QUOTIENT-RESIDUAL TEMPORAL MEMORY FOR SCALABLE CONTINUOUS-TIME GRAPH LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continuous-time temporal graph models maintain evolving node states and repeatedly propagate information as timestamped interactions arrive. Their accuracy comes with a persistent cost: the state and propagation domain grow with the original graph. Graph coarsening can reduce this domain, but prior coarse-to-fine pipelines recover accuracy through full-graph fine-tuning and memory reconstruction, which reintroduces the resource that coarsening was intended to remove. We introduce the Quotient-Residual Temporal Network (QRTN), a fine-tuning-free framework that preserves every timestamped event while separating shared quotient dynamics from a small identity-preserving residual. QRTN provides three operating points. QRTN-M processes original-node events throughout training while storing high-dimensional state only on supernodes. QRTN-X trains shared temporal operators on an event-preserving quotient and transfers parameters, but no runtime state, to a fresh original graph. QRTN-X-Gated uses the same quotient training but initialises original-node runtime state through a fixed, attenuated lift of the learned quotient state, without replay or original-graph gradient updates. These modes expose an auditable choice among persistent memory compression, strict parameter-only transfer, and state-initialised transfer. We provide a causal formulation, exact state accounting, and time-complexity bounds that distinguish node compression from event reduction. The accompanying evaluation protocol tests accuracy, inductive prediction, oversmoothing, cold-start behaviour, and end-to-end resource use across coarsening ratios.

1 INTRODUCTION

Continuous-time dynamic graphs represent a system as an ordered stream of interactions rather than as a sequence of snapshots. This representation retains exact event times and causal order, which are important in communication, transaction, and recommendation data. Temporal Graph Networks (TGNs) and related models exploit this structure through recurrent memories, time encoders, and temporal neighborhood aggregation (Rossi et al., 2020; da Xu et al., 2020; Kumar et al., 2019; Trivedi et al., 2019). The same mechanisms make training expensive. Every event may read and update memory, encode elapsed time, and trigger graph propagation. At scale, persistent node state and repeated temporal computation become primary bottlenecks (Zhou et al., 2022).

Graph coarsening offers a natural response: replace the original graph with a smaller quotient whose nodes represent groups of structurally related nodes. This strategy is well established for static graphs (Loukas, 2019; Chen et al., 2022; Huang et al., 2021). A direct application to temporal learning, however, creates a mismatch. Training occurs between supernodes, while evaluation asks for predictions between original nodes. A common remedy is to transfer coarse-trained parameters to a full model and fine-tune on the original stream. That remedy weakens the resource claim because it recreates original-node memories and performs a second optimization phase on the full graph.

This paper asks a more specific question: *can a continuous-time learner preserve original-node predictive behavior while moving most recurrent state off the original graph?* Equivalently, what must remain at original-node resolution for temporal link prediction without a full-graph recovery phase? Our answer is a state-allocation principle: high-dimensional quotient state captures dynamics shared by nodes in the same supernode, while a low-dimensional residual preserves the private history

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.

required to distinguish them. Both states are updated causally and every timestamped event remains in the stream. QRTN compresses the state domain, not the temporal evidence.

This decouples three resolutions usually tied together in continuous-time graph models: event observation, recurrent-state ownership, and prediction. QRTN keeps the first and third at original-node resolution while allowing high-dimensional temporal state to live at quotient resolution.

We study three questions: whether high-dimensional recurrent state can remain compressed throughout deployment (QRTN-M); whether quotient-trained temporal operators can be instantiated at original-node resolution without replay or fine-tuning (QRTN-X); and whether a transfer gap reflects learned operators or missing historical runtime state (QRTN-X-Gated, via a bounded one-time lift). Neither transfer protocol replays the original training stream or performs original-graph gradient updates.

The distinction matters because "compression" otherwise hides several different quantities. A method may reduce nodes, persistent state, unique endpoint pairs, raw events, or full-pipeline time. QRTN preserves all raw events by design, so its event-processing cost remains linear in the stream length. Its savings come from the state domain and from batching high-dimensional quotient updates. We state these costs explicitly and report them separately in the evaluation protocol.

The contributions are:

- **a state-resolution principle for continuous-time graph learning:** original-node event and prediction resolution can be preserved while high-dimensional recurrent state is owned at quotient resolution, with a small private residual retaining node-specific temporal history;
- **two operational realizations and one diagnostic:** QRTN-M for persistent compressed execution, QRTN-X for zero-replay, zero-adaptation coarse-to-original operator transfer, and QRTN-X-Gated for isolating historical-state mismatch;
- **evidence that the decomposition is not merely a coarsening heuristic:** removing the private residual sharply degrades prediction, while historical-negative, cold-start, transfer-capacity, and ratio controls isolate when shared state succeeds and when it fails;
- **resource-transparent evaluation:** event volume, state ownership, state dimensionality, quotient-update count, accelerator memory, throughput, and predictive retention are reported separately, including a million-event state-scaling profile.

2 Related Work and Positioning

TGN-style models use recurrent memory and temporal message passing to learn from continuous-time streams (Rossi et al., 2020; da Xu et al., 2020; Kumar et al., 2019; Trivedi et al., 2019); systems work accelerates their full-graph sampling and synchronization (Zhou et al., 2022). Memory-free temporal sequence models, including GraphMixer and DyGFormer, provide important accuracy context but do not own persistent recurrent node state in the same way (Cong et al., 2023; Yu et al., 2023). QRTN is complementary: it changes the ownership and resolution of recurrent state, and transfers recurrent update operators, rather than compressing a memory-free sequence encoder.

QRTN is not a temporal application of graph coarsening alone. Its central distinction is that event resolution, recurrent-state resolution, and prediction resolution are separated: events and predictions remain at original-node resolution, while shared high-dimensional temporal state may live on a quotient. Static coarsening and hybrid shared-private models motivate the quotient-residual decomposition (Chen et al., 2022; Loukas, 2019; Huang et al., 2021; Joly et al., 2026; Li et al., 2026). Their mechanism is different: static methods jointly propagate on original and coarsened structure, whereas QRTN assigns causal private updates to original endpoints, updates shared quotient state at controlled temporal resolution, and defines a no-recovery transfer boundary for one chronological stream. Coarsening can regularize redundant propagation, but it can also erase identity; QRTN makes that trade-off measurable rather than assuming it prevents oversmoothing (Li et al., 2018; Oono & Suzuki, 2020). Detailed positioning and evaluation-protocol context appear in Appendix B.

3 PROBLEM FORMULATION

3.1 CONTINUOUS-TIME EVENT STREAM AND LINK PREDICTION

A continuous-time dynamic graph (CTDG) represents a system as an ordered sequence of timestamped interactions rather than as a sequence of discrete static snapshots. Let V denote the universe of $N_{\text{full}} = |V|$ original nodes observed across the entire timeline. The dynamic interaction stream is defined as:

$$\mathcal{E} = \{e_i = (u_i, v_i, t_i, \mathbf{e}_i)\}_{i=1}^{M}, \qquad t_1 \le t_2 \le \cdots \le t_M, \tag{1}$$

where $u_i, v_i \in V$ are the interacting source and destination nodes, $t_i \in \mathbb{R}_{\ge 0}$ is the non-decreasing event timestamp, and $\mathbf{e}_i \in \mathbb{R}^{d_e}$ is an optional dynamic edge feature vector (e.g., interaction text attributes or transaction metadata), distinct from static node features $\mathbf{x}_u \in \mathbb{R}^{d_x}$ (set to zero vectors when node features are absent).

At time t_i, the objective is dynamic link prediction: given the causal history of events prior to t_i, predict whether edge (u_i, v_i) occurs at timestamp t_i. Let $s_\theta : V \times V \times \mathbb{R}_{\ge 0} \to \mathbb{R}$ denote a temporal link scoring function parameterized by learnable weights θ. For each observed positive event $e_i = (u_i, v_i, t_i, \mathbf{e}_i)$, the model scores the true destination v_i against a negative candidate v_i^-. Parameters θ are trained by minimizing the binary cross-entropy ranking loss:

$$\mathcal{L}(e_i) = -\log \sigma(s_\theta(u_i, v_i, t_i)) - \log\big(1 - \sigma\big(s_\theta(u_i, v_i^-, t_i)\big)\big)\,, \tag{2}$$

where $\sigma(z) = 1/(1 + e^{-z})$ is the standard logistic sigmoid activation. By design, all model states are updated only *after* link prediction scores for timestamp t_i have been evaluated, preventing future temporal information leakage.

Negative candidates. Each positive event is ranked against a candidate destination sampled before scoring; state is updated only afterward. The primary comparison uses persisted random candidates shared across methods, while historical and inductive candidates are robustness controls. Role-constrained candidate pools and all sampler definitions are given in Appendix B.

