CTDG-THOR: QUERY-SIDE SECOND-ORDER TEMPORAL HISTORY ROUTING FOR CONTINUOUS-TIME DYNAMIC GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Events in a continuous-time dynamic graph (CTDG) arrive as a stream of timestamps and messages. Most CTDG encoders build a node representation from that node's own events, so a neighbor's trajectory reaches a link score only when the model routes it across nodes. We present **CTDG-THOR**, which isolates candidate-independent routing of second-order temporal history from the query side. The model combines a TGN-style memory, history-conditioned attention, and pair-level aggregation of first-hop contexts for the query's neighbors. It expands second-order context only for the query, so the negative candidate set does not multiply path computation. We formalize an incident-only endpoint and a one-hop endpoint, and prove a witness separation: exponentially many event streams that the incident-only endpoint cannot distinguish become separable once second-order history is routed. Under matched seeds and a shared backbone, CTDG-THOR improves test MRR by $+0.0653$ on `tgbl-wiki` and $+0.0368$ on `tgbl-enron`. Fixed-checkpoint interventions also indicate that the trained predictor uses this route. Under the fixed-negative TGB-v1 protocol, it reaches test MRR 0.6416 ± 0.0123, 0.2478 ± 0.0255, and 0.1223 ± 0.0084 on these three datasets. The second-order route adds one scalar and a mean aggregation over the one-hop control and keeps peak memory below 90 MB. The comparisons show that where cross-node history is routed affects prediction in memory-based CTDGs.

1 INTRODUCTION

A continuous-time dynamic graph (CTDG) records interactions (u, v, t, m) at arbitrary times. It underlies temporal link prediction, recommendation, and anomaly detection, all of which depend on representations learned from the event stream.

Most CTDG models follow one of two designs. Memory-based models such as TGN Rossi et al. (2020) update a state for each node as events arrive. Attention-based models such as TGAT Xu et al. (2020) and DyGFormer Yu et al. (2023) encode temporal neighborhoods without a persistent memory. Both build a node representation mainly from that node's own interactions. However, the correct pair score also depends on how trajectories flow across nodes: a neighbor's history can determine the score even when it never appears in the query's own events. Current models leave this routing choice implicit.

We therefore study where cross-node history should enter a CTDG pipeline without expanding every candidate. At the incident-only endpoint, a node representation depends only on its own events and cannot contain trajectories that stay outside its history. Capacity cannot close this gap: exponentially many event streams map to the same representation, and adding width, depth, or attention heads leaves the missing signal missing. We use this endpoint as a controlled reference from which to study where second-order history belongs and how to route it to the pair score at bounded cost.

We test this claim on synthetic CTDGs. We build graphs whose label is controlled by $\rho \in [0, 1]$, the fraction of predictive signal that comes from neighbor history rather than the node's own state. As ρ grows toward one, a memory-only encoder should lose the signal and rank near chance. It

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

did, while CTDG-THOR retained high rank under the same parameter budget. The construction separates learnability of the route from the model definition.

We implement the idea in **CTDG-THOR**. The model combines a TGN-style memory with three components:

- History-conditioned attention makes attention depend on neighbor trajectories rather than only on current event features.
- Query-side second-order context builds a neighbor-history summary for the query endpoint and adds it at pair scoring through an adaptive fusion gate.
- Query-only computation expands second-order context only for the query, not for every negative candidate, so path computation does not grow with the candidate set.

The fusion gate starts near zero. CTDG-THOR therefore begins close to the history-conditioned attention model and learns how much neighbor history to use during training.

This paper makes four contributions.

1. **(Model)** We introduce candidate-independent query-side second-order temporal history routing for CTDGs. The context is computed once per query and reused across candidates. CTDG-THOR combines this route with a TGN-style memory and history-conditioned attention.
2. **(Theory)** We formalize an incident-only endpoint and a one-hop endpoint, then prove a separation witness, an exponential separation result, a fixed-backbone characterization, and a second-order separation (Theorems A_{ct}–D_{ct} and Corollaries 1–2). Proposition 1 establishes learnability of the second-order signal under the explicit linear-readout construction.
3. **(Mechanism evidence)** Under matched seeds and an identical backbone, CTDG-THOR improves over the one-hop control by $+0.0653$ MRR on `tgbl-wiki` and $+0.0368$ on `tgbl-enron`. Fixed-checkpoint interventions indicate that the trained predictor uses this route.
4. **(External validation and cost)** Under the fixed-negative TGB-v1 protocol, CTDG-THOR reaches test MRR 0.6416 ± 0.0123 on `tgbl-wiki`, 0.2478 ± 0.0255 on `tgbl-enron`, and 0.1223 ± 0.0084 on `tgbl-lastfm`. The second-order route adds one scalar and a mean aggregation over the one-hop control and keeps peak memory below 90 MB.

We test whether the second-order route adds information absent from the one-hop representation. Matched seeds and a shared backbone measure predictive value; fixed-checkpoint interventions measure route dependence; the TGB-v1 runs measure transfer.

2 Related Work

Dynamic-graph learning spans static, snapshot-based, and event-based designs Kazemi et al. (2020); Feng et al. (2026). We follow these three lines below.

Static graph neural networks. GCN Kipf & Welling (2017) propagates node features over a fixed graph, and GAT Veličković et al. (2018) learns attention weights over neighbors. Both assume that the graph is static, so neither represents when an interaction occurred or in what order events arrived.

Discrete-time dynamic graphs. Snapshot-based methods take different routes. EvolveGCN Pareja et al. (2020) evolves GCN parameters over time, GC-LSTM Chen et al. (2022) places graph convolution inside LSTM gates, and DySAT Sankar et al. (2020) combines structural and temporal attention. ROLAND You et al. (2022) updates static GNN states hierarchically. Recent scalable designs fuse input snapshots Qi et al. (2025) or maintain evolving neighbor stores Wu et al. (2026), but retain the fixed snapshot grid: event order within a snapshot is lost.

Continuous-time dynamic graphs. TGAT Xu et al. (2020) applies time-encoded attention to a node's temporal neighbors without maintaining memory. TGN Rossi et al. (2020) adds a trainable state to that design. JODIE Kumar et al. (2019) predicts future embedding trajectories in interaction networks, while DyRep Trivedi et al. (2019) models structural and temporal evolution together.

DyGFormer processes source and destination histories with a transformer, and TGFormer models interaction trajectories with an auto-correlation mechanism Yu et al. (2023); Chen et al. (2026). Several later models extend propagation over longer horizons Gravina et al. (2024), project temporal walk matrices Lu et al. (2024), simplify the backbone to a transformer Wu et al. (2024), or refresh stale memory states Pang et al. (2025). These methods change how node states are computed, but they do not specify where neighbor history should enter the pair score.

Another line of work routes temporal paths directly. CAW Wang et al. (2021b) uses causal anonymous walks, TCL Wang et al. (2021a) combines transformers with contrastive learning, and NAT Luo & Li (2022) scales neighborhood-aware attention. GraphMixer Cong et al. (2023) shows that a simple MLP mixer can remain competitive. TNCN Zhang et al. (2024) builds a candidate-conditioned pairwise representation from multi-hop common neighbors. CTDG-THOR instead isolates candidate-independent routing: the second-order query context is computed once and reused, while only the final pair score is candidate-specific. These designs differ in candidate dependency and computation reuse, as well as in where the information path enters prediction.

Evaluation protocols add another source of variation: temporal splits and repeated edges can change rankings Poursafaei et al. (2022), and standardized protocols expose failures that single-dataset comparisons miss Huang et al. (2023); Gastinger et al. (2024); Huang et al. (2024).

