Kernelized Edge Attention: Addressing Semantic Attention Blurring in Temporal Graph Neural Networks

Govind Waghmare[1], Srini Rohan Gujulla Leel[1], Nikhil Tumbde*[1], Sumedh B G*[1], Sonia Gupta[1], Srikanta Bedathur[2]

[1]Mastercard
[2]Indian Institute of Technology, Delhi
{govind.waghmare, srinirohan.gujullaleel, nikhil.tumbde, sumedh.bg, sonia.gupta}@mastercard.com, srikanta@cse.iitd.ac.in

Abstract

Temporal Graph Neural Networks (TGNNs) aim to capture the evolving structure and timing of interactions in dynamic graphs. Although many models incorporate time through encodings or architectural design, they often compute attention over entangled node and edge representations, failing to reflect their distinct temporal behaviors. Node embeddings evolve slowly as they aggregate long-term structural context, while edge features reflect transient, timestamped interactions (e.g. messages, trades, or transactions). This mismatch results in semantic attention blurring, where attention weights cannot distinguish between slowly drifting node states and rapidly changing, information-rich edge interactions. As a result, models struggle to capture fine-grained temporal dependencies and provide limited transparency into how temporal relevance is computed. This paper introduces KEAT (Kernelized Edge Attention for Temporal Graphs), a novel attention formulation that modulates edge features using a family of continuous-time kernels, including Laplacian, RBF, and learnable MLP variant. KEAT preserves the distinct roles of nodes and edges, and integrates seamlessly with both Transformer-style (e.g., DyGFormer) and message-passing (e.g., TGN) architectures. It achieves up to 18% MRR improvement over the recent DyGFormer and 7% over TGN on link prediction tasks, enabling more accurate, interpretable and temporally aware message passing in TGNNs.

Introduction

Learning from time-evolving graph data is critical to applications such as recommendation, event forecasting, and fraud detection. While Graph Neural Networks (GNNs) have proven effective on static graphs (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017; Liao et al. 2019), adapting them to continuous-time dynamics poses fundamental challenges (Kazemi et al. 2020; Longa et al. 2023; Zheng, Yi, and Wei 2025). TGNNs address this by modeling both structural and temporal dependencies, often through specialized architectures like temporal walks (Wang et al. 2021b), sequence patching (Yu et al. 2023), and alternative message-passing (Luo and Li 2022; Cong et al. 2023), or via time encodings (Chen et al. 2025; Xu et al. 2020, 2019).

*These authors contributed equally.

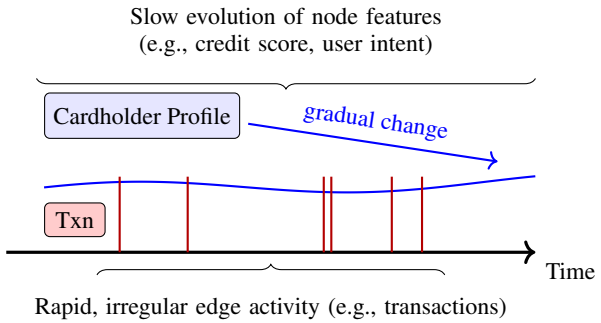

Figure 1: Temporal mismatch in dynamic graphs. While user profiles (nodes) change gradually (e.g., credit behavior), interactions like transactions (edges) vary rapidly and irregularly. Attention mechanisms in existing TGNNs often entangle these signals, leading to semantic attention blurring.

To model temporal relationships in graph, attention-based TGNNs have gained popularity, particularly those inspired by Transformer architectures (Wu, Fang, and Liao 2024; Yu et al. 2023; Rossi et al. 2020). These models extend self-attention to graphs by assigning learnable importance scores to neighboring nodes and edges. While effective, most existing approaches compute attention over entangled node and edge representations. They fail to explicitly modulate edge features based on complex temporal dynamics, overlooking the fact that nodes and edges often evolve at different temporal rates. See Figure 1 for an illustration of this mismatch.

For instance, in financial transaction graphs, cardholder profiles (nodes) evolve gradually, reflecting changes like creditworthiness or spending behavior. Whereas, transaction records (edges) vary rapidly and irregularly, especially in the presence of suspicious or high-frequency activity. Capturing these short-term edge patterns without overwhelming them with slowly drifting node context is crucial for temporal precision. In particular, these models do not directly encode how time should modulate attention weights or influence message aggregation. This leads to *semantic attention blurring*, where slowly evolving node embeddings, capturing long-term structural context, are mixed indiscriminately with sparse, transient edge features that carry rich, recent

information. As a result, attention mechanisms struggle to prioritize temporally and semantically relevant interactions, reducing both temporal fidelity and interpretability.

While prior works focus on improving time encodings to address these issues, such efforts may offer limited returns under current attention formulations, such as those implemented in `TransformerConv`[1], where attention scores are computed over the sum of node and edge projections. This formulation, or slight variations of it, is adopted in models such as TGN (Rossi et al. 2020), DyGFormer (Yu et al. 2023), and others (Xu et al. 2020), which similarly suffer from an inability to disentangle temporally distinct signals. As a result, such aggregation blurs temporal distinctions, making it difficult to separate recent, informative edge events from older or semantically unrelated interactions. This phenomenon is illustrated in Figure 2. In contrast, we shift the focus from time encoding design to the attention mechanism itself. By modulating only edge features, **comprising raw edge attributes and time encodings**, with continuous-time kernels, we introduce explicit temporal awareness into the attention computation and message aggregation. Crucially, our formulation is compatible with, and often enhances, existing time encoding schemes, making it broadly applicable and encoding-agnostic.

