TEMPORAL RESIDUAL MEMORY FOR MITIGATING CATASTROPHIC FORGETTING IN CONTINUAL KNOWLEDGE GRAPH EMBEDDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual Knowledge Graph Embedding (CKGE) aims to sequentially incorporate newly arriving facts into a knowledge graph while preserving the knowledge learned from previous snapshots. A key challenge is catastrophic forgetting, where the knowledge acquired from past snapshots can be rapidly lost as the model adapts to a new snapshot. Existing studies have mainly focused on preserving past knowledge and transferring it stably across snapshots to mitigate such forgetting. This work starts from the observation that relations and entities may exhibit different degrees of semantic stability as a knowledge graph grows. Based on this perspective, we propose Temporal Residual Memory (TRM). TRM introduces snapshot-wise entity-side residual memory through the BoxE bump path, freezes memories from previous snapshots, and aggregates valid memories according to each entity's birth snapshot for current prediction. Experimental results show that the proposed method achieves strong performance on diverse CKGE benchmarks and improves knowledge retention metrics. In addition, experiments extending the core idea of TRM to TransE and DistMult also show a tendency toward mitigating forgetting. These results suggest that leveraging entity-side memory formed from past snapshots for current prediction can serve as an effective strategy for mitigating forgetting in CKGE.

1 INTRODUCTION

Knowledge Graph Embedding (KGE) maps entities and relations into continuous vector spaces for reasoning tasks such as Knowledge Graph Completion (KGC) Wang et al. (2017); Cao et al. (2024b). However, large-scale real-world knowledge graphs such as DBpedia Bizer et al. (2009) continuously grow as new facts arrive Ji et al. (2022); Xie et al. (2020), making full retraining at every time step inefficient. Continual Knowledge Graph Embedding (CKGE) instead learns sequential graph snapshots while preserving previously acquired knowledge Daruna et al. (2021); Cui et al. (2023). A key challenge in continual learning is catastrophic forgetting Kemker et al. (2018); Ramasesh et al. (2021); Wang et al. (2024b). In CKGE, adapting to a new snapshot can drift representations learned from previous snapshots and rapidly degrade prediction on past knowledge Liu et al. (2021); Zhou & Cao (2021); Wang et al. (2024a). Existing CKGE studies have addressed this problem from various perspectives Yuan et al. (2026). Most of these studies have focused on how to preserve past representations. In contrast, this work views CKGE not only as a problem of parameter preservation, but also as a problem of how to effectively leverage entity-side information formed in past snapshots for current learning and prediction.

This work starts from the perspective that relations and entities may exhibit different degrees of semantic stability as a knowledge graph grows Yang et al. (2025). Relations tend to preserve relatively stable structural meanings across snapshots, whereas entities undergo more direct changes in semantic position and context as new facts arrive. Therefore, it is natural in CKGE to treat relations and entities separately rather than updating or preserving them in the same way. From this perspective, we propose Temporal Residual Memory (TRM), which maintains the backbone relation-side scoring structure while managing dynamically changing entity-side representations through snapshot-wise entity residual memory. TRM stores snapshot-wise entity residual memories, aggregates them over time, and combines the aggregated residual with the current entity representation to reuse past patterns

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.

for current prediction. Thus, TRM treats past knowledge not only as something to protect, but as a resource to retrieve and integrate into current prediction. We build TRM on BoxE Abboud et al. (2020), whose point-bump decomposition enables flexible modeling of entity changes induced by newly arriving facts while maintaining the relation-side structure. TRM injects snapshot-wise entity residual memories through the BoxE bump path and recomposes them for current prediction.

To better analyze forgetting behavior, we define Retention Trajectory Evaluation (RTE), a supplementary metric that measures relative performance changes across subsequent snapshot boundaries rather than only at the final snapshot, complementing BWT Lopez-Paz & Ranzato (2017b) and ROD Yang et al. (2025). Experiments on diverse CKGE benchmarks show that TRM achieves strong final performance and positive retention scores on most datasets. Extensions of the core memory reuse idea to TransE Bordes et al. (2013) and DistMult Yang et al. (2014) also improve retention metrics, suggesting that entity-side memory reuse is not limited to the BoxE setting. The main contributions of this paper are summarized as follows:

- We reinterpret CKGE as a problem of reusing entity-side information formed in past snapshots for current learning and prediction, and propose TRM, which manages snapshot-wise entity residual memory while maintaining the relation-side scoring structure.
- We introduce RTE as a supplementary trajectory-aware retention metric and show that TRM achieves strong final performance with positive retention behavior on most CKGE benchmarks.
- We extend the core memory reuse idea to TransE and DistMult, showing that entity-side residual memory can reduce forgetting tendencies beyond the BoxE setting.

2 Related work

Traditional KGE. Knowledge Graph Embedding (KGE) represents entities and relations in continuous vector spaces for reasoning tasks such as link prediction Lin et al. (2015); Cao et al. (2024a). Translation-based models such as TransE Bordes et al. (2013) have been widely used as backbones in Continual Knowledge Graph Embedding (CKGE) due to their simple structure and computational efficiency. DistMult Yang et al. (2014) is a representative semantic matching-based model that models relations through diagonal bilinear interactions and computes triple scores via multiplicative interactions between entities and relations. BoxE Abboud et al. (2020) is a spatio-translational embedding model that represents entities with points and bumps and relations with role-specific boxes, enabling relational structures and entity-side interactions to be modeled separately within a geometric embedding space.

Continual models for KG. Many CKGE baselines inherit general continual learning strategies, including regularization-based methods Kirkpatrick et al. (2017); Zenke et al. (2017), replay-based methods Lopez-Paz & Ranzato (2017a); Wang et al. (2019), and dynamic architecture-based methods Rusu et al. (2016). Early CKGE studies mainly focused on transferring representations across snapshots and stabilizing parameter changes Song & Park (2018). LKGE combines masked KG autoencoding, embedding transfer, and regularization Cui et al. (2023), while IncDE uses hierarchical ordering, incremental distillation, and two-stage training to balance new knowledge learning and old knowledge preservation Liu et al. (2024). SAGE adjusts the embedding space through scale-aware gradual evolution Li et al. (2025b), and BAKE formulates CKGE as sequential Bayesian inference with posterior updates and clustering-based constraints Li et al. (2025a). Recent work also considers efficiency and scalability; ETT-CKGE performs snapshot transfer through task-driven tokens Zhu et al. (2025). Other studies redesign the update mechanism. ERPP avoids direct entity embedding transfer and accumulates relational information through relation path passing, inheritance, and evolution Yang et al. (2025). STCKGE uses spatial transformation-based dual-component entity representations and relation-aware spatial regions Wang et al. (2025). In contrast, this work retrieves snapshot-wise entity residual memories for current prediction, treating past representations as reusable memory rather than merely preserving or transferring them.

