AdaTKG: Adaptive Memory for Temporal Knowledge Graph Reasoning

Anonymous Author(s)

Abstract

Temporal knowledge graphs (TKGs) represent time-stamped relational facts and support a wide range of reasoning tasks over evolving events. However, existing methods produce entity representations that are *static* at the entity level, in that each representation is a function of learned parameters *only* and retains *no trace of the interactions* in which the entity has participated. In this paper, we depart from this static view and propose that each entity be modeled as an ***adaptive process*** whose representation is refined every time the entity participates in a fact. To this end, we propose **AdaTKG**, which maintains a per-entity *memory* that is updated with every observed interaction, with the memory accumulating online and predictions improving as more interactions arrive. Specifically, we instantiate the memory update as a learnable exponential moving average governed by a single *shared* scalar instead of using learnable parameters *for each entity*, enabling AdaTKG to handle entities unseen during training. Extensive experiments confirm consistent gains over TKG baselines, demonstrating the effectiveness of adaptive memory. Code and appendix are available at: https://anonymous.4open.science/r/adatkg_anon-4637/.

1 Introduction

Temporal knowledge graphs (TKGs) organize real-world facts into time-stamped relational quadruples and have become a foundation for reasoning about events over time [2]. They support applications such as event forecasting and risk analysis, each framed as predicting a missing entity or relation at a future timestamp. A substantial body of work has investigated learning temporal dynamics over such graphs [5, 9, 17–19, 32, 37], and these models form the backbone of TKG reasoning pipelines.

Despite this rapid progress, essentially every existing TKG reasoning method shares a common design choice that has been left largely unexamined: the representation of any given entity is *static* at the entity level, in the sense that it is a function of the learned parameters alone and *carries no trace of the interactions in which the entity has participated*. This holds even for the recent line of inductive methods [8, 22, 24, 40], which are designed to handle entities unseen during training.

While recent work has focused on whether a TKG reasoner can handle emerging entities unseen at training time (*transductive* vs. *inductive*), we focus on a different aspect, whether *each entity's representation is refined whenever entity participates in a fact*, formalizing as ***static*** vs. ***adaptive*** distinction. We propose that every entity should be modeled as an *adaptive process* whose representation is refined with each interaction, as illustrated in Figure 1.

To this end, we propose **AdaTKG**, an *Adaptive Inductive* TKG reasoning method that refines each entity's representation through a *per-entity memory* updated whenever the entity participates in a fact. Specifically, we instantiate the memory update as a learnable exponential moving average (EMA) that blends previous memory

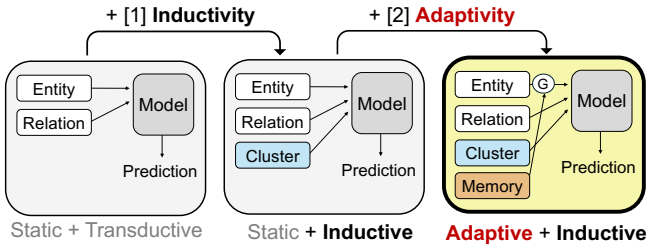

Figure 1: Adaptive TKG reasoning. Our framework enables *adaptivity* through a *memory* refined with each interaction.

with each newly observed interaction via a single learnable scalar. As the parameter is *shared* across entities and *not assigned per entity*, AdaTKG enables *inductive reasoning* on entities unseen during training while remaining efficient. Our main contributions are:

- We argue that TKG reasoning should be *adaptive* at the entity level, with each entity's representation refined through its own interactions rather than read from a fixed lookup. This frames TKG reasoning along a new axis beyond transductive vs. inductive, namely *static* vs. *adaptive*.
- We instantiate this principle with **AdaTKG** in which a learnable EMA blends each entity's past memory with new interactions via a *single shared scalar parameter*. Since no per-entity learnable parameter is introduced, the mechanism is directly applicable to entities unseen during training.
- We conduct extensive experiments showing that AdaTKG consistently outperforms strong TKG baselines, with the gain growing as more interactions of each entity are observed. This provides direct evidence that *adaptivity* at the entity level is a crucial component of TKG reasoning.

2 Related Work

Reasoning on TKGs. Reasoning on temporal knowledge graphs (TKGs) aims to infer missing or future facts by modeling the temporal evolution of entities and relations [2]. Prior work is typically split into *interpolation* [11, 13, 34] and *extrapolation* [5, 17, 18, 37, 42], with more recent approaches leveraging Transformer architectures and language models [9, 19, 32]. These methods, however, operate under a closed-world assumption and initialize the embedding of any new entity randomly, leading to representation collapse [38] whenever an entity emerges without historical interactions.

Inductive reasoning on KGs. Inductive learning on static KGs seeks to generalize to entities unseen during training, with representative methods exploiting subgraph structure [3, 20, 28], meta-learning [4], relation-aware attention [15], or zero-shot transfer across graphs [10]. Extending inductivity to the temporal setting, ALRE-IR [22], zrLLM [8], and POSTRA [24] target unseen relations or entities under limited supervision, and TransFIR [40] recently

Table 1: Representative formulations on TKGs. We argue that the entity should not be *static*, but instead an *adaptive* process whose representation is refined whenever it participates in a fact. We omit the (t_q) superscript for simplicity.

Paradigm		Representation $\mathbf{z}_e$
Representation	Generalization	
Static	Transductive	$\mathbf{z}_e^{\mathrm{ST}} = \mathbf{h}_e$
Static	Inductive	$\mathbf{z}_e^{\mathrm{SI}} = \mathbf{z}_e^{\mathrm{ST}} + \omega_e \cdot \mathbf{c}_{\pi(e)}$
Adaptive	Inductive	$\mathbf{z}_e^{\mathrm{AI}} = (1-g_e) \odot \mathbf{z}_e^{\mathrm{SI}} + g_e \odot \mathbf{m}_e$

establishes the state of the art by transferring *type-level* behavioral prototypes from the same cluster to each emerging entity.

Inductive reasoning without temporal state. A complementary line of inductive methods maintains *no temporal per-entity state*. Static KG foundation models [3, 10, 15, 20, 28] re-derive each entity representation from local graph topology for every query, while PLM- and Transformer-based TKG reasoners [9, 14, 36] cast reasoning as a text-conditioned or sequence-modeling task whose inductive behavior hinges on an entity's textual surface form. In both cases, the representation is recomputed per query rather than refined as the entity accumulates interactions.

Temporal memory and meta-adaptation. Other methods maintain or adapt per-entity state, but not in the forward-only, cold-start inductive regime we target. Meta-learning approaches [4] treat each emerging entity as a meta-task and adapt via gradient updates on a small support set at inference, requiring test-time optimization, whereas dynamic graph neural networks [26, 29, 30, 35, 41] maintain a time-evolving per-node memory but assume a closed entity set observed at training time. In contrast, AdaTKG allocates memory at inference and scores an entity from its very first interaction through a forward-only update, with a strict cold-start guarantee and no test-time optimization.

Across the methods above, entity representations are *static at the entity level*, which are a function of the learned parameters alone. To the best of our knowledge, AdaTKG is the first TKG reasoning method that is *adaptive at the entity level*, maintaining a memory for each entity that is updated with every interaction.

3 Problem Definition

Temporal knowledge graph (TKG). TKG is a sequence of timestamped snapshots

$$\mathcal{G} = \{\mathcal{G}_t\}_{t\in\mathcal{T}}, \qquad \mathcal{G}_t = (\mathcal{E}_{1:t}, \mathcal{R}, \mathcal{F}_t), \tag{1}$$

where $\mathcal{T}$ is the discrete set of timestamps, $\mathcal{E}_{1:t}$ denotes the set of entities observed up to time t, $\mathcal{R}$ is the relation set, and $\mathcal{F}_t \subseteq \mathcal{E}_{1:t} \times \mathcal{R} \times \mathcal{E}_{1:t} \times \{t\}$ is the set of timestamped facts that hold at time t. Each fact is written as a quadruple (e_s, r, e_o, t), with $e_s, e_o \in \mathcal{E}_{1:t}$ the subject and object entities and $r \in \mathcal{R}$ their relation. The cumulative history available up to time t_q is $\mathcal{H}_{t_q} = \bigcup_{i<t_q} \mathcal{F}_i$.

Link prediction task in TKG. Given a query $(e_s, r, ?, t_q)$ whose timestamp t_q lies in the future, the TKG link prediction task is to recover the missing entity from the entity pool by ranking every candidate $e \in \mathcal{E}$ under a scoring function $\phi(\cdot)$ conditioned on the history $\mathcal{H}_{t_q}$. Following the extrapolation protocol, the dataset is split chronologically so that no future information is leaked during training. As test queries are temporally ordered, the protocol breaks the i.i.d. assumption by design, as earlier ground-truth facts enter $\mathcal{H}_{t_q}$ when scoring later queries.

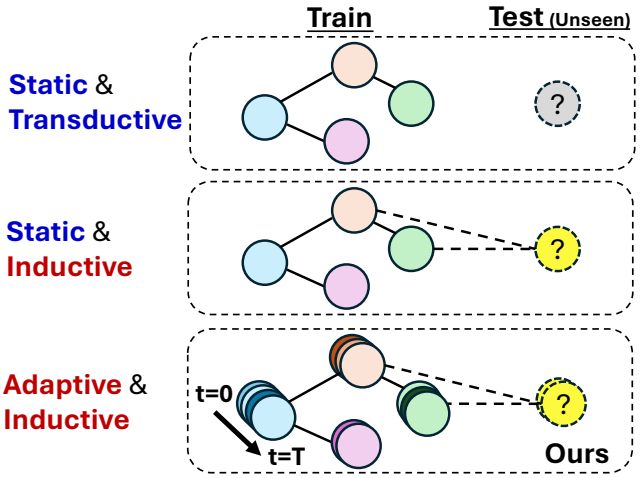

Figure 2: Ours: Adaptive + Inductive. AdaTKG fuses a static inductive prior with a per-entity memory through an adaptive gate, enabling both generalization to unseen entities and refinement upon each interaction.

Emerging entities. Let the first-appearance time of an entity e:

$$t_e(e) \;=\; \min\{\, t \in \mathcal{T} \,|\, e \text{ appears in some fact of } \mathcal{F}_t \,\}. \tag{2}$$

At timestamp t, an entity e is an *emerging entity* if $e \in \mathcal{E}_{1:t} \setminus \mathcal{E}_{1:t-1}$, and its participation at $t_q = t_e(e)$ is accompanied by no prior history. Following previous works [15, 40], we adopt this setting as our primary challenge, focusing on answering queries at the moment an entity enters the graph. This arises frequently in real-world TKGs and makes reasoning difficult as no interaction history is available for the entity [2].