In this work, we propose a novel temporal attention formulation that directly addresses the mismatch in temporal dynamics between node and edge features in TGNNs. We call our framework **KEAT** *(**K**ernelized **E**dge **A**ttention for **T**emporal Graphs)*. KEAT applies a family of continuous-time kernels exclusively to edge features, thereby incorporating temporal sensitivity into attention without altering node semantics. The key idea is simple yet effective: edge-time features are modulated via kernel functions such as Laplacian, RBF, or a learnable MLP-based variant. This modulation is applied within the key and value projections of attention, allowing recent interactions to be emphasized while preserving the relatively stable, slow-evolving role of node embeddings. KEAT integrates seamlessly with Transformer-style and message-passing TGNNs and remains agnostic to specific time encoding schemes.

Existing models like DyGFormer (Yu et al. 2023) leverage temporal patches and time-interval encodings to capture evolving graph dynamics. However, they still rely on conventional attention formulations and inherit similar limitations. We show that integrating KEAT into DyGFormer and other TGNN architectures improves temporal precision, interpretability, and predictive performance. These gains validate the broad applicability and plug-and-play nature of KEAT across diverse temporal modeling settings. Our key contributions are summarized as follows:

- We formally identify the issue of *semantic attention blurring* in Transformer-based TGNNs, where attention scores over combined node and edge projections hinder temporal precision and interpretability.
- To resolve this, we introduce KEAT, a time-aware attention mechanism that applies continuous-time kernel modulation exclusively to edge features, preserving the distinct temporal roles of nodes and edges.
- KEAT is agnostic to time encoding schemes and compatible with a range of TGNN architectures. It requires minimal architectural changes and no additional supervision, making it practical for integration into existing systems.
- Through extensive experiments on dynamic graph benchmarks, we demonstrate that KEAT improves accuracy, enhances temporal fidelity, and produces interpretable attention patterns aligned with evolving edge dynamics.

[1]Documentation for `TransformerConv`: https://pytorch-geometric.readthedocs.io/en/stable/generated/torch_geometric.nn.conv.TransformerConv.html (Fey and Lenssen 2019)

Related Work

Temporal Graph Neural Networks. TGNNs extend GNNs to dynamic graphs by modeling both structure and time. Early methods such as TGN (Rossi et al. 2020), TGAT (Xu et al. 2020), and CAWN (Wang et al. 2021b) incorporate time using continuous-time encodings or time-difference embeddings. Other approaches adopt positional or sinusoidal encodings inspired by language models (Xu et al. 2019). Recent work like LeTE (Chen et al. 2025) introduces a learnable transformation framework for time encoding using nonlinear mappings such as Fourier and splines, aiming to unify and generalize existing encoding schemes. Despite these advances, most of these models assume that better time encodings alone will yield better performance. In contrast, we argue that time encoding improvements have limited effect if the attention formulation itself cannot separate the dynamics of node and edge features. Our work departs from this direction by keeping time encodings fixed and focusing instead on making attention computations explicitly time-aware through kernel-based modulation.

Attention Mechanisms in TGNNs. Transformer-inspired TGNNs such as TGAT (Xu et al. 2020), TGN (Rossi et al. 2020), and DyGFormer (Yu et al. 2023) use attention to weigh the influence of temporal neighbors during message passing. These models often follow the `TransformerConv` formulation, a standardized implementation from the PyTorch Geometric, computing the attention scores from the sum of projected node and edge features. This design simplifies computation but introduces semantic attention blurring, wherein temporally distinct messages are mixed. Some methods attempt to refine attention using structural bias (e.g., CAWN's walk-based decay (Wang et al. 2021b)), and a few incorporate time signals into attention projections through time encodings (Xu et al. 2020), but explicit time-aware modulation, such as continuous-time scaling of attention inputs, remains underexplored. Our kernel-based attention addresses this gap by decoupling edge dynamics from node semantics via time-dependent scaling, providing a lightweight yet effective way to inject temporal information into the attention mechanism.

Modulated Attention Beyond TGNNs. Time-aware attention has been studied in other domains such as NLP, time series and credit risk modeling. In language models, temporal self-attention augments Transformer-based architec-