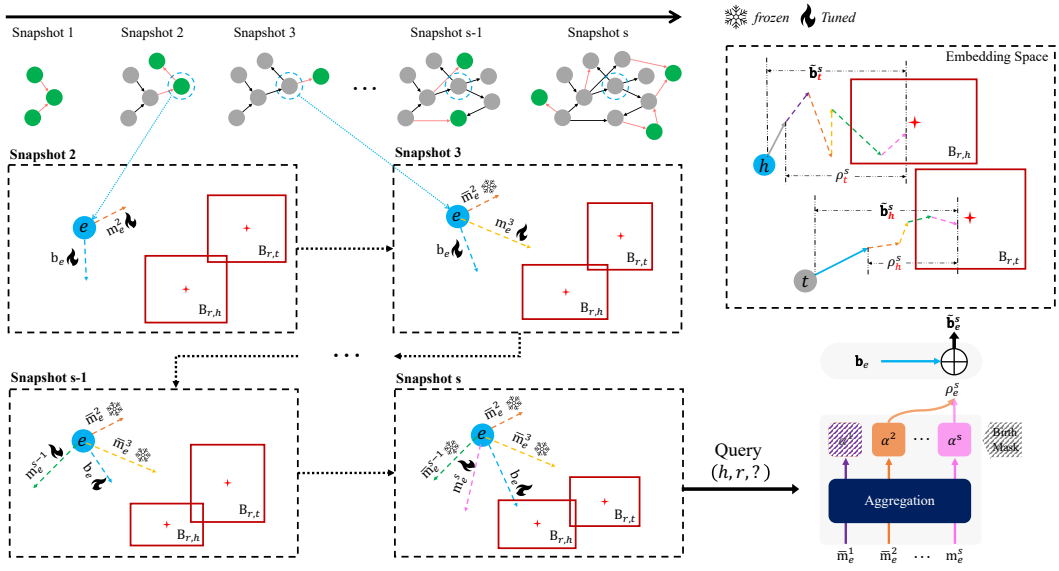

Figure 1: Overall framework of TRM.

3 METHODOLOGY

3.1 PRELIMINARIES

We follow the standard continual knowledge graph embedding (CKGE) setting on growing knowledge graphs. Detailed definitions and notation used throughout this paper are provided in Appendix A.

Knowledge transfer and retention metrics. Forward Transfer (FWT) measures how much the knowledge acquired from previous snapshots helps the learning of subsequent snapshots, while Backward Transfer (BWT) evaluates how well the knowledge learned from past snapshots is retained after learning later snapshots Lopez-Paz & Ranzato (2017b). In addition, previous work has used Rate of Decline (ROD) to complement BWT by considering both test set size and initial performance Yang et al. (2025). However, both BWT and ROD compare only a limited set of time points for each snapshot, and therefore do not fully capture how forgetting progresses across snapshot boundaries. For example, if a model suffers a large performance drop at intermediate snapshots but partially recovers at the final snapshot, metrics based only on the final time point may fail to distinguish such behavior. To address this limitation, we define Retention Trajectory Evaluation (RTE) as a new supplementary metric. Specifically, let $m_{s,i} = \mathrm{MRR}(s, i)$ denote the performance on the test set of snapshot i after the model has been trained up to snapshot s. We define the relative change with respect to the performance immediately after learning snapshot i as

$$\delta_i(s) = \frac{m_{s,i} - m_{i,i}}{m_{i,i}}, \tag{1}$$

where $\delta_i(s) < 0$ indicates forgetting, while $\delta_i(s) > 0$ indicates retention or improvement. Based on this quantity, RTE is defined as follows:

$$\mathrm{RTE} = \sum_{i=1}^{S-1} w_i \cdot \frac{1}{S-i} \sum_{s=i+1}^{S} \delta_i(s), \qquad w_i = \frac{|Q_i|}{|Q| - |Q_S|}. \tag{2}$$

Here, $\mathrm{MRR}(s, i)$ denotes the MRR on the test set Q_i after training up to snapshot s, and $|Q_i|$ and $|Q|$ denote the size of the test set for snapshot i and the total size of all test sets, respectively. Since RTE averages the relative performance changes over all subsequent snapshot boundaries for each past snapshot, it can more directly evaluate retention and forgetting behavior across the entire snapshot trajectory than metrics that compare only the final time point.

3.2 TEMPORAL RESIDUAL MEMORY

Model overview. The overview of TRM is shown in Figure 1. TRM approaches the CKGE problem from the perspective of memory reuse. To this end, it maintains snapshot-wise entity residual memory and reuses it when forming predictions at each snapshot. In other words, TRM treats past knowledge not merely as something to be protected, but as a resource that can be actively retrieved and integrated into current prediction. Based on the view that relations and entities may exhibit different degrees of semantic stability as a knowledge graph grows, TRM keeps the relatively stable backbone relation-side scoring structure and introduces entity residual memory only for entity-side representations, which change more directly with newly arriving facts. To examine this memory-scope configuration in more detail, a further analysis including the entity-side setting adopted in TRM is provided in Appendix F. At each snapshot, TRM stores snapshot-wise entity residual memory formed during training, while keeping memories from past snapshots fixed. When scoring triples in a new snapshot, TRM retrieves only the valid past entity residual memories according to each entity's birth snapshot, aggregates them, and recomposes them with the current entity representation. This allows the model to reuse entity-side representation patterns formed in past snapshots for current prediction while preserving the original scoring mechanism of the backbone.

BoxE-based TRM. TRM is built on BoxE. Since BoxE decomposes entity representations into points and bumps, it can model changing entity representations while maintaining stable relation-side structure as new facts are introduced. On top of this BoxE structure, TRM introduces snapshot-wise residual bumps as temporal memories and injects their aggregation through the BoxE bump path for current prediction. In BoxE, each entity e is represented by a point vector $\mathbf{p}_e \in \mathbb{R}^d$ and a bump vector $\mathbf{b}_e \in \mathbb{R}^d$, while each relation r is represented by two role-specific boxes corresponding to the head and tail roles. For each role $i \in \{h, t\}$, the relation box is constructed as $\mathbf{B}_{r,i} = [\boldsymbol{\ell}_{r,i}, \boldsymbol{u}_{r,i}]$ from the center vector $\mathbf{c}_{r,i}$ and the delta vector $\boldsymbol{\delta}_{r,i}$, where $\boldsymbol{\ell}_{r,i} = \min\big(\mathbf{c}_{r,i} - \frac{1}{2}\boldsymbol{\delta}_{r,i}, \mathbf{c}_{r,i} + \frac{1}{2}\boldsymbol{\delta}_{r,i}\big)$, $\boldsymbol{u}_{r,i} = \max\big(\mathbf{c}_{r,i} - \frac{1}{2}\boldsymbol{\delta}_{r,i}, \mathbf{c}_{r,i} + \frac{1}{2}\boldsymbol{\delta}_{r,i}\big)$. For a triple (h, r, t), BoxE uses a cross-bump structure to construct $\mathbf{q}_h = \mathbf{p}_h + \mathbf{b}_t$ and $\mathbf{q}_t = \mathbf{p}_t + \mathbf{b}_h$. Let $\boldsymbol{\mu}_{r,i} = (\boldsymbol{\ell}_{r,i} + \boldsymbol{u}_{r,i})/2$, $\boldsymbol{\omega}_{r,i} = \boldsymbol{u}_{r,i} - \boldsymbol{\ell}_{r,i}$, and $\boldsymbol{\omega}^+_{r,i} = \boldsymbol{\omega}_{r,i} + \mathbf{1}$. The coordinate-wise distance between a point $\mathbf{x}$ and a box $\mathbf{B}_{r,i}$ is defined as follows:

$$\Delta(\mathbf{x}, \mathbf{B}_{r,i}) = \begin{cases} |\mathbf{x} - \boldsymbol{\mu}_{r,i}| \oslash \boldsymbol{\omega}^+_{r,i}, & \text{if } \boldsymbol{\ell}_{r,i} \le \mathbf{x} \le \boldsymbol{u}_{r,i}, \\ \boldsymbol{\omega}^+_{r,i} \odot |\mathbf{x} - \boldsymbol{\mu}_{r,i}| - \dfrac{\boldsymbol{\omega}_{r,i}}{2} \odot \left(\boldsymbol{\omega}^+_{r,i} - (\mathbf{1} \oslash \boldsymbol{\omega}^+_{r,i})\right), & \text{otherwise.} \end{cases} \tag{3}$$