4 From Static TKG to Adaptive TKG

In this section, we cast representative prior methods and our proposal into a common scoring form. Let $\mathbf{h}_e \in \mathbb{R}^d$ denote the *base* representation of entity e and $\mathbf{h}_r \in \mathbb{R}^d$ the embedding of relation r. For a query $(e_q, r_q, ?, t_q)$, the score assigned to a candidate e_o takes the unified form $\phi_{t_q}(e_q, r_q, e_o) \;=\; f\big(\mathbf{z}_{e_q}^{(t_q)},\ \mathbf{h}_{r_q},\ \mathbf{z}_{e_o}^{(t_q)}\big)$, where $f(\cdot)$ is a relational decoder (e.g., ConvTransE [27]) and $\mathbf{z}_e^{(t_q)}$ is the *effective* representation of entity e used for reasoning at time t_q. These formulations differ solely in how $\mathbf{z}_e^{(t_q)}$ is constructed, with comparison show in Table 1.

Paradigm 1: Static + Transductive. Conventional TKG reasoning methods [5, 17, 18, 37] learn a lookup embedding for every entity directly from the training facts as

$$\mathbf{z}_e^{(t_q)} \;=\; \mathbf{h}_e, \qquad \mathbf{h}_e \in \mathbb{R}^d, \quad e \in \mathcal{E}_{1:T_{\mathrm{tr}}}, \tag{3}$$

where T_{tr} is the final training timestamp. The representation is **static** in that $\mathbf{h}_e$ is a function of the learned parameters alone and *carries no trace of the interactions of e itself*, and **transductive** in that

only entities seen during training have a well-defined embedding. However, on emerging entities, $\mathbf{h}_e$ is either undefined or initialized randomly, leading to representation collapse [38].

Paradigm 2: Static + Inductive. To handle emerging entities, recent TKG methods [8, 22, 24] obtain $\mathbf{h}_e$ from an external source (e.g., a pretrained text encoder) and augment it with a *type-level signal* derived from semantically similar entities. Let $\pi(e) \in \{1, \dots, K\}$ be the cluster index of e under a learned codebook and let $\mathbf{c}_{k,t}$ denote the prototype of cluster k at time t, pooled from the interaction patterns of its members [40]. Then, the effective representation is computed using a parametric transfer gate $\Psi(\cdot)$ as

$$\mathbf{z}_e^{(t_q)} = \mathbf{h}_e + \omega_e \cdot \mathbf{c}_{\pi(e),t_q}, \qquad \omega_e = \Psi\big([\mathbf{h}_e \,\|\, \mathbf{c}_{\pi(e),t_q}]\big). \quad (4)$$

This paradigm is **inductive** in that it *allows an emerging entity to still be scored*, since $\mathbf{c}_{\pi(e),t_q}$ is derived from cluster mates rather than from e itself. However, the transfer is **static** in that $\mathbf{z}_e^{(t_q)}$ is still a function of $\mathbf{h}_e$ and $\mathbf{c}_{\pi(e),t_q}$, and *does not incorporate any per-entity state beyond them.*

Paradigm 3: Adaptive + Inductive. We argue that an entity should be treated not as a *static* representation but as an **adaptive** process whose representation is refined every time the entity participates in a fact. To this end, we introduce a *memory* $\mathbf{m}_e^{(\tau)} \in \mathbb{R}^d$ for each entity which is updated via a forward pass whenever e participates in a fact, where τ counts the number of updates. Given a signal $\mathbf{x}_e^{(\tau)}$ summarizing the interaction that triggers the τ-th update of e, the memory is updated as

$$\mathbf{m}_e^{(\tau)} = \mathcal{U}\big(\mathbf{m}_e^{(\tau-1)}, \mathbf{x}_e^{(\tau)}\big), \qquad \mathbf{m}_e^{(0)} = \mathbf{0}, \quad (5)$$

where $\mathcal{U}(\cdot,\cdot)$ is a stateful update operator.

At query time t_q we read $\mathbf{m}_e \equiv \mathbf{m}_e^{(\tau(e,t_q))}$, where $\tau(e,t_q)$ counts the interactions of e strictly before t_q. Note that following the standard TKG protocol [18, 40], training facts are presented in chronological order of t. Memory is thus updated *chronologically*, so no future information leaks into a query. Concretely, $\mathbf{x}^{(\tau)}e$ is built only from *observed* facts of e, so the update never sees the ground-truth object. For an emerging entity e^*, $\mathbf{m}_{e^*} = \mathbf{0}$ and the gate is zero-masked, so AdaTKG reduces exactly to TransFIR (Corollary 1, Appendix E). The effective representation is then produced by an *adaptive gate* $g_e^{(t_q)} \in [0,1]^d$ that fuses the static embedding from Eq. (4) with the memory as

$$\mathbf{z}_e^{(t_q)} = \big(1 - g_e^{(t_q)}\big) \odot \big(\mathbf{h}_e + \omega_e \cdot \mathbf{c}_{\pi(e),t_q}\big) + g_e^{(t_q)} \odot \mathbf{m}_e. \quad (6)$$

Although the memory is maintained on a per-entity basis and may appear inherently transductive, it can be designed to remain **inductive**, by governing its update through *entity-agnostic* parameters rather than parameters indexed by individual entities, making it applicable to emerging entities.

5 Proposed Method: AdaTKG

5.1 Model Architecture

In this section, we instantiate the components in **Paradigm 3: Adaptive + Inductive** in Section 4:

- **[1] Backbone**: Generates static embedding & interaction signal.
- **[2] Memory update**: Generates memory with interaction signal.
- **[3] Adaptive gate**: Fuses static embedding & memory.

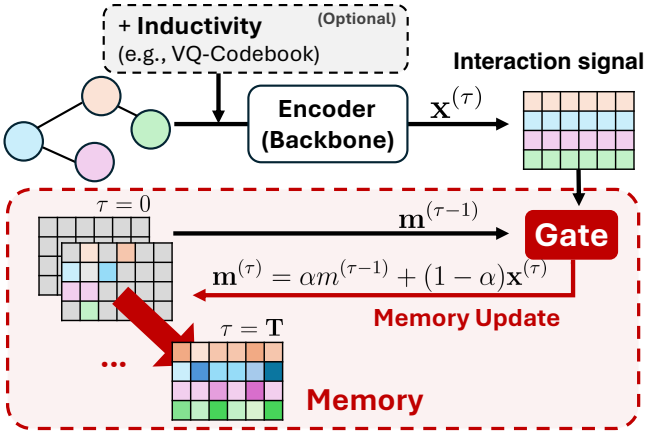

Figure 3: AdaTKG architecture. An encoder produces an interaction signal $\mathbf{x}^{(\tau)}$ which updates the entity's memory $\mathbf{m}^{(\tau)}$ via a learnable EMA. The decay rate α controls how strongly the previous memory is retained relative to the new signal.

[1] Backbone. We adopt the backbone of prior work [40], which for each query $q = (e_q, r_q, ?, t)$ generates the three quantities:

$$\textsc{Backbone}(q) \rightarrow \big(\mathbf{h}_e,\ \omega_e \cdot \mathbf{c}_{\pi(e),t},\ \mathbf{x}_e^{(\tau)}\big),$$

namely *1) static entity embedding* $\mathbf{h}_e$ from a pretrained text encoder, *2) type-level inductive prior* $\omega_e \mathbf{c}_{\pi(e),t}$ from a VQ codebook applied to $\mathbf{h}_e$ (Eq. (4)), and *3) interaction signal* $\mathbf{x}_e^{(\tau)}$ produced by a Transformer-based encoder over the recent interaction chain of e. Details are provided in Appendix D.

[2] Memory update. To make the per-entity memory inductive, it is crucial that ***memory is treated as an internal state, not as a learnable parameter***. That is, the buffer $\{\mathbf{m}_e\}_{e \in \mathcal{E}}$ itself carries no trainable weight; instead, all learnable capacity resides in the *shared* update rule, which is parameterized in an *entity-agnostic* manner. Consequently, AdaTKG's gain does *not* arise from per-entity knowledge stored in $\mathbf{m}_e$, but from the shared update rule learning to exploit a *new information channel* that opens at inference time and thus remains effective even for emerging entities.

To realize this principle, we instantiate the stateful update operator $\mathcal{U}$ of Eq. (5) as a simple yet effective design, a learnable EMA, as shown in Figure 3:

$$\mathbf{m}_e^{(\tau)} = \alpha\, \mathbf{m}_e^{(\tau-1)} + (1-\alpha)\, \mathbf{x}_e^{(\tau)}, \quad (7)$$

with $\mathbf{m}_e^{(0)} = \mathbf{0}$, where $\alpha = \sigma(\rho)$, $\rho \in \mathbb{R}$ is a single learnable scalar shared across all entities, and $\sigma(\cdot)$ is the sigmoid, yielding a decay rate $\alpha \in (0,1)$. Crucially, ρ is the only learnable parameter, and it is entity-agnostic by construction, so the same update rule applies to every entity including emerging entities. Note that EMA is merely one of possible instantiations of $\mathcal{U}$, and ***our focus is not on the EMA operator itself but on endowing each entity with adaptivity through a per-entity memory***. Details of other possible instantiations are provided in Appendix F.

Since the memory is updated online over the training stream, three protocol decisions are required to keep the update strictly

Algorithm 1 Training of AdaTKG (Single-epoch view).

Require: Facts $\mathcal{F}_{\text{tr}}$; Parameters $\rho, \mathbf{W}_g$ (+ backbone Θ_{bb}); Per-entity state $\{(\mathbf{m}_e, \tau_e)\}_{e\in\mathcal{E}}$.

$\alpha \leftarrow \sigma(\rho)$ and $\mathbf{m}_e \leftarrow \mathbf{0},\ \tau_e \leftarrow 0$ **for all** $e \in \mathcal{E}$
for timestamp $t = 1$ **to** T_{tr} **do**
  for each query $q = (e_q, r_q, ?, t)$ from the facts at t **do**
    ▷ *[1] Embeddings (w/ inductivity)*
    $\left(\mathbf{h}_{e_q},\ \omega_{e_q}\cdot\mathbf{c}_{\pi(e_q),t},\ \mathbf{x}_{e_q}^{(\tau_{e_q}+1)}\right) \leftarrow \textsc{Backbone}(q)$
    $\widetilde{\mathbf{h}}_e = \left(\mathbf{h}_e + \omega_e\cdot\mathbf{c}_{\pi(e),t}\right)$
    ▷ *[2] Memory update*
    $\widetilde{\mathbf{m}}_{e_q} \leftarrow \text{stop\_grad}(\mathbf{m}_{e_q})$
    $\mathbf{m}_{e_q} \leftarrow \alpha\,\widetilde{\mathbf{m}}_{e_q} + (1-\alpha)\,\mathbf{x}_{e_q}^{(\tau_{e_q}+1)}$ ▷ Eq. (7)
    ▷ *[3] Adaptive-gate fusion*
    $\mathbf{z}_e^{(t)} \leftarrow \left(1 - g_e^{(t)}\right)\odot\widetilde{\mathbf{h}}_e + g_e^{(t)}\odot\mathbf{m}_e$ for $e\in\{e_q\}\cup\mathcal{E}$ ▷ Eq. (6)
    Score $\phi(e_q, r_q, e_o) = f(\mathbf{z}_{e_q}^{(t)}, \mathbf{h}_{r_q}, \mathbf{z}_{e_o}^{(t)})$
    $\tau_{e_q} \leftarrow \tau_{e_q} + 1$
  end for
end for

causal and the learning signal consistent, with details provided in Appendix E.