3.2 QUOTIENT GRAPH PARTITION AND COMPRESSION RATIO

A quotient partition $\pi : V \to V_{\text{c}}$ maps each original node $u \in V$ to a supernode $c \in V_{\text{c}} = \{1, \ldots, N_{\text{c}}\}$, where $N_{\text{c}} = |V_{\text{c}}|$ is the number of coarsened supernodes. In continuous-time streams, nodes appear dynamically. To prevent future temporal leakage, the partition π is constructed strictly using the chronological training split $\mathcal{E}_{\text{train}}$.

Let N_{train} be the nodes observed in the training split. A nominal ratio r allocates $N_{\text{c}} = \lceil r N_{\text{train}} \rceil$ supernodes, giving realized ratio $r' = N_{\text{c}}/N_{\text{full}}$. Thus r' can differ across datasets at the same nominal r; exact counts and the full derivation are in Appendix I.

Mapped self-loop events advance temporal state but are masked from the link-ranking loss, preserving timing without creating trivial positive links. The partition is a fixed, training-only preprocessing artifact, with role-preserving clustering in bipartite streams and deterministic assignment for nodes first seen after training. Appendix B gives the complete construction and leakage controls.

4 QUOTIENT-RESIDUAL TEMPORAL NETWORKS

4.1 SHARED TEMPORAL ARCHITECTURE

The central design constraint is that compressing state ownership must not collapse original-node identity. A quotient-only recurrent state violates this constraint: two original nodes assigned to the same supernode can become indistinguishable despite having different private interaction histories. QRTN therefore factorizes temporal state into a shared high-dimensional component and a small node-private residual. Each original node $u \in V$ at timestamp t is represented by:

$$\mathbf{z}_u(t) = \mathrm{LN}\big(P_{\text{c}}\mathbf{m}_{\pi(u)}(t) + P_{\text{r}}\mathbf{r}_u(t) + P_\phi \boldsymbol{\phi}(\Delta t_u) + P_x \mathbf{x}_u\big)\,, \tag{3}$$

where $\mathrm{LN}(\cdot)$ denotes Layer Normalization (Ba et al., 2016). The components of this representation are defined as follows: Here $\mathbf{m}_{\pi(u)} \in \mathbb{R}^{d_{\text{c}}}$ is shared by a supernode, $\mathbf{r}_u \in \mathbb{R}^{d_{\text{r}}}$ is private to node u,

$\phi(\Delta t_u)$ is a learnable harmonic encoding of elapsed time (da Xu et al., 2020), and $\mathbf{x}_u$ is an optional static feature. Linear projections map these terms to a common hidden width ($d = d_c = 128$, $d_r = 16$ by default). An MLP scores the two endpoint representations and their interaction features. The private term is indispensable: without it, nodes with the same quotient state, features, and elapsed time are indistinguishable before the decoder. The exact separation observation and decoder specification are in Appendix C.

The residual update records endpoint-specific innovations that a shared quotient state cannot retain. After scoring event $e_i = (u_i, v_i, t_i, \mathbf{e}_i)$, node u_i's private residual is updated using a Gated Recurrent Unit (GRU) cell (Cho et al., 2014):

$$\mathbf{r}_{u_i}^{+} = U_r\big(\mathbf{r}_{u_i}^{-},\ \big[\,\mathbf{z}_{v_i}^{-}(t_i) \,\|\, \psi(\mathbf{e}_i) \,\|\, \phi(\Delta t_{u_i}) \,\|\, 1\,\big]\big)\,, \tag{4}$$

where $\psi : \mathbb{R}^{d_e} \to \mathbb{R}^{d_\psi}$ is a linear edge-feature projection layer, and 1 is a source role indicator (0 for the destination update $\mathbf{r}_{v_i}^{+}$). The updater U_r is shared across nodes. Consequently, it can be trained on supernodes and later applied to an arbitrary number of original nodes.

Quotient memories evolve at coarser temporal grains. Within a chronological micro-block of q events, dependency-free residual updates are evaluated in parallel, while any read-after-write dependency is ordered exactly as in the sequential residual reference. Each event produces messages for its endpoint supernodes; messages addressed to supernode c in block $\mathcal{B}$ form $I_c(\mathcal{B})$. At the boundary, each touched supernode receives one GRU update:

$$\mathbf{m}_c^{+} = U_c\left(\mathbf{m}_c^{-}, \frac{1}{|I_c(\mathcal{B})|}\sum_{j \in I_c(\mathcal{B})} \mathbf{a}_j\right). \tag{5}$$

All events in $\mathcal{B}$ read the pre-block quotient state $\mathbf{m}_c^{-}$. This is causal, but later events do not read quotient updates caused earlier in the block. Hence q is an explicit accuracy-throughput control, evaluated in Appendix 13. The schedule and message construction are specified in Appendix C.

4.2 QRTN-M: PERSISTENT MEMORY COMPRESSION

QRTN-M uses the original event stream in Equation 1 from training through test. For event $(u, v, t, \mathbf{e})$, the decoder receives the original identities u and v, while the quotient channel reads $\mathbf{m}_{\pi(u)}$ and $\mathbf{m}_{\pi(v)}$. The private residuals $\mathbf{r}_u$ and $\mathbf{r}_v$ distinguish nodes that share a supernode. The model is optimized once. Validation and test continue from the causally accumulated state, with no parameter transfer, replay stage, or full-graph fine-tuning.

This mode provides persistent compressed state ownership. The high-dimensional state and any quotient-domain propagation remain compressed during inference; only the small private residual remains at original-node resolution. It deliberately preserves the number of raw training events, and its residual updater therefore still executes at original-event resolution.

4.3 QRTN-X: PARAMETER-ONLY COARSE-TO-ORIGINAL TRANSFER

QRTN-X asks whether temporal operators learned entirely on a quotient can operate on an arbitrary original graph. It maps every training event

$$(u_i, v_i, t_i, \mathbf{e}_i) \longmapsto (\pi(u_i), \pi(v_i), t_i, \mathbf{e}_i). \tag{6}$$

Timestamps, event features, repeated interactions, and chronological order are retained. Unlike pair-window aggregation, Equation 6 produces exactly M temporal updates. If $\pi(u_i) = \pi(v_i)$, the event advances quotient and residual state but its link-prediction loss is masked. This retains activity and inter-event timing without teaching the decoder that a quotient self-pair is an informative positive edge.

During coarse training, each supernode owns both a main state and a residual state. The update functions, encoders, projections, and decoder are graph-size independent. After checkpoint selection, QRTN-X creates a new network with identity assignment $\pi(u) = u$, initializes one main and residual state per original node, and copies the learned parameters θ. It does not copy the coarse runtime state. Validation advances the fresh original-node state causally, and test continues from the selected validation state. The original training stream is not replayed.

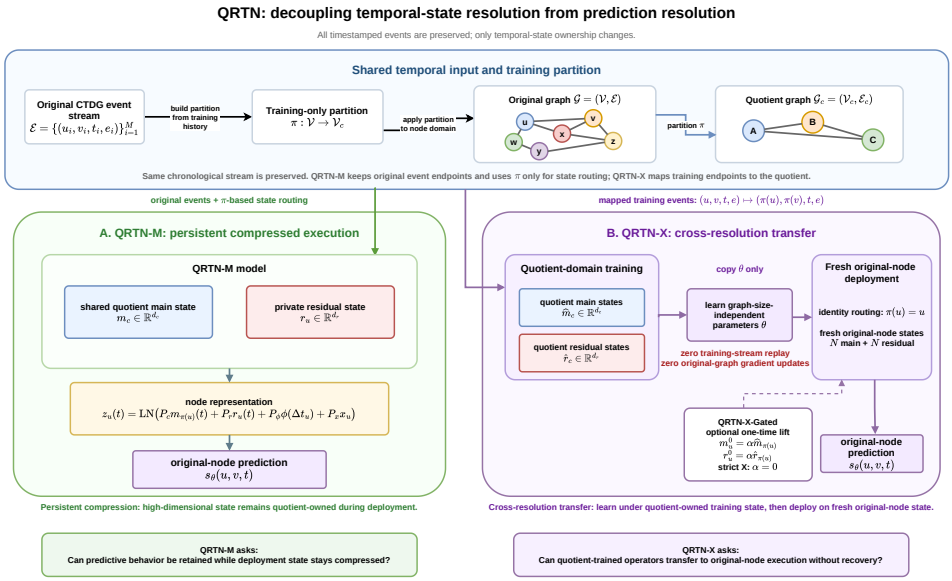

Figure 1: QRTN decouples temporal-state resolution from original-node prediction resolution. A training-only partition is shared by two operating modes: QRTN-M preserves the original event stream while keeping high-dimensional state on quotient nodes and a private residual at each original node; QRTN-X learns operators at quotient resolution and deploys them on fresh original-node state without training-stream replay or original-graph adaptation. QRTN-X-Gated adds an optional one-time bounded lift at the transfer boundary.