This leaves open where cross-node history should enter a memory-based CTDG pipeline. CTDG-THOR routes second-order context only on the query endpoint, so the context is computed once per query and its cost is independent of the number of negative candidates. We use official TGB reference models for external validation and same-code one-hop/second-order controls to isolate the effect of that route. The analysis below targets a specific question: can a node state built only from its own history represent trajectories that never enter it? Time-aware Weisfeiler–Lehman and universal-approximation results concern a broader class and leave this routing question open.

3 Problem Definition

We consider a continuous-time dynamic graph (CTDG) represented as an event stream

$$E = \{(u_i, v_i, t_i, m_i)\}_{i=1}^{n}, \tag{1}$$

where $u_i, v_i \in \mathcal{V}$ are nodes, t_i is a timestamp, and m_i is an edge feature or message. The node set $\mathcal{V}$ has size N and is fixed across time.

3.1 Temporal Link Prediction

Given events with timestamps before t, the goal is to score candidate edges at time t. A model computes

$$p(u, v, t) = g\big(\mathbf{h}_u(t), \mathbf{h}_v(t), \Theta\big), \tag{2}$$

where $\mathbf{h}_u(t)$ is the representation of node u at time t obtained from the event history, and g is a decoder. We evaluate with the original fixed-negative TGB-v1 protocol and the MRR metric. The released TGB-v1 negative sampler files contain 999 negative candidates per positive on `tgbl-wiki` and 1,000 on `tgbl-uci`. Current TGB-v2 leaderboards use different negative sets and are not directly comparable with these numbers. We use the v1 protocol because it provides a stable, matched-seed setting for controlled route comparisons. The protocol follows the Temporal Graph Benchmark Huang et al. (2023).

3.2 Temporal Node Property Prediction

At label timestamps $\{t_\ell\}$, each node u has a label vector $y_u(t_\ell) \in \mathbb{R}^C$. Given the event history before t_ℓ, the model predicts

$$\hat{y}_u(t_\ell) = \phi\big(\mathbf{h}_u(t_\ell)\big), \tag{3}$$

with a linear head ϕ and evaluation by NDCG@10 over the C classes.

Both tasks use the same temporal encoder; only the decoder differs.

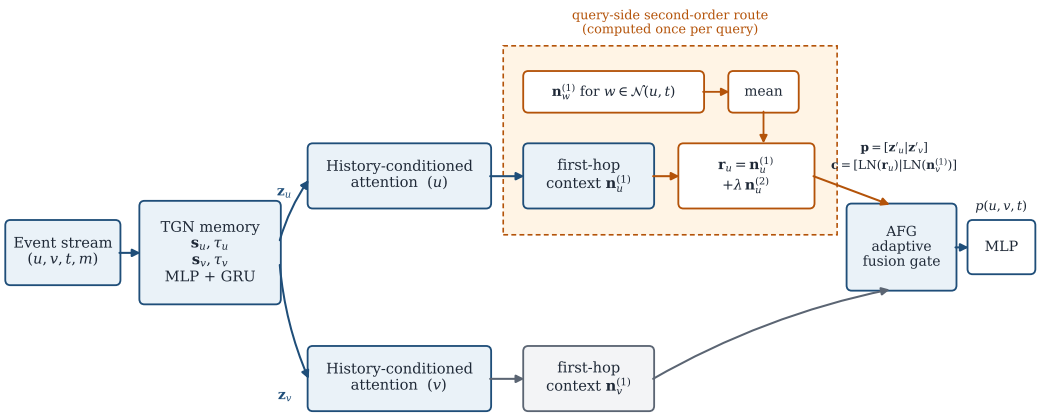

Figure 1: Overview of CTDG-THOR. A shared memory and history-conditioned attention encoder produces endpoint representations for the query and candidate. At scoring time, only the query endpoint receives a second-order context, formed by averaging the first-hop contexts of its neighbors; the candidate retains its first-hop context. The adaptive fusion gate combines these contexts before the decoder.

4 CTDG-THOR

CTDG-THOR uses a TGN-style memory backbone with history-conditioned attention and pair-level context routing. The encoder returns node representations $\mathbf{h}_u(t)$ for either a link predictor or a node-property head. Figure 1 gives an overview of the resulting pipeline.

4.1 Memory Backbone

Each node u maintains a memory state $\mathbf{s}_u$ and a last-update time τ_u. When an event (u, v, t, m) occurs, the source and destination memories are updated through a message function and a GRU:

$$\mathbf{msg}_u = \mathrm{MLP}([\mathbf{s}_u \| m \| \mathrm{TimeEnc}(t - \tau_u)]), \tag{4}$$

$$\mathbf{s}_u \leftarrow \mathrm{GRU}\big(\mathbf{s}_u, \mathrm{agg}_u(\mathbf{msg}_u)\big), \qquad \tau_u \leftarrow t, \tag{5}$$

and symmetrically for v with $\mathbf{s}_v$ and τ_v. Each endpoint update uses only its own memory and the current event message, so the backbone does not route the other endpoint's trajectory into the representation. At query time the node input is $\mathbf{z}_u = [\mathbf{s}_u \| \mathbf{e}_u]$, where $\mathbf{e}_u$ is a learned node embedding.

4.2 History-Conditioned Attention

At time t, a recent-neighbor loader returns the neighbors w of u whose events occur before t. We compute structural attention with logits conditioned on each neighbor's historical state:

$$e_{uw} = \mathrm{LeakyReLU}\big(\mathbf{a}^\top [\mathbf{q}_u \| \mathbf{W}_k \mathbf{z}_w \| \mathrm{TimeEnc}(t_{uw} - \tau_u) \| m_{uw}]\big), \tag{6}$$

$$\alpha_{uw} = \mathrm{softmax}_w(e_{uw}), \tag{7}$$

$$\mathbf{z}'_u = \mathbf{W}_{\mathrm{out}}\big([\mathbf{q}_u \| \textstyle\sum_w \alpha_{uw} \mathbf{W}_v \mathbf{z}_w]\big), \tag{8}$$

where $\mathbf{q}_u = \mathbf{W}_{\mathrm{base}} \mathbf{z}_u$ is a memory-based query and t_{uw} is the timestamp of the neighbor event. We encode the gap between the neighbor event and the last update of the endpoint being read, which yields $\mathrm{TimeEnc}(t_{uw} - \tau_u)$ in history-conditioned attention. The historical state $\mathbf{z}_w$ affects both which neighbors receive attention and the values being aggregated.

4.3 Query-Side Second-Order Context

For a query endpoint u, the second-order context aggregates the first-hop contexts of u's neighbors and is injected at the link-scoring level. For a node u, the first-hop context is an attention-weighted aggregation

$$\begin{aligned}\mathbf{n}_u^{(1)} &= \sum_{w \in \mathcal{N}(u,t)} \beta_{uw} \mathbf{W}_n \mathbf{z}_w, \\ \beta_{uw} &\propto \exp\big(\mathrm{LeakyReLU}\big(\mathbf{a}_n^\top [\mathbf{s}_u \| \mathbf{W}_n \mathbf{z}_w \\ &\qquad \| \mathrm{TimeEnc}(t_{uw} - \tau_w) \| m_{uw}]\big)\big).\end{aligned} \tag{9}$$

CTDG-THOR then averages the first-hop contexts of the current neighbors to form a second-order context,

$$\mathbf{n}_u^{(2)} = \frac{1}{|\mathcal{N}(u,t)|} \sum_{w \in \mathcal{N}(u,t)} \mathbf{n}_w^{(1)}, \qquad \mathbf{r}_u = \mathbf{n}_u^{(1)} + \lambda \mathbf{n}_u^{(2)}, \tag{10}$$

where λ is a learnable scalar initialized to 0.5. The first-hop context and second-order context share the same backbone and attention modules.