- **a) Epoch reset.** Buffer is reset at every epoch start, so $\rho, \mathbf{W}_g$ receive a consistent learning signal.
- **b) Chronological replay.** Facts are replayed in time order, so each query reads EMA of past facts.
- **c) Detached update.** With $\mathbf{m}_e^{(\tau-1)}$ detached before the update, no recurrence gradients flow.

[3] Adaptive gate. The gate $g_e^{(t_q)} \in [0,1]^d$ in Eq. (6) fuses the *static* entity embedding with the *adaptive* memory. Note that $g_e^{(t_q)}$ is a d-dimensional gate, so the fusion in Eq. (6) is a coordinate-wise interpolation and the (cold-start) zero-mask $g_e^{(t_q)} = \mathbf{0}$ refers to the d-dimensional zero vector. We parameterize it by a sigmoid-gated linear projection with explicit zero-masking when the memory buffer is empty, where $\mathbf{W}_g \in \mathbb{R}^{d\times 2d}$ is a learnable projection,

$$g_e^{(t_q)} = \begin{cases} \mathbf{0} & \text{if } \mathbf{m}_e = \mathbf{0}, \\ \sigma\left(\mathbf{W}_g\,[\mathbf{h}_e \,\|\, \mathbf{m}_e]\right) & \text{otherwise.} \end{cases} \tag{8}$$

The zero-masking realizes the hypothesis of Corollary 1 in Appendix E, guaranteeing that AdaTKG reduces to the Static + Inductive paradigm (Eq. (4)) whenever no interaction of e has been observed. For the loss function, we follow that of the previous work [40] without modification.

5.2 Properties of the Adaptive Memory

[Generality] Update-operator agnostic. The adaptive memory framework is not tied to EMA, as any stateful update operator $\mathcal{U}$ that maintains a per-entity state can be plugged in without changing the training protocol. It is important to note that our focus is *not on the choice of EMA itself* but on the *adaptive property* it enables; to highlight this, we additionally instantiate $\mathcal{U}$ as a GRU cell and as a cross-attention readout over a bounded per-entity buffer, with details provided in Appendix F.

[Efficiency] Shared parameters. AdaTKG introduces two learnable components: (i) the EMA module, which contributes a shared scalar ρ that governs the memory dynamics, and (ii) the adaptive gate, which contributes a shared projection $\mathbf{W}_g$ that controls the fusion. Crucially, neither ρ nor $\mathbf{W}_g$ is indexed by the entity, so the buffer $\{\mathbf{m}_e\}$ itself carries no learnable parameter. AdaTKG thus applies uniformly to every entity, remaining effective on those never seen during training.

[Online update] Accumulation at inference. The per-entity memory $\mathbf{m}_e$ accumulates online with every new fact at inference time, so the predictions improve as more interactions arrive. This online dynamic also underlies the consistent gain on emerging/unknown entities as shown in Table 3, where the $\mathbf{h}_e$ provides no entity-specific signal and $\mathbf{m}_e$ supplies a substantial new source of information. Note that $\mathbf{m}_e$ is carried chronologically across train/valid/test phases (resetting only at training-epoch boundaries), so the memory accumulated during training remains available at test time.

[Interpretability] Transition across regimes. The adaptive fusion behaves predictably across the cold-start and warm regimes. With no interaction of e observed, we enforce $g_e^{(t_q)} = \mathbf{0}$ via a zero-mask on the gate, so that Eq. (6) recovers the *Static + Inductive* formulation in Eq. (4). As more interactions of e are observed, $\mathbf{m}_e$ grows informative and the gate shifts weight toward the memory branch.

6 Experiments

Datasets. We evaluate AdaTKG on four widely used temporal knowledge graph benchmarks: ICEWS14, ICEWS18, ICEWS05-15 [1], and GDELT [16]. Each dataset is a chronologically ordered sequence of timestamped quadruples (e_s, r, e_o, t). Details of datasets are discussed in Appendix A.

Evaluation protocol. Following prior work on inductive TKG reasoning [22, 40], we evaluate link prediction using Mean Reciprocal Rank (MRR) and Hits@k with $k \in \{3, 10\}$, computed over the filtered ranking of every candidate entity. While we focus on *emerging* entities following previous works [15, 40], we additionally evaluate on the *unknown* slice to capture entities that, although unseen in training, may accumulate a few interactions before the query timestamp, defined as:

- *Emerging*: Queries on entities that appear for the *very first time* at the query timestamp.
- *Unknown*: Queries on entities that were never observed during training but *may have appeared in validation or earlier in the test stream*, i.e., $s \in \mathcal{V}\setminus\mathcal{V}_{\text{train}}$ or $o \in \mathcal{V}\setminus\mathcal{V}_{\text{train}}$.

Note that these two slices are not disjoint: every *Emerging* query is also *Unknown*. We follow the standard chronological split to prevent leakage of future information into training.

Baselines. We adopt the same baseline set as TransFIR [40], with methods grouped into three categories. *(i) Graph-based*: CyGNet [42], RE-GCN [18], HiSMatch [17], MGESL [23], LogCL [5], HisRes [39], and MLEMKD [25]. *(ii) Path-based*: TLogic [21], TILP [33], ECEformer [9], and GenTKG [19]. *(iii) Static inductive*: CompGCN [31], ICL [14], PPT [36], MorsE [4], InGram [15], and TransFIR [40]. All baseline numbers are taken from the TransFIR paper [40], with TransFIR itself reproduced from its official GitHub release for fair comparison.

Table 2: Comparison with the various TKG reasoning methods across the emerging entities. The table shows the link-prediction performance for queries whose query entity first appears at the query timestamp. Best scores are in bold red, second-best in underlined blue.

	ICEWS14			ICEWS18			ICEWS05-15			GDELT		
	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10
(i) Graph-based												
CyGNet [42]	.0111	.0098	.0202	.0031	.0020	.0047	.0031	.0020	.0048	.0067	.0031	.0147
RE-GCN [18]	.1175	.1263	.2232	.0947	.1004	.1797	.0887	.0961	.1803	.0222	.0209	.0393
HiSMatch [17]	.0284	.0285	.0418	.0055	.0058	.0076	.0242	.0238	.0443	.0159	.0141	.0270
MGESL [23]	.0309	.0361	.0603	.0747	.0809	.1031	.1069	.1166	.1563	.0516	.0471	.0815
LogCL [5]	.1354	.1770	.2273	.0903	.1064	.1548	.1917	.2452	.2855	.0473	.0479	.0973
HisRes [39]	.1169	.1107	.2132	.0445	.0434	.0735	.1325	.1332	.1407	.0416	.0737	.0932
MLEMKD [25]	.0685	.0728	.1303	.0402	.0382	.0831	.0833	.0848	.1717	.0229	.0215	.0436
(ii) Path-based												
TLogic [21]	.0122	.0107	.0257	.0141	.0131	.0262	.0121	.0108	.0285	.0733	.0749	.1131
TILP [33]	.0397	.0471	.1114	.0498	.0669	.1659	.0358	.0374	.1030	.0053	.0025	.0084
ECEformer [9]	.0461	.0496	.0915	.0323	.0680	.0454	.0587	.0642	.1141	.0426	.0410	.0872
GenTKG [19]	—	.0983	.1919	—	.0540	.1512	—	.1105	.1873	—	.0734	.1013
(iii) Static inductive												
CompGCN [31]	.0682	.0906	.1213	.0638	.0745	.1049	.1885	.2103	.2479	.0472	.0791	.0934
ICL [14]	.0252	.0261	.0388	.0639	.0727	.0938	.0254	.0302	.0373	.0277	.0326	.0362
PPT [36]	.0093	.1062	.1716	.0368	.0386	.0650	.0015	.0005	.0022	.0406	.0425	.0764
MorsE [4]	.0136	.0074	.0185	.0078	.0075	.0126	.0381	.0167	.0439	.0039	.0040	.0152
InGram [15]	.0563	.0596	.1138	.0254	.0265	.0518	.0771	.0793	.1454	.0471	.0430	.0847
TransFIR [40]	.1763	.2096	.3413	.1114	.1230	.2252	.2177	.2530	.3708	.1013	.0994	.2131
AdaTKG (Ours)	**.2011**	**.2250**	**.3621**	**.1379**	**.1543**	**.2612**	**.2270**	**.2573**	**.3850**	**.1051**	**.1129**	**.2301**
Δ *over SoTA (%)*	**+14.1**	**+7.3**	**+6.1**	**+23.8**	**+25.4**	**+16.0**	**+4.3**	**+1.7**	**+3.8**	**+3.8**	**+13.6**	**+8.0**

Table 3: Comparison with the SoTA baseline across the emerging/unknown entities. The table shows the link-prediction performance across the two test slices: *Emerging* and *Unknown*.

Entity	Method	ICEWS14			ICEWS18			ICEWS05-15			GDELT		
		MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10
Emerging	TransFIR	.1763	.2096	.3413	.1114	.1230	.2252	.2177	.2530	.3708	.1013	.0994	.2131
	AdaTKG	**.2011**	**.2250**	**.3621**	**.1379**	**.1543**	**.2612**	**.2270**	**.2573**	**.3850**	**.1051**	**.1129**	**.2301**
	Δ *(%)*	**+14.1**	**+7.3**	**+6.1**	**+23.8**	**+25.4**	**+16.0**	**+4.3**	**+1.7**	**+3.8**	**+3.8**	**+13.6**	**+8.0**
Unknown	TransFIR	.2136	.2483	.3901	.1348	.1517	.2591	.3013	.3425	.4706	.1080	.1140	.2261
	AdaTKG	**.2293**	**.2590**	**.4029**	**.1687**	**.1897**	**.3027**	**.3044**	**.3458**	**.4790**	**.1148**	**.1208**	**.2343**
	Δ *(%)*	**+7.4**	**+4.3**	**+3.3**	**+25.1**	**+25.1**	**+16.8**	**+1.0**	**+1.0**	**+1.8**	**+6.3**	**+6.0**	**+3.6**

Implementation details. AdaTKG initializes $\mathbf{h}_e$ from a frozen BERT [7] encoder applied to the entity's textual surface form. Relational scoring is performed with ConvTransE [27], the decoder adopted by recent TKG baselines. We use Adam optimizer [12] and select hyperparameters via validation MRR with early stopping. Results averaged over three random seeds, along with standard deviations, are in Appendix H. Further details are in Appendix C.

6.1 Main Results

To demonstrate the effectiveness of AdaTKG, we present two complementary views of performance:

(1) Comparison with various methods across emerging entities. Following prior work on inductive TKG [15, 40], we focus our comparison on emerging entities. Table 2 demonstrates that our method outperforms all baselines across four benchmarks on link-prediction tasks. This is consistent with our design, where AdaTKG learns *how to integrate new interactions online* rather than storing learnable memory for seen entities, so even unseen entities accumulate useful signal at inference time.