This is zero-replay, zero-adaptation transfer: no original-graph gradient step occurs and no original training event is used to reconstruct memory. The trade-off is that inference uses original-node main states. QRTN-X therefore targets quotient-domain learning and cross-resolution operator transfer rather than persistent quotient memory.

4.4 QRTN-X-Gated: State-initialized coarse-to-original transfer

Strict parameter-only transfer deliberately exposes a cold-start problem: the original-node network begins validation with learned update operators but no state summarizing the training period. QRTN-X-Gated tests whether a bounded transfer of already learned runtime state resolves this mismatch without fine-tuning or replay. Let $\widehat{\mathbf{m}}_c$ and $\widehat{\mathbf{r}}_c$ be the quotient states at the selected checkpoint. For an original node assigned during training, the transferred initialization is

$$\mathbf{m}_u^0 = \alpha \widehat{\mathbf{m}}_{\pi(u)}, \qquad \mathbf{r}_u^0 = \alpha \widehat{\mathbf{r}}_{\pi(u)}, \qquad 0 \le \alpha \le 1. \tag{7}$$

Nodes without a training assignment remain at zero state. After Equation 7, the assignment is discarded, the graph uses identity routing, and every original-node state evolves independently. No original training event is replayed and no original-graph gradient is taken. The gate α is fixed using validation data; the benchmark uses $\alpha = 0.25$ and reports strict QRTN-X as the $\alpha = 0$ control.

QRTN-X-Gated is a state-initialized, fine-tuning-free protocol that uses the partition once at the boundary. Its purpose is to separate the behavior of transferred operators from the absence of historical runtime state. It therefore identifies whether a strict-transfer gap is attributable to the transfer boundary itself or to cold state at deployment.

Worked event mapping. An explicit four-event example, including quotient self-loops and the parameter-only transfer boundary, is provided in Appendix C. The main point is that mapping changes endpoints but never removes, merges, or reorders event records.

Table 1: Resource profile at $r = 0.15$, averaged over three seeds. Peak VRAM is measured during training; throughput is chronological inference rate. Total training-time comparisons are omitted because archived runs have non-comparable cache and partition conditions.

Dataset	Operating Point	State Ratio ρ_{M}	Peak VRAM	Throughput
Wikipedia	QRTN-M	0.170	166.2 MB	1115 ev/s
	QRTN-X-Gated	0.844	177.3 MB	1051 ev/s
	QRTN-X	0.844	191.9 MB	1130 ev/s
Reddit	QRTN-M	0.189	140.1 MB	1319 ev/s
	QRTN-X-Gated	0.844	152.8 MB	1299 ev/s
	QRTN-X	0.844	176.2 MB	1413 ev/s
MOOC	QRTN-M	0.192	67.2 MB	873 ev/s
	QRTN-X-Gated	0.844	77.1 MB	916 ev/s
	QRTN-X	0.844	82.9 MB	810 ev/s
LastFM	QRTN-M	0.189	70.9 MB	984 ev/s
	QRTN-X-Gated	0.844	74.4 MB	906 ev/s
	QRTN-X	0.844	79.4 MB	802 ev/s

5 Memory and Time Complexity

QRTN distinguishes four quantities: event volume M, state owners (N or N_c), state width, and high-dimensional quotient-update count U_c. With b bytes per scalar, a full-node memory has $S_{\text{full}} = bN_{\text{full}}(d_{\text{f}} + 1)$ bytes, while QRTN-M has $S_{\text{M}} = b[N_{\text{c}}(d_{\text{c}} + 1) + N_{\text{full}}(d_{\text{r}} + 1)]$, so only its private d_{r}-dimensional term grows with every original node. QRTN-X and QRTN-X-Gated use quotient-owned state during training and original-node state after transfer. All modes retain the complete stream, giving $O(M[C_{\text{enc}} + C_{\text{r}}(d_{\text{r}})] + U_{\text{c}}C_{\text{c}}(d_{\text{c}}) + T_{\pi})$ event-processing cost. Thus QRTN targets recurrent-state ownership and expensive shared updates, not sublinear event processing. Detailed derivations, realized ratios, and operating-point accounting appear in Appendix I.

6 Experimental Design

The experiments test a structural hypothesis: preserving temporal evidence while reducing the resolution of high-dimensional state can retain useful original-node predictive behavior without a second full-graph optimization phase. We report multi-seed runs across four continuous-time dynamic graph benchmarks, comparing the QRTN capabilities against full-node, heuristic, and native external references. Results that share chronological splits and stored negatives are compared directly; native external-toolkit results provide accuracy context under their native protocol.

6.1 Datasets, splits, and metrics

We evaluate Wikipedia, Reddit, MOOC, and LastFM using standard chronological training (70%), validation (15%), and test (15%) splits. All models were trained and benchmarked on an NVIDIA RTX A6000 GPU (48 GB GDDR6, CUDA 12.8). The primary evaluation metrics are Average Precision (AP, computing the area under the precision-recall curve) and Area Under the Receiver Operating Characteristic curve (ROC-AUC) (Poursafaei et al., 2022; Rossi et al., 2020), evaluated under pre-generated, stored negative samples shared identically across all compared models. The default micro-block aggregation size is fixed at $q = 32$ across all primary experiments, balancing accelerator parallelism with causal update resolution.

Table 1 reports accelerator measurements from an NVIDIA RTX A6000 GPU (48 GB GDDR6 VRAM, CUDA 12.8, Driver 570.144). Peak allocated GPU memory is below 200 MB in these configurations, while chronological evaluation throughput ranges from 800 to 1,413 events per

Table 2: Original-node link prediction at $r = 0.15$: mean $\pm$ standard deviation over three seeds under shared chronological splits and stored negatives. † Official TGN new-node AP uses its native reserved-node subset and sampler. FT denotes original-graph fine-tuning. State ratio is relative to 172d full-node TGN memory; QRTN-X "train" is the quotient-domain ratio before transfer.

Dataset	Method	FT	Test AP	New-node AP	State Ratio
Wikipedia	Full TGN (172d)	no	0.9812 ± 0.0009	0.9765 ± 0.0011	1.00
	EdgeBank (unlimited)	no	0.9345 ± 0.0005	0.8993 ± 0.0007	N/A (memory-free)
	QRTN-M (persistent quotient)	no	0.9741 ± 0.0012	0.9709 ± 0.0010	0.170
	QRTN-X (parameter-only)	no	0.9464 ± 0.0069	0.9387 ± 0.0076	0.844 (train: 0.081)
	QRTN-X-Gated (gated lift)	no	0.9632 ± 0.0038	0.9571 ± 0.0051	0.844 (train: 0.081)
Reddit	QRTN-M (persistent quotient)	no	0.9698 ± 0.0019	0.9548 ± 0.0027	0.189
	QRTN-X (parameter-only)	no	0.9067 ± 0.0031	0.8917 ± 0.0065	0.844 (train: 0.103)
	QRTN-X-Gated (gated lift)	no	0.9378 ± 0.0047	0.9274 ± 0.0049	0.844 (train: 0.103)
MOOC	QRTN-M (persistent quotient)	no	0.9207 ± 0.0093	0.9175 ± 0.0086	0.192
	QRTN-X (parameter-only)	no	0.8030 ± 0.0202	0.8040 ± 0.0188	0.844 (train: 0.107)
	QRTN-X-Gated (gated lift)	no	0.8222 ± 0.0151	0.8236 ± 0.0080	0.844 (train: 0.107)
LastFM	QRTN-M (persistent quotient)	no	0.7693 ± 0.0729	0.8228 ± 0.0854	0.189
	QRTN-X (parameter-only)	no	0.8070 ± 0.0155	0.8725 ± 0.0130	0.844 (train: 0.103)
	QRTN-X-Gated (gated lift)	no	0.7832 ± 0.0115	0.8541 ± 0.0110	0.844 (train: 0.103)

second. QRTN-M and the transfer modes all retain the complete chronological event stream; their difference is state ownership and initialization, not a reduction in the number of observed training events. We therefore make no end-to-end training-time claim and do not compare total runtime between QRTN-X and QRTN-X-Gated from the archived traces.

To thoroughly diagnose temporal generalization and prevent metric conflation, we formally evaluate four distinct event subsets during the chronological test stream:

1. **Transductive (Full test):** Chronological evaluation over all positive test events and matched negatives.
2. **Train-inductive (New-node):** Test events where at least one endpoint was completely absent from the training split $\mathcal{E}_{\text{train}}$ (da Xu et al., 2020).
3. **Split-inductive (Unseen-before-split):** Test events where at least one endpoint had zero prior interactions anywhere before the test split boundary.
4. **First-observation:** The initial chronological interaction of each newly appearing node in the test stream. This isolates pure zero-shot / cold-start capability before any node-specific residual update can occur.