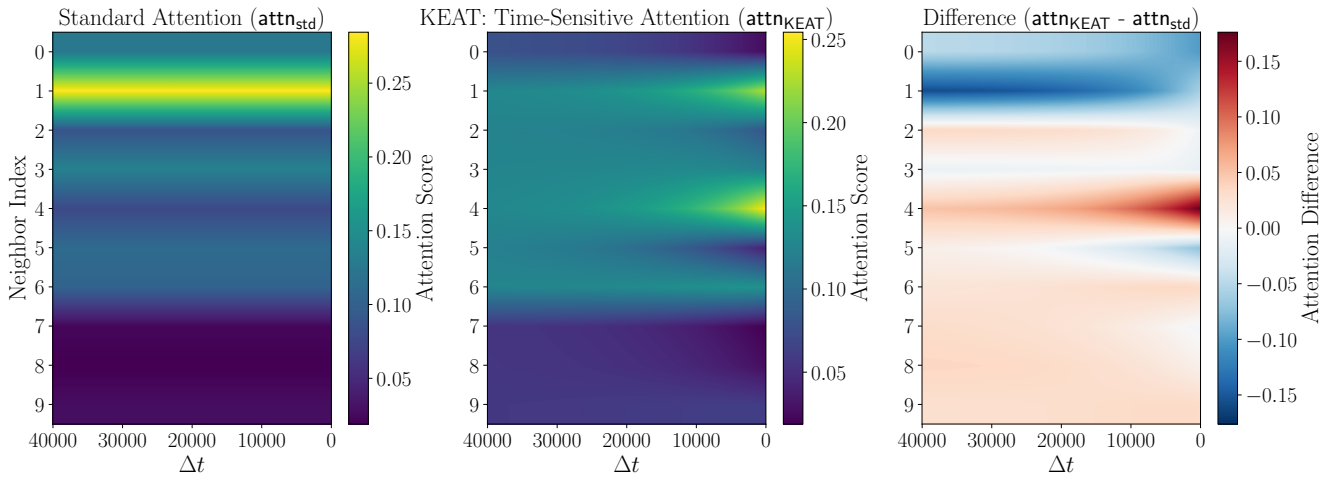

Figure 2: Temporal attention on `tgbl-wiki` illustrating **semantic attention blurring**. We plot attention over ten neighbors across relative time Δt. (Left) Standard attention assigns uniform attention values across time for each neighbor, leading to semantic blurring. This shows invariance of attention to time encodings or Δt. (Middle) KEAT introduces time-aware modulation, emphasizing recent (smaller Δt value) and contextually relevant edges. (Right) The difference highlights how KEAT corrects the blurred focus of standard attention. See experimental section for details.

tures with timestamped text (Rosin and Radinsky 2022). Powerformer (Hegazy, Mahoney, and Erichson 2025) and (Niu et al. 2024) treat attention as the primary representation for time series. In credit risk assessment, DGNN-SR (Yuan et al. 2025) integrates multiple time perspectives directly into the attention module. In static graphs, Gradformer (Liu et al. 2024) applies exponential decay to graph transformer. Works like (Press, Smith, and Lewis 2022; Chi et al. 2022) study attention in NLP for extrapolation. While these works make attention more expressive, our approach introduces a modular, temporally kernel-based attention mechanism that applies time-conditioned scaling exclusively to edge representations, thus decoupling node and edge dynamics. This formulation allows time to play a more explicit role in the attention computation and helps reduce semantic attention blurring in a more interpretable way.

Temporal Information Modeling. Several TGNNs incorporate temporal information through auxiliary mechanisms rather than within attention. JODIE (Kumar, Zhang, and Leskovec 2019) applies time-conditioned projections to evolve user embeddings, and DyRep (Trivedi et al. 2019) models temporal interactions via point processes intensities. TGN (Rossi et al. 2020) maintains time-aware memory modules, while CAWN (Wang et al. 2021b) integrates temporal signals during neighbor sampling. In contrast, our approach embeds temporal characteristics directly into the attention computation by modulating edge features with kernel-weighted time encodings. This enables interpretable, differentiable, and architecture-agnostic modulation.

Modular and Extendable TGNN Designs. Recent architectures such as DyGFormer (Yu et al. 2023) adopt patch-based attention blocks to process temporally localized subgraphs. While these designs improve scalability and temporal resolution, they still compute attention using mixed node-edge projections. Our kernel-based formulation can be seamlessly integrated into such architectures, enhancing their temporal fidelity without disrupting their tokenization, encoding, or aggregation pipelines. This highlights the flexibility of our method as a plug-and-play module for improving temporal sensitivity in a wide range of TGNNs.

Methodology

Problem Setting and Notation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, T)$ be a dynamic graph, where $\mathcal{V}$ is the set of nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of timestamped edges, and T is the set of event times. Each interaction is a tuple $(i, j, \mathbf{e}_{ij}, t_{ij})$, with $i, j \in \mathcal{V}$, $\mathbf{e}_{ij} \in \mathbb{R}^{d_e}$ representing raw edge features, and $t_{ij} \in \mathbb{R}_+$ denoting the interaction time. TGNNs typically capture temporal context by computing the time difference $\Delta t = t_i - t_{ij}$ between the current query time t_i and a past interaction time t_{ij}. This Δt is encoded using a time encoding function $\phi : \mathbb{R}_+ \to \mathbb{R}^{d_t}$, producing $\phi(\Delta t)$. The temporal and raw edge features are then concatenated to form: $\bar{\mathbf{e}}_{ij} = [\mathbf{e}_{ij} \,\|\, \phi(\Delta t)] \in \mathbb{R}^{d_e + d_t}$. Given the temporal neighborhood $\mathcal{N}(i)$ of node i, the objective is to compute updated node embeddings $\mathbf{h}'_i \in \mathbb{R}^d$ that are both structurally informative and temporally sensitive.

Transformer-style Temporal Message Passing

Transformer-based attention mechanisms are widely adopted in TGNNs for aggregating messages from temporal neighbors. A representative implementation, `TransformerConv` (Fey and Lenssen 2019), computes attention from a source node i to a target node j as:

$$\alpha_{ij} = \text{softmax}_j \left(\frac{(\mathbf{W}_q \mathbf{h}_i)^\top (\mathbf{W}_k \mathbf{h}_j + \mathbf{W}_e \bar{\mathbf{e}}_{ij})}{\sqrt{d_k}} \right), \quad (1)$$

where $\mathbf{h}_i, \mathbf{h}_j \in \mathbb{R}^d$ are node embeddings, $\bar{\mathbf{e}}_{ij} \in \mathbb{R}^{d_e+d_t}$ is the concatenated edge-time feature, and $\mathbf{W}_q, \mathbf{W}_k \in \mathbb{R}^{d' \times d}$, $\mathbf{W}_e \in \mathbb{R}^{d' \times (d_e+d_t)}$ are learnable projection matrices. The attention is scaled by $d_k = d'$ as in standard Transformers (Vaswani et al. 2017). The final embedding of node i is:

$$\mathbf{h}'_i = \mathbf{W}_{\text{self}}\mathbf{h}_i + \sum_{j \in \mathcal{N}(i)} \alpha_{ij} \cdot (\mathbf{W}_v \mathbf{h}_j + \mathbf{W}'_e \bar{\mathbf{e}}_{ij}), \quad (2)$$

where $\mathbf{W}_v \in \mathbb{R}^{d' \times d}$, $\mathbf{W}'_e \in \mathbb{R}^{d' \times (d_e+d_t)}$, and $\mathbf{W}_{\text{self}} \in \mathbb{R}^{d' \times d}$ are learnable weight matrices. This formulation, or variants of it, is used in several TGNNs such as TGN (Rossi et al. 2020), DyGFormer (Yu et al. 2023), and others.