(2) Comparison with SoTA across emerging/unknown entities. Table 3 compares AdaTKG with TransFIR [40] across the two slices of unseen entities (*Emerging*, *Unknown*), jointly characterizing model behavior on cold-start and few-shot regimes. AdaTKG consistently outperforms TransFIR on both slices across all four benchmarks, indicating that the gain extends beyond the strict cold-start setting to entities merely seen rarely during training.

6.2 Ablation Study

To isolate the contribution of each design choice in AdaTKG, we perform a controlled ablation on the emerging slice across all four benchmarks, organized along four axes:

(1) Effectiveness of Adaptive + Inductive paradigm. The four rows in Table 4 traverse the 2×2 paradigm grid: *Static + Inductive* (≡ TransFIR [40]) keeps only the cluster prior, *Adaptive + Transductive*

Table 4: [Ablation 1] Paradigm grid. The 2×2 grid of Static/Adaptive × Transductive/Inductive.

Paradigm		ICEWS14			ICEWS18			ICEWS05-15			GDELT		
		MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10
Static	**Transductive**	.1478	.1795	.3147	.0930	.1016	.2002	.2084	.2517	.3663	.0866	.0835	.1883
Adaptive	**Transductive**	.1612	.1872	.3236	.1172	.1320	.2175	.2129	.2455	.3652	.0894	.0940	.1868
Static	**Inductive**	.1763	.2096	.3413	.1114	.1230	.2252	.2177	.2530	.3708	.1013	.0994	.2131
Adaptive	**Inductive**	**.2011**	**.2250**	**.3621**	**.1379**	**.1543**	**.2612**	**.2270**	**.2573**	**.3850**	**.1051**	**.1129**	**.2100**

Table 5: [Ablation 2] Adaptive gate. Effect of an adaptive gate g_e vs. a constant gate $g = 0.5$.

Gate	ICEWS14			ICEWS18			ICEWS05-15			GDELT		
	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10
w/o Adaptive gate	.1839	.2138	.3544	.1291	.1440	.2426	.2192	.2530	.3850	.0982	.1032	.1976
w/ Adaptive gate	**.2011**	**.2250**	**.3621**	**.1379**	**.1543**	**.2612**	**.2270**	**.2573**	**.3850**	**.1051**	**.1129**	**.2301**
Δ *(%)*	**+9.4**	**+5.2**	**+2.2**	**+6.8**	**+7.2**	**+7.7**	**+3.6**	**+1.7**	**+0.0**	**+7.0**	**+9.4**	**+16.4**

Table 6: [Ablation 3] Update operator $\mathcal{U}$. Three instantiations of the memory update rule.

Update operator		ICEWS14			ICEWS18			ICEWS05-15			GDELT		
		MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10
Base (w/o Adaptivity) [40]		.1763	.2096	.3413	.1114	.1230	.2252	.2177	.2530	.3708	.1013	.0994	.2131
AdaTKG	EMA *(default)*	**.2011**	**.2250**	**.3621**	**.1379**	**.1543**	**.2612**	**.2270**	**.2573**	**.3850**	**.1051**	**.1129**	**.2301**
	Δ *over Base (%)*	**+14.1**	**+7.3**	**+6.1**	**+23.8**	**+25.4**	**+16.0**	**+4.3**	**+1.7**	**+3.8**	**+3.8**	**+13.6**	**+8.0**
	GRU	**.1955**	**.2192**	**.3582**	**.1428**	**.1599**	**.2605**	**.2330**	**.2700**	**.3925**	**.1112**	**.1141**	**.2243**
	Δ *over Base (%)*	**+10.9**	**+4.6**	**+4.9**	**+28.2**	**+30.0**	**+15.7**	**+7.0**	**+6.7**	**+5.9**	**+9.8**	**+14.8**	**+5.3**
	Cross-attention	**.1913**	**.2296**	**.3609**	**.1454**	**.1712**	**.2761**	**.2243**	**.2573**	**.3815**	**.1297**	**.1396**	**.2544**
	Δ *over Base (%)*	**+8.5**	**+9.5**	**+5.7**	**+30.5**	**+39.2**	**+22.6**	**+3.0**	**+1.7**	**+2.9**	**+28.0**	**+40.4**	**+19.4**

Table 7: [Ablation 4] EMA decay parameter. Three parameterizations of the EMA decay rate.

EMA decay rate		ICEWS14			ICEWS18			ICEWS05-15			GDELT		
		MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10
Base (w/o Adaptivity) [40]		.1763	.2096	.3413	.1114	.1230	.2252	.2177	.2530	.3708	.1013	.0994	.2131
Shared scalar	$\alpha = \sigma(\rho)$	**.2011**	**.2250**	**.3621**	.1379	**.1543**	.2612	**.2270**	**.2573**	**.3850**	**.1051**	**.1129**	**.2301**
Per-entity scalar	$\alpha_e = \sigma(\rho_e)$	.1888	.2165	.3525	**.1389**	.1529	**.2615**	.2199	.2562	.3808	.1008	.1056	.2046
Per-dim. vector	$\boldsymbol{\alpha} = \sigma(\boldsymbol{\rho})$	.1885	.2192	.3532	.1358	.1462	.2556	.2129	.2442	.3753	.0990	.1052	.2057

keeps only the per-entity memory, and *Adaptive + Inductive* is the AdaTKG that combines both. Removing the per-entity memory collapses AdaTKG to TransFIR, while removing the cluster prior breaks the cold-start match with TransFIR. The proposed Adaptive + Inductive paradigm achieves the best results, demonstrating the effectiveness of adaptivity in TKG reasoning.

(2) Effectiveness of adaptive gate. As shown in Table 5, replacing the learned gate by the constant $g = 0.5$ removes its ability to track the varying reliability of the memory and degrades performance, confirming that the gate is a necessary mechanism for the graceful cold-start behavior.

(3) Robustness to update operator. We compare three instantiations of the stateful update operator $\mathcal{U}$, the 1) learnable EMA, a 2) GRU cell, and a 3) cross-attention readout over a bounded per-entity buffer. All three retain the adaptive gate of Eq. (8) for fusion, so *only the memory update rule itself differs*. As shown in Table 6, every variant improves over the static baseline [40], indicating that the gain is driven by the *adaptive property* of per-entity memory rather than by any particular implementation. It is important to note that our focus is *not on the choice of EMA itself but on this adaptive property*, and we adopt EMA as a lightweight yet effective realization of $\mathcal{U}$.

(4) Variants of EMA decay parameterization. The default AdaTKG uses a single shared learnable scalar $\alpha = \sigma(\rho)$ for the EMA decay. Two natural relaxations enlarge the parameterization: a 1) *per-entity scalar*[1] $\alpha_e = \sigma(\rho_e)$ that lets each entity learn its own forgetting rate and a 2) *per-dimension vector* $\boldsymbol{\alpha} = \sigma(\boldsymbol{\rho}) \in [0, 1]^d$ that decays each coordinate at a different rate. As shown in Table 7, while all designs outperform the baseline (w/o Adaptivity), the shared scalar outperforms both relaxations, indicating that a *single shared scalar* already suffices for the memory branch.

[1]This breaks inductivity for emerging entities, where we fall back to the cluster-prototype decay $\rho_{\pi(e)}$.

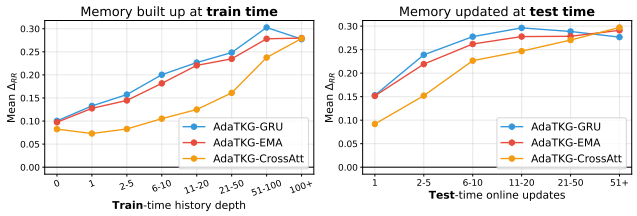

Figure 4: Performance by # interactions at train and test time. We measure the memory's contribution by $\Delta_{RR} = \mathrm{RR}_{\mathrm{full}} - \mathrm{RR}_{\mathrm{zero}}$ (i.e., the difference between the per-query RR of the model w/ and w/o its memory branch) and stratify it along two axes: (Left) Train-time history depth of the subject and the (Right) Test-time online updates accumulated during inference. The increase of Δ_{RR} along both axes, consistent across all three operators, indicates that more observed interactions translate into a *larger memory contribution* and, in turn, into *higher predictive performance*.

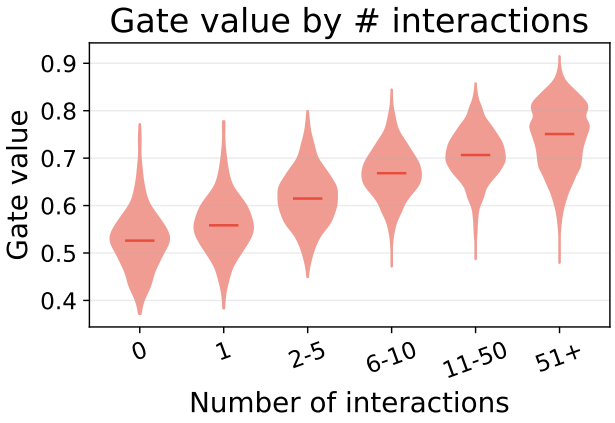

Figure 5: Gate value by # interactions. Distribution of the learned gate $g_e^{(t_q)}$, stratified by the number of interactions.

6.3 Analysis

(1) Effectiveness of memory by # interactions. (Figure 4). To demonstrate that the *per-entity memory is the actual source of AdaTKG's gain*, we score each AdaTKG variant on ICEWS14 in two configurations: 1) with the trained memory branch active ($\mathrm{RR}_{\mathrm{full}}$), and 2) with the memory branch bypassed by setting the gate to zero ($\mathrm{RR}_{\mathrm{zero}}$). The per-query difference $\Delta_{RR} = \mathrm{RR}_{\mathrm{full}} - \mathrm{RR}_{\mathrm{zero}}$ is thus *the performance gain by the memory branch*, and we stratify it along two complementary axes in Figure 4: (Left) train-time history depth of the subject and (Right) test-time online updates accumulated during inference. Δ_{RR} is positive across every bin and grows along both axes for all three update operators, confirming that more interactions translate into a larger memory contribution and a corresponding gain in MRR. Results across all four benchmarks are provided in Appendix J.

(2) Gate activation by # interactions (Figure 5). The adaptive gate $g_e^{(t_q)}$ controls how much of the representation comes from the *static inductive prior* versus the *per-entity memory*, and its trajectory offers a direct window into what the model has learned. As shown in Figure 5 with ICEWS14 and the EMA update operator, the gate value grows with the number of observed interactions. See Appendix K for results across all benchmarks and update operators.

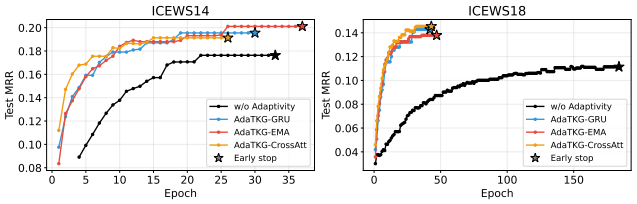

Figure 6: Training curves on test MRR. Per-epoch test emerging MRR for the static-inductive baseline (*w/o Adaptivity*) and the three AdaTKG update operators (*w/ Adaptivity*). AdaTKG update operators yield faster convergence and higher performance.