Table 2 and the ratio sweeps report results over three random seeds; the in-depth Wikipedia component ablation in Section 7 reports five random seeds.

The main coarsening sweep evaluates $r \in \{0.15, 0.40, 0.70, 1.00\}$. The locked QRTN architecture uses coarse width $d_{\text{c}} = 128$, hidden width 128, time encoder width 16, edge encoder width 32, and private residual width $d_{\text{r}} = 16$. Hyperparameters and early stopping are selected using validation data only; test data are evaluated exactly once on the selected checkpoint.

6.2 Primary benchmark results

Table 2 presents the primary link-prediction results across all four benchmark datasets at the default coarsening ratio $r = 0.15$.

Q1: Predictive retention under persistent state compression. On Wikipedia, QRTN-M reaches 0.9741 ± 0.0012 test AP and 0.9709 ± 0.0010 new-node AP, compared with 0.9812 ± 0.0009 AP for the same-runtime 172-dimensional full-node control, at a persistent-state ratio of 0.170. It therefore retains 99.3% of the full-node AP while removing 83.0% of persistent recurrent-state bytes, for an absolute AP difference of 0.0071. This is the primary QRTN-M operating point: original-node predictions and every chronological event are retained, while high-dimensional temporal state is shared on the quotient. Under historical negatives, it reaches 0.9352 AP versus 0.9366 for the full-node control, while EdgeBank falls to 0.5736 AP (Appendix 12).

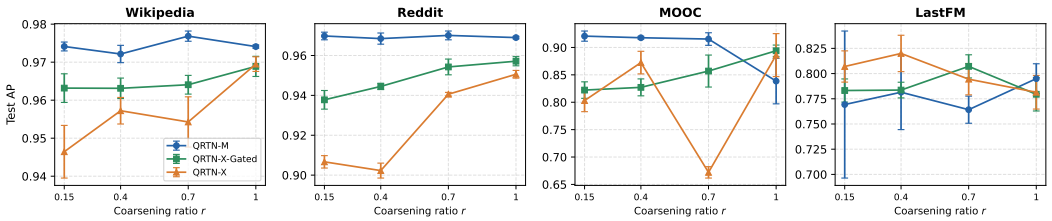

Figure 2: Coarsening-ratio sweep across four datasets. Error bars are standard deviation over three seeds for $r \in \{0.15, 0.40, 0.70, 1.00\}$. The curves identify stable regimes and dataset-specific failure boundaries.

Official full-node TGN baseline. We additionally run the vendored TGN implementation (Rossi et al., 2020) on all four datasets with three seeds, the same chronological splits, and the exact persisted random negative destination for every aggregate validation and test event. Official TGN reaches 0.9787 ± 0.0017 AP on Wikipedia and 0.9788 ± 0.0008 on Reddit, placing QRTN-M 0.0046 and 0.0090 AP lower, respectively. On MOOC and LastFM, the corresponding official-TGN results are 0.8946 ± 0.0214 and 0.6561 ± 0.0208, whereas QRTN-M reaches 0.9207 ± 0.0093 and 0.7693 ± 0.0729. We also report TGN's native new-node AP, marked † in Table 2, but do not treat it as a shared-protocol comparison. These are matched aggregate test comparisons, but they are not a capacity-matched architecture search: TGN uses its native 172-dimensional memory, temporal attention, training-negative sampler, and early-stopping trajectory. We therefore use them as a reproducible full-node baseline rather than claiming uniform architectural superiority. Full per-seed AP/AUC results and the protocol boundary for native new-node evaluation are in Appendix F.

Q2: Cross-resolution operator transfer without recovery. QRTN-X learns shared temporal operators while recurrent training state is owned only by quotient nodes, then deploys them on fresh original-node state with neither training-stream replay nor original-graph gradients. At $r = 0.15$, its quotient-training state is only 0.081–0.107 of the full-node reference across the four datasets (Appendix 11); its original-node inference state is 0.844 because deployment restores a 128-dimensional main state and a 16-dimensional private state per node. QRTN-X is therefore not a persistent inference-memory compression method; it tests whether temporal update operators learned under compressed state ownership survive a change in state resolution. On LastFM, strict transfer reaches 0.8070 ± 0.0155 AP at $r = 0.15$, above QRTN-M's 0.7693 ± 0.0729, and remains stronger at $r = 0.40$ and 0.70. Thus learned quotient-domain dynamics can transfer successfully without a recovery phase, while the other datasets delineate the cold-state boundary.

Q3: Historical-state mismatch at the transfer boundary. Comparing QRTN-X and QRTN-X-Gated holds learned operators fixed while changing state initialization. On Wikipedia, strict transfer achieves 0.9464 ± 0.0069 AP; the attenuated lift ($\alpha = 0.25$) yields 0.9632 ± 0.0038 AP without replay or original-graph gradients. The same ordering holds on Reddit and MOOC, but not on LastFM, where strict transfer is stronger. The gate therefore identifies a dataset-dependent contribution of historical-state mismatch, rather than supplying a universally better transfer mode.

Q3 continued: original-resolution transfer control. The $r = 1$ limiting case isolates the effect of runtime-state history from graph compression. We compare the same-runtime 172-dimensional full-node control with its ordinary carried state and with all runtime memory reset independently before validation and test, alongside QRTN-X with fresh original-node state and QRTN-X-Gated with a one-time $\alpha = 0.25$ lift. On Wikipedia, carrying state improves the full-node control from 0.9739 ± 0.0006 to 0.9795 ± 0.0015 AP. QRTN-X and QRTN-X-Gated achieve 0.9636 ± 0.0023 and 0.9646 ± 0.0014 AP, respectively. On Reddit, the carried-state advantage is larger, 0.9742 ± 0.0031 versus 0.9449 ± 0.0034 AP. Here the gated transfer reaches 0.9433 ± 0.0039 AP, within 0.0016 AP of the reset full-node control, while strict QRTN-X reaches 0.9290 ± 0.0015. This control separates transfer quality from an otherwise confounded comparison against models that retain the entire training-period runtime state. The gate does not materially improve first-observation AP in this test, which is consistent with a shared lifted state not supplying private node identity. Full per-seed metrics and the exact reset protocol appear in Appendix E.

Transfer, cold-start, and ratio boundaries. At $r = 1$, gating improves transfer on Wikipedia, Reddit, and MOOC, with the largest gain on Reddit (0.9290 to 0.9434), while strict transfer is stronger on LastFM. At $r = 0.15$, LastFM favors strict transfer (0.8070 ± 0.0155) over QRTN-M (0.7693 ± 0.0729), whereas QRTN-M is stronger on Reddit and MOOC. First-observation, split-inductive, $r = 1$, gate-sensitivity, and complete ratio results are in Appendices E, G, and H.

Coarsening ratio sweeps. Figure 2 characterizes predictive retention versus state allocation. QRTN-M is stable on Wikipedia and Reddit: Wikipedia AP varies by less than 0.005 from $r = 0.15$ to 1.00 while its state ratio changes from 0.170 to 0.578. MOOC falls from 0.9207 at $r = 0.15$ to 0.8386 at $r = 1.00$; strict transfer also declines at $r = 0.70$. The complete numeric table is in Appendix H.

7 Component Ablations, Baselines, and Robustness Controls

Necessity of the private residual. Removing the residual channel decreases Test AP from 0.9700 ± 0.0017 to 0.8544 ± 0.0023 on the five-seed Wikipedia control. This large drop shows that the gains do not arise from quotient coarsening alone: shared high-dimensional state captures reusable temporal dynamics, but the node-private channel is required to preserve endpoint-specific history. The complete component and heuristic-baseline table is in Appendix 7.

Comparison with heuristic and external baselines. DyGFormer reaches 0.9881 ± 0.0004 AP on Wikipedia and 0.9835 ± 0.0003 on Reddit under its native DyGLib protocol. These full original-graph, memory-free models provide external accuracy context, not direct stored-negative rankings or the recurrent-state object QRTN compresses. Appendix 8 reports all external results and protocol boundaries.

Micro-block throughput scaling. The completed three-seed sweep uses $q \in \{1, 8, 32, 128\}$ on Wikipedia and LastFM (Appendix 13). Here q controls shared quotient-state resolution while private residual chronology remains causal. On Wikipedia, throughput rises from 136.6 to 1418.5 events/s with AP unchanged within seed variation, validating a substantial within-block approximation without measurable degradation; LastFM rises from 167.1 to 868.1 events/s with a dataset-specific accuracy–throughput response.

TGB online state-scaling profile. At 638,486 nodes, the one-million-event `tgbl-coin` prefix verifies that state-ownership savings persist beyond the benchmark graphs: peak VRAM falls from 484.3 MB to 112.0 MB at $r = 0.15$ and 87.2 MB at $r = 0.01$, or $4.3\times$ and $5.6\times$ reductions (Appendix 14). This is an online state-update memory profile, not a training-speed or predictive-accuracy claim.