For a scored pair (u, v), we apply second-order routing to the query endpoint u. The candidate endpoint v retains its first-hop context. The pair-level adaptive fusion gate (AFG) combines the endpoint embeddings with the normalized contexts:

$$\mathbf{p} = [\mathbf{z}'_u \| \mathbf{z}'_v], \tag{11}$$

$$\mathbf{c} = [\mathrm{LayerNorm}(\mathbf{r}_u) \| \mathrm{LayerNorm}(\mathbf{n}_v^{(1)})], \tag{12}$$

$$\mathbf{g} = \sigma\big(\mathbf{W}_g[\mathbf{p} \| \mathbf{c}]\big), \tag{13}$$

$$\mathbf{p}^* = \mathbf{p} + \mathbf{g} \odot \mathbf{W}_c \mathbf{c}. \tag{14}$$

The gate is initialized near-closed by zeroing $\mathbf{W}_g$ and setting its bias to -3, so $g \approx 0.047$ at initialization. CTDG-THOR starts close to the history-conditioned attention model and learns how far to open the neighbor-history route during training.

4.4 Decoders and Training

For link prediction, a two-layer MLP produces the score $p(u, v) = \sigma(\mathrm{MLP}(\mathbf{p}^*))$ and is trained with binary cross-entropy against one negative candidate per positive edge. For node property prediction, a linear head predicts $\hat{y}_u = \phi(\mathbf{z}'_u)$ and is trained with binary cross-entropy against the C-dimensional label vector.

Training follows a chronological replay. Events are processed in timestamp order, memory is updated after each batch, and the updated memory is detached so gradients do not propagate across the entire history. Each validation pass continues in chronological order, and the best validation checkpoint stores both parameters and the TGN memory buffers. Test scoring loads those buffers and continues without resetting them. Appendix A reports the replay diagnostic.

5 Second-Order Routing Analysis

We study an event stream $E = \{(u_i, v_i, t_i, m_i)\}_{i=1}^n$ with increasing timestamps. The analysis uses two boundary cases: an incident-only model and a one-hop model. Comparing them isolates the effect of making length-two paths available. Intermediate routing designs are outside the scope of this comparison. Full constructions, proofs, margin conditions, and complexity accounting appear in Appendix D.

Incident-only endpoint. Class A_{ct}, the incident-only class, contains models whose representation of a node u at time t depends on events incident to u before t and on the current event features, but not on the memories or trajectories of other nodes. Memory-only predictors and TGN without graph attention are examples.

Theorem $\mathbf{A}_{\mathrm{ct}}$ (Temporal blindness). If two streams have the same events incident to v before t, including timestamps and messages, then every class-A_{ct} model produces the same representation of v at t, regardless of the rest of the graph.

Theorem $\mathbf{B}_{\mathrm{ct}}$ (Exponential separation). For every $k \geq 1$ there is a family of 2^k streams over $2k+1$ nodes such that all class-A_{ct} representations of a designated node are identical, while the pair-level input produced by the Direction-A context and gate separates all streams under the explicit construction parameters.

Theorem $\mathbf{C}_{\mathrm{ct}}$ (Fixed backbone). The collapse holds for arbitrary representation width, depth, and number of attention heads. The idealized construction separates the family with raw context margin at least $1/k$ and $O(d^2)$ additional parameters. The margin applies before LayerNorm; the implemented normalized readout is evaluated empirically.

Proposition 1 (Learnability). For bounded linear readouts on the raw context, the empirical Rademacher complexity is $O(\sqrt{1/m})$. A margin-maximizing readout therefore requires $O(B^2k^2\varepsilon^{-2}\log(1/\delta))$ samples for error ε with probability at least $1-\delta$.

One-hop endpoint. Class B_{ct}, the one-hop class, contains models whose query representation can use the query's own history and the current states of immediate neighbors, but not summaries computed from those neighbors' own neighborhoods. The one-hop control is in this class.

Theorem $\mathbf{D}_{\mathrm{ct}}$ (Second-order separation). For every $k \geq 1$ there are 2^k streams over $4k+2$ nodes such that every class-B_{ct} model maps the designated query node to the same representation. The candidate endpoint has identical history in all streams, while the raw second-order context of CTDG-THOR separates all query configurations. The construction uses uniform first-hop weights and an open second-order gate, with a raw-context margin of at least $1/k$.

Proof sketch. For each bit b_i, the stream contains (x_i, w_i) at time 1, (x_i, y_i) at time 2 if and only if $b_i = 1$, and (v, w_i) at time 3. A class-B_{ct} model reads the state of w_i, but that state depends only on the identical event at time 1, so the 2^k streams collapse at v. In CTDG-THOR, the first-hop context of w_i reads the memory of x_i, which changes only when $b_i = 1$. Averaging these contexts forms the second-order context of v and exposes all k bits. The open gate and generic projections then separate the resulting pair inputs. Because the candidate endpoint has the same history in every stream, the pair-level input preserves the query separation; any decoder that distinguishes these inputs also separates the pair scores.

Corollary 1 (Scale independence). For the class-A_{ct} construction, increasing width, depth, or heads cannot distinguish any two streams, because the distinguishing signal never enters the query's incident history.

Corollary 2 (Information content, informal). The class-A_{ct} representation has zero mutual information with the construction's label bits. Any model that routes the k bits into the query carries up to k bits of information. The gain is therefore tied to the number of information channels opened into the query, not to scale alone.

Interpretation and cost. The bounds give a direct test: remove or shuffle the second-order path on a fixed checkpoint and measure the change in MRR. The added projections cost $O(d^2)$ parameters; with bounded neighborhoods, second-order evaluation costs $O(\sum_{w \in \mathcal{N}(u,t)} |\mathcal{N}(w,t)|d)$ work per query. The implemented query-only route remains below 90 MB of peak memory and is within 0.5 s per UCI epoch of the one-hop control.

Construction check. We instantiated the theorem construction with the implemented modules outside the training loop. For $k = 3$, eight streams, and five random parameter draws, the memory-only encoder had minimum node and pair-score distance 0 in every draw. The attention-only, one-hop, and second-order variants separated all eight streams in every draw. For the one-hop control, the mean minimum pair-score distance was 1.10×10^{-4} at the near-closed initialization and 3.85×10^{-4} for the ideal open-gate construction.

Table 1: Matched one-hop/second-order comparison, five seeds, test MRR. The primary comparison is against the one-hop control. Dashes in the final column mark the CTDG-THOR rows, for which no self-comparison is defined.

Dataset	Model	Test MRR	p vs CTDG-THOR
tgbl-wiki	one-hop control	0.5762 ± 0.0177	0.0002
tgbl-wiki	**CTDG-THOR**	$\mathbf{0.6416 \pm 0.0123}$	—
tgbl-enron	one-hop control	0.2110 ± 0.0108	0.0094
tgbl-enron	**CTDG-THOR**	$\mathbf{0.2478 \pm 0.0255}$	—
tgbl-lastfm	one-hop control	0.1119 ± 0.0048	0.0778
tgbl-lastfm	**CTDG-THOR**	$\mathbf{0.1223 \pm 0.0084}$	—

6 Experiments

The experiments identify the second-order signal, test route dependence with matched and fixed-checkpoint comparisons, check transfer under TGB-v1, measure cost, and test the shared encoder on node properties.