Here, $\odot$ and $\oslash$ denote element-wise multiplication and division, respectively. The BoxE triple distance is then computed as $D_{\text{Box}}(h, r, t) = \|\Delta(\mathbf{q}_h, \mathbf{B}_{r,h})\|_2 + \|\Delta(\mathbf{q}_t, \mathbf{B}_{r,t})\|_2$. Therefore, TRM intervenes not in the relation boxes themselves, but in the entity-side bump path that forms $\mathbf{q}_h$ and $\mathbf{q}_t$.

Snapshot-wise entity residual memory. TRM maintains a snapshot-wise entity residual memory $\mathbf{m}^\tau_e \in \mathbb{R}^d$ for each snapshot τ and entity e. This memory is not an independent entity embedding, but is used as a correction term added to the current entity representation. When training at snapshot s, memories from past snapshots are used for retrieval but are fixed so that they are not updated by the current loss, while only the memory of the current snapshot remains trainable. Thus, for $\tau < s$, TRM uses frozen memories denoted by $\bar{\mathbf{m}}^\tau_e$, whereas the current memory is used as the trainable memory $\mathbf{m}^s_e$. Since not every entity exists at every snapshot, we define the birth snapshot of entity e as $b(e) = \min\{s \mid e \in \mathcal{E}_{\le s}\}$, and define the valid memory set available at snapshot s as $\mathcal{V}^s_e = \{\tau \mid 1 \le \tau \le s,\ \tau \ge b(e)\}$. TRM uniformly aggregates these valid memories. Accordingly, the temporally aggregated residual $\boldsymbol{\rho}^s_e$ and the final bump $\tilde{\mathbf{b}}^s_e$ of entity e are defined as:

$$\boldsymbol{\rho}^s_e = \frac{1}{|\mathcal{V}^s_e|}\left(\sum_{\tau \in \mathcal{V}^s_e,\ \tau < s} \bar{\mathbf{m}}^\tau_e + \mathbf{m}^s_e\right), \qquad \tilde{\mathbf{b}}^s_e = \mathbf{b}_e + \boldsymbol{\rho}^s_e. \tag{4}$$

In this way, TRM does not aggregate memories indiscriminately, but averages only the memories that are valid for the entity at the current snapshot.

Parameter initialization and expansion. At the first snapshot, entity points, base bumps, and role-specific relation-box parameters are initialized uniformly, while snapshot-wise entity residual

memories are initialized with small zero-mean Gaussian noise so that TRM starts close to the base BoxE scoring path. When new entities or relations appear in later snapshots, previously learned parameters are copied unchanged, while only newly introduced parameters are initialized in the same way. Newly allocated entity residual memories are also initialized with the same small Gaussian noise and are accessed according to the birth-aware valid memory set.

Recomposition and training. The aggregated memory is additively injected into the original cross-bump scoring path of BoxE. Thus, for a triple (h, r, t) at snapshot s, the two role-specific representations are given by $\tilde{\mathbf{q}}_h^s = \mathbf{p}_h + \tilde{\mathbf{b}}_t^s$ and $\tilde{\mathbf{q}}_t^s = \mathbf{p}_t + \tilde{\mathbf{b}}_h^s$, and the TRM triple distance is computed as

$$D_{\mathrm{TRM}}^s(h, r, t) = \|\Delta(\tilde{\mathbf{q}}_h^s, \mathbf{B}_{r,h})\|_2 + \|\Delta(\tilde{\mathbf{q}}_t^s, \mathbf{B}_{r,t})\|_2. \tag{5}$$

That is, TRM does not modify the relation boxes $\mathbf{B}_{r,h}, \mathbf{B}_{r,t}$ or the point-to-box distance of BoxE itself, but injects snapshot-wise entity residual memory only through the entity-side bump path that serves as the input to BoxE scoring. For a given query $(h, r, ?)$, TRM computes $D_{\mathrm{TRM}}^s(h, r, t)$ for each candidate tail entity $t \in \mathcal{E}_{\leq s}$. Because the memory-augmented bump $\tilde{\mathbf{b}}_t^s$ depends on the candidate entity, the head-role representation is recomposed separately for each candidate. Therefore, entity-side interaction patterns formed in past snapshots are directly reflected in current tail prediction.

We use a margin-based ranking loss for TRM, as it showed the most stable balance between final performance and retention metrics. Since the loss function can interact with the scoring function and backbone structure, Appendix E further analyzes this sensitivity by comparing additional loss settings. Training at snapshot s is performed by combining the KGE loss for the current snapshot with inter-snapshot drift regularization, written as

$$\mathcal{L}^s = \mathcal{L}_{\mathrm{kge}}^s + \lambda_{\mathrm{evo}}(\mathcal{L}_{\mathrm{evo}}^{\mathrm{rel}} + \mathcal{L}_{\mathrm{evo}}^{\mathrm{ent}}). \tag{6}$$

Given a positive triple $x_i^+ = (h_i, r_i, t_i)$ and its corresponding corrupted triples $\{x_{i,j}^-\}_{j=1}^n$, let $d_i^+ = D_{\mathrm{TRM}}^s(x_i^+)$ and $d_{i,j}^- = D_{\mathrm{TRM}}^s(x_{i,j}^-)$. The KGE loss for the current snapshot is then defined as

$$\mathcal{L}_{\mathrm{kge}}^s = \sum_i \left[\gamma + d_i^+ - \sum_{j=1}^n w_{i,j} d_{i,j}^-\right]_+, \qquad w_{i,j} = \frac{\exp(-d_{i,j}^-)}{\sum_{k=1}^n \exp(-d_{i,k}^-)}. \tag{7}$$

Here, γ is the margin value, $w_{i,j}$ denotes the self-adversarial negative weight, and $[\cdot]_+ = \max(\cdot, 0)$. The entity and relation drift regularization terms are respectively defined as

$$\begin{aligned} \mathcal{L}_{\mathrm{evo}}^{\mathrm{ent}} &= \sum_{e \in \mathcal{E}_{\leq s-1}} \left(\left\|\mathbf{p}_e^s - \mathbf{p}_e^{s-1}\right\|_2^2 + \left\|\mathbf{b}_e^s - \mathbf{b}_e^{s-1}\right\|_2^2 \right), \\ \mathcal{L}_{\mathrm{evo}}^{\mathrm{rel}} &= \sum_{r \in \mathcal{R}_{\leq s-1}} \sum_{i \in \{h,t\}} \left(\left\|\mathbf{c}_{r,i}^s - \mathbf{c}_{r,i}^{s-1}\right\|_2^2 + \left\|\boldsymbol{\delta}_{r,i}^s - \boldsymbol{\delta}_{r,i}^{s-1}\right\|_2^2 \right). \end{aligned} \tag{8}$$

The former prevents the point and base bump of previously observed entities from drifting excessively during adaptation to the current snapshot, while the latter prevents the role-specific relation boxes from changing abruptly. Negative samples are generated based on the Partial Completeness Assumption (PCA) Galárraga et al. (2013) by corrupting entities in positive triples.