Table 8: Robustness to training horizon. *Training horizon* ($k\%$) denotes the fraction of the most recent training time window used; AdaTKG's gain over the baseline persists at every horizon and is largest at the shortest.

	Training horizon			
	25%	50%	75%	100%
Base (w/o Adaptivity) [40]	.1089	.1408	.1357	.1763
AdaTKG	**.1435**	**.1676**	**.1727**	**.2011**
Δ over SoTA (%)	**+31.8**	**+19.0**	**+27.3**	**+14.1**

(3) Faster convergence and higher final performance (Figure 6). Figure 6 traces the per-epoch test emerging MRR for the static-inductive baseline (*w/o Adaptivity*) and the three AdaTKG update operators. All three variants 1) *converge in fewer epochs* and to a 2) *higher final MRR* than the baseline. This indicates that adaptivity yields both 1) *training efficiency* and 2) *predictive performance*, and that these gains stem from the memory itself rather than from any specific update operator.

(4) Robustness to training horizon (Table 8). We additionally train both methods on only the last $k\%$ of the training time window to test whether the gain depends on long training history. AdaTKG outperforms TransFIR at every horizon, with the gain *largest at the shortest horizon*. This is consistent with the asymmetric roles of the two branches: the static branch is *parametric* and its quality scales with how much training data it sees, while the memory branch is *non-parametric* and accumulates online from the test stream, so its marginal value is highest when the static prior is weakest.

(5) Sensitivity to memory-related hyperparameters (Figure 7). For each of ICEWS14 and ICEWS18, we sweep the full hyperparameter grid (chain length, hidden dim, # layers, codebook size) and plot the resulting test emerging MRR for every AdaTKG operator against the best static-inductive baseline. Every AdaTKG HP point lies above the baseline reference line on both benchmarks, indicating that the adaptive memory mechanism is *robust to hyperparameter choices*.

Table 9: Efficiency comparison. We compare TransFIR and the three AdaTKG update operators along 1) the number of parameters, 2) the training time per epoch, and 3) the FLOPs per query. AdaTKG delivers a substantial performance gain over TransFIR with only marginal extra computation.

Method		[1] Efficiency						[2] Performance		
		# Parameters (M)		Training time (s/epoch)		FLOPs (M/query)		MRR	H@3	H@10
		Value	Δ (%)	Value	Δ (%)	Value	Δ (%)			
Base (w/o Adaptivity) [40]		41.15	–	56.1	–	399.8	–	.1763	.2096	.3413
AdaTKG	EMA (*default*)	44.10	**+7.2%**	64.1	**+14.3%**	405.7	**+1.5%**	**.2011**	**.2250**	**.3621**
	GRU	47.65	**+15.8%**	67.6	**+20.5%**	408.8	**+2.2%**	**.1955**	**.2192**	**.3582**
	Cross-attention	46.47	**+12.9%**	62.7	**+11.8%**	445.9	**+11.5%**	**.1913**	**.2296**	**.3609**

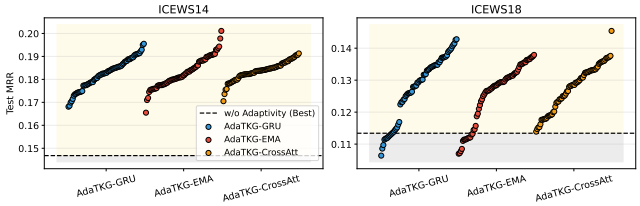

Figure 7: Sensitivity to memory-related hyperparameters. Each point is one hyperparameter setting of AdaTKG, compared against the *w/o Adaptivity* baseline at its best setting (dashed line).

(6) Efficiency analysis (Table 9). We compare AdaTKG with TransFIR, the SoTA static-inductive baseline, on three computational dimensions on ICEWS14: 1) number of parameters, 2) training time per epoch, and 3) FLOPs per query, evaluating all four methods under the same hyperparameter configuration (= AdaTKG-EMA's best HP on ICEWS14) for fair comparison. As shown in Table 9, the three update operators all add only a *small overhead* over the base while *substantially improving performance*, confirming that adaptivity is the source of the gain rather than added model capacity. Among them, we adopt EMA as our default operator as it offers the best balance between efficiency and performance. Comparison on the other three benchmarks is provided in Appendix I.

(7) Qualitative ex: How memory adapts (Figure 8). While Figure 5 reports the *average* gate trajectory, here we trace the gate $g_e^{(t)}$ of two individual entities on ICEWS18 with AdaTKG across their successive test-time appearances. The red trajectory (↑) corresponds to an entity whose gate climbs rapidly, indicating that its observed actions diverge from the *static inductive prior* and the model accordingly leans on the *per-entity memory branch*; the blue trajectory (↓) corresponds to an entity whose gate remains low throughout, indicating that the *static inductive prior* already explains its behavior well and the model accordingly trusts it. This entity-level contrast confirms that the gate is not a function of time or interaction count alone, but of the *entity-specific deviation* from the *static prior*, which is the behavior anticipated by the design.

7 Conclusion

We propose AdaTKG, which shifts TKG reasoning from *static* to *adaptive* representations by equipping every entity with a memory governed by an adaptive gate. Crucially, the memory is treated as an internal state rather than a learnable parameter, with all learnable capacity concentrated in a single shared update rule, making it applicable to unseen entities. Across various datasets, AdaTKG yields consistent improvements, with the gain growing in the number of interactions per entity.

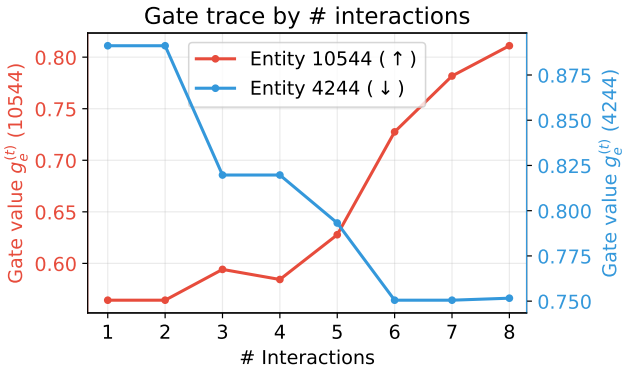

Figure 8: Qualitative gate trace.

Limitation and Future Work. AdaTKG's per-entity memory must be persisted between inference calls, introducing a simple bookkeeping requirement absent in stateless inductive methods. Coupling the memory with a continually adapting codebook lets the static prior itself accommodate slow distribution drift on long-horizon TKGs, which we view as a natural next step.

GenAI Usage Disclosure

Generative AI tools were used in preparing this paper solely for light editing, such as grammar checking and minor wording refinements, and not to generate research ideas, design or conduct experiments, produce results, or write any substantive technical content, all of which was authored and verified by the authors.

Impact Statement

AdaTKG targets a practically consequential setting: reasoning about entities that continuously emerge in temporal knowledge graphs without any training-time footprint, which can broaden analyses in domains such as financial monitoring and public-health surveillance. Because event-based TKGs inherit reporting biases that may propagate into AdaTKG's per-entity memories, its outputs should serve as decision support validated by human experts.

References

[1] Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. ICEWS Coded Event Data. doi:10.7910/DVN/28075

[2] Borui Cai, Yong Xiang, Longxiang Gao, He Zhang, Yunfeng Li, and Jianxin Li. 2023. Temporal knowledge graph completion: a survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. 6545–6553.

[3] Jiajun Chen, Huarui He, Feng Wu, and Jie Wang. 2021. Topology-aware correlations between relations for inductive link prediction in knowledge graphs. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 6271–6278.

[4] Mingyang Chen, Wen Zhang, Yushan Zhu, Hongting Zhou, Zonggang Yuan, Changliang Xu, and Huajun Chen. 2022. Meta-knowledge transfer for inductive knowledge graph embedding. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 927–937.

[5] Wei Chen, Huaiyu Wan, Yuting Wu, Shuyuan Zhao, Jiayaqi Cheng, Yuxin Li, and Youfang Lin. 2024. Local-global history-aware contrastive learning for temporal knowledge graph reasoning. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 733–746.

[6] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. 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)*. 1724–1734.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 4171–4186.

[8] Zifeng Ding, Heling Cai, Jingpei Wu, Yunpu Ma, Ruotong Liao, Bo Xiong, and Volker Tresp. 2024. zrllm: Zero-shot relational learning on temporal knowledge graphs with large language models. In *Proceedings of the 2024 conference of the North American chapter of the association for computational linguistics: Human language technologies (Volume 1: Long papers)*. 1877–1895.

[9] Zhiyu Fang, Shuai-Long Lei, Xiaobin Zhu, Chun Yang, Shi-Xue Zhang, Xu-Cheng Yin, and Jingyan Qin. 2024. Transformer-based reasoning for learning evolutionary chain of events on temporal knowledge graph. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*. 70–79.

[10] Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. 2024. Towards Foundation Models for Knowledge Graph Reasoning. In *The Twelfth International Conference on Learning Representations*.

[11] Alberto Garcia-Duran, Sebastijan Dumančić, and Mathias Niepert. 2018. Learning Sequence Encoders for Temporal Knowledge Graph Completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 4816–4821.

[12] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.

[13] Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. 2020. Tensor Decompositions for Temporal Knowledge Base Completion. In *International Conference on Learning Representations*.

[14] Dong-Ho Lee, Kian Ahrabian, Woojeong Jin, Fred Morstatter, and Jay Pujara. 2023. Temporal Knowledge Graph Forecasting Without Knowledge Using In-Context Learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 544–557.

[15] Jaejun Lee, Chanyoung Chung, and Joyce Jiyoung Whang. 2023. InGram: Inductive knowledge graph embedding via relation graphs. In *International conference on machine learning*. PMLR, 18796–18809.

[16] Kalev Leetaru and Philip A Schrodt. 2013. Gdelt. In *ISA annual convention*, Vol. 2. Citeseer, 1–49.

[17] Zixuan Li, Zhongni Hou, Saiping Guan, Xiaolong Jin, Weihua Peng, Long Bai, Yajuan Lyu, Wei Li, Jiafeng Guo, and Xueqi Cheng. 2022. HiSMatch: Historical Structure Matching based Temporal Knowledge Graph Reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. 7328–7338.

[18] Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. 2021. Temporal knowledge graph reasoning based on evolutional representation learning. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 408–417.

[19] Ruotong Liao, Xu Jia, Yangzhe Li, Yunpu Ma, and Volker Tresp. 2024. GenTKG: Generative Forecasting on Temporal Knowledge Graph with Large Language Models. In *Findings of the Association for Computational Linguistics: NAACL 2024*. 4303–4317.

[20] Shuwen Liu, Bernardo Grau, Ian Horrocks, and Egor Kostylev. 2021. Indigo: Gnn-based inductive knowledge graph completion using pair-wise encoding. *Advances in Neural Information Processing Systems* 34 (2021), 2034–2045.