6.1 Diagnostic: Synthetic Routing Tasks

In the controllable-ρ task, the label shifts from the node's own signal to neighbor history. The memory-only encoder moved toward chance as $\rho \to 1$, while models that injected neighbor history retained high rank. In the depth-2 task, the label is available only through a neighbor-of-neighbor event. Across five seeds, MRR was 0.5833 ± 0.0241 for memory-only, 0.5931 ± 0.0418 for the one-hop control, and 0.7708 ± 0.1201 for CTDG-THOR. The paired gain over one-hop was $+0.1778$ (95% CI $[+0.0485, +0.3071]$, $p = 0.0188$). The real-data comparison below tests whether the same gap appears on benchmark graphs.

6.2 Controlled Second-Order Comparison

We trained every variant from the same code base with identical dimensions, seeds, validation subsets, negative samplers, and evaluator. The one-hop control kept the TGN memory, history-conditioned attention, and first-hop context, but removed the second-order term. On the 10,000-node configuration the two models had 843,873 and 843,874 trainable parameters. The second-order route thus adds one learned scalar and one mean aggregation.

Table 1 reports the matched comparison. CTDG-THOR improved over one-hop by $+0.0653$ on wiki (95% CI $[+0.0524, +0.0783]$) and $+0.0368$ on enron (95% CI $[+0.0150, +0.0585]$). On lastfm the gain was $+0.0104$ (95% CI $[-0.0018, +0.0227]$), which we report as positive but non-significant. The matched-edge gain on wiki was $+0.0668$ (95% CI $[+0.0536, +0.0800]$; $p = 0.00015$), and a shuffled-content control reached 0.5847 ± 0.0195, below CTDG-THOR by $+0.0569$ ($p = 0.00147$). The same-code TGN-style control reached 0.5557 ± 0.0116 on wiki and 0.1960 ± 0.0100 on enron.

To separate architecture effects from protocol effects, we also ran a uniform internal protocol that fixes batch, dimension, seed, and validation settings across models. The one-hop control reaches the highest test MRR among these comparisons, 0.6594, compared with 0.5898 for TGN, 0.5311 for DyGFormer, 0.4249 for TGAT, and 0.3827 for the memory-only encoder. The second-order gain is therefore measured on top of the strongest internal baseline.

6.3 External Validation: Fixed-Negative TGB-v1 Protocol

The TGB-v1 runs used the released negative sampler files, evaluator, chronological replay, 20 epochs, batch size 200, and five seeds. Checkpoint selection used the first five validation batches; `tgbl-uci` used the full validation split. Table 2 summarizes the training and evaluation settings.

Table 3 reports the fixed-negative results. With the released sampler and evaluator, CTDG-THOR has the highest test MRR among the reference configurations reported here on all three datasets. The ordering therefore holds under the released sampler and evaluator.

Table 2: Training and evaluation protocol for the fixed-negative TGB-v1 runs.

Dataset	Model	Epochs	Batch	Dim.	LR	Neg./pos.
`tgbl-wiki`	CTDG-THOR	20	200	64	10^{-4}	999
`tgbl-wiki`	TGN	20	200	100	10^{-4}	999
`tgbl-enron`	CTDG-THOR	20	200	64	10^{-4}	183
`tgbl-enron`	TGN	20	200	100	10^{-4}	183
`tgbl-uci`	CTDG-THOR	20	200	64	5×10^{-4}	1,000
`tgbl-uci`	TGN	20	200	64	10^{-4}	1,000
`tgbl-lastfm`	CTDG-THOR	20	200	64	10^{-4}	999
`tgbl-lastfm`	TGN	20	200	100	10^{-4}	999

Table 3: Fixed-negative TGB-v1 reference protocol, five seeds for trained models, best validation and test MRR. EdgeBank is deterministic and has no checkpoint selection, so its validation entry is not applicable.

Dataset	Model	Best validation MRR	Test MRR
tgbl-wiki	TGN (official)	0.3974 ± 0.0133	0.2931 ± 0.0103
tgbl-wiki	TGAT-style	0.6050 ± 0.0076	0.5731 ± 0.0085
tgbl-wiki	DyGFormer (official)	0.5322 ± 0.0047	0.4741 ± 0.0035
tgbl-wiki	**CTDG-THOR**	$\mathbf{0.6866 \pm 0.0069}$	$\mathbf{0.6416 \pm 0.0123}$
tgbl-enron	TGN (official)	0.0112 ± 0.0004	0.0116 ± 0.0003
tgbl-enron	EdgeBank	—	0.1008
tgbl-enron	**CTDG-THOR**	$\mathbf{0.5083 \pm 0.0163}$	$\mathbf{0.2478 \pm 0.0255}$
tgbl-lastfm	TGN (official)	0.0487 ± 0.0080	0.0294 ± 0.0085
tgbl-lastfm	**CTDG-THOR**	$\mathbf{0.1315 \pm 0.0012}$	$\mathbf{0.1223 \pm 0.0084}$

To rule out an implementation artifact behind the low TGN enron score, we repeated the official TGN implementation with 100 and 64 dimensions. Validation MRR remained at 0.0110, test MRR was 0.0115, and the same code reached validation MRR 0.335 on wiki after five epochs. The flat enron curve is therefore specific to the reference configuration and dataset. Across seeds, the paired gaps to CTDG-THOR are significant on wiki ($p = 1.5 \times 10^{-6}$) and enron ($p = 3.2 \times 10^{-5}$); EdgeBank reaches test MRR 0.1008 on enron.

Table 4 reports additional reference configurations. Matching TGN's dimension does not explain its result on wiki (0.2337 vs. 0.2931 for the official dimension). TGAT-style is the strongest additional baseline on both datasets, at 0.5731 on wiki and 0.1953 on enron, while CTDG-THOR reaches 0.6416 and 0.2478. DyRep, TCL, and GraphMixer provide further reference points without changing the ordering.

6.4 Route Dependence and Query Regime

To test whether the trained predictor depends on the route, we intervened on five fixed wiki checkpoints while keeping the test edges and candidate sets unchanged. Table 5 reports the intervention results. Removing second-order context reduced MRR by 0.0105 ± 0.0047; shuffling its content across queries reduced MRR by 0.0452 ± 0.0088. Moving the query-side route to the candidate lowered performance by 0.0830, and bidirectional placement lowered it by 0.1347.

The route mattered most where second-order paths were available. Removing the context reduced MRR by 0.0554 with one or two valid paths, by 0.0460 with three to five paths, and by 0.0069 with six to ten paths. Without a path, the two conditions were identical. Queries with degree 0–1 and 2–4 gained $+0.0144$ and $+0.0134$, respectively. Cold queries in the rich-path group gained $+0.0097$, compared with $+0.0027$ for warm queries in that group.

6.5 Efficiency

In a same-process UCI benchmark, one epoch took 5.9 s for the memory backbone, 6.7 s for CTDG-THOR, and 19.4 s for DyGFormer. CTDG-THOR used 0.41M parameters and 56.8 MB peak mem-

Table 4: Additional reference configurations and adaptations under the fixed-negative TGB-v1 protocol, five seeds, best validation and test MRR.