4 Experiments

4.1 Datasets and baselines

To assess the effectiveness of TRM, we conducted experiments on seven benchmark datasets: ENTITY, RELATION, FACT, HYBRID, GraphEqual, GraphHigher, and GraphLower. These datasets cover distinct growth patterns, including entity expansion, relation expansion, fact accumulation, hybrid growth, and snapshot-wise variations in triple increments, making them suitable for jointly evaluating adaptation in CKGE settings and mitigating forgetting. Table 1 summarizes, for each dataset and snapshot, the cumulative numbers of entities and relations, together with the numbers of newly added triples. All experiments were conducted under a five-snapshot setting. We compared TRM with seven previously proposed CKGE methods, namely LKGE Cui et al. (2023), IncDE Liu et al. (2024),

Table 1: The statistics of the dataset.

Dataset	Snapshot 1			Snapshot 2			Snapshot 3			Snapshot 4			Snapshot 5		
	$\lvert\mathcal{E}_{\leq 1}\rvert$	$\lvert\mathcal{R}_{\leq 1}\rvert$	$\lvert\Delta\mathcal{T}^{(1)}\rvert$	$\lvert\mathcal{E}_{\leq 2}\rvert$	$\lvert\mathcal{R}_{\leq 2}\rvert$	$\lvert\Delta\mathcal{T}^{(2)}\rvert$	$\lvert\mathcal{E}_{\leq 3}\rvert$	$\lvert\mathcal{R}_{\leq 3}\rvert$	$\lvert\Delta\mathcal{T}^{(3)}\rvert$	$\lvert\mathcal{E}_{\leq 4}\rvert$	$\lvert\mathcal{R}_{\leq 4}\rvert$	$\lvert\Delta\mathcal{T}^{(4)}\rvert$	$\lvert\mathcal{E}_{\leq 5}\rvert$	$\lvert\mathcal{R}_{\leq 5}\rvert$	$\lvert\Delta\mathcal{T}^{(5)}\rvert$
ENTITY	2,909	233	46,388	5,817	236	72,111	8,275	236	73,785	11,633	237	70,506	14,541	237	47,326
RELATION	11,560	48	98,819	13,343	96	93,535	13,754	143	66,136	14,387	190	30,032	14,541	237	21,594
FACT	10,513	237	62,024	12,779	237	62,023	13,586	237	62,023	13,894	237	62,023	14,541	237	62,023
HYBRID	8,628	86	57,561	10,040	102	20,873	12,779	151	88,017	14,393	209	103,339	14,541	237	40,326
GraphEqual	2,908	226	57,636	5,816	235	62,023	5,816	235	62,023	11,632	237	62,023	14,541	237	66,411
GraphHigher	900	197	10,000	1,838	221	20,000	3,714	234	40,000	7,467	237	80,000	14,541	237	160,116
GraphLower	7,505	237	160,000	11,258	237	80,000	13,134	237	40,000	14,072	237	20,000	14,541	237	10,116

$\lvert\mathcal{E}_{\leq s}\rvert$, $\lvert\mathcal{R}_{\leq s}\rvert$, and $\lvert\Delta\mathcal{T}^{(s)}\rvert$ denote the cumulative number of entities, relations, and newly introduced triples at snapshot s, respectively.

BAKE Li et al. (2025a), SAGE Li et al. (2025b), ETT-CKGE Zhu et al. (2025), STCKGE Wang et al. (2025), and ERPP Yang et al. (2025). Most baselines were evaluated directly using publicly available code and their reported hyperparameter settings under an identical random seed. For studies whose code was not publicly released, we adopted the performance values reported in the original papers. In addition, the implementation setting of TRM is described in detail in Appendix C.

4.2 RANKING PERFORMANCE

Table 2: Ranking performance comparison on ENTITY, RELATION, FACT, and HYBRID.

	ENTITY			RELATION			FACT			HYBRID		
Model	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
LKGE	0.243	0.143	0.439	0.174	0.091	0.340	0.209	0.122	0.389	0.202	0.114	0.375
IncDE	0.249	0.147	0.444	0.200	0.112	0.369	0.217	0.127	0.393	0.222	0.129	0.399
SAGE	0.252	0.155	0.438	0.193	0.108	0.356	0.220	0.133	0.392	0.212	0.125	0.377
BAKE	0.277	0.169	0.483	0.203	0.124	0.350	0.189	0.117	0.321	0.226	0.144	0.383
ETT-CKGE	0.255	0.153	0.450	0.200	0.112	0.366	0.216	0.125	0.396	0.224	0.132	0.401
STCKGE†	0.269	0.206	0.398	0.244	0.180	0.374	0.249	0.192	0.367	0.248	0.189	0.361
ERPP†	<u>0.386</u>	<u>0.283</u>	<u>0.583</u>	<u>0.303</u>	<u>0.215</u>	<u>0.475</u>	**0.307**	**0.215**	<u>0.487</u>	<u>0.316</u>	<u>0.223</u>	<u>0.502</u>
TRM	**0.410**	**0.294**	**0.635**	**0.334**	**0.230**	**0.538**	<u>0.294</u>	<u>0.198</u>	**0.489**	**0.337**	**0.234**	**0.540**

† denotes results from original papers. **Bold** is best and <u>underline</u> is second best.

Table 2 and Table 3 compare final ranking performance across seven CKGE benchmarks. We report MRR, Hits@1, and Hits@10 for the link prediction task. The first table covers ENTITY, RELATION, FACT, and HYBRID, which reflect different growth patterns in entities, relations, facts, and their combinations. The second table covers GraphEqual, GraphHigher, and GraphLower, which vary the amount of newly added triples across snapshots.

On ENTITY, RELATION, and HYBRID, TRM achieves the best MRR, Hits@1, and Hits@10, showing strong performance under entity-driven, relation-driven, and mixed growth settings. These results support the effectiveness of reusing entity-side residual memory when the evolving KG can still benefit from interaction patterns formed in previous snapshots. On FACT, however, ERPP records the highest MRR and Hits@1, while TRM ranks second on these metrics and achieves the highest Hits@10. This suggests that TRM remains competitive in fact-accumulation settings, but relation-path based accumulation can be more effective when the relation set remains stable and new facts are continuously added. A similar tendency appears on GraphEqual, GraphHigher, and GraphLower. ERPP achieves the best MRR and Hits@1 on all three datasets. TRM remains competitive on GraphEqual and GraphLower, where it records second-best performance. However, on GraphHigher, where the number of newly added triples increases over snapshots, TRM shows lower performance than ERPP and STCKGE. This suggests that the benefit of reusing past residual memory may become less pronounced when later snapshots introduce increasingly large amounts of new factual information.

Another notable observation is that methods using TransE as their backbone generally show lower ranking performance than ERPP, STCKGE, and TRM. This trend is observed not only for earlier methods such as LKGE and IncDE, but also for more recent TransE-based approaches such as SAGE,

Table 3: Ranking performance on GraphEqual, GraphHigher, and GraphLower.