[21] Yushan Liu, Yunpu Ma, Marcel Hildebrandt, Mitchell Joblin, and Volker Tresp. 2022. Tlogic: Temporal logical rules for explainable link forecasting on temporal knowledge graphs. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 36. 4120–4127.

[22] Xin Mei, Libin Yang, Xiaoyan Cai, and Zuowei Jiang. 2022. An Adaptive Logical Rule Embedding Model for Inductive Reasoning over Temporal Knowledge Graphs. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 7304–7316. doi:10.18653/v1/2022.emnlp-main.493

[23] Shi Mingcong, Chunjiang Zhu, Detian Zhang, Shiting Wen, and Li Qing. 2024. Multi-Granularity History and Entity Similarity Learning for Temporal Knowledge Graph Reasoning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. 5232–5243.

[24] Jiaxin Pan, Mojtaba Nayyeri, Osama Mohammed, Daniel Hernandez, Rongchuan Zhang, Cheng Cheng, and Steffen Staab. 2025. Towards Foundation Model on Temporal Knowledge Graph Reasoning. *arXiv preprint arXiv:2506.06367* (2025).

[25] Ye Qian, Xiaoyan Wang, Fuhui Sun, and Li Pan. 2025. Compressing transfer: Mutual learning-empowered knowledge distillation for temporal knowledge graph reasoning. *IEEE Transactions on Neural Networks and Learning Systems* (2025).

[26] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond*.

[27] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 3060–3067.

[28] Komal Teru, Etienne Denis, and Will Hamilton. 2020. Inductive relation prediction by subgraph reasoning. In *International conference on machine learning*. PMLR, 9448–9457.

[29] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *International conference on machine learning*. PMLR, 3462–3471.

[30] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. DyRep: Learning Representations over Dynamic Graphs. In *International Conference on Learning Representations (ICLR)*.

[31] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P Talukdar. 2020. Composition-based Multi-Relational Graph Convolutional Networks. In *ICLR*.

[32] Jiapu Wang, Sun Kai, Linhao Luo, Wei Wei, Yongli Hu, Alan Wee-Chung Liew, Shirui Pan, and Baocai Yin. 2024. Large language models-guided dynamic adaptation for temporal knowledge graph reasoning. *Advances in Neural Information Processing Systems* 37 (2024), 8384–8410.

[33] Siheng Xiong, Yuan Yang, Faramarz Fekri, and James Clayton Kerce. 2023. TILP: Differentiable Learning of Temporal Logical Rules on Knowledge Graphs. In *The Eleventh International Conference on Learning Representations (ICLR)*.

[34] Siheng Xiong, Yuan Yang, Ali Payani, James C Kerce, and Faramarz Fekri. 2024. Teilp: Time prediction over knowledge graphs via logical reasoning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 38. 16112–16119.

[35] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive Representation Learning on Temporal Graphs. In *International Conference on Learning Representations (ICLR)*.

[36] Wenjie Xu, Ben Liu, Miao Peng, Xu Jia, and Min Peng. 2023. Pre-trained Language Model with Prompts for Temporal Knowledge Graph Completion. In *Findings of the Association for Computational Linguistics: ACL 2023*. 7790–7803.

[37] Yi Xu, Junjie Ou, Hui Xu, and Luoyi Fu. 2023. Temporal Knowledge Graph Reasoning with Historical Contrastive Learning. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, Brian Williams, Yiling Chen, and Jennifer Neville (Eds.). AAAI Press, 4765–4773. doi:10.1609/AAAI.V37I4.25601

[38] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. 2021. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*. PMLR, 12310–12320.

[39] Jinchuan Zhang, Ming Sun, Chong Mu, Jinhao Zhang, Quanjiang Guo, and Ling Tian. 2025. Historically relevant event structuring for temporal knowledge graph reasoning. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. IEEE, 3179–3192.

[40] Ze Zhao, Yuhui He, Lyuwen Wu, Gu Tang, Bin Lu, Xiaoying Gan, Luoyi Fu, Xinbing Wang, and Chenghu Zhou. 2026. Inductive Reasoning for Temporal Knowledge Graphs with Emerging Entities. In *The Fourteenth International Conference on Learning Representations (ICLR)*.

[41] Yanping Zheng, Lu Yi, and Zhewei Wei. 2025. A survey of dynamic graph neural networks. *Frontiers of Computer Science* 19, 6 (2025), 196323.

[42] Cunchao Zhu, Muhao Chen, Changjun Fan, Guangquan Cheng, and Yan Zhang. 2021. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 4732–4740.

A Dataset Details

Dataset statistics. We conduct experiments on four temporal knowledge graph benchmarks: three derived from the ICEWS event corpus [1] (**ICEWS14**, **ICEWS18**, **ICEWS05-15**) and **GDELT** [16]. The ICEWS benchmarks record timestamped geopolitical events at a daily granularity, while GDELT records global events at a 15-minute granularity, and all four share the extrapolation protocol commonly used in prior TKG work [5, 18, 40]. Table A.1 summarizes the per-benchmark statistics. In the table, *#Snapshots* is the number of distinct timestamps in the graph and *#Emerging* is the number of entities that first appear in the validation or test split with no training-time interactions. To enable a direct comparison with TransFIR [40], we adopt the same chronological train/validation/test split with ratio 5:2:3, so that emerging-entity queries remain a meaningful fraction of the evaluation.

Table A.1: Statistics of the four TKG benchmarks used in our experiments.

Dataset	#Entities	#Relations	#Snapshots	#Total Triples	#Emerging Entities
ICEWS14	7,128	230	365	90,730	1,301
ICEWS18	23,033	256	304	468,558	3,434
ICEWS05-15	10,488	251	4,017	461,329	1,954
GDELT	7,691	240	2,976	2,277,405	875

Data splits and preprocessing. We adopt the chronological train / validation / test split released with each benchmark, ensuring that no future information leaks into training. Following common practice [5, 40], we augment each fact (e_s, r, e_o, t) with its inverse (e_o, r^{-1}, e_s, t) and evaluate both directions, reporting the average across the two directions. For entities that possess textual descriptions, we use the officially released entity-name strings as input to the text encoder described in Appendix C, while entities without descriptions receive a special placeholder token.

Emerging-entity statistics. The #Emerging column of Table A.1 reports the number of entities that appear for the first time only in the validation or test split (i.e., having no training-time interaction). Across the four benchmarks, emerging entities account for a non-trivial fraction of unique entities, which justifies treating the *Emerging* slice as the primary evaluation regime.

B Baselines

We adopt the same baseline set as TransFIR [40], comprising prior TKG reasoning methods grouped into three categories (graph-based, path-based, and static inductive), with TransFIR itself included as our direct baseline within the static-inductive group. Short descriptions of each method are provided below. Implementation sources follow the official releases when available, otherwise the reimplementations used in TransFIR for fair comparability.

Graph-based.

- **CyGNet** [42]: a sequential copy-generation network that predicts future facts from recurrent historical patterns under a closed entity vocabulary.
- **RE-GCN** [18]: a recurrent evolutional GCN that learns time-evolving entity and relation representations from adjacent snapshots.
- **HiSMatch** [17]: a historical-structure matching model that aligns query- and candidate-side historical subgraphs for link prediction.
- **MGESL** [23]: a multi-granularity history and entity similarity learning framework that captures temporal patterns at multiple levels of abstraction.
- **LogCL** [5]: a local-global history-aware contrastive-learning model that combines entity-aware attention with granularity-aware contrastive signals.
- **HisRes** [39]: a historically relevant event structuring framework with multi-granularity evolutionary and global relevance encoders.
- **MLEMKD** [25]: a mutual learning-empowered knowledge distillation method that compresses a TKG reasoner through adaptive distillation.

Path-based.

- **TLogic** [21]: an explainable temporal-rule forecasting framework that extracts cyclic rules via time-constrained random walks.
- **TILP** [33]: a differentiable learning framework for temporal logical rules on knowledge graphs.
- **ECEformer** [9]: a Transformer that encodes evolutionary chains of events through intra-quadruple representation learning and inter-quadruple context mixing.
- **GenTKG** [19]: a retrieval-augmented generation framework that combines temporal logical-rule retrieval with few-shot instruction tuning of a large language model.

Static inductive.

- **CompGCN** [31]: a composition-based multi-relational GCN that unifies node and relation embeddings under a shared message-passing framework.
- **ICL** [14]: an in-context-learning approach that performs temporal KG forecasting by prompting a frozen large language model with historical quadruples.
- **PPT** [36]: a pretrained-language-model method that reformulates TKG completion as a cloze-style task with soft prompts.
- **MorsE** [4]: a meta-knowledge transfer framework that learns entity-independent structural patterns for inductive KG embedding of unseen entities.
- **InGram** [15]: an inductive KG embedding model that exploits a learned relation graph to generalize to unseen entities and relations.
- **TransFIR** [40]: our direct baseline and the current state of the art on the emerging slice, which maps each entity to a learned codebook and transfers type-level behavioral prototypes from semantically similar cluster members.

C Implementation Details

Static encoder. For every entity $e \in \mathcal{E}$, we obtain a frozen textual representation $\mathbf{h}_e$ by feeding the entity surface form to a pretrained BERT-base encoder [7] and taking the `[CLS]` token, with the encoder weights not updated during training. For a direct comparison with TransFIR [40], we use the same encoder choice.

Relational decoder. We score candidate triples with ConvTransE [27], which has been widely adopted by recent TKG reasoning methods [5, 18, 40]. The decoder's kernel size, number of filters, and dropout are kept at their default values from the public ConvTransE implementation.

Hyperparameter grid. We select hyperparameters by grid search on the validation MRR of the *Emerging* slice, which is the setting most directly aligned with our research goal. Table C.1 lists the search ranges shared with TransFIR [40] and the configuration selected per dataset for AdaTKG-EMA (default operator). The EMA decay rate $\alpha = \sigma(\rho)$ is a learnable scalar (Section 5.1) and therefore is not searched over.

Table C.1: Hyperparameter search ranges and the selected configuration for AdaTKG-EMA. The ranges for shared hyperparameters (chain length, GNN layers, hidden dimensionality, codebook size) match those of TransFIR [40] for a direct comparison.

Hyperparameter	Search Range	Selected (AdaTKG-EMA)			
		ICEWS14	ICEWS18	ICEWS05-15	GDELT
Interaction-chain length L	{10, 15, 30}	10	30	30	30
Number of layers	{2, 3}	3	3	3	2
Hidden dimensionality d	{256, 512, 768, 1024}	768	1024	1024	1024
Codebook size K	{30, 50, 100}	100	100	50	30

Training objective. We follow the loss formulation of TransFIR [40] without modification, and our only design choice is the absence of any auxiliary loss on the memory state or the gate, so that both modules are trained purely through the link-prediction signal. AdaTKG is optimized end-to-end under the cross-entropy loss and a vector-quantization commitment loss applied to the codebook as

$$\mathcal{L} \;=\; \mathcal{L}_{\mathrm{LP}} \;+\; \lambda\,\mathcal{L}_{\mathrm{VQ}}, \qquad \lambda = 0.1. \tag{C.1}$$

Training protocol. All models are optimized with Adam [12]. We train for up to 200 epochs with early stopping (patience of 10 epochs on validation MRR) and report the best checkpoint. Each experiment is repeated with three random seeds and we report the mean across seeds in the main paper, with standard deviations relegated to the Additional Experiments appendix.