8 Discussion and Limitations

Continuous-time graph learning need not align event resolution, recurrent-state resolution, and prediction resolution. QRTN changes temporal-memory ownership while retaining the complete chronology and original-node task: QRTN-M compresses persistent inference state, whereas QRTN-X and QRTN-X-Gated compress quotient-domain training state and restore original-node state at deployment. Its target is recurrent-state scaling, not event deletion; the event-processing term remains linear in M. Appendix B provides reproducibility, partition-sensitivity, and execution-equivalence controls.

9 Conclusion

Continuous-time graph learning need not bind event resolution, recurrent-state resolution, and prediction resolution to the same node set. QRTN preserves the complete original-node event stream and prediction task while relocating shared high-dimensional dynamics to quotient resolution and retaining a small private state per node. QRTN-M demonstrates persistent state compression; QRTN-X establishes zero-replay, zero-adaptation transfer of quotient-trained temporal operators; and QRTN-X-Gated isolates historical-state initialization. More broadly, temporal evidence can remain fine-grained even when temporal-state ownership is not.

Reproducibility Statement. An implementation of QRTN and the experiment runners is provided at `https://anonymous.4open.science/r/QRTN-71AF/`. The main text specifies the state-allocation contracts, evaluation tasks, and primary protocol; the appendix provides the complete algorithm, dataset-processing and benchmark specifications, full tables, and supporting controls. The released repository contains the corresponding source code and scripts; datasets and pretrained checkpoints are obtained from their cited public sources and are not redistributed.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Jie Chen, Yousef Saad, and Zechen Zhang. Graph coarsening: from scientific computing to machine learning. *SeMA Journal*, 79(1):187–223, 2022.

Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gulçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1724–1734, 2014.

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? *arXiv preprint arXiv:2302.11636*, 2023.

da Xu, chuanwei ruan, evren korpeoglu, sushant kumar, and kannan achan. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rJeW1yHYwH`.

Julia Gastinger, Shenyang Huang, Mikhail Galkin, Erfan Loghmani, Ali Parviz, Farimah Poursafaei, Jacob Danovitch, Emanuele Rossi, Ioannis Koutis, Heiner Stuckenschmidt, et al. Tgb 2.0: A benchmark for learning on temporal knowledge graphs and heterogeneous graphs. *Advances in neural information processing systems*, 37:140199–140229, 2024.

Jin Guo, Zhen Han, Su Zhou, Jiliang Li, Volker Tresp, and Yuyi Wang. Continuous temporal graph networks for event-based graph data. In *Proceedings of the 2nd Workshop on Deep Learning on Graphs for Natural Language Processing (DLG4NLP 2022)*, pp. 22–29, 2022.

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. *Advances in Neural Information Processing Systems*, 36:2056–2073, 2023.

Zengfeng Huang, Shengzhong Zhang, Chong Xi, Tang Liu, and Min Zhou. Scaling up graph neural networks via graph coarsening. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 675–684, 2021.

Antonin Joly, Nicolas Keriven, and Aline Roumy. Core-gnn: Multilevel message passing on coarsened graphs. *arXiv preprint arXiv:2608.02128*, 2026.

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

Guoming Li, Jian Yang, Xukun Wang, Zixiao Wang, Shangsong Liang, and Yifan Chen. Remedying coarsening-based gnn training under heterophily via adaptive complementary enhancement. *arXiv preprint arXiv:2607.21885*, 2026.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Andreas Loukas. Graph reduction with spectral and cut guarantees. *Journal of Machine Learning Research*, 20(116):1–42, 2019.

Yang Meng and Yuting Liu. Neu: Continuous-time memory evolution for temporal interaction graph networks. *Neural Networks*, pp. 108998, 2026.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1ldO2EFPr.

Farimah Poursafaei, Shenyang Huang, Kellin Pelrine, and Reihaneh Rabbany. Towards better evaluation for dynamic link prediction. *Advances in Neural Information Processing Systems*, 35: 32928–32941, 2022.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *International conference on learning representations*, 2019.

Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. Towards better dynamic graph learning: New architecture and unified library. *Advances in Neural Information Processing Systems*, 36:67686–67700, 2023.

Hongkuan Zhou, Da Zheng, Israt Nisa, Vasileios Ioannidis, Xiang Song, and George Karypis. Tgl: a general framework for temporal gnn training on billion-scale graphs. *Proc. VLDB Endow.*, 15 (8):1572–1580, April 2022. ISSN 2150-8097. doi: 10.14778/3529337.3529342. URL https://doi.org/10.14778/3529337.3529342.

A LLM USAGE DISCLOSURE

In this work, generative AI tools were used only as writing assistants for textual refinement, organization, and readability improvement, and for generating and revising figure and diagram drafts. They were later used to identify possible errors or inconsistencies in the manuscript that could have been missed during manual review. All suggested improvements were evaluated by the authors and implemented manually. Generative AI tools were not used to generate benchmark data, produce scientific results, or make final scientific decisions. No language model is an author. The authors take complete responsibility for the final contents of this paper, including all text, claims, figures, and other artifacts produced with AI assistance.

B DETAILED PROTOCOL AND PARTITION CONSTRUCTION

Related-work context. Temporal link-prediction results depend on chronological splits, negative construction, and new-node definitions (Poursafaei et al., 2022). TGB and TGB 2.0 emphasize reproducible negative protocols and realistic temporal scale (Huang et al., 2023; Gastinger et al., 2024). QRTN follows this principle by persisting validation and test candidates, using candidate-independent routing before scoring, and reporting first-observation events separately from broader inductive subsets. Related temporal architectures also include continuous-time memory evolution, temporal neural ODEs, and scalable mailbox systems (Meng & Liu, 2026; Guo et al., 2022; Zhou et al., 2022); QRTN changes recurrent-state ownership rather than replacing these systems techniques.

Negative candidate policies. For each positive (u_i, v_i, t_i), a negative destination v_i^- is sampled from a candidate pool $\mathcal{D} \subseteq V$ before scoring. The primary protocol samples uniformly from $\mathcal{D} \setminus \{v_i\}$ and persists each validation and test draw. In bipartite streams, $\mathcal{D}$ is restricted to the destination role. Historical negatives are sampled from destinations previously observed with source u_i before t_i, falling back to the random pool when none exist. Inductive negatives are sampled from nodes absent from the training split. Historical partner sets are updated only after their associated event, preserving causality.

Realized ratio and partition. Writing $\gamma = N_{\text{train}}/N_{\text{full}}$, the requested supernode count is $N_{\text{c}} = \lceil rN_{\text{train}} \rceil$ and the realized full-universe ratio is $r' = N_{\text{c}}/N_{\text{full}} = \gamma r$ up to rounding. The partition is constructed only from $\mathcal{E}_{\text{train}}$: temporal-affinity clusters are formed within overlapping training windows and merged to the target count. Source and destination roles are clustered separately for bipartite streams. Nodes absent from training receive a deterministic assignment at evaluation. A mapped self-loop updates recurrent state but is excluded from the ranking loss and batch-normalization denominators; thus it preserves event timing without becoming a trivial positive target.

Reproducibility controls. On the Wikipedia partition control, the tested temporal-affinity variants yield AP between 0.9692 and 0.9718; this is limited sensitivity evidence for that family and dataset, not a general partition-robustness claim. A deterministic CPU rerun reproduces the logged loss trajectory, event-level logits, states, and gradients for the designated configuration. This is an execution-equivalence check rather than another accuracy result.

C Method Details and Worked Mapping

The decoder applies an MLP to $[\mathbf{z}_u \| \mathbf{z}_v \| \mathbf{z}_u \odot \mathbf{z}_v \| |\mathbf{z}_u - \mathbf{z}_v|]$. Before LayerNorm, two nodes in the same supernode with equal features and elapsed times differ only through $P_{\text{r}}(\mathbf{r}_u - \mathbf{r}_v)$. Thus a quotient-only construction cannot encode different private event histories under these conditions; this is an architectural observation, not an accuracy guarantee.

Within a micro-block, an event message to source supernode c_u is

$$\mathbf{a}_{u \to c_u} = \text{MLP}_{\text{c}}\big([\mathbf{z}_u^- \| \mathbf{z}_v^- \| \psi(\mathbf{e}) \| \boldsymbol{\phi}(\Delta t_{c_u}) \| 1]\big)\,, \tag{8}$$

with the destination message defined symmetrically. A topological dependency schedule batches only events whose private-state reads do not depend on each other. Its layer writes are committed before dependent layers execute. Hence private residual state matches the sequential event order, while quotient state is intentionally aggregated once per block.

For a concrete mapping, let $\pi(a) = \pi(b) = A$ and $\pi(c) = \pi(d) = B$. The original events $(a, b, t_1), (a, c, t_2), (b, c, t_3), (c, d, t_4)$ map respectively to $(A, A, t_1), (A, B, t_2), (A, B, t_3), (B, B, t_4)$. All four records remain distinct and ordered. The two self-loops update state but are masked from the ranking loss; after quotient training, QRTN-X copies only learned parameters into fresh states for a, b, c, d.