	GraphEqual			GraphHigher			GraphLower		
Model	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
LKGE	0.225	0.127	0.423	0.218	0.126	0.400	0.215	0.121	0.410
IncDE	0.235	0.133	0.436	0.227	0.132	0.414	0.223	0.124	0.420
SAGE	0.231	0.134	0.424	0.237	0.142	0.423	0.213	0.121	0.398
BAKE	0.253	0.150	0.458	0.240	0.140	0.440	0.248	0.146	0.452
ETT-CKGE	0.229	0.130	0.425	0.226	0.131	0.414	0.206	0.114	0.390
STCKGE†	0.288	0.215	0.436	0.331	0.253	0.489	0.251	0.186	0.385
ERPP†	**0.340**	**0.237**	0.547	**0.361**	**0.255**	**0.572**	**0.346**	**0.238**	**0.564**
TRM	0.339	0.228	**0.568**	0.323	0.212	0.547	0.307	0.200	0.530

BAKE, and ETT-CKGE. This suggests that, although continual learning mechanisms can improve knowledge transfer and mitigate forgetting, final ranking performance may still be constrained by the expressiveness of the underlying TransE scoring function.

Overall, the main ranking results show both the strengths and limitations of TRM. TRM is highly effective when entity-driven, relation-driven, or mixed changes are prominent, as shown by its best performance on ENTITY, RELATION, and HYBRID. However, its advantage becomes weaker when the main challenge is the continual accumulation of new facts, especially when the amount of newly added triples grows over time. This is because TRM combines current entity representations with residual memories from past snapshots, whose relative benefit may decrease when later snapshots are dominated by newly added factual information. These results suggest that entity-side memory reuse is an effective strategy for CKGE, while also indicating that additional mechanisms may be needed to better handle rapidly increasing factual updates.

4.3 KNOWLEDGE RETENTION CAPABILITY

Table 4: Knowledge retention capability comparison on CKGE benchmarks in terms of ROD and RTE.

	ENTITY		RELATION		FACT		HYBRID		GraphEqual		GraphHigher		GraphLower	
Model	ROD	RTE	ROD	RTE	ROD	RTE	ROD	RTE	ROD	RTE	ROD	RTE	ROD	RTE
LKGE	-0.008	-0.006	-0.217	-0.184	-0.074	-0.039	-0.081	-0.055	-0.007	-0.005	-0.004	-0.002	-0.027	-0.022
IncDE	-0.066	-0.045	-0.124	-0.101	-0.020	0.000	-0.062	-0.044	-0.017	-0.017	-0.008	-0.006	-0.018	-0.024
SAGE	-0.053	-0.042	-0.210	-0.178	-0.060	-0.031	-0.092	-0.063	-0.033	-0.026	-0.013	-0.009	-0.070	-0.051
BAKE	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ETT-CKGE	0.004	0.003	-0.147	-0.130	-0.038	-0.013	-0.014	-0.010	-0.047	-0.036	0.007	0.007	-0.088	-0.083
STCKGE†	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERPP†	0.003	-	-0.001	-	**0.001**	-	-0.011	-	0.001	-	**0.033**	-	0.002	-
TRM	**0.025**	**0.020**	**0.023**	**0.019**	-0.010	**0.009**	**0.002**	**0.003**	**0.033**	**0.028**	0.020	**0.018**	**0.009**	**0.016**

Table 4 compares the knowledge retention capability of each model in terms of ROD and RTE. This experiment evaluates how stably the knowledge formed in earlier snapshots is preserved after learning new snapshots. The comparison is particularly important because it directly contrasts ERPP, which first reported positive knowledge retention metrics on some benchmarks Yang et al. (2025), with TRM, which recombines entity-side residual memory for current prediction. ROD compares the performance measured immediately after learning each past snapshot with that measured at the final stage and can therefore be interpreted as a metric of ultimate retention Yang et al. (2025). By contrast, RTE is computed by averaging the relative changes across all snapshot boundaries after each snapshot as shown in Eq. 2. It therefore serves as a trajectory-aware retention metric that reflects not only whether performance is eventually recovered, but also how long and how severely it deteriorates during the intermediate process. Taken together, the two metrics capture both final retention and retention stability throughout training.

Overall, TRM shows stronger and more consistent retention than existing baselines. LKGE, IncDE, and SAGE record negative ROD and RTE on most benchmarks, and ETT-CKGE also shows mostly

negative values except on ENTITY and GraphHigher. Among the baselines, ERPP shows the strongest ROD behavior, achieving positive ROD on five of the seven benchmarks. Against this strong baseline, TRM achieves positive ROD on six benchmarks, with FACT being the only negative case. TRM records the highest ROD on ENTITY, RELATION, HYBRID, GraphEqual, and GraphLower, showing particularly strong final retention when growth involves entities, relations, or mixed changes. ERPP remains stronger on FACT and GraphHigher, suggesting that relation-centered accumulation can be more effective when new facts continuously accumulate or when the number of newly added triples increases over snapshots.

From the perspective of RTE, TRM shows the most stable retention trajectory overall. It records positive RTE on all seven benchmarks and achieves the highest RTE on all of them. This indicates that TRM does not merely improve final-stage retention, but also maintains a stable retention trajectory across intermediate snapshot boundaries.

In summary, the retention results show that TRM consistently contributes to forgetting mitigation across diverse growth scenarios. Even in FACT and GraphHigher, where TRM is not the strongest model in final ranking performance, it maintains positive RTE, and its ROD is positive on all benchmarks except FACT. Thus, these results highlight TRM as a retention-oriented mechanism that stabilizes past knowledge during continual KG growth.

4.4 Ablation study

Table 5: Ablation study on TRM.

	ENTITY		RELATION		FACT		HYBRID		GraphEqual		GraphHigher		GraphLower	
Setting	MRR	RTE	MRR	RTE	MRR	RTE	MRR	RTE	MRR	RTE	MRR	RTE	MRR	RTE
TRM	**0.410**	**0.020**	**0.334**	0.019	**0.294**	**0.009**	**0.337**	0.003	**0.339**	**0.028**	**0.323**	0.018	**0.307**	**0.016**
w/o Memory	0.393	0.017	0.316	**0.023**	0.277	-0.027	0.323	**0.009**	0.329	0.019	0.321	**0.030**	0.301	-0.004
w/o Freeze	0.376	-0.067	0.276	-0.115	0.244	-0.154	0.320	-0.021	0.311	-0.056	0.316	-0.031	0.287	-0.028

Table 5 presents an ablation study on how the core components of TRM affect final performance and retention trajectory. Here, 'w/o Memory' denotes a setting in which snapshot-wise entity residual memory is removed, leaving only the basic point-bump structure of BoxE and drift regularization. In this setting, $\boldsymbol{\rho}_e^s$ is not used, and the final bump is simplified to $\tilde{\mathbf{b}}_e^s = \mathbf{b}_e$. 'w/o Freeze' denotes a setting in which entity residual memory aggregation is retained, but past entity residual memories are not frozen, so they can also be updated during training on the current snapshot. In addition, an experiment replacing uniform memory aggregation with dynamically weighted aggregation is discussed in Appendix D.