Time Encodings in Temporal Graphs

Popular time encoding functions $\phi(\Delta t)$ include:

- **Fixed encodings**: GraphMixer (Cong et al. 2023) employs non-trainable sinusoidal time encodings, similar to the positional encodings used in standard Transformers.
- **Relative and learnable encodings**: TGN (Rossi et al. 2020) uses learnable sinusoidal embeddings for relative time, similar to TGAT (Xu et al. 2020), applied in both message computation and memory updates. LeTE (Chen et al. 2025) learns flexible time encoding functions using splines and Fourier-based transformations.

Limitations of Time Encodings. Time encodings $\phi(\Delta t)$ in TGNNs are often concatenated with node or edge features (see Eq.1 and 2) but do not explicitly influence the attention weights. As a result, they weakly capture temporal patterns like frequency or semantic relevance (Figure 2), limiting the model's ability to differentiate diverse temporal interactions.

A common approach is to encode time differences Δt using sinusoidal functions such as $\phi(\Delta t) = \cos(\omega \Delta t)$ or $\sin(\omega \Delta t)$. As shown in Appendix B, these simple periodic encodings capture only a limited subset of the moment spectrum of the inter-arrival distribution $p(\Delta t)$:

$$\mathbb{E}_{\Delta t}[\cos(\omega \Delta t)] = \sum_{n=0}^{\infty} \frac{(-1)^n \omega^{2n}}{(2n)!} \mathbb{E}[(\Delta t)^{2n}]. \quad (3)$$

While some prior works use both sine and cosine terms to capture more statistical moments (Chen et al. 2025), these encodings remain passive and fail to directly shape attention weights. As a result, they often miss temporal asymmetries important for aggregation. This limitation is exacerbated in real-world temporal graphs, where the distribution $p(\Delta t)$ often shifts across training and validation sets (refer Appendix C). In many graphs, early interactions are typically sparse, while later ones are denser, causing skewed distributions and shifts in both the mean and higher-order moments.

KEAT addresses this challenge by introducing kernel-modulated encodings of the form $f(\Delta t) = \psi(\Delta t) \cdot \bar{\mathbf{e}}_{ij}$, where $\psi(\Delta t)$ is a temporal kernel. This formulation expands into a full polynomial over Δt and becomes sensitive to all moments of $p(\Delta t)$ (see Appendix B.2). Importantly, the modulated signal $f(\Delta t)$ is used to directly influence attention computation, ensuring that time information shapes both the neighbor weighting and the final embedding output.

Design Criteria for Temporal Kernels

To enable time-aware attention, we modulate edge features using a continuous temporal kernel $\psi(\Delta t)$ that scales contributions based on the elapsed time Δt. The kernel should satisfy several desirable properties: (i) it should decay monotonically with increasing Δt to prioritize recent events; (ii) it must be bounded within $[0, 1]$ to ensure numerical stability; (iii) it should be continuous to support smooth optimization; and (iv) it should be flexible. Parameterized MLPs can model richer, potentially non-monotonic (vs. Laplacian and RBF) temporal patterns when interpretability is less critical.

Family of Temporal Kernels

We instantiate the temporal modulation function $\psi(\Delta t)$ using the following kernel families:

- **Laplacian kernel:** $\psi(\Delta t) = \exp\left(-\frac{\Delta t}{\sigma}\right)$
- **RBF kernel:** $\psi(\Delta t) = \exp\left(-\frac{\Delta t^2}{\sigma^2}\right)$
- **Learned kernel (MLP):** $\psi(\Delta t) = \text{MLP}(\Delta t)$

Here, σ denotes the standard deviation of inter-event time differences computed from the training set. The Laplacian and RBF kernels are parameter-free and introduce interpretable temporal biases by smoothly attenuating the influence of older interactions. In contrast, the MLP-based kernel offers greater flexibility, enabling the model to learn complex, data-driven modulation patterns. All variants are differentiable and integrate seamlessly into end-to-end training.

Method: KEAT – Kernelized Attention over Time

We introduce KEAT, a mechanism that integrates temporal modulation into attention by applying a kernel $\psi(\Delta t)$ to the edge-time feature $\bar{\mathbf{e}}_{ij}$ *before* it influences attention or embedding updates. In Transformer-style architectures (Eq. 1) attention scores are computed from projected node embeddings and edge features. Node embeddings evolve slowly and capture long-term structural information, while edge features encode temporally localized, interaction-specific signals. Thus, modulating the edge component using a continuous-time kernel $\psi(\Delta t)$ provides a principled way to inject temporal sensitivity. KEAT requires only a lightweight modification to edge projections, introducing negligible overhead during training or inference.