D Protocol Checklist

Every reported result records the dataset and split fingerprints, stored-negative fingerprint, partition fingerprint, resolved device, seed, batch size, event-engine version, routing policy, checkpoint path, and exact state accounting. A run qualifies as fine-tuning-free only if it performs no gradient step on the original training graph after quotient training. QRTN-X additionally forbids original-stream replay and coarse-memory lifting. QRTN-X-Gated forbids replay and gradients but records the transfer gate, assignment fingerprint, number of initialized nodes, and transferred-state bytes.

E Runtime-State Initialization Fairness

Tables 3 and 4 report every completed seed from the $r = 1$ initialization control. The full-node rows use the 172-dimensional full-node control implemented in the QRTN runtime, rather than the separately implemented official TGN baseline. "Carried" retains the training and validation stream state under the ordinary chronological protocol. "Reset" sets runtime state to zero independently before validation and before test. QRTN-X also begins each held-out stream with fresh original-node memory. QRTN-X-Gated independently applies its selected training-state lift before validation and before test, so validation events never warm-start test.

F Official Full-Node TGN Baseline

Table 5 reports every completed seed from the direct, vendored TGN implementation (Rossi et al., 2020). It uses the same 70/15/15 chronological split as QRTN. For aggregate validation and test, the

Table 3: Wikipedia runtime-state initialization fairness at $r = 1$. Each row is one completed seed. All held-out metrics use the same chronological splits and stored negatives.

Method	Seed	Val. AP	Test AP	Test AUC	New-node AP	First-observation AP
Full-node 172d, carried	0	0.9822	0.9783	0.9759	0.9738	0.8138
Full-node 172d, carried	1	0.9818	0.9790	0.9760	0.9755	0.8310
Full-node 172d, carried	2	0.9826	0.9812	0.9787	0.9766	0.8129
Full-node 172d, reset	0	0.9775	0.9732	0.9692	0.9701	0.8171
Full-node 172d, reset	1	0.9775	0.9743	0.9704	0.9723	0.8273
Full-node 172d, reset	2	0.9778	0.9742	0.9705	0.9702	0.8194
QRTN-X, fresh	0	0.9672	0.9617	0.9569	0.9581	0.8087
QRTN-X, fresh	1	0.9704	0.9662	0.9616	0.9638	0.8108
QRTN-X, fresh	2	0.9672	0.9629	0.9579	0.9580	0.7663
QRTN-X-Gated, $\alpha = 0.25$	0	0.9694	0.9641	0.9595	0.9602	0.8046
QRTN-X-Gated, $\alpha = 0.25$	1	0.9709	0.9662	0.9617	0.9626	0.8093
QRTN-X-Gated, $\alpha = 0.25$	2	0.9695	0.9635	0.9586	0.9567	0.7834

Table 4: Reddit runtime-state initialization fairness at $r = 1$. Each row is one completed seed. All held-out metrics use the same chronological splits and stored negatives.

Method	Seed	Val. AP	Test AP	Test AUC	New-node AP	First-observation AP
Full-node 172d, carried	0	0.9712	0.9757	0.9738	0.9649	0.8677
Full-node 172d, carried	1	0.9666	0.9707	0.9690	0.9563	0.8376
Full-node 172d, carried	2	0.9706	0.9763	0.9747	0.9640	0.8408
Full-node 172d, reset	0	0.9470	0.9454	0.9365	0.9418	0.8654
Full-node 172d, reset	1	0.9424	0.9413	0.9322	0.9388	0.8394
Full-node 172d, reset	2	0.9462	0.9480	0.9405	0.9432	0.8472
QRTN-X, fresh	0	0.9292	0.9273	0.9189	0.9233	0.8518
QRTN-X, fresh	1	0.9328	0.9304	0.9236	0.9243	0.8583
QRTN-X, fresh	2	0.9288	0.9292	0.9221	0.9237	0.8397
QRTN-X-Gated, $\alpha = 0.25$	0	0.9436	0.9403	0.9358	0.9253	0.8563
QRTN-X-Gated, $\alpha = 0.25$	1	0.9455	0.9419	0.9384	0.9242	0.8564
QRTN-X-Gated, $\alpha = 0.25$	2	0.9478	0.9478	0.9447	0.9328	0.8405

TGN evaluator consumes the exact pre-generated random negative destination arrays used by QRTN, in chronological order, and computes AP/AUC over the concatenated held-out event set. TGN retains its native random negative sampler during optimization. Its native new-node AP/AUC are included for completeness, but use a reserved-node subset and native negative sampler that differ from QRTN's shared new-node protocol.

G Transfer Capacity Controls

Table 6 extends the $r = 1$ transfer endpoint to all four datasets. It holds the graph representation at original-node resolution and therefore separates the effect of the transfer mechanism from compression.

H Supplementary Experiments

The following numerical tables support the sensitivity and systems controls summarized in the main text. They are collected here to keep the central benchmark comparison focused on Table 2.

Table 5: Official full-node TGN, per seed. Aggregate Test AP/AUC use the shared split and stored random negatives; native new-node metrics use TGN's reserved-node subset and sampler.

Dataset	Seed	Test AP	Test AUC	Native new-node AP	Native new-node AUC
Wikipedia	0	0.9801	0.9771	0.9644	0.9596
Wikipedia	1	0.9792	0.9761	0.9636	0.9589
Wikipedia	2	0.9768	0.9749	0.9634	0.9586
Reddit	0	0.9779	0.9761	0.9553	0.9535
Reddit	1	0.9792	0.9776	0.9576	0.9557
Reddit	2	0.9792	0.9772	0.9565	0.9550
MOOC	0	0.9122	0.9240	0.8948	0.9056
MOOC	1	0.8708	0.8900	0.8532	0.8681
MOOC	2	0.9006	0.9146	0.8808	0.8955
LastFM	0	0.6574	0.6745	0.7477	0.7523
LastFM	1	0.6762	0.6883	0.6377	0.6693
LastFM	2	0.6347	0.6656	0.7078	0.7249

I State Accounting and Complexity Details

I.1 Persistent-state accounting

The main-text expressions exclude parameters because parameter count is independent of the number of nodes for the compared recurrent operators. They also exclude raw event storage, which is $O(Md_e)$ and shared by QRTN modes when features are materialized. Temporary activation memory depends on batch size, causal-layer width, and implementation, so it is measured rather than folded into persistent-state formulas.

Let b be bytes per state scalar, d_{f} the full-node memory width, and $d_{\mathrm{c}}, d_{\mathrm{r}}$ the main and residual widths. A full-node model requires

$$S_{\mathrm{full}} = bN_{\mathrm{full}}(d_{\mathrm{f}} + 1). \tag{9}$$

For QRTN-M,

$$S_{\mathrm{M}} = b\left[N_{\mathrm{c}}(d_{\mathrm{c}} + 1) + N_{\mathrm{full}}(d_{\mathrm{r}} + 1)\right], \qquad \rho_{\mathrm{M}} = \frac{r'(d_{\mathrm{c}} + 1) + (d_{\mathrm{r}} + 1)}{d_{\mathrm{f}} + 1}, \tag{10}$$

where $r' = N_{\mathrm{c}}/N_{\mathrm{full}} = \gamma r$ and $\gamma = N_{\mathrm{train}}/N_{\mathrm{full}}$. The exact dataset-specific quantities used for the reported state ratios appear in Table 15.

Table 15: Dataset statistics, training-active node fractions $\gamma = N_{\mathrm{train}}/N_{\mathrm{full}}$, and supernode counts at nominal coarsening ratio $r = 0.15$. These quantities determine the realized QRTN-M state ratio through Equation 10.

Dataset	Total Nodes N	Train Nodes N_{train}	Fraction γ	N_{c} ($r = 0.15$)	Total Events M	Feat. d_e
Wikipedia	8,228	5,293	0.6433	794	124,821	172
Reddit	9,450	7,698	0.8146	1,155	100,340	172
MOOC	7,145	6,015	0.8418	902	351,010	4
LastFM	1,981	1,612	0.8137	242	1,110,689	2

Table 6: All-dataset $r = 1$ transfer endpoint: mean $\pm$ sample standard deviation over three seeds. Strict QRTN-X uses fresh original-node state; Gated adds the one-time lift.