The memory-removed variant shows relatively stable retention on several datasets even without explicit memory. This suggests that the point-bump decomposition of BoxE provides a useful basis for handling changing entity representations in the continual setting. Nevertheless, the full TRM achieves higher MRR on all seven datasets and maintains positive RTE throughout. This demonstrates that aggregating snapshot-wise entity residual memory and recombining it into the current scoring path consistently improves final ranking performance while preserving positive retention trajectories across all datasets. However, the memory-removed variant shows higher RTE on RELATION, HYBRID, and GraphHigher, suggesting that residual memory reuse does not uniformly maximize retention under all growth patterns. Even so, the memory-removed variant fails to maintain positive RTE on FACT and GraphLower, whereas the full TRM keeps RTE positive on all datasets. The comparison with the non-frozen variant shows that freezing past memory is critical in TRM. The non-frozen variant records lower MRR than the full TRM on all seven datasets, and its RTE becomes negative on every dataset. This indicates that simply accumulating entity residual memory and continuing to update it is insufficient for stable preservation of past knowledge. Overall, these results suggest that the point-bump structure of BoxE provides a useful foundation in the continual setting, while freezing snapshot-wise entity residual memory allows past entity-side representations to be utilized more stably.

Table 6: Extension of TRM to TransE and DistMult.

	ENTITY		RELATION		FACT		HYBRID		GraphEqual		GraphHigher		GraphLower	
Model	MRR	RTE	MRR	RTE	MRR	RTE	MRR	RTE	MRR	RTE	MRR	RTE	MRR	RTE
TransE	0.181	-0.402	0.123	-0.491	0.153	-0.266	0.143	-0.358	0.189	-0.205	0.205	-0.218	0.185	-0.171
TransE + Memory	0.248	-0.116	0.157	-0.325	0.193	-0.101	0.193	-0.142	0.212	-0.046	0.219	-0.104	0.217	-0.033
Δ	**+0.067**	**+0.286**	**+0.034**	**+0.166**	**+0.040**	**+0.165**	**+0.050**	**+0.216**	**+0.023**	**+0.159**	**+0.014**	**+0.114**	**+0.032**	**+0.138**
DistMult	0.287	-0.069	0.213	-0.030	0.187	0.099	0.230	-0.019	0.260	0.015	0.232	-0.023	0.227	-0.008
DistMult + Memory	0.321	-0.011	0.247	0.042	0.218	0.064	0.250	0.019	0.275	0.064	0.251	0.004	0.241	0.048
Δ	**+0.034**	**+0.058**	**+0.034**	**+0.072**	**+0.031**	-0.035	**+0.020**	**+0.038**	**+0.015**	**+0.049**	**+0.019**	**+0.027**	**+0.014**	**+0.056**

Δ denotes the difference between each memory-extended model and its corresponding base model.

4.5 Extension to other backbones

To examine whether the entity-side memory reuse idea underlying TRM is specific to BoxE or can also mitigate forgetting in other KGE backbones, we apply this memory idea to TransE and DistMult. In these variants, snapshot-wise residual memory is aggregated and added to each entity representation before applying the original scoring function of the corresponding backbone. Detailed integration schemes and loss sensitivity analysis are provided in Appendix E. Table 6 compares the original backbones with their memory-extended variants.

For TransE, the memory-extended variant improves both MRR and RTE on all seven datasets, indicating that entity-side temporal residuals are effective even under translation-based scoring. For DistMult, the memory-extended variant improves MRR on all datasets and RTE on six datasets except FACT, suggesting that memory reuse also benefits semantic-matching scoring in most cases. Moreover, even without memory extension, the base DistMult model already achieves competitive ranking performance compared with recent TransE-based CKGE methods. This suggests that final ranking performance depends not only on the continual learning strategy, but also on the scoring geometry of the underlying backbone. Overall, these results show that the memory reuse strategy underlying TRM generalizes beyond BoxE and contributes to both final performance and retention.

5 Conclusion

This study targets catastrophic forgetting in Continual Knowledge Graph Embedding by reframing it not merely as preserving past knowledge, but as reusing entity-side information formed in past snapshots for current learning and prediction. Based on this view, we proposed Temporal Residual Memory (TRM). TRM maintains the backbone relation-side scoring structure while handling changes in entity-side representations through snapshot-wise entity residual memory. It then injects the aggregation of these entity residual memories through the BoxE bump path into the current scoring path, thereby incorporating entity-side information from past snapshots into current prediction.

Experimental results show that TRM is especially effective under entity-driven, relation-driven, and mixed growth settings, while achieving stable retention behavior across diverse CKGE benchmarks. In particular, TRM achieves positive ROD and RTE on most datasets, indicating that snapshot-wise entity residual memory reuse consistently contributes to forgetting mitigation. The ranking results also indicate that TRM is less dominant in fact-accumulation and triple-increment variation settings, where relation-path accumulation can be more advantageous for final prediction accuracy. Nevertheless, TRM maintains stable retention behavior across these settings.

In addition, experiments extending the core memory reuse idea of TRM to TransE and DistMult show a general tendency toward forgetting mitigation, suggesting that entity-side residual memory reuse is not limited to the BoxE setting. Future work should investigate how to reduce the additional computational cost introduced by the cross-bump structure of BoxE and how to combine entity-side memory reuse with mechanisms that better handle rapidly increasing factual updates.

AI use statement

In this work, we used generative AI tools to assist with English translation and to improve the grammar, readability, and writing style of the manuscript. We also used generative AI tools for

retrieval and discovery purposes, such as identifying potentially relevant related work, and to assist with research execution through code-related tasks, including code refactoring, writing experimental scripts, and implementing utilities for logging experimental results. We did not use generative AI tools for research ideation, generating synthetic datasets, or proving mathematical claims. All AI-assisted writing and code were reviewed and revised by the authors, and the code was tested to verify its correctness and consistency with the intended experimental procedures. Literature identified with the aid of generative AI was manually verified against the original sources before being cited or discussed in the manuscript. We take responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the aid of generative AI.

Reproducibility statement

Implementation details and experimental settings are provided in the Appendix C. The source code for reproducing our experiments is included in the supplementary material.

References

Ralph Abboud, Ismail Ceylan, Thomas Lukasiewicz, and Tommaso Salvatori. Boxe: A box embedding model for knowledge base completion. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9649–9661. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6dbbe6abe5f14af882ff977fc3f35501-Paper.pdf.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - a crystallization point for the web of data. *Web Semant.*, 7(3):154–165, September 2009. ISSN 1570-8268. doi: 10.1016/j.websem.2009.07.002. URL https://doi.org/10.1016/j.websem.2009.07.002.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf.

Jiahang Cao, Jinyuan Fang, Zaiqiao Meng, and Shangsong Liang. Knowledge graph embedding: A survey from the perspective of representation spaces. *ACM Comput. Surv.*, 56(6), March 2024a. ISSN 0360-0300. doi: 10.1145/3643806. URL https://doi.org/10.1145/3643806.

Jiahang Cao, Jinyuan Fang, Zaiqiao Meng, and Shangsong Liang. Knowledge graph embedding: A survey from the perspective of representation spaces. *ACM Computing Surveys*, 56(6):1–42, 2024b.

Yuanning Cui, Yuxin Wang, Zequn Sun, Wenqiang Liu, Yiqiao Jiang, Kexin Han, and Wei Hu. Lifelong embedding learning and transfer for growing knowledge graphs. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i4.25539. URL https://doi.org/10.1609/aaai.v37i4.25539.