Hardware and runtime. All experiments are conducted on a single NVIDIA L40S GPU (48 GB). A full training run of AdaTKG-EMA on ICEWS14 at the selected hyperparameters of Table C.1 takes about 1 hour, with peak GPU memory under 30 GB. Training time on the larger benchmarks (ICEWS18, ICEWS05-15, GDELT) is reported per-epoch in the efficiency tables of Appendix I.

D Backbone Details

For completeness we reproduce the three backbone outputs $\textsc{Backbone}(q) \rightarrow (\mathbf{h}_e,\ \omega_e \cdot \mathbf{c}_{\pi(e),t},\ \mathbf{x}_e^{(\tau)})$ that AdaTKG consumes (Section 5.1 [1]). All three modules are taken from [40] unchanged.

(a) Static entity embedding $\mathbf{h}_e$. $\mathbf{h}_e \in \mathbb{R}^d$ is obtained by feeding the textual surface form of entity e into a frozen pretrained BERT [7] encoder and reading the `[CLS]` token. Because the encoder is not fine-tuned, $\mathbf{h}_e$ is well-defined for every entity that has a textual description — in particular, for entities never seen at training time — which is the source of the model's inductivity.

(b) Type-level inductive prior $\omega_e \cdot \mathbf{c}_{\pi(e),t}$. A learnable vector-quantized codebook $C = \{\mathbf{c}_k\}_{k=1}^{K}$ is applied to $\mathbf{h}_e$ to produce a cluster assignment $\pi(e) = \arg\min_k \|\mathbf{h}_e - \mathbf{c}_k\|_2^2$. At every query timestamp t, the cluster prototype $\mathbf{c}_{\pi(e),t}$ is recomputed by pooling the IC-encoder outputs of all cluster mates, so the prototype evolves as the graph evolves. The transfer gate $\omega_e = \Psi([\mathbf{h}_e \,\|\, \mathbf{c}_{\pi(e),t}]) \in [0,1]^d$ regulates how much of the prototype each entity inherits. The codebook is trained jointly with the rest of the model through the commitment loss $\mathcal{L}_{\mathrm{VQ}}$ in Eq. (C.1).

(c) Interaction signal $\mathbf{x}_e^{(\tau)}$. For each fact in which e participates, the event is summarized as

$$\mathbf{x}_e^{(\tau)} \;=\; \mathbf{W}_2\,\mathrm{GELU}\big(\mathbf{W}_1\,[\mathbf{h}_e^{\mathrm{IC}} \,\|\, \mathbf{h}_{r_\tau}]\big), \tag{D.1}$$

with learnable projections $\mathbf{W}_1 \in \mathbb{R}^{d\times 2d}$ and $\mathbf{W}_2 \in \mathbb{R}^{d\times d}$. The two inputs are (i) $\mathbf{h}_e^{\mathrm{IC}}$, the embedding of e's *interaction chain* — the time-ordered sequence of recent facts involving e, filtered by relation similarity to the query [40] and encoded with a Transformer — and (ii) $\mathbf{h}_{r_\tau}$, the relation embedding of the τ-th observed fact involving e (*not* the relation of an open query).

Both inputs are known at prediction time, so $\mathbf{x}_e^{(\tau)}$ never observes the ground-truth object of an open query. At the first appearance of an emerging entity, the chain is empty and $\mathbf{m}_e = \mathbf{0}$, which the adaptive gate of Eq. (8) converts into a hard fall-back to the static-inductive prior (Corollary 1, Appendix E).

E Reduction of AdaTKG to the Static-Inductive Setting at Cold Start

In this appendix, we formalize the observation, stated informally in Section 4 of the main text, that AdaTKG strictly generalizes the *Static + Inductive* setting (whose representative instantiation is TransFIR [40]), and we make precise the sense in which AdaTKG incurs no cold-start cost relative to that setting. The analysis concerns only the gated fusion rule, and therefore holds independently of the specific stateful update operator $\mathcal{U}$ used to maintain the per-entity memory.

AdaTKG generalizes the static-inductive setting. We first show that AdaTKG recovers the static-inductive prior exactly when the adaptive gate vanishes.

Proposition 1 (*Reduction to the static-inductive setting at zero gate*). *Let $\mathbf{z}_e^{(t_q),\mathrm{Ada}}$ and $\mathbf{z}_e^{(t_q),\mathrm{SI}}$ denote, respectively, the effective entity representation produced by AdaTKG via Eq.* (6) *and by the static-inductive setting via Eq.* (4) *at query time t_q. If the adaptive gate satisfies $g_e^{(t_q)} = \mathbf{0}$ and the static-inductive prior used by AdaTKG coincides with the one defined in Eq.* (4)*, then*

$$\mathbf{z}_e^{(t_q),\mathrm{Ada}} \;=\; \mathbf{z}_e^{(t_q),\mathrm{SI}},$$

and consequently the score $\phi_{t_q}(e_q, r_q, e_o)$ of AdaTKG equals that of the static-inductive setting for every query triple (e_q, r_q, e_o).

Proof. Substituting $g_e^{(t_q)} = \mathbf{0}$ into the AdaTKG fusion rule of Eq. (6) yields

$$\mathbf{z}_e^{(t_q),\mathrm{Ada}} \;=\; (1-\mathbf{0})\odot\big(\mathbf{h}_e + \omega_e\cdot\mathbf{c}_{\pi(e),t_q}\big) + \mathbf{0}\odot\mathbf{m}_e \;=\; \mathbf{h}_e + \omega_e\cdot\mathbf{c}_{\pi(e),t_q} \;=\; \mathbf{z}_e^{(t_q),\mathrm{SI}},$$

where the last equality is by the *Static + Inductive* definition in Eq. (4). Because the relational decoder $f(\cdot)$ in the unified scoring form (Section 4) depends on entity representations only through $\mathbf{z}_e^{(t_q)}$, replacing $\mathbf{z}_e^{(t_q),\,\mathrm{Ada}}$ by $\mathbf{z}_e^{(t_q),\,\mathrm{SI}}$ yields identical scores. □

Cold-start parity. Proposition E has an immediate consequence at the moment each entity first appears in the graph.

Corollary 1 (*Cold-start parity with the static-inductive setting*). *At the query time $t_q = t_e(e)$ of an emerging entity e, the per-entity memory is zero by the initialization in Eq. (5), namely $\mathbf{m}_e = \mathbf{0}$. If the adaptive gate is parameterized so that $g_e^{(t_q)} = \mathbf{0}$ whenever $\mathbf{m}_e = \mathbf{0}$ (a design choice readily enforced by zero-masking the gate output when the memory buffer is empty), then, by Proposition E, the prediction of AdaTKG coincides exactly with that of the static-inductive setting.*

Corollary 1 formalizes the graceful-degradation property emphasized in the main text: AdaTKG never pays a cold-start cost relative to the static-inductive setting (instantiated by TransFIR [40] in our experiments), and all empirical gains reported in Section 6 stem purely from information retained in the per-entity memory.

F Stateful Update Operators

We compared three instantiations of the stateful update operator $\mathcal{U}$ in Section 6.2: a learnable EMA (used by AdaTKG), a GRU cell, and a cross-attention readout over a bounded buffer. All three follow the unified template of Eq. (5),

$$\mathbf{m}_e^{(\tau)} \;=\; \mathcal{U}\big(\mathbf{m}_e^{(\tau-1)},\, \mathbf{x}_e^{(\tau)}\big), \qquad \mathbf{m}_e^{(0)} \;=\; \mathbf{0},$$

and share the same interaction signal

$$\mathbf{x}_e^{(\tau)} \;=\; \mathrm{MLP}_{\mathrm{enc}}\big([\,\mathbf{c}_e^{(\tau)} \,\|\, \mathbf{r}_e^{(\tau)}\,]\big), \tag{F.1}$$

where $\mathbf{c}_e^{(\tau)}$ is the chain summary produced by the TransFIR backbone at the τ-th interaction of e and $\mathbf{r}_e^{(\tau)}$ is the relation embedding of that interaction. The three variants differ only in how $\mathcal{U}$ aggregates the past memory state with the new signal.

(a) Learnable EMA (AdaTKG, Section 5.1). The default instantiation realizes $\mathcal{U}$ as an exponential moving average parameterized by a single shared scalar:

$$\mathbf{m}_e^{(\tau)} \;=\; \alpha\,\mathbf{m}_e^{(\tau-1)} \;+\; (1-\alpha)\,\mathbf{x}_e^{(\tau)}, \qquad \alpha \;=\; \sigma(\rho),\ \rho\in\mathbb{R}. \tag{F.2}$$

The decay α is shared across all entities, so the operator introduces only one learnable scalar beyond TransFIR.

(b) GRU cell. A standard GRU cell [6] treats $\mathbf{m}_e^{(\tau-1)}$ as the hidden state and $\mathbf{x}_e^{(\tau)}$ as the input, applying a learned reset and update gate:

$$\mathbf{m}_e^{(\tau)} \;=\; \mathrm{GRUCell}\big(\mathbf{x}_e^{(\tau)},\, \mathbf{m}_e^{(\tau-1)}\big). \tag{F.3}$$

The GRU cell adds three weight matrices in $\mathbb{R}^{d\times d}$ and three biases, roughly $3d^2$ extra parameters.

(c) Cross-attention readout over a bounded buffer. This variant maintains a per-entity FIFO buffer $\mathbf{B}_e \in \mathbb{R}^{K\times d}$ of the most recent K interaction signals (default $K=16$). The buffer is updated by appending the new signal at position $\tau \bmod K$,

$$\mathbf{B}_e[\tau \bmod K] \;\leftarrow\; \mathbf{x}_e^{(\tau)},$$

and the memory state read at query time is computed by cross-attending the query embedding $\mathbf{q}_e$ to the buffer,

$$\mathbf{m}_e^{(\tau)} \;=\; \mathrm{MultiHeadAttn}\big(\mathbf{q}_e,\, \mathbf{B}_e,\, \mathbf{B}_e\big). \tag{F.4}$$

This variant adds $O(d^2)$ attention parameters and an additional $O(K\,d)$ buffer per entity.

Adaptive gate fusion (shared). Regardless of the choice of $\mathcal{U}$, the resulting memory $\mathbf{m}_e$ is fused with the inductive prior through the same adaptive gate of Eq. (8):

$$\mathbf{z}_e^{(t_q)} \;=\; (1-g_e^{(t_q)}) \odot (\mathbf{h}_e + \omega_e\,\mathbf{c}_{\pi(e),\,t_q}) \;+\; g_e^{(t_q)} \odot \mathbf{m}_e,$$

with the cold-start zero-mask $g_e^{(t_q)} = \mathbf{0}$ active whenever $\mathbf{m}_e = \mathbf{0}$. The (P1)–(P3) protocol of Section 5.1 (epoch reset, chronological replay, detached update) is also applied identically to all three variants, so the comparison in Section 6.2 isolates the effect of $\mathcal{U}$ alone.