Dataset	Strict QRTN-X Test AP	Gated QRTN-X Test AP	Strict New-node AP	Gated New-node AP
Wikipedia	0.9636 ± 0.0022	0.9653 ± 0.0013	0.9602 ± 0.0033	0.9605 ± 0.0029
Reddit	0.9290 ± 0.0016	0.9434 ± 0.0039	0.9237 ± 0.0005	0.9275 ± 0.0047
MOOC	0.8589 ± 0.0196	0.8710 ± 0.0078	0.8670 ± 0.0179	0.8768 ± 0.0070
LastFM	0.7937 ± 0.0094	0.7858 ± 0.0041	0.8632 ± 0.0084	0.8564 ± 0.0037

Table 7: Five-seed Wikipedia component and baseline context at $r = 0.15$. Values are mean $\pm$ sample standard deviation under shared splits and stored negatives.

Method	Seeds	Test AP	Test AUC	New-node AP
Full-node TGN (172d)	5	0.9812 ± 0.0009	0.9790 ± 0.0011	0.9765 ± 0.0011
QRTN-M	5	0.9700 ± 0.0017	0.9669 ± 0.0017	0.9663 ± 0.0015
Coarse-only ablation	5	0.8544 ± 0.0023	0.8456 ± 0.0017	0.8749 ± 0.0042
EdgeBank (unlimited)	5	0.9345 ± 0.0005	0.9374 ± 0.0002	0.8993 ± 0.0007
EdgeBank (time-window)	5	0.9166 ± 0.0004	0.9176 ± 0.0002	0.8905 ± 0.0003
Full TGN (legacy)	5	0.9372 ± 0.0038	0.9266 ± 0.0039	0.9260 ± 0.0044
DyGFormer	5	0.9871 ± 0.0011	0.9849 ± 0.0015	0.9788 ± 0.0016
GraphMixer	5	0.9471 ± 0.0025	0.9441 ± 0.0023	0.9318 ± 0.0033
TGAT	5	0.9436 ± 0.0020	0.9370 ± 0.0022	0.9330 ± 0.0020

Table 8: Native DyGLib external baselines. They use DyGLib's native sampler and reserved-node split, so are not directly ranked against Table 2. Values are mean $\pm$ standard deviation over completed runs.

Dataset	Model	Test AP	Test AUC	Native new-node AP	Native new-node AUC
Wikipedia	DyGFormer	0.9881 ± 0.0004	0.9861 ± 0.0005	0.9804 ± 0.0003	0.9775 ± 0.0006
	GraphMixer	0.9594 ± 0.0013	0.9545 ± 0.0012	0.9461 ± 0.0023	0.9399 ± 0.0023
	TGAT	0.9197 ± 0.0016	0.9149 ± 0.0018	0.9100 ± 0.0035	0.9025 ± 0.0047
Reddit	DyGFormer	0.9835 ± 0.0003	0.9808 ± 0.0001	0.9734 ± 0.0005	0.9686 ± 0.0008
	GraphMixer	0.9526 ± 0.0006	0.9498 ± 0.0003	0.9197 ± 0.0006	0.9157 ± 0.0010
MOOC	DyGFormer	0.8204 ± 0.0961	0.8233 ± 0.0983	0.7970 ± 0.1288	0.8041 ± 0.1279
	GraphMixer	0.8263 ± 0.0004	0.8393 ± 0.0002	0.8123 ± 0.0008	0.8267 ± 0.0002
LastFM	DyGFormer	0.9273 ± 0.0015	0.9286 ± 0.0012	0.9403 ± 0.0010	0.9393 ± 0.0010
	GraphMixer	0.7540 ± 0.0014	0.7359 ± 0.0008	0.8176 ± 0.0029	0.8038 ± 0.0004

For QRTN-X and QRTN-X-Gated, the quotient-training and original-node inference state sizes are

$$S_{\mathrm{X,train}} = bN_{\mathrm{c}}(d_{\mathrm{c}} + d_{\mathrm{r}} + 2), \qquad S_{\mathrm{X,infer}} = bN_{\mathrm{full}}(d_{\mathrm{c}} + d_{\mathrm{r}} + 2). \tag{11}$$

Consequently, their inference ratio is $(d_{\mathrm{c}} + d_{\mathrm{r}} + 2)/(d_{\mathrm{f}} + 1) = 146/173 = 0.844$ under the reported dimensions, while the quotient-training ratio is $r' \cdot 146/173$. QRTN-X-Gated adds one state lift of $O(N_{\mathrm{full}}(d_{\mathrm{c}} + d_{\mathrm{r}}))$ at transfer.

Table 16: State and transfer accounting for the three QRTN operating points. M denotes raw training events.

Property	QRTN-M	QRTN-X	QRTN-X-Gated
Original training events retained	M	M	M
Original labels used during training	yes	no, quotient labels	no, quotient labels
High-dimensional training state owners	N_{c}	N_{c}	N_{c}
Private training residual owners	N	N_{c}	N_{c}
Parameter transfer	no	yes	yes
Runtime-state transfer	no	no	gated lift
Assignment used at transfer	no	no	yes, once
Original training-stream replay	no	no	no
Original-graph gradient steps	no second phase	none	none
High-dimensional inference state owners	N_{c}	N	N
Persistent quotient-memory compression	yes	training only	training only

Table 9: Transfer-gate sensitivity at coarsening ratio $r = 0.15$ over three seeds. The table compares strict transfer, the attenuated default, and full state lifting across aggregate and cold-start subsets.

Dataset	Transfer Gate α	Test AP	First-Obs AP	Unseen-Split AP
Wikipedia	$\alpha = 0$	0.9464 ± 0.0069	0.7562 ± 0.0215	0.9239 ± 0.0100
	$\alpha = 0.25$	0.9632 ± 0.0038	0.7848 ± 0.0270	0.9482 ± 0.0094
	$\alpha = 1$	0.9627 ± 0.0020	0.7856 ± 0.0146	0.9514 ± 0.0049
Reddit	$\alpha = 0$	0.9067 ± 0.0031	0.8541 ± 0.0121	0.8802 ± 0.0065
	$\alpha = 0.25$	0.9378 ± 0.0047	0.8304 ± 0.0241	0.8996 ± 0.0035
	$\alpha = 1$	0.9363 ± 0.0039	0.8288 ± 0.0116	0.8967 ± 0.0072
MOOC	$\alpha = 0$	0.8030 ± 0.0202	0.9498 ± 0.0093	0.7964 ± 0.0233
	$\alpha = 0.25$	0.8222 ± 0.0151	0.9563 ± 0.0087	0.8339 ± 0.0139
	$\alpha = 1$	0.8282 ± 0.0364	0.9535 ± 0.0074	0.8278 ± 0.0464
LastFM	$\alpha = 0$	0.8070 ± 0.0155	0.6484 ± 0.0170	0.9574 ± 0.0062
	$\alpha = 0.25$	0.7832 ± 0.0115	0.5528 ± 0.0054	0.9432 ± 0.0069
	$\alpha = 1$	0.7872 ± 0.0154	0.5364 ± 0.0211	0.9505 ± 0.0054

Table 10: Cold-start and inductive link prediction at $r = 0.15$ over three seeds. First observation has zero prior private state; unseen-before-split requires an endpoint unseen before the test boundary.

Dataset	Operating Point	Full Test AP	First-Observation AP	Unseen-Before-Split AP
Wikipedia	QRTN-M	0.9741 ± 0.0012	0.8004 ± 0.0047	0.9636 ± 0.0014
	QRTN-X-Gated	0.9632 ± 0.0038	0.7848 ± 0.0270	0.9482 ± 0.0094
	QRTN-X	0.9464 ± 0.0069	0.7562 ± 0.0215	0.9239 ± 0.0100
Reddit	QRTN-M	0.9698 ± 0.0019	0.8474 ± 0.0079	0.9326 ± 0.0058
	QRTN-X-Gated	0.9378 ± 0.0047	0.8304 ± 0.0241	0.8996 ± 0.0035
	QRTN-X	0.9067 ± 0.0031	0.8541 ± 0.0121	0.8802 ± 0.0065
MOOC	QRTN-M	0.9207 ± 0.0093	0.9537 ± 0.0038	0.9150 ± 0.0043
	QRTN-X-Gated	0.8222 ± 0.0151	0.9563 ± 0.0087	0.8339 ± 0.0139
	QRTN-X	0.8030 ± 0.0202	0.9498 ± 0.0093	0.7964 ± 0.0233
LastFM	QRTN-M	0.7693 ± 0.0729	0.5450 ± 0.0169	0.9041 ± 0.0864
	QRTN-X-Gated	0.7832 ± 0.0115	0.5528 ± 0.0054	0.9432 ± 0.0069
	QRTN-X	0.8070 ± 0.0155	0.6484 ± 0.0170	0.9574 ± 0.0062

I.2 Event-processing derivation

Let C_{enc} denote per-event encoding and decoding cost, $C_{\mathrm{r}}(d_{\mathrm{r}})$ a residual update, and $C_{\mathrm{c}}(d_{\mathrm{c}})$ a quotient update. For micro-blocks $\mathcal{B}_1, \ldots, \mathcal{B}_K$ of at most q events, define

$$U_{\mathrm{c}} = \sum_{k=1}^{K} \left| \{ c : I_c(\mathcal{B}_k) \neq \varnothing \} \right|. \tag{12}$$

Since each event touches at most two positive endpoint clusters,

$$U_{\mathrm{c}} \leq \left\lceil \frac{M}{q} \right\rceil \min(N_{\mathrm{c}}, 2q) \leq 2M. \tag{13}$$

The resulting cost is

$$T_{\mathrm{QRTN}} = O(M[C_{\mathrm{enc}} + C_{\mathrm{r}}(d_{\mathrm{r}})] + U_{\mathrm{c}} C_{\mathrm{c}}(d_{\mathrm{c}}) + T_{\pi}). \tag{14}$$

For dense gated updates, $C_{\mathrm{r}}(d_{\mathrm{r}}) = O(d_{\mathrm{r}}^2)$ and $C_{\mathrm{c}}(d_{\mathrm{c}}) = O(d_{\mathrm{c}}^2)$ up to input projections. QRTN-X adds no full-graph optimization after quotient training; QRTN-X-Gated adds only the linear state lift before chronological evaluation.