Angel Daruna, Mehul Gupta, Mohan Sridharan, and Sonia Chernova. Continual learning of knowledge graph embeddings. *IEEE Robotics and Automation Letters*, 6(2):1128–1135, 2021. doi: 10.1109/LRA.2021.3056071.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, pp. 413–422, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450320351. doi: 10.1145/2488388.2488425. URL https://doi.org/10.1145/2488388.2488425.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, 2022. doi: 10.1109/TNNLS.2021.3070843.

Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. doi: 10.1609/aaai.v32i1.11651. URL https://ojs.aaai.org/index.php/AAAI/article/view/11651.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

Linyu Li, Zhi Jin, Yuanpeng He, Dongming Jin, Yichi Zhang, Haoran Duan, Xuan Zhang, Zhengwei Tao, and Nyima Tash. Learning to evolve: Bayesian-guided continual knowledge graph embedding. *arXiv preprint arXiv:2508.02426*, 2025a.

Yifei Li, Lingling Zhang, Hang Yan, Tianzhe Zhao, Zihan Ma, Muye Huang, and Jun Liu. Sage: Scale-aware gradual evolution for continual knowledge graph embedding. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, pp. 1600–1611, New York, NY, USA, 2025b. Association for Computing Machinery. ISBN 9798400714542. doi: 10.1145/3711896.3737115. URL https://doi.org/10.1145/3711896.3737115.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), Feb. 2015. doi: 10.1609/aaai.v29i1.9491. URL https://ojs.aaai.org/index.php/AAAI/article/view/9491.

Huihui Liu, Yiding Yang, and Xinchao Wang. Overcoming catastrophic forgetting in graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 8653–8661, 2021.

Jiajun Liu, Wenjun Ke, Peng Wang, Ziyu Shang, Jinhua Gao, Guozheng Li, Ke Ji, and Yanhe Liu. Towards continual knowledge graph embedding via incremental distillation. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i8.28722. URL https://doi.org/10.1609/aaai.v38i8.28722.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6470–6479, Red Hook, NY, USA, 2017a. Curran Associates Inc. ISBN 9781510860964.

David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017b. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/f87522788a2be2d171666752f97ddebb-Paper.pdf.

Vinay Venkatesh Ramasesh, Ethan Dyer, and Maithra Raghu. Anatomy of catastrophic forgetting: Hidden representations and task semantics. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=LhY8QdUGSuw.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Hyun-Je Song and Seong-Bae Park. Enriching translation-based knowledge graph embeddings through continual learning. *IEEE Access*, 6:60489–60497, 2018. doi: 10.1109/ACCESS.2018.2874656.

Hong Wang, Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Shiyu Chang, and William Yang Wang. Sentence embedding alignment for lifelong relation extraction. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *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)*, pp. 796–806, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1086. URL https://aclanthology.org/N19-1086/.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024a. doi: 10.1109/TPAMI.2024.3367329.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024b. doi: 10.1109/TPAMI.2024.3367329.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12): 2724–2743, 2017. doi: 10.1109/TKDE.2017.2754499.

Xinyan Wang, Jinshuo Liu, Kaijian Xie, Meng Wang, Cheng Bi, Juan Deng, Donghong Ji, and Jeff Z. Pan. Stckge: Continual knowledge graph embedding based on spatial transformation. *Knowledge-Based Systems*, 329:114337, 2025. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2025.114337. URL https://www.sciencedirect.com/science/article/pii/S0950705125013760.

Yu Xie, Chunyi Li, Bin Yu, Chen Zhang, and Zhouhua Tang. A survey on dynamic network embedding. *arXiv preprint arXiv:2006.08093*, 2020.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

Jing Yang, Xinfa Jiang, Xiaowen Jiang, Yuan Gao, Laurence T. Yang, Shaojun Zou, and Shundong Yang. From knowledge forgetting to accumulation: Evolutionary relation path passing for lifelong knowledge graph embedding. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, pp. 1197–1206, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400715921. doi: 10.1145/3726302.3729982. URL https://doi.org/10.1145/3726302.3729982.

Qiao Yuan, Sheng-Uei Guan, Pin Ni, Tianlun Luo, Prudence Wong, Victor Chang, and Ka Lok Man. Continual graph learning: A survey. *Pattern Recognition*, 179:113600, 2026. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2026.113600. URL https://www.sciencedirect.com/science/article/pii/S0031320326005662.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 3987–3995. JMLR.org, 2017.

Fan Zhou and Chengtai Cao. Overcoming catastrophic forgetting in graph neural networks with experience replay. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 4714–4722, 2021.

Lijing Zhu, Qizhen Lan, Qing Tian, Wenbo Sun, Li Yang, Lu Xia, Yixin Xie, Xi Xiao, Tiehang Duan, Cui Tao, and Shuteng Niu. Ett-ckge: Efficient task-driven tokens for continual knowledge graph embedding. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2025, Porto, Portugal, September 15–19, 2025, Proceedings, Part VI*, pp. 481–496, Berlin, Heidelberg, 2025. Springer-Verlag. ISBN 978-3-032-06105-8. doi: 10.1007/978-3-032-06106-5_28. URL https://doi.org/10.1007/978-3-032-06106-5_28.

A PRELIMINARIES

Growing knowledge graph. We represent a growing knowledge graph as a sequence of cumulative snapshots $\mathcal{G} = \{\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \ldots, \mathcal{G}^{(S)}\}$, where each snapshot is defined as $\mathcal{G}^{(s)} = (\mathcal{E}_{\leq s}, \mathcal{R}_{\leq s}, \mathcal{T}_{\leq s})$. Here, $\mathcal{E}_{\leq s}$, $\mathcal{R}_{\leq s}$, and $\mathcal{T}_{\leq s}$ denote the sets of entities, relations, and triples observed up to snapshot s, respectively. Each triple is written as $(h, r, t) \in \mathcal{T}_{\leq s}$, where $h, t \in \mathcal{E}_{\leq s}$ and $r \in \mathcal{R}_{\leq s}$. The graph expands over time such that $\mathcal{E}_{\leq s-1} \subseteq \mathcal{E}_{\leq s}$, $\mathcal{R}_{\leq s-1} \subseteq \mathcal{R}_{\leq s}$, and $\mathcal{T}_{\leq s-1} \subseteq \mathcal{T}_{\leq s}$. We denote the newly introduced entities, relations, and triples at snapshot s by $\Delta\mathcal{E}^{(s)} = \mathcal{E}_{\leq s} - \mathcal{E}_{\leq s-1}$, $\Delta\mathcal{R}^{(s)} = \mathcal{R}_{\leq s} - \mathcal{R}_{\leq s-1}$, and $\Delta\mathcal{T}^{(s)} = \mathcal{T}_{\leq s} - \mathcal{T}_{\leq s-1}$, respectively, where $\mathcal{E}_{\leq 0} = \mathcal{R}_{\leq 0} = \mathcal{T}_{\leq 0} = \emptyset$.

Continual knowledge graph embedding. Given a knowledge graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, knowledge graph embedding aims to learn low-dimensional representations of entities and relations. For each triple $(h, r, t) \in \mathcal{T}$, the corresponding embeddings are denoted by $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^d$, where d is the embedding dimension.