Parameter count summary.

- EMA: 1 scalar (shared ρ).
- GRU: $\sim 3d^2$ parameters in the cell.
- Attention: $\sim 4d^2$ attention parameters and a per-entity buffer of size $K\,d$.

The single-scalar parameterization of EMA is the most parsimonious operating point along this frontier and, as reported in Section 6.2, also the most accurate on the emerging slice.

G Sensitivity to Memory Update Timing

By default, AdaTKG updates the per-entity memory $\mathbf{m}_e$ with the current interaction signal $\mathbf{x}_e^{(\tau)}$ *before* reading $\mathbf{m}_e$ for scoring (the `before` ordering used throughout the main paper). To probe the sensitivity of this design choice, we additionally train AdaTKG with the alternative `after` ordering, in which $\mathbf{m}_e$ is read for scoring *first*,so the score depends only on the strictly past memory and the EMA update is committed only afterwards. Both orderings share the per-dataset best HP of AdaTKG (Table C.1); only the order of the read and the update differs. Table G.1 reports the comparison on the *Emerging* slice, demonstarting that across all four benchmarks, the default `before` ordering consistently outperforms the `after` ordering, supporting our design choice in the main paper.

Table G.1: Sensitivity to memory update timing. AdaTKG on the *Emerging* slice with the memory update applied either `before` (default) or `after` scoring.

Update timing	ICEWS14			ICEWS18			ICEWS05-15			GDELT		
	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10
before (default)	**.2011**	**.2250**	**.3621**	**.1379**	**.1543**	**.2612**	**.2270**	**.2573**	**.3850**	**.1051**	**.1129**	**.2301**
after	.1777	.2011	.3505	.1322	.1470	.2482	.1914	.2110	.3290	.0814	.0797	.1663

H Multi-Seed Robustness

To ensure fair comparison across all baselines, the main paper reports single-seed results. In this section, we further evaluate AdaTKG and the static-inductive baseline ([40]) under three random seeds at the per-dataset best HP of Table C.1, reporting the mean

and standard deviation on the *Emerging* slice (Table H.1). AdaTKG continues to outperform the SoTA under this multi-seed setting.

Table H.1: Multi-seed robustness on the *Emerging* slice. Mean ± standard deviation over three random seeds.

Dataset	Metric	Method		
		TransFIR [40]	**AdaTKG (Ours)**	Δ (%)
ICEWS14	MRR	.1768 ± .0012	**.2008** ± .0014	**+13.6**
	H@3	.2090 ± .0015	**.2256** ± .0017	**+7.9**
	H@10	.3417 ± .0019	**.3618** ± .0021	**+5.9**
ICEWS18	MRR	.1118 ± .0011	**.1382** ± .0013	**+23.6**
	H@3	.1224 ± .0014	**.1538** ± .0016	**+25.7**
	H@10	.2257 ± .0017	**.2607** ± .0019	**+15.5**
ICEWS05-15	MRR	.2173 ± .0010	**.2274** ± .0012	**+4.6**
	H@3	.2534 ± .0013	**.2569** ± .0015	**+1.4**
	H@10	.3712 ± .0016	**.3853** ± .0018	**+3.8**
GDELT	MRR	.1009 ± .0014	**.1054** ± .0013	**+4.5**
	H@3	.0998 ± .0017	**.1124** ± .0016	**+12.6**
	H@10	.2127 ± .0020	**.2305** ± .0019	**+8.4**

I Efficiency Comparison Across Benchmarks

Tables I.1, I.2, and I.3 extend the main-paper Table 9 to ICEWS18, ICEWS05-15, and GDELT. The pattern observed on ICEWS14 consistently holds across all benchmarks, with every update operator incurring only modest overhead while improving emerging-entity performance, indicating that the cost–benefit profile stems from the *adaptivity principle* itself rather than any specific operator.

Memory footprint at inference. Beyond the cost metrics in the tables above, AdaTKG keeps one additional piece of state at inference: a per-entity memory buffer of size $|\mathcal{E}| \times d$. With our largest configuration ($d = 1024$ on ICEWS18, which has 23,033 entities), this buffer takes about 94 MB, and stays under 100 MB on every benchmark.

Table I.1: Efficiency comparison on ICEWS18.

Method		[1] Efficiency						[2] Performance		
		# Parameters (M)		**Training time** (s/epoch)		**FLOPs** (M/query)		**MRR**	**H@3**	**H@10**
		Value	Δ (%)	Value	Δ (%)	Value	Δ (%)			
Base (w/o Adaptivity) [40]		72.77	–	330.7	–	1643.9	–	.1114	.1230	.2252
AdaTKG	EMA (*default*)	78.02	**+7.2%**	361.3	**+9.2%**	1654.4	**+0.6%**	**.1379**	**.1543**	**.2612**
	GRU	84.32	**+15.9%**	378.0	**+14.3%**	1658.8	**+0.9%**	**.1428**	**.1599**	**.2605**
	Cross-attention	82.22	**+13.0%**	360.5	**+9.0%**	1725.8	**+5.0%**	**.1454**	**.1712**	**.2761**

Table I.2: Efficiency comparison on ICEWS05-15.

Method		[1] Efficiency						[2] Performance		
		# Parameters (M)		**Training time** (s/epoch)		**FLOPs** (M/query)		**MRR**	**H@3**	**H@10**
		Value	Δ (%)	Value	Δ (%)	Value	Δ (%)			
Base (w/o Adaptivity) [40]		72.71	–	420.5	–	1913.9	–	.2177	.2530	.3708
AdaTKG	EMA (*default*)	77.96	**+7.2%**	466.3	**+10.9%**	1924.4	**+0.5%**	**.2270**	**.2573**	**.3850**
	GRU	84.25	**+15.9%**	491.0	**+16.8%**	1930.2	**+0.8%**	**.2330**	**.2700**	**.3925**
	Cross-attention	82.15	**+13.0%**	459.8	**+9.4%**	1995.8	**+4.3%**	**.2243**	**.2573**	**.3815**

Table I.3: Efficiency comparison on GDELT.

Method		[1] Efficiency						[2] Performance		
		# Parameters (M)		**Training time** (s/epoch)		**FLOPs** (M/query)		**MRR**	**H@3**	**H@10**
		Value	Δ (%)	Value	Δ (%)	Value	Δ (%)			
Base (w/o Adaptivity) [40]		60.08	–	1384.6	–	1062.6	–	.1013	.0994	.2131
AdaTKG	EMA (*default*)	65.33	**+8.7%**	1488.7	**+7.5%**	1073.1	**+1.0%**	**.1051**	**.1129**	**.2301**
	GRU	71.63	**+19.2%**	1549.0	**+11.9%**	1075.9	**+1.3%**	**.1112**	**.1141**	**.2243**
	Cross-attention	69.53	**+15.7%**	1503.2	**+8.6%**	1144.4	**+7.7%**	**.1297**	**.1396**	**.2544**

J Memory Ablation Across Benchmarks

Figures J.1, J.2, and J.3 extend the main-paper Figure 4 (ICEWS14) to ICEWS18, ICEWS05-15, and GDELT. In each figure, Δ_{RR} is stratified by 1) train-time history depth (left) and 2) test-time online updates (right). Across all four benchmarks, every AdaTKG variant lies above the Base in every bin and the gap widens monotonically with the history depth, indicating that the per-entity memory contributes consistently across datasets.

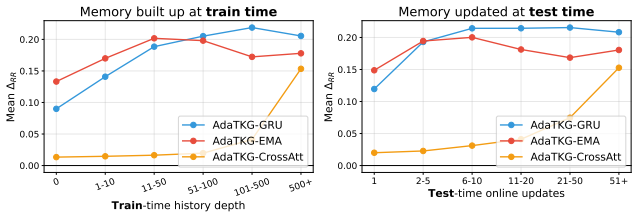

Figure J.1: Memory ablation on ICEWS18.

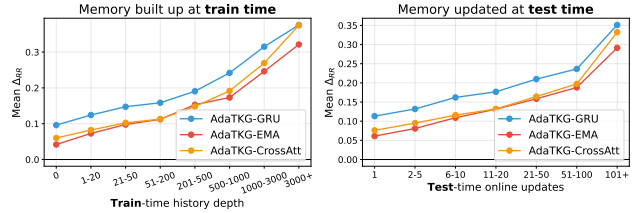

Figure J.2: Memory ablation on ICEWS05-15.

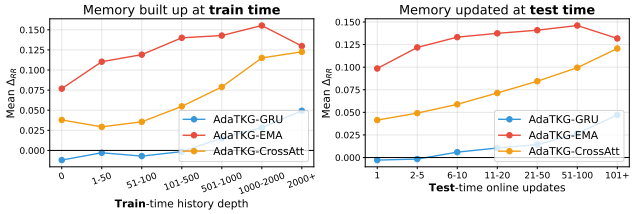

Figure J.3: Memory ablation on GDELT.

K Gate Distribution Across Benchmarks

Figure K.1 extends the main-paper Figure 5 by showing the full train-time gate distribution stratified by the number of observed interactions, for every (*dataset*, *update operator*) pair. The same monotonic upward shift in the gate value with more interactions holds across all four benchmarks and all three update operators, confirming that the adaptive gate consistently learns to lean on the per-entity memory branch as evidence accumulates, rather than being an artifact of a particular dataset or operator.

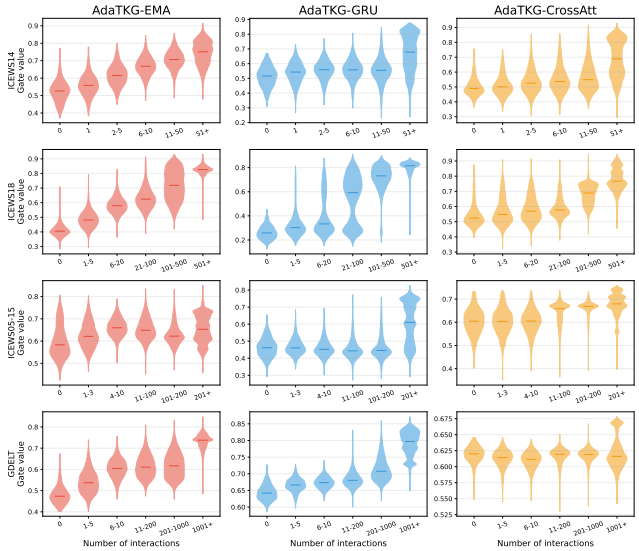

Figure K.1: Gate value by # interactions. Each subplot is a (*dataset, operator*) pair, with rows as datasets and columns as AdaTKG operators (EMA, GRU, CrossAtt). Within each subplot, violins show the train-time distribution of the learned gate $g_e^{(t_q)}$ stratified by the number of subject interactions observed before the query, with the median marked.