Dataset	Model	Best validation MRR	Test MRR
tgbl-wiki	TGN, dim 64 alignment	0.3287 ± 0.0551	0.2337 ± 0.0594
tgbl-wiki	DyRep (dim 64)	0.0437 ± 0.0085	0.0254 ± 0.0078
tgbl-wiki	TGAT-style adaptation	0.6050 ± 0.0076	0.5731 ± 0.0085
tgbl-wiki	TCL	0.2590 ± 0.0120	0.2215 ± 0.0103
tgbl-wiki	GraphMixer	0.0719 ± 0.0046	0.0462 ± 0.0051
tgbl-enron	DyRep (dim 64)	0.0238 ± 0.0079	0.0215 ± 0.0097
tgbl-enron	TGAT-style adaptation	0.4717 ± 0.0250	0.1953 ± 0.0117

Table 5: Fixed-checkpoint route interventions on tgbl-wiki, five seeds.

Condition	Test MRR	Change
query-side second-order	0.6380 ± 0.0095	—
remove second-order	0.6275 ± 0.0088	-0.0105
shuffle second-order	0.5933 ± 0.0170	-0.0452
candidate-side route	0.5551 ± 0.0353	-0.0830
bidirectional route	0.5033 ± 0.0433	-0.1347

ory; DyGFormer used 0.98M and 991.8 MB. The query-only second-order route adds 0.8 s per UCI epoch over the memory backbone and does not scale with the negative candidate set. Relative to the TGN backbone, the complete routing module adds about 0.025M parameters, roughly 6.5%; the second-order term itself adds one scalar and one mean aggregation over the matched one-hop control.

Node-property pilot. To check whether the shared encoder extends beyond link prediction, we ran the node-property task from Section 3.2 on un_trade. The memory encoder reaches NDCG@10 0.3439 ± 0.0069, while the one-hop routing model reaches 0.3514 ± 0.0095 over three seeds. The same encoder therefore supports the node-property head. We report this as a preliminary check outside link prediction.

7 Conclusion

We presented CTDG-THOR, which combines TGN-style memory, history-conditioned attention, and candidate-independent query-side second-order temporal history routing. We prove that the second-order route expands the representable function class while backbone width, depth, and attention heads remain fixed. With the backbone and seeds held constant, CTDG-THOR improved over the one-hop control by $+0.0653$ MRR on tgbl-wiki and $+0.0368$ on tgbl-enron.

The fixed-checkpoint interventions indicate that the learned predictor uses this route. Removing second-order context reduced MRR by 0.0105, and shuffling it reduced MRR by 0.0452. Candidate-side and bidirectional placement performed worse. Under the fixed-negative TGB-v1 protocol, CTDG-THOR reaches test MRR 0.6416 ± 0.0123 on tgbl-wiki, 0.2478 ± 0.0255 on tgbl-enron, and 0.1223 ± 0.0084 on tgbl-lastfm. Under this protocol, it reaches the highest test MRR among the reported reference configurations, and the route remains query-only. In memory-based CTDGs, where history is routed into the prediction matters alongside what history is stored.

Reproducibility Statement

The supplementary archive contains the first-party model and training code, runner scripts, result JSONs, dependency lock, seed-to-run map, and exact commands. The appendix specifies the fixed-negative TGB-v1 protocol, hyperparameters, memory replay diagnostic, and the construction check for the separation results. Trained models in the main link-prediction tables use five seeds; EdgeBank is deterministic and uses one run. The UCI stability check uses three.

AI USE STATEMENT

AI-based tools assisted with language editing and literature verification. All scientific claims, mathematical arguments, experimental results, citations, and final manuscript content were reviewed and verified by the authors.

REFERENCES

Hongjiang Chen, Pengfei Jiao, Ming Du, Xuan Guo, Zhidong Zhao, Di Jin, and Xiao Liu. Tgformer: Towards temporal graph transformer with auto-correlation mechanism. *Pattern Recognition*, 170: 112053, 2026. doi: 10.1016/j.patcog.2025.112053.

Jinyin Chen, Xueke Wang, and Xuanheng Xu. Gc-lstm: Graph convolution embedded lstm for dynamic network link prediction. *Applied Intelligence*, 52(7):7513–7528, 2022. doi: 10.1007/s10489-021-02518-9.

Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. Do we really need complicated model architectures for temporal networks? In *ICLR*, 2023. URL https://arxiv.org/abs/2302.11636.

ZhengZhao Feng, Rui Wang, TianXing Wang, Mingli Song, Sai Wu, and Shuibing He. A comprehensive survey of dynamic graph neural networks: Models, frameworks, benchmarks, experiments and challenges. *IEEE Transactions on Knowledge and Data Engineering*, 38(1):26–46, 2026. doi: 10.1109/TKDE.2025.3621291.

Julia Gastinger, Shenyang Huang, Mikhail Galkin, Erfan Loghmani, Ali Parviz, Farimah Poursafaei, Jacob Danovitch, Emanuele Rossi, Ioannis Koutis, Heiner Stuckenschmidt, Reihaneh Rabbany, and Guillaume Rabusseau. Tgb 2.0: A benchmark for learning on temporal knowledge graphs and heterogeneous graphs. In *NeurIPS Datasets and Benchmarks Track*, volume 37, pp. 140199–140229, 2024. doi: 10.52202/079017-4450.

Alessio Gravina, Giulio Lovisotto, Claudio Gallicchio, Davide Bacciu, and Claas Grohnfeldt. Long range propagation on continuous-time dynamic graphs. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *PMLR*, pp. 16206–16225, 2024. URL https://proceedings.mlr.press/v235/gravina24a.html.

Qiang Huang, Xin Wang, Susie Xi Rao, Zhichao Han, Zitao Zhang, Yongjun He, Quanqing Xu, Yang Zhao, Zhigao Zheng, and Jiawei Jiang. Benchtemp: A general benchmark for evaluating temporal graph neural networks. In *IEEE International Conference on Data Engineering (ICDE)*, pp. 4044–4057, 2024. doi: 10.1109/ICDE60146.2024.00310.

Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany. Temporal graph benchmark for machine learning on temporal graphs. In *NeurIPS*, 2023. doi: 10.52202/075280-0099.

Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020. URL https://www.jmlr.org/papers/v21/19-447.html.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *KDD*, pp. 1269–1278, 2019. doi: 10.1145/3292500.3330895.

Xiaodong Lu, Leilei Sun, Tongyu Zhu, and Weifeng Lv. Improving temporal link prediction via temporal walk matrix projection. In *NeurIPS*, volume 37, pp. 141153–141182, 2024. doi: 10.52202/079017-4482.

Yuhong Luo and Pan Li. Neighborhood-aware scalable temporal network representation learning. In *Learning on Graphs Conference*, 2022. URL https://proceedings.mlr.press/v198/luo22a.html.

Jianye Pang, Xinjie Zhu, and Xiaofei Xiong. Dyfmvp: Say goodbye to staleness! fresh memory vigorous preserver for continuous-time dynamic graph. In *IEEE International Conference on Data Engineering (ICDE)*, pp. 1622–1635, 2025. doi: 10.1109/ICDE65448.2025.00125.

Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *AAAI*, pp. 5363–5370, 2020. doi: 10.1609/aaai.v34i04.5984.

Farimah Poursafaei, Shenyang Huang, Kellin Pelrine, and Reihaneh Rabbany. Towards better evaluation for dynamic link prediction. In *NeurIPS*, volume 35, pp. 32928–32941, 2022. doi: 10.52202/068431-2386.

QingGuo Qi, Hongyang Chen, Minhao Cheng, and Han Liu. Input snapshots fusion for scalable discrete-time dynamic graph neural networks. In *KDD*, pp. 1138–1149, 2025. doi: 10.1145/3690624.3709316.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs, 2020.

Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *WSDM*, pp. 519–527, 2020. doi: 10.1145/3336191.3371845.

Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Representation learning over dynamic graphs. In *ICLR*, 2019. URL https://openreview.net/forum?id=hCkI0hRgq.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Lu Wang, Xiaofu Chang, Shuang Li, Yunfei Chu, Hui Li, Wei Zhang, Xiaofeng He, Le Song, Jingren Zhou, and Hongxia Yang. Tcl: Transformer-based dynamic graph modelling via contrastive learning. arXiv preprint arXiv:2105.07944, 2021a.

Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. Inductive representation learning in temporal networks via causal anonymous walks. In *ICLR*, 2021b. URL https://arxiv.org/abs/2101.05974.

Danni Wu, Yuanyuan Xu, Xuemin Lin, Wenjie Zhang, and Ying Zhang. Understanding evolving graph structures for large discrete-time dynamic graph representation. *Proceedings of the VLDB Endowment*, 19(5):862–875, 2026. doi: 10.14778/3796195.3796201.

Yuxia Wu, Yuan Fang, and Lizi Liao. On the feasibility of simple transformer for dynamic graph modeling. In *WWW*, pp. 870–880, 2024. doi: 10.1145/3589334.3645622.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. In *ICLR*, 2020. URL https://openreview.net/forum?id=rJe2syrtvS.

Jiaxuan You, Tianlong Du, and Jure Leskovec. Roland: Graph learning framework for dynamic graphs. In *KDD*, pp. 2358–2366, 2022. doi: 10.1145/3534678.3539300.

Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. Towards better dynamic graph learning: New architecture and unified library. In *NeurIPS*, 2023. doi: 10.52202/075280-2960.

Xiaohui Zhang, Yanbo Wang, Xiyuan Wang, and Muhan Zhang. Efficient neural common neighbor for temporal graph link prediction, 2024.

APPENDIX

A SUPPLEMENTARY EXPERIMENTAL DETAILS

All link-prediction experiments use the original fixed-negative TGB-v1 sampler files (one-vs-many, `hist_rnd` strategy). For each positive edge, the model scores the positive destination against all fixed-negative candidates, and MRR is computed with the released TGB-v1 evaluator. For `tgbl-uci`, the official files contain 1,000 negatives per positive, and we use the full validation split for checkpoint selection because a five-batch subset is too noisy for that dataset. An anonymized supplementary archive containing the model code, runner scripts, result JSONs, and exact reproduction commands accompanies this submission. It also contains the dependency lock and the seed-to-run map used for every paired comparison.

The best validation checkpoint contains both model parameters and the TGN memory buffers used during chronological replay. Test scoring loads those buffers and does not reset them. The optional replay mode is a diagnostic only; it cannot reconstruct the memory trajectory written by earlier training weights, so it is not used for the reported numbers.

Table 6: Wall-clock per 20-epoch run (RTX 2060 SUPER).

Dataset	CTDG-THOR (min)	Official TGN (min)
tgbl-wiki	7–8	8–10
tgbl-enron	4–5	4
tgbl-uci	6	5–6

Table 6 reports wall-clock per 20-epoch run, including training and the reported validation schedule; absolute times vary with GPU load, and parameter counts and peak memory are reported in the main-text efficiency table.

B EVIDENCE PROTOCOL TIERS

Table 7 separates the sources of the reported numbers. The distinction matters because the TGB main benchmark and the TGB supplementary datasets share the negative-sampler and evaluator protocol, but not their provenance. No DGB-derived rebuild is used in the paper version.

Table 7: Protocol and data-source tiers.

Tier	Data	Use in this paper
TGB-v1 main benchmark	Original `tgbl-wiki` files and fixed-negative sampler/evaluator	Main five-seed link-prediction table
TGB-v1 supplementary	Original `tgbl-enron`, `tgbl-lastfm`, and `tgbl-uci` files and fixed-negative sampler/evaluator	Main and appendix link-prediction rows; UCI stability check
DGB-derived rebuild	Not used in the current paper	Historical only; excluded from the submission package
Uniform internal protocol	TGB-v1 data and negative files with shared batch, dimension, seed, and validation settings	Controlled ablation and `un_trade` pilot

C TGBL-UCI STABILITY CHECK

Table 8 summarizes this stability check. Under the fixed-negative TGB-v1 protocol, `tgbl-uci` uses 1,000 negatives per positive and the full validation split for checkpoint selection. CTDG-THOR reaches test MRR 0.0778 ± 0.0095 versus 0.0380 ± 0.0385 for the official TGN implementation over three seeds ($p = 0.168$); the smaller standard deviation indicates lower run-to-run variance, while three seeds do not support a formal mean-significance claim.

Table 8: `tgbl-uci` under the fixed-negative TGB-v1 reference protocol, 20 epochs, 3 seeds.

Model	Best validation MRR	Test MRR
TGN (official, dim 64)	0.0189 ± 0.0142	0.0380 ± 0.0385
CTDG-THOR (dim 64)	0.0721 ± 0.0097	$\mathbf{0.0778 \pm 0.0095}$

D Second-Order Routing Analysis

We analyze continuous-time dynamic graphs (CTDGs), where the input is an event stream $E = \{(u_i, v_i, t_i, m_i)\}_{i=1}^{n}$ with increasing timestamps.

The main object is the distinction between one-hop and second-order temporal history routing. A one-hop representation may depend on the query node's own history and the current states of its immediate neighbors. A second-order representation may additionally depend on summaries computed at those neighbors from their own neighborhoods. The implemented one-hop control belongs to the first class. CTDG-THOR computes the second-order context $\mathbf{n}_v^{(2)}$ in Eq. (11) and is the primary construction in Theorem D_{ct} below. The incident-only endpoint is retained as a background analysis because it identifies the information that is absent before any cross-node routing is introduced.

We treat the formal classes below as restricted endpoints of the temporal receptive field. The construction isolates what changes when a path of length two is made available; it is not intended as a complete description of every partial-routing temporal model. The route-dependence interventions in Section 6.4 provide the operational test of the mechanism.

D.1 Incident-Only Endpoint

Define **class $\mathbf{A}_{\mathrm{ct}}$** as models whose representation of a node u at time t depends on:

1. the events incident to u before t (its own memory/trajectory), and
2. the current event features and timestamp,

but *not* on the memories or trajectories of other nodes at t. Memory-only link predictors and TGN without the graph-attention module belong to this class; attention models that aggregate neighbor features are partial-routing and lie outside it.

Theorem $\mathbf{A}_{\mathrm{ct}}$ (Event-level temporal blindness). Let M be any model in class A_{ct}. If two event streams have the same events incident to node v up to time t, with the same timestamps and messages, then M produces the same representation of v at time t, regardless of how the rest of the graph evolved.

Proof. The representation of v at time t is, by definition of class A_{ct}, a function only of the events incident to v before t and the current event features. These are identical in the two streams.

D.2 Exponential Separation for Event Streams

Theorem $\mathbf{B}_{\mathrm{ct}}$ (Exponential separation). For every $k \geq 1$ there exists a family of 2^k event streams over $2k + 1$ nodes such that:

1. every model in class A_{ct} produces the same representation of a designated node v;
2. the pair-level scoring input used by CTDG-THOR with query-side second-order context and the Adaptive Fusion Gate separates the family under the explicit construction parameters below. The node embedding of v is produced by history-conditioned attention and is not the object modified by the query-side second-order context.

D.2.1 Construction

Nodes: $v, w_1, \ldots, w_k, x_1, \ldots, x_k$. For a bit vector $b \in \{0, 1\}^k$:

- at time 1: event (w_i, x_i) occurs iff $b_i = 1$, and event (v, w_i) occurs for every i;

- the message of (w_i, x_i) is a distinct one-hot signature e_i.