KEAT explicitly biases or modulates the attention toward different temporal interactions by scaling edge-time features via $\psi(\Delta t)$. The modified attention score becomes:

$$\alpha_{ij} = \text{softmax}_j \left(\frac{(\mathbf{W}_q \mathbf{h}_i)^\top \left(\mathbf{W}_k \mathbf{h}_j + \mathbf{W}_e \left[f(\Delta t)\right]\right)}{\sqrt{d_k}} \right), \quad (4)$$

where, $f(\Delta t) = \psi(\Delta t) \cdot \bar{\mathbf{e}}_{ij}$ and the corresponding node embedding update formula is:

$$\mathbf{h}'_i = \mathbf{W}_{\text{self}}\mathbf{h}_i + \sum_{j \in \mathcal{N}(i)} \alpha_{ij} \cdot (\mathbf{W}_v \mathbf{h}_j + \mathbf{W}'_e \left[f(\Delta t)\right]) \quad (5)$$

This design ensures that temporally closer interactions contribute significantly to attention computation and feature aggregation, while preserving the original Transformer architecture. Notably, node features remain unaltered, allowing a clean decoupling of structural and temporal contributions.

In DyGFormer (Yu et al. 2023), interaction histories are divided into patches, each summarizing a local neighborhood around source or destination nodes. To integrate KEAT, we modulate only the edge features within each patch using a representative patch timestamp (e.g., mean edge time). Specifically, the query is scaled by $\exp(t_{\text{patch}})$ and the key by $\exp(-t_{\text{patch}})$, resulting in attention scores of the form $\exp(t_{\text{query}} - t_{\text{key}})$. This introduces a directional temporal bias, enabling time-sensitive attention without altering DyGFormer's architecture. See Appendix D for details.

Advantages of Kernel-Based Attention

KEAT offers several desirable properties: • **Temporal precision**: (Laplacian and RBF kernels) emphasizes recent interactions by downweighting distant ones; • **Semantic disentanglement**: isolates evolving edge patterns while preserving node semantics; • **Architectural generality**: seamlessly integrates into diverse TGNNs (e.g., TGN, TGAT, DyGFormer); • **Simplicity and efficiency**: lightweight, differentiable, and requires no additional preprocessing.

Beyond demonstrating empirical utility, we provide theoretical justification for using temporal kernels in attention. In particular, we show that kernel-based modulation suppresses contributions from higher-order moments of the time distribution, improving robustness to distribution shifts across training and inference. The following theorem formally captures this effect for the Laplacian or RBF kernel $\psi(t)$:

Theorem 1 *Let $p(t)$ be a probability density function supported on $[0, \infty)$ such that $\mathbb{E}[t^n] < \infty$ and $\mathbb{E}[\psi(t)t^n] < \infty$ for all $n \geq 0$, where $\psi(t)$ is a non-negative, monotonically decreasing kernel function. Let $\phi(t) = \sum_{n=0}^{\infty} c_n t^n$ be an analytic time encoding function with coefficients $\{c_n\}$. Define the kernel-to-base ratio for each moment order n as:*

$$R_n = \frac{\mathbb{E}[\psi(t)t^n]}{\mathbb{E}[t^n]}, \quad \textit{with } R_0 = \mathbb{E}[\psi(t)].$$

Then, the sequence $\{R_n\}_{n=0}^{\infty}$ is strictly decreasing and converges to zero: $\lim_{n\to\infty} R_n = 0$. As a result, the expected kernel-weighted time encoding $\mathbb{E}[\psi(t) \cdot \phi(t)]$ becomes increasingly dominated by lower-order terms[2] .

The proof is provided in Appendix E. In addition, we establish a complementary result: incorporating temporal kernels into attention reduces the variance of the attention logits. This promotes more stable neighbor weighting, particularly under temporally imbalanced or skewed interaction histories. This variance-reduction effect further underscores the robustness and interpretability benefits of kernel-based attention. See Appendix E for a complete derivation.

Experiments

We empirically validate the effectiveness of our proposed temporal kernel modulation framework across multiple dynamic graph learning tasks. Our evaluation focuses on three key aspects: (1) compatibility and integration with existing attention-based TGNN architectures, (2) the ability to capture fine-grained temporal dependencies through moment-aware attention modulation, and (3) performance gains under varying temporal encodings and graph dynamics.

Experiments are primarily conducted on the Temporal Graph Benchmark (TGB) (Huang et al. 2023), which provides large-scale, realistic datasets with diverse temporal characteristics. To further evaluate the generality and robustness of our kernelized attention mechanism, we extend the benchmark suite by incorporating additional datasets from the JODIE (Kumar, Zhang, and Leskovec 2019) framework for dynamic link prediction, as well as the DGraphFin dataset (Huang et al. 2022) for dynamic node classification. This extended evaluation suite enables us to test KEAT across a wider range of temporal graph settings, including financial graphs, and user-item interactions.

KEAT is implemented as a lightweight, modular enhancement to existing Transformer-based TGNNs without requiring architectural changes. We report results using standardized splits and evaluation metrics to isolate the impact of temporal attention modulation. Unless stated otherwise, we use the Laplacian kernel by default. A detailed description of datasets along with statistics is provided in Appendix C.

Task Definitions

We consider two standard tasks on temporal graphs $\mathcal{G}$. In **dynamic link prediction**, the objective is to estimate the probability of a link between two nodes at a given time, framed as a ranking problem with Mean Reciprocal Rank (MRR) as the metric. For JODIE datasets, we measure performance using AUC and Average Precision (AP). In **dynamic node classification**, the goal is to predict a node's label at a specific time based on its interaction history and temporal context, capturing evolving patterns such as user preferences or risk. Performance is evaluated using NDCG@10 for `tgbn` dasets and using AUC for the DGraphFin dataset.

