the test set, we randomly sample 10% nodes as 'new nodes' for inductive tasks, and mask down all their links in the training set. Both the number of self-attention layers and the number of heads in each layer of the backbone network are 2. The length of timeline is chosen from [2, 3, 4] (we only report the best result). During training, we use Adam optimizer with learning rate 1e-4. The dimension of time encoding vectors is set to 172, which is same to the dimension of link feature vectors. We have conducted experiments to verify the effect of different aggregate functions in the Timeline Aggregator module. The result is shown in Table 7 (timeline length is 2 and all experiments are conducted on a RTX 2080Ti GPU). Taking both the performance and efficiency into consideration, we decide to deploy a 1-layer MLP as the timeline aggregate function because it achieves comparable performance while having faster convergence rate and smaller parameter size than other aggregate functions. Readers can implement the self-attention mechanism for better performance.

## Conclusion

In this paper, we propose a simple but effective frame-level timeline modeling method for temporal graph representation learning, where the main contributions are made to the way that temporal neighborhoods are constructed and neighboring information is aggregated. Technically, we break down a temporal sequence of graph-structured data into individual frames, and model the evolution of successive frames to mine deeper into the dynamics of nodes and links. Experimental results demonstrate the effectiveness of FTM. Meanwhile, our experiments empirically reveal that even state-of-the-art GNNs have critical weakness in modeling temporal graphs; but FTM helps to derive generalizable knowledge during training and thus greatly improves both the robustness and the domain generality of baseline methods, especially when there are outliers/noise in the data (*cf.* Figure 4(a), Table 3), or the amount of data and computational resources are insufficient (*cf.* Table 5). The efficacy of FTM may provide insights that could facilitate the design of more advanced representation learning methods on temporal graphs.

## Acknowledgements

This paper was partially supported by Shenzhen Science & Technology Research Program (No: GXWD20201231165807007-20200814115301001) and NSFC (No: 62176008)

## References

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Cheng, X.; Zhu, Z.; Li, H.; Li, Y.; and Zou, Y. 2023. SSVMR: Saliency-based Self-training for Video-Music Retrieval. *CoRR*, abs/2302.09328.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

García-Durán, A.; Dumancic, S.; and Niepert, M. 2018. Learning Sequence Encoders for Temporal Knowledge Graph Completion. In *EMNLP*. Brussels, Belgium: ACL.

Graepel, T.; Candela, J. Q.; Borchert, T.; and Herbrich, R. 2010. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *Proceedings of the 27th International Conference on Machine Learning*, 13–20. Haifa, Israel: Omnipress.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*, 855–864. San Francisco, CA, USA: ACM.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 1025–1035. Long Beach, CA, USA.

He, X.; Pan, J.; Jin, O.; Xu, T.; Liu, B.; Xu, T.; Shi, Y.; Atallah, A.; Herbrich, R.; Bowers, S.; et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*. New York City, New York, USA: ACM.

Kipf, T. N.; and Welling, M. 2016a. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations*. Toulon, France: OpenReview.net.

Kipf, T. N.; and Welling, M. 2016b. Variational Graph Auto-Encoders. *CoRR*, abs/1611.07308.

Kumar, S.; Spezzano, F.; Subrahmanian, V.; and Faloutsos, C. 2016. Edge weight prediction in weighted signed networks. In *16th International Conference on Data Mining (ICDM)*, 221–230. Barcelona, Spain: IEEE Computer Society.

Kumar, S.; Zhang, X.; and Leskovec, J. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th International Conference on Knowledge Discovery & Data Mining*, 1269–1278. Anchorage, AK, USA: ACM.

Li, H.; Li, X.; Karimi, B.; Chen, J.; and Sun, M. 2022. Joint Learning of Object Graph and Relation Graph for Visual Question Answering. In *IEEE International Conference on Multimedia and Expo, ICME 2022, Taipei, Taiwan, July 18-22, 2022*, 1–6. IEEE.

Nguyen, G. H.; Lee, J. B.; Rossi, R. A.; Ahmed, N. K.; Koh, E.; and Kim, S. 2018. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference*, 969–976. Lyon, France: International World Wide Web Conferences Steering Committee.

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. New York, USA: AAAI.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th International Conference on Knowledge Discovery & Data Mining*, 701–710. New York, USA: ACM.

Rossi, E.; Chamberlain, B.; Frasca, F.; Eynard, D.; Monti, F.; and Bronstein, M. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. In *ICML 2020 Workshop on Graph Representation Learning*.

Singer, U.; Guy, I.; and Radinsky, K. 2019. Node embedding over temporal graphs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 4605–4612. Macao, China: ijcai.org.

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

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph attention networks. In *6th International Conference on Learning Representations*. Vancouver, BC, Canada: OpenReview.net.

Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; and Achan, K. 2020. Inductive representation learning on temporal graphs. In *8th International Conference on Learning Representations*. Addis Ababa, Ethiopia: OpenReview.net.

Zhu, Z.; Xu, W.; Cheng, X.; Song, T.; and Zou, Y. 2022. A Dynamic Graph Interactive Framework with Label-Semantic Injection for Spoken Language Understanding. *CoRR*, abs/2211.04023.