For a static partition constructed over W windows with event sets $\mathcal{E}^{(w)}$, an affinity implementation that sorts candidate edges has preprocessing cost

$$T_{\pi} = O\left(\sum_{w=1}^{W} |\mathcal{E}^{(w)}| \log |\mathcal{E}^{(w)}| + T_{\mathrm{merge}} \right), \tag{15}$$

Table 11: Master coarsening-ratio results for all completed QRTN modes. Values are Test AP $\pm$ standard deviation over three seeds. ρ_{M} is persistent QRTN-M state; $\rho_{\mathrm{X}}^{\mathrm{train}}$ is quotient-domain QRTN-X training state. QRTN-X inference ratio is 0.844; non-varying full-node and external baselines are in Table 2.

Dataset	r	ρ_{M}	$\rho_{\mathrm{X}}^{\mathrm{train}}$	QRTN-M AP	QRTN-X-Gated AP	QRTN-X AP
Wikipedia	0.15	0.170	0.081	0.9741 ± 0.0012	0.9632 ± 0.0038	0.9464 ± 0.0069
	0.4	0.290	0.217	0.9721 ± 0.0023	0.9631 ± 0.0027	0.9572 ± 0.0035
	0.7	0.434	0.380	0.9768 ± 0.0013	0.9641 ± 0.0025	0.9543 ± 0.0066
	1	0.578	0.543	0.9741 ± 0.0004	0.9689 ± 0.0027	0.9695 ± 0.0019
Reddit	0.15	0.189	0.103	0.9698 ± 0.0019	0.9378 ± 0.0047	0.9067 ± 0.0031
	0.4	0.341	0.275	0.9685 ± 0.0028	0.9445 ± 0.0016	0.9023 ± 0.0037
	0.7	0.523	0.481	0.9701 ± 0.0022	0.9543 ± 0.0040	0.9406 ± 0.0009
	1	0.706	0.687	0.9690 ± 0.0006	0.9571 ± 0.0023	0.9506 ± 0.0019
MOOC	0.15	0.192	0.106	0.9207 ± 0.0093	0.8222 ± 0.0151	0.8030 ± 0.0202
	0.4	0.349	0.284	0.9177 ± 0.0025	0.8272 ± 0.0155	0.8724 ± 0.0206
	0.7	0.538	0.498	0.9154 ± 0.0115	0.8569 ± 0.0292	0.6720 ± 0.0104
	1	0.726	0.711	0.8386 ± 0.0415	0.8940 ± 0.0104	0.8860 ± 0.0392
LastFM	0.15	0.189	0.103	0.7693 ± 0.0729	0.7832 ± 0.0115	0.8070 ± 0.0155
	0.4	0.341	0.275	0.7815 ± 0.0371	0.7836 ± 0.0078	0.8200 ± 0.0180
	0.7	0.523	0.481	0.7642 ± 0.0135	0.8072 ± 0.0115	0.7944 ± 0.0155
	1	0.705	0.687	0.7950 ± 0.0148	0.7795 ± 0.0166	0.7815 ± 0.0167

Table 12: Robustness under negative sampling protocols (Wikipedia, $r = 0.15$, 3 seeds). EdgeBank collapses under historical negatives, while QRTN-M remains robust and tracks the full-node model.

Negative Protocol	Method	Test AP	New-node AP
Random	EdgeBank (unlimited)	0.9345 ± 0.0002	0.8995 ± 0.0004
	Full-node TGN (172d)	0.9818 ± 0.0007	0.9773 ± 0.0009
	QRTN-M	0.9699 ± 0.0014	0.9657 ± 0.0016
Historical	EdgeBank (unlimited)	0.5736 ± 0.0000	0.5920 ± 0.0000
	Full-node TGN (172d)	0.9366 ± 0.0036	0.9229 ± 0.0051
	QRTN-M	0.9352 ± 0.0015	0.9249 ± 0.0020
Inductive	EdgeBank (unlimited)	0.9361 ± 0.0003	0.8998 ± 0.0001
	Full-node TGN (172d)	0.9789 ± 0.0025	0.9732 ± 0.0022
	QRTN-M	0.9673 ± 0.0028	0.9624 ± 0.0028

where T_{merge} depends on the target-ratio enforcement rule. This cost is paid once per partition and is reported separately from model training.

J Accelerator Execution

The benchmark implementation supports profiling on an NVIDIA RTX A6000 GPU (48 GB GDDR6 VRAM, Driver 570.144, CUDA 12.8, PyTorch). Accuracy sweeps may schedule independent jobs concurrently; such runs are not used as evidence for isolated wall-clock speed. The resource table is therefore an end-to-end execution profile, while the TGB experiment is an online state-update profile. Neither is presented as a general optimizer-throughput claim.

The implementation separates causal scheduling from neural execution. For each chronological batch, endpoint conflicts are resolved on the host before any event tensor enters the accelerator. The resulting dependency layers are encoded as one contiguous index tensor and transferred once. The model then executes each conflict-free layer through batched projection, decoding, residual GRU, and quotient-message operations. This design avoids a device-to-host synchronization inside the forward pass while preserving the event order of the sequential reference implementation.

Table 13: Three-seed QRTN-M micro-block sweep at $r = 0.15$ with $q \in \{1, 8, 32, 128\}$ on Wikipedia and LastFM.

Dataset	Micro-event block q	Test AP	New-node AP	Throughput (events/sec)
Wikipedia	$q = 1$	0.9655 ± 0.0026	0.9608 ± 0.0026	136.6 ± 17.9
	$q = 8$	0.9659 ± 0.0024	0.9611 ± 0.0035	654.4 ± 138.9
	$q = 32$	0.9658 ± 0.0023	0.9615 ± 0.0028	1074.6 ± 168.5
	$q = 128$	0.9655 ± 0.0025	0.9613 ± 0.0038	1418.5 ± 225.5
LastFM	$q = 1$	0.7605 ± 0.0015	0.8346 ± 0.0002	167.1 ± 42.4
	$q = 8$	0.8161 ± 0.0303	0.8731 ± 0.0188	631.8 ± 35.7
	$q = 32$	0.8368 ± 0.0323	0.8881 ± 0.0214	714.1 ± 55.7
	$q = 128$	0.8442 ± 0.0186	0.8934 ± 0.0136	868.1 ± 69.4

Table 14: Online state-update profile on a one-million-event `tgbl-coin` prefix. All rows use $q = 1024$ without backpropagation; the 172d full-node control is implemented in the QRTN runtime.

Method	Ratio r	Supernodes	State Memory	Peak VRAM	Throughput	Status
Full-node 172d control	1.0 (Full)	638,486	421.4 MB	484.3 MB	4,813 ev/s	✓
QRTN-M ($r = 0.15$)	0.15	95,772	65.2 MB	112.0 MB	3,404 ev/s	✓
QRTN-M ($r = 0.05$)	0.05	31,924	49.3 MB	94.9 MB	3,416 ev/s	✓
QRTN-M ($r = 0.01$)	0.01	6,384	43.0 MB	87.2 MB	3,439 ev/s	✓

Following the data placement used by TGN, the complete edge-feature table is moved to the accelerator once when a run starts. Individual batches gather feature rows on device rather than copying dense feature matrices from NumPy at every optimization step. Node and quotient memories also remain device resident and are updated with indexed tensor operations. Stored-negative generation and online routing remain host-side because they are discrete protocol operations, but their outputs move to the accelerator before neural execution.

We verify execution equivalence against an event-by-event reference engine. The test compares positive and negative logits, original-node residual states, quotient states, last-update times, and parameter gradients under identical weights and events. Resource records include the selected engine, accelerator feature-cache bytes, events per second, and logged run duration. Direct wall-clock conclusions are restricted to measurements with the same isolated execution protocol.