D.2.2 Collapse for Class $\mathrm{A_{ct}}$

Node v has exactly the events (v, w_i) at time 1 for all i, identical across all b. Hence its own memory/trajectory is identical across all configurations, so every model in class $\mathrm{A_{ct}}$ produces the same representation for v.

D.2.3 Separation for CTDG-THOR

After time 1, the query-side second-order context computes

$$n_v = \sum_{i=1}^{k} \beta_{v,w_i} W_n \, h_{w_i}(1), \tag{15}$$

where $h_{w_i}(1)$ encodes whether (w_i, x_i) occurred. For generic β, W_n, and gate parameters, the context n_v is injective in b, so the pair-level scoring inputs are pairwise distinct. A later event involving v inherits this separation through the link predictor.

D.2.4 Generic Parameters

Scope. Lemma 1 is stated for the analytically parameterized readout used in the construction. It is not asserted for the full training pipeline, whose activations are only piecewise analytic.

Lemma 1 (Generic separation). Let $f : \Theta \times X \to \mathbb{R}^d$ be real-analytic in θ for every x. If there exists θ_0 and $x \neq x'$ with $f(\theta_0, x) \neq f(\theta_0, x')$, then for Lebesgue-almost every θ, $f(\theta, x) \neq f(\theta, x')$.

Proof. The set $\{\theta : f(\theta, x) = f(\theta, x')\}$ is the zero set of a nonzero real-analytic function, hence has Lebesgue measure zero. A finite union over pairs (b, b') still has measure zero.

An explicit θ_0 exists by continuity: take $\beta_{v,w_i} = 1/k$, $W_n = I$, $W_q = I$, and the gate fixed open in the idealized readout; then the raw neighbor context distinguishes exactly which coordinates $b_i = 1$. Applying Lemma 1 gives the generic statement. This generic statement concerns the constructed readout only; it is not used to assert parameter-genericity of the trained CTDG-THOR pipeline.

D.3 Fixed Backbone-Capacity Separation and Learnability

Theorem $\mathrm{C_{ct}}$ (Fixed backbone-capacity separation). Fix width $d \geq k$, depth L, and number of heads H. Every model in class $\mathrm{A_{ct}}$ with d-dimensional node representations and any Lipschitz link predictor collapses the family of Theorem $\mathrm{B_{ct}}$: the representation of v is identical across all 2^k configurations, so no positive separation margin is achievable even if the model is given exactly the same width, depth, and heads as CTDG-THOR. The idealized construction separates the family with raw-context margin at least $1/k$ for the explicit parameters of Theorem $\mathrm{B_{ct}}$, using $O(d^2)$ additional parameters and $O(\sum_{w \in \mathcal{N}(u,t)} |\mathcal{N}(w,t)|d)$ neighbor-attention work beyond its TGN backbone.

Proof. The collapse follows directly from the definition of class $\mathrm{A_{ct}}$: the representation of v depends only on events incident to v, which are identical across all configurations, irrespective of width, depth, or heads. For the margin, set $\beta_{v,w_i} = 1/k$, $W_n = I$, $W_q = I$, and the gate to 1 in the construction of Theorem $\mathrm{B_{ct}}$. The query-side second-order context is

$$n_v(b) = \frac{1}{k} \sum_{i:b_i=1} e_i, \tag{16}$$

so for $b \neq b'$,

$$\|n_v(b) - n_v(b')\|_2 \;\geq\; \frac{1}{k}. \tag{17}$$

For the raw context, there exists a 1-Lipschitz readout that assigns distinct scores to b and b', giving margin at least $1/k$ before normalization. We do not claim that LayerNorm preserves this lower bound: finite Lipschitzness is an upper-bound property, and normalization can identify distinct raw

contexts. The implemented normalization and readout are therefore evaluated empirically rather than covered by the margin guarantee. The additional parameters are the projection, query, context, and gate matrices, i.e. $O(d^2)$. ■

Remark on margin realizability. For clarity, the margin construction sets the gate to exactly 1. In the trained model the gate is a sigmoid, so this value is approached only in the limit. The margin statement applies to the raw context before LayerNorm; Proposition 1 likewise treats the query-side second-order context in this idealized construction. Its margin condition requires $1/k \geq 1/(2k)$, which is satisfied.

Proposition 1 (Learnability of the second-order signal). Let $\mathcal{H}$ be the class of linear readouts w with $\|w\|_2 \leq B$ applied to the raw second-order context. In the explicit construction these contexts lie in the unit ball. The empirical Rademacher complexity of $\mathcal{H}$ on m samples is at most $B\sqrt{2/m}$. Consequently, when the raw-context separation margin of Theorem $\mathrm{C_{ct}}$ is at least $1/(2k)$, a margin-maximizing linear readout achieves generalization error at most ε with probability at least $1-\delta$ using

$$m \;=\; O\!\left(\frac{B^2k^2}{\varepsilon^2}\log\frac{1}{\delta}\right) \tag{18}$$

samples. In contrast, every class $\mathrm{A_{ct}}$ readout is constant on the family and has error $1/2$ on the balanced decision problem, for any number of samples.

Proof. For the linear class with $\|w\|_2 \leq B$ and inputs in the unit ball, the standard Rademacher bound is $\widehat{\mathfrak{R}}_m(\mathcal{H}) \leq B\sqrt{2/m}$; the sample-complexity claim follows from the usual margin-based generalization bound. The zero-information claim is immediate from Theorem $\mathrm{A_{ct}}$, since the readout sees only the constant class-A representation of v. ■

Remark. Theorem $\mathrm{C_{ct}}$ and Proposition 1 address the two natural reviewer objections to an expressiveness-only claim: the separation is not an artifact of giving CTDG-THOR a larger parameter budget, and the explicit linear readout construction is learnable from few samples rather than being a purely existential construction. The construction is verified empirically in the supplementary code.

D.4 Second-Order Separation for CTDG-THOR

Define **class $\mathbf{B_{ct}}$** as models whose representation of a query node v can depend on v's own incident history and on the memory states of its immediate neighbors, but not on the memory states of nodes at distance two or more. This class captures memory-free one-hop temporal attention baselines in which a neighbor contributes its current embedding, not the history that produced that embedding. It is a restricted endpoint used to isolate the architectural difference; it is the one-hop end of a temporal receptive-field spectrum, and we do not claim that every one-hop model belongs to this class. The implemented one-hop control is in this class: its context is a function of the states $\mathbf{z}_w$ of immediate neighbors, and does not evaluate first-hop contexts of those neighbors. CTDG-THOR lies outside the class because its term $\mathbf{n}_v^{(2)}$ explicitly aggregates the first-hop contexts $\mathbf{n}_w^{(1)}$.

Theorem $\mathbf{D_{ct}}$ (Second-order temporal separation). For every $k \geq 1$ there exist 2^k event streams over $4k+2$ nodes such that every class-$\mathrm{B_{ct}}$ model produces the same representation of v. The candidate endpoint c has identical history across the streams, while the raw second-order context in CTDG-THOR separates all query configurations. The resulting pair inputs therefore preserve the separation; any decoder that distinguishes these inputs also separates the pair scores.