Baselines Our proposed kernel-based attention mechanism is designed to enhance Transformer-style TGNNs. To evaluate its impact, we integrate it into two widely used attention-based architectures, TGN and DyGFormer, serving as our primary backbones. The resulting models, KEAT-TGN and KEAT-DyGFormer, are compared against their vanilla counterparts to isolate the effect of our contribution. We also report results from several strong baselines, including DyRep (Trivedi et al. 2019), TNCN (Zhang et al. 2024), and CTAN (Gravina et al. 2024), as well as memory-based methods $\text{EdgeBank}_{\text{tw}}$ and EdgeBank_{∞} (Poursafaei et al. 2022), included in the TGB leaderboard. While some of these baselines do not explicitly model temporal attention, they provide valuable context for evaluating our method. Other baselines, such as TCL (Wang et al. 2021a), NAT (Luo and Li 2022), CAWN (Wang et al. 2021b), and GraphMixer (Cong et al. 2023) underperform compared to DyGFormer and/or TGB and are evaluated on a narrower set of datasets, as shown in the TGB leaderboard[3]. Appendix F summarizes baseline selection criteria, ranking on the link prediction task, their scalability and dataset coverage.

[2]This formulation naturally extends to expectations of the form $\mathbb{E}[\psi(t) \cdot \bar{\mathbf{e}}_{ij}]$, as used in Equations 4 and 5.

[3]https://tgb.complexdatalab.com/docs/leader_linkprop/

Method	Split	tgbl-wiki	tgbl-review	tgbl-coin	tgbl-comment
EdgeBank$_{tw}$	Val	0.600	0.024	0.492	0.124
	Test	0.571	0.025	0.580	0.149
EdgeBank$_{\infty}$	Val	0.527	0.023	0.315	0.109
	Test	0.495	0.023	0.359	0.129
DyRep	Val	0.072 ± 0.009	0.216 ± 0.031	0.512 ± 0.014	0.291 ± 0.028
	Test	0.050 ± 0.017	0.220 ± 0.030	0.452 ± 0.046	0.289 ± 0.033
TNCN	Val	0.741 ± 0.001	0.325 ± 0.003	0.740 ± 0.002	0.643 ± 0.003
	Test	0.718 ± 0.001	0.377 ± 0.010	0.762 ± 0.004	0.697 ± 0.006
CTAN	Val	—	—	—	—
	Test	0.668 ± 0.007	0.405 ± 0.004	0.748 ± 0.004	0.671 ± 0.067
TGN	Val	0.435 ± 0.069	0.313 ± 0.012	0.607 ± 0.014	0.356 ± 0.019
	Test	0.396 ± 0.060	0.349 ± 0.020	0.586 ± 0.037	0.379 ± 0.021
KEAT-TGN	Val	0.515 ± 0.053	0.324 ± 0.004	0.624 ± 0.012	0.368 ± 0.019
	Test	0.474 ± 0.031	0.380 ± 0.007	0.632 ± 0.023	0.400 ± 0.020
Improvement	Test	+7.76%	+3.10%	+4.62%	+2.10%
DyGFormer	Val	0.816 ± 0.005	0.219 ± 0.017	0.730 ± 0.002	0.613 ± 0.003
	Test	0.798 ± 0.004	0.224 ± 0.015	0.752 ± 0.004	0.670 ± 0.001
KEAT-DyGFormer	Val	0.829 ± 0.003	0.335 ± 0.020	—	—
	Test	0.815 ± 0.005	0.412 ± 0.012	0.803 ± 0.003	0.776 ± 0.001
Improvement	Test	+1.69%	+18.80%	+5.10%	+10.60%

Table 1: Link prediction results (MRR) on the TGBL benchmark datasets. We report both validation and test MRR for each method. All baseline results are taken from TGB leaderboard (Huang et al. 2023). **KEAT** denotes our kernel-based attention mechanism integrated into TGN and DyGFormer backbones. The top three test results for each dataset are highlighted using green (first), yellow (second), and gray (third). Entries marked with — indicate that baseline results are not available.

Implementation Details. We implement our model using `PyTorch` and `PyTorch Geometric`, and the official TGB framework (Huang et al. 2023). All experiments follow the default hyperparameters provided by the TGB benchmark to ensure fair and reproducible comparisons. Each experiment is run five times with different random seeds. The experimentation settings and hyperparameters for JODIE datasets and `DGraphFin` are provided in Appendix G.

Link Prediction Results. Table 1 presents the MRR for link prediction on the `tgbl` benchmark datasets. Our method, Kernel-based Attention (KEAT), is applied to two popular TGNN backbones: TGN and DyGFormer. Across all datasets, KEAT variants demonstrate consistent improvements over their respective base models. Notably, KEAT-DyGFormer achieves the best test performance on all four datasets, `tgbl-wiki`, `tgbl-review`, `tgbl-coin`, and `tgbl-comment`, demonstrating the generality and effectiveness of our kernelized attention mechanism. KEAT-TGN also shows gains over TGN on all datasets, with improvements ranging from +2.1% to +7.8% in test MRR. These results affirm that augmenting TGNNs with time-sensitive, structure-aware attention yields measurable and consistent performance benefits. KEAT also demonstrates consistently stronger performance on the JODIE datasets across multiple evaluation metrics, highlighting its effectiveness in handling time-evolving interactions. Details are in Appendix G.

Effect of Kernel Choice. We evaluate the effect of different kernel choices within the KEAT framework through an ablation study using three variants: *Laplacian*, *RBF*, *MLP*, with a baseline (no kernel modulation). All experiments are conducted using the TGN backbone on three benchmark datasets, `tgbl-wiki`, `tgbl-review`, and `tgbl-coin` (see Table 2). The results consistently indicate that temporal kernels enhance performance. For instance, on `tgbl-wiki`, the Laplacian kernel yields an MRR of 0.474±0.031, marking a significant improvement of +7.8% over the no-kernel baseline (0.396±0.060). Similarly, the *RBF kernel* achieves the highest MRR on `tgbl-coin` at 0.636±0.021, surpassing the baseline by +5.0%. These gains underscore the benefit of temporally modulating edge features when computing attention.