Proof. Let the nodes be c, v, w_i, x_i, and y_i for $i \in [k]$. The candidate endpoint c has no incident events in any stream. For a bit vector $b \in \{0,1\}^k$, construct the following stream:

- at time 1, event (x_i, w_i) occurs for every i, with a message that does not depend on b_i;
- at time 2, event (x_i, y_i) occurs if and only if $b_i = 1$, with message e_i chosen as a one-hot signature;
- at time 3, event (v, w_i) occurs for every i, with the same message for all b.

For class-B_{ct} models, the representation of v at time 3 depends only on (v, w_i) and on the memory of each w_i. The events (x_i, w_i) and (v, w_i) are identical across b, so each w_i's memory also cannot distinguish b_i; the distinguishing event (x_i, y_i) is not incident to w_i. Hence every class-B_{ct} model maps all 2^k streams to the same representation of v. Since c has no incident events, its representation is also identical across the streams; a deterministic pair decoder therefore receives the same input for every b.

In CTDG-THOR, the first-hop context of w_i aggregates the state of x_i. The memory of x_i changes only when (x_i, y_i) occurs, so this first-hop context distinguishes b_i. The second-order context of v aggregates the first-hop contexts of its neighbors w_i, thereby routing all k bits to v. With generic projections and a positive second-order gate, the resulting pair-level context separates the 2^k streams. Because the candidate representation is fixed, the concatenated pair input preserves this separation, and any decoder that distinguishes the pair inputs yields distinct scores. The separation is therefore not available to class-B_{ct}, although CTDG-THOR opens it with $O(d^2)$ additional parameters and $O(\sum_{w \in \mathcal{N}(u,t)} |\mathcal{N}(w,t)|d)$ neighbor-attention work. The construction is an existence result for the idealized readout with uniform first-hop weights and an open second-order gate. The trainable softmax/GRU/AFG implementation is evaluated empirically; the theorem does not assert that gradient training finds this parameter configuration.

Fixed-budget separation. Let $d \geq k$, fix arbitrary width d, depth L, and number of heads H. Every class-B_{ct} model with d-dimensional node representations and any Lipschitz pair decoder maps the family of Theorem D_{ct} to the same representations of both v and the fixed candidate c, hence the same pair inputs. The explicit CTDG-THOR construction separates the family with raw-context margin at least $1/k$. The argument follows Theorem C_{ct}: the raw second-order context is $\frac{1}{k}\sum_{i:b_i=1} e_i$, so distinct bit vectors differ by at least $1/k$ before normalization. We do not extend this lower bound through the implemented LayerNorm; the normalized learned readout is evaluated empirically.

Learnability of the second-order readout. For linear readouts with bounded Euclidean norm B, the empirical Rademacher complexity on m samples is at most $B\sqrt{2/m}$. With margin $1/k$, a margin-maximizing linear readout requires

$$O\big(B^2 k^2 \varepsilon^{-2} \log(1/\delta)\big)$$

samples for generalization error at most ε with probability at least $1-\delta$. This mirrors Proposition 1 and applies to the explicit linear construction of Theorem D_{ct}, not to every trained component of the full model.

D.5 Why the Separation Is an Information-Routing Property

The blindness of class A_{ct} is not a definitional convenience but a consequence of an information-flow constraint that is directly testable. Restate the class operationally: a model is in class A_{ct} if and only if, for every node v, time t, and pair of event streams E, E' that agree on the events incident to v before t and on the current event features, the model returns the same representation of v at t, even when E and E' are permuted on all events not incident to v. This is a property one can check at runtime on a single model: fix the incident events and reorder the distant ones; a class-A_{ct} model is invariant, while every model that routes cross-node history into v changes.

Corollary 1 (Scale-independence of the blindness). Let M be any model in class A_{ct} with representation width d, depth L, and H attention heads, and let $b \in \{0,1\}^k$ be the bit vector of the Theorem B_{ct} construction. For every d, L, H, the mapping $b \mapsto \mathbf{h}_v(t)$ is constant. Consequently no width, depth, or head count, no amount of training, and no data size can make a class-A_{ct} encoder separate even two of the 2^k streams.

Proof. The events incident to v in the construction are exactly (v, w_i) at time 1 for all i, which are identical for every b; the current event features are likewise identical. By the operational definition of class A_{ct}, $\mathbf{h}_v(t)$ depends only on these. The bound holds uniformly over (d, L, H). ■

Corollary 1 makes precise the sense in which the obstacle is the *absence of a cross-node information path* rather than limited capacity. The information that distinguishes the 2^k streams—which neighbors w_i fired, i.e. the values of b_i—lives on the edges $w_i \to x_i$ at time 2 and never reaches

v's incident history. No enlargement of the readout that observes v alone can expose it. The minimum repair is to route at least one bit per neighbor from the neighborhood into the query readout, which is exactly what the second-order context and history-conditioned attention do. Adding width to a class-A_{ct} encoder and adding a single cross-node injection are therefore structurally different operations: the former grows a function of an input that is already information-poor, while the latter adds information. This is why the claim is about *information routing*, and why it does not reduce to a larger-capacity model being compared against a smaller one (the two models in Theorem C_{ct} have identical width, depth, and heads).

Class A_{ct} is best read as the incident-only endpoint of a spectrum rather than as an exact description of any one baseline. In practice, temporal encoders route a partial slice of cross-node history: a TGAT-style model aggregates neighbor features, TGN with graph attention reads neighbor memories, and DyGFormer, TCL, and GraphMixer encode neighbor sequences. These models therefore sit strictly between the incident-only endpoint and the second-order-context injection—they are not blind, but they open only a limited set of routing channels into a node's representation. The construction identifies the bit of information those channels can miss (the neighbor's own evolving trajectory), and the second-order context is the minimal mechanism that can route it. The empirical gain over the reference TGN and the memory baseline is consistent with a routing view only when paired with the same-code ablation in Section 6.2. Official-TGN gaps alone do not establish the routing interpretation. The theorem bounds the blind endpoint rather than predicting each baseline's exact partial-routing position.

Corollary 2 (Information content), informal. For the family of Theorem B_{ct}, split the representation of a well-trained model into the part determined by the node's own history and the part routed from cross-node history. A class-A_{ct} model has zero cross-node component, hence its representation carries mutual information 0 with b. Any model that routes the k firing bits into v carries mutual information up to k bits. The achievable gain is therefore proportional to the number of distinct information channels the model opens between the neighborhood and v, not to the model's scale.

D.6 Efficiency–Expressiveness Joint View

The expressiveness gain must be weighed against cost. The second-order context and history-conditioned attention add $O(d^2)$ parameters to a TGN backbone: $d \times d$ projections $\mathbf{W}_n$, $\mathbf{W}_k$, $\mathbf{W}_v$, a pair-level gate $\mathbf{W}_g$, and the context projections used by the pair scorer. Concretely, on the real-data configuration ($d = 64$, $k = 10$) the measured overhead is 0.025M parameters, about 6.5% relative to the TGN-style backbone. The second-order term requires evaluating first-hop contexts for the direct neighbors, giving $O(k^2 d)$ work per query when both neighborhoods are capped at k; it does not add a second encoder. Section 6.5 in the main text shows the resulting wall-clock and peak-memory costs stay within 0.5 s and 35 MB of the one-hop control on `tgbl-uci`, while DyGFormer is roughly $3\times$ slower and uses about 1 GB of peak memory. In the backbone-capacity separation of Theorem C_{ct}, the injection buys a positive separation margin $1/k$ that no class-A_{ct} model with the same width, depth, and heads can achieve. For fixed d, the added parameter cost does not grow with the number of attended neighbors, while the marginal raw-context separation decreases as $1/k$ and the query-time attention cost grows as $O(k^2 d)$ under the bounded-neighborhood implementation.