Laplacian and RBF perform most consistently, benefiting from positive-definite, distance-based temporal modulation. MLP is more expressive but has higher variance, as it lacks built-in inductive bias and depends heavily on available data. Overall, these results highlight the generalization advantage of structured kernels in time-sensitive tasks.

Choosing the Right Kernel. Laplacian prioritizes recency, ideal for short-term temporal dynamics. In contrast, the RBF kernel provides a smoother, more balanced decay that captures both recent and mid-range dependencies. MLP is suited for high-data regimes with complex, non-

Method	Kernel	tgbl-wiki		tgbl-review		tgbl-coin	
		Val	Test	Val	Test	Val	Test
KEAT-TGN	Laplacian	0.515 ± 0.053	**0.474** ± 0.031	**0.324** ± 0.004	**0.378** ± 0.007	**0.624** ± 0.012	0.632 ± 0.023
	RBF	0.512 ± 0.027	0.437 ± 0.035	0.322 ± 0.008	0.374 ± 0.011	0.623 ± 0.015	**0.636** ± 0.021
	MLP	**0.524** ± 0.017	0.456 ± 0.027	**0.324** ± 0.007	0.372 ± 0.008	0.620 ± 0.022	0.617 ± 0.024
TGN	w/o Kernel	0.435 ± 0.069	0.396 ± 0.060	0.313 ± 0.012	0.349 ± 0.020	0.607 ± 0.014	0.586 ± 0.037

Table 2: MRR results on three TGBL datasets using different kernel types. Each cell reports mean ± standard deviation for validation and test sets. Best results in each column (based on mean) are shown in bold.

monotonic temporal patterns but may underperform when supervision is limited. Overall, Laplacian and RBF offer strong, low-variance performance across settings. Unless otherwise specified, we use Laplacian as the default kernel.

Node Classification Results. KEAT consistently improves node classification performance when integrated into TGN across diverse `tgbn` datasets (refer Appendix G). On `tgbn-trade` and `tgbn-genre`, KEAT-TGN achieves substantial gains of +5.30% and +5.29% in test NDCG@10, respectively. These improvements highlight KEAT's ability to inject temporal structure into attention mechanism, aligning node representations more closely with their evolving interaction context. Even on challenging datasets like `tgbn-token`, KEAT improves performance by +1.1%, indicating robustness. The slight but consistent gain on `tgbn-reddit` suggests that KEAT complements existing signals without degrading strong baselines. These findings reinforce KEAT's effectiveness in stabilizing node-level predictions and improving temporal generalization. Refer Appendix G for results on `DGraphFin`.

Impact of KEAT Across Time Encoding Strategies. We evaluate the robustness of KEAT with various time encoding methods, including `GraphMixer` (non-learnable), `TGN/TGAT`(learnable), and `LeTE`(learnable, Fourier and Spline-based), on three datasets. Results in Appendix G (on `tgbl-wiki`, `tgbl-review` and `tgbl-coin`) demonstrate that KEAT consistently improves test MRR, regardless of the encoding strategy. For instance, on `tgbl-wiki`, KEAT improves test MRR by +7.8% with `TGN/TGAT` and +7.3% with `LeTE`. Even when the raw encodings are strong (e.g., `GraphMixer` on `tgbl-review`), KEAT improves or maintains performance. Importantly, we observe reduced variance across runs when KEAT is used, suggesting it helps regularize attention with structured inductive bias. These results indicate that KEAT is orthogonal to the choice of time encoding and acts as a plug-and-play enhancement for diverse temporal message-passing models.

Attention plots in Appendix G highlights how KEAT enables edge-modulated, time-sensitive attention over neighbors. Unlike standard attention, which yields flat, temporally agnostic weights, KEAT dynamically adjusts attention based on both edge timing and feature context. This allows attention to vary meaningfully with Δt, yielding sharper, context-aware focus that mitigates semantic attention blurring and enhances temporal discrimination. Furthermore, attention weight heatmaps over time (refer to Appendix G) illustrate the interpretability of KEAT: they reveal how attention shifts gradually as edge timestamps grow older, offering insights into how the model prioritizes recent and semantically relevant interactions.

Computational Complexity. With Laplacian or RBF kernels, KEAT incurs no additional computational overhead and retains the same time and space complexity as its underlying backbone architectures, TGN and DyGFormer. MLP kernel operates on scalar time gaps with a few parameters. Under standard TGBL settings, we observe similar inference complexity of MLP kernel compared to other kernels.

Extended results. For completeness, we provide detailed dataset statistics, extended ablations (e.g., node vs. edge modulation, kernel width sensitivity), results on additional datasets (including JODIE and DGraphFin), full experimental settings, and theoretical proofs in Appendix. We encourage readers to refer to these sections for further insights. Together, these empirical results establish KEAT as a general-purpose enhancement for TGNNs. By embedding temporal structure directly into the attention computation, KEAT improves both predictive performance and training stability across datasets, time encoding schemes, and kernel types.

Limitations of KEAT. While KEAT demonstrates strong performance across datasets, it may be less effective in scenarios when edge activity is extremely sparse or highly clustered in time, which can limit the informativeness of temporal modulation. Addressing such challenging regimes remains an important and promising direction for future work.

Conclusion

We presented KEAT, a kernel-based attention framework that addresses semantic attention blurring in TGNNs. By introducing temporal kernels into the attention mechanism, KEAT preserves fine-grained distinctions between recent and older edge interactions, enabling more precise temporal reasoning. The method is lightweight, model-agnostic, and compatible with diverse time encoding strategies. Empirical results on diverse datasets show consistent performance gains across multiple tasks and model backbones such as TGN and DyGFormer. Extensive ablations further confirm the impact of kernel choice, edge-level modulation, and reduced attention variance. Overall, KEAT offers a simple yet effective solution to improve both the accuracy and generalization of TGNNs with minimal overhead.

Acknowledgments

Srikanta Bedathur acknowledges his DS Chair Professor of AI fellowship and the funding support from Mastercard AI Garage for this work.

References

Chen, X.; Tang, Y.; Xu, J.; Zhang, J.; Zhang, S.; Peng, S.; Zheng, X.; and Xiong, Y. 2025. Rethinking Time Encoding via Learnable Transformation Functions. In *International Conference on Machine Learning*.

Chi, T.-C.; Fan, T.-H.; Ramadge, P. J.; and Rudnicky, A. I. 2022. KERPLE: Kernelized Relative Positional Embedding for Length Extrapolation. In *International Conference on Neural Information Processing Systems*.

Cong, W.; Zhang, S.; Kang, J.; Yuan, B.; Wu, H.; Zhou, X.; Tong, H.; and Mahdavi, M. 2023. Do We Really Need Complicated Model Architectures For Temporal Networks? In *International Conference on Learning Representations*.

Fey, M.; and Lenssen, J. E. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *International Conference on Learning Representations Workshop on Representation Learning on Graphs and Manifolds*.

Gravina, A.; Lovisotto, G.; Gallicchio, C.; Bacciu, D.; and Grohnfeldt, C. 2024. Long Range Propagation on Continuous-Time Dynamic Graphs. *International Conference on Machine Learning*.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *International Conference on Neural Information Processing Systems*.

Hegazy, K.; Mahoney, M.; and Erichson, N. B. 2025. Powerformer: A Transformer with Weighted Causal Attention for Time-series Forecasting.

Huang, S.; Poursafaei, F.; Danovitch, J.; Fey, M.; Hu, W.; Rossi, E.; Leskovec, J.; Bronstein, M.; Rabusseau, G.; and Rabbany, R. 2023. Temporal Graph Benchmark for Machine Learning on Temporal Graphs. *Advances in Neural Information Processing Systems*.

Huang, X.; Yang, Y.; Wang, Y.; Wang, C.; Zhang, Z.; Xu, J.; Chen, L.; and Vazirgiannis, M. 2022. DGraph: A Large-Scale Financial Dataset for Graph Anomaly Detection. *Advances in Neural Information Processing Systems*.

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

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

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 and data mining*.

Liao, R.; Zhao, Z.; Urtasun, R.; and Zemel, R. 2019. LanczosNet: Multi-Scale Deep Graph Convolutional Networks. In *International Conference on Learning Representations*.

Liu, C.; Yao, Z.; Zhan, Y.; Ma, X.; Pan, S.; and Hu, W. 2024. Gradformer: graph transformer with exponential decay. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*.

Longa, A.; Lachi, V.; Santin, G.; Bianchini, M.; Lepri, B.; Lio, P.; Scarselli, F.; and Passerini, A. 2023. Graph neural networks for temporal graphs: State of the art, open challenges, and opportunities. *arXiv preprint arXiv:2302.01018*.

Luo, Y.; and Li, P. 2022. Neighborhood-aware Scalable Temporal Network Representation Learning. *Learning on Graphs Conference*.

Niu, P.; Zhou, T.; Wang, X.; Sun, L.; and Jin, R. 2024. Attention as Robust Representation for Time Series Forecasting.

Poursafaei, F.; Huang, S.; Pelrine, K.; and Rabbany, R. 2022. Towards better evaluation for dynamic link prediction. *Advances in Neural Information Processing Systems*.

Press, O.; Smith, N.; and Lewis, M. 2022. Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation. In *International Conference on Learning Representations*.

Rosin, G. D.; and Radinsky, K. 2022. Temporal attention for language models. *The Nations of the Americas Chapter of the Association for Computational Linguistics*.

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

Trivedi, R.; Farajtabar, M.; Biswal, P.; and Zha, H. 2019. DyRep: Learning Representations over Dynamic Graphs. In *International Conference on Learning Representations*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc.

Wang, L.; Chang, X.; Li, S.; Chu, Y.; Li, H.; Zhang, W.; He, X.; Song, L.; Zhou, J.; and Yang, H. 2021a. TCL: Transformer-based Dynamic Graph Modelling via Contrastive Learning. *Computing Research Repository*.

Wang, Y.; Chang, Y.-Y.; Liu, Y.; Leskovec, J.; and Li, P. 2021b. Inductive Representation Learning in Temporal Networks via Causal Anonymous Walks. In *International Conference on Learning Representations*.

Wu, Y.; Fang, Y.; and Liao, L. 2024. On the Feasibility of Simple Transformer for Dynamic Graph Modeling. *WWW*.

Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; and Achan, K. 2019. Self-attention with functional time representation learning. *Advances in neural information processing systems*.

Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; and Achan, K. 2020. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations*.

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*.

Yuan, Q.; Liu, Y.; Tang, Y.; Chen, X.; Zheng, X.; He, Q.; and Ao, X. 2025. Dynamic Graph Learning with Static Relations for Credit Risk Assessment. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zhang, X.; Wang, Y.; Wang, X.; and Zhang, M. 2024. Efficient Neural Common Neighbor for Temporal Graph Link Prediction. *arXiv preprint arXiv:2406.07926*.

Zheng, Y.; Yi, L.; and Wei, Z. 2025. A survey of dynamic graph neural networks. *Frontiers of Computer Science*.