Given a growing KG represented as the snapshot sequence $\{\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \ldots, \mathcal{G}^{(S)}\}$, continual knowledge graph embedding aims to continually update the embeddings of previously observed entities and relations while learning embeddings for newly introduced entities and relations. At snapshot s, the model updates the old embeddings in $\mathcal{E}_{\leq s-1}$ and $\mathcal{R}_{\leq s-1}$, and learns new embeddings for $\Delta\mathcal{E}^{(s)}$ and $\Delta\mathcal{R}^{(s)}$ based on the newly arrived triples $\Delta\mathcal{T}^{(s)}$. As a result, embeddings for all entities in $\mathcal{E}_{\leq s}$ and relations in $\mathcal{R}_{\leq s}$ are obtained after training on snapshot s.

B LIMITATIONS

Despite these findings, several limitations remain. First, although TRM improves retention by reusing snapshot-wise entity residual memory, the BoxE cross-bump structure requires memory-augmented representations to be recomposed for each candidate entity before scoring. This candidate-wise recomposition increases evaluation cost, as discussed in Appendix C.4, and may limit scoring efficiency on large-scale knowledge graphs. Second, our experiments used existing CKGE benchmarks with diverse growth patterns, but these benchmarks remain relatively small compared with real-world knowledge graphs and may not fully capture their scale and complexity. More realistic large-scale CKGE benchmarks are therefore needed to further validate the practical applicability of TRM, particularly benchmarks with larger numbers of entities, relations, and facts, as well as imbalanced graph growth and long-term evolution. Third, while TRM provides stable retention, its ranking advantage is less pronounced when growth is dominated by newly added facts or changing triple increments. This suggests that entity-side residual memory reuse may need to be combined with mechanisms that better exploit newly expanded factual structure. Future work should therefore examine the scalability and stability of TRM in more realistic large-scale settings, develop efficient scoring methods that reduce the cost of candidate-wise recomposition, and explore mechanisms that better handle rapidly increasing factual updates.

C ADDITIONAL EXPERIMENTAL DETAILS

C.1 IMPLEMENTATION DETAILS

TRM was implemented in PyTorch, and all experiments were conducted using an RTX 4080 (16GB) GPU. All benchmarks were evaluated under a continual learning setting consisting of five snapshots. Among the baselines reported in the main text, methods with publicly available code were directly reproduced using the same fixed random seed and benchmark splits. For methods whose code was not publicly released, we used the values explicitly reported in the original papers.

TRM was trained in a 700-dimensional embedding space using the Adam optimizer with a learning rate of 5×10^{-4}. Entity points, base bumps, and role-specific relation-box parameters were initialized uniformly in the range $[-0.5/\sqrt{d}, 0.5/\sqrt{d}]$, while snapshot-wise entity residual memories were initialized from a zero-mean Gaussian distribution with standard deviation 10^{-3}. When the graph expanded, parameters of previously observed entities and relations were copied from the previous snapshot, and only newly introduced parameters were randomly initialized. The batch size was set

to 512, and the maximum number of epochs was set to 1000. For each positive triple, 10 negative triples were sampled during training. Validation performance was measured every 10 epochs, and early stopping was applied with a patience of 2. During evaluation, the evaluation batch size was set to 100, and the score batch chunk size and score chunk size were set to 512 and 1024, respectively, to control GPU memory usage during scoring. Snapshot-wise entity residual memory was aggregated using birth-aware uniform aggregation, and residual memory from past snapshots was frozen during training on the current snapshot. Drift regularization across snapshots was applied with $\lambda_{\text{evo}} = 0.01$.

Evaluation was performed following the filtered link prediction protocol. Tail prediction was evaluated with the query $(h, r, ?)$, and head prediction was converted into the query $(t, r^{-1}, ?)$ using the inverse relation and evaluated through the same tail-ranking procedure. At a given snapshot s, candidate tails were selected from the entity set observed up to the current snapshot, $\mathcal{E}_{\leq s}$, and positive triples present in the training data were filtered out during ranking. Final ranking performance was reported using MRR, Hits@1, and Hits@10, and knowledge retention capability was evaluated using ROD and the RTE defined in Eq. 2.

C.2 Embedding dimension analysis

Table 7: Embedding dimension sweep on seven CKGE benchmarks. MRR, ROD, and RTE are averaged over the seven datasets. Training time is reported in seconds.

d	Avg. MRR	Avg. ROD	Avg. RTE	Pos. ROD	Pos. RTE	Avg. Time (s)
100	0.3189	-0.0137	-0.0028	3/7	3/7	2462.3
300	**0.3356**	0.0077	0.0126	5/7	6/7	5571.6
500	**0.3356**	0.0098	0.0132	4/7	7/7	8045.3
700	0.3349	**0.0145**	**0.0163**	**6/7**	**7/7**	10390.3
900	0.3329	0.0144	0.0160	5/7	6/7	14718.2

To analyze the effects of embedding dimension on the performance, retention, and training cost of TRM, we conducted additional experiments with $d \in \{100, 300, 500, 700, 900\}$. Table 7 summarizes the average MRR, average ROD, average RTE, the numbers of datasets with positive ROD and RTE values, and the average training time over seven CKGE benchmarks. All MRR, ROD, and RTE values were rounded to four decimal places.

As shown in Table 7, increasing the embedding dimension from the 100-dimensional configuration to the 300-dimensional configuration substantially improves the average MRR. However, the 500-dimensional configuration shows the same average MRR as the 300-dimensional configuration, and the average MRR slightly decreases when moving from the 700-dimensional configuration to the 900-dimensional configuration. In contrast, the retention metrics generally improve as the embedding dimension increases up to the 700-dimensional configuration. The 700-dimensional configuration achieves the highest average ROD and average RTE, while also maintaining positive RTE on all seven datasets and positive ROD on six datasets. Moving from 700 to 900 dimensions does not improve average MRR or retention, but increases training time substantially. Therefore, we used the 700-dimensional configuration as the main setting, considering that it provides the most stable balance between final performance, retention stability, and training cost.

C.3 HYPERPARAMETER SENSITIVITY

Table 8: Effect of the margin value on TRM. MRR, ROD, and RTE are averaged over the seven datasets.

Value	Avg. MRR	Avg. ROD	Avg. RTE	Pos. ROD	Pos. RTE
γ=3	0.3291	**0.0183**	**0.0193**	5/7	6/7
γ=5	0.3349	0.0145	0.0163	**6/7**	**7/7**
γ=7	**0.3381**	0.0091	0.0127	5/7	**7/7**

Margin value. Table 8 summarizes the performance changes of TRM as a function of the margin value γ in terms of the average MRR, average ROD, and average RTE over the seven datasets, as well as the number of datasets in which each retention metric has a positive value. Overall, the average MRR increases as γ becomes larger, and $\gamma = 7$ records the highest average MRR. This suggests that a larger margin can enforce a stronger separation between positive and negative triples, thereby improving final ranking performance. However, the retention metrics show a different pattern. $\gamma = 3$ records the highest average ROD and average RTE, indicating that a smaller margin can be beneficial for average retention. At the same time, it shows the lowest average MRR, and ROD and RTE are positive on only five and six of the seven datasets, respectively. This suggests that, although a smaller margin may help preserve past knowledge on average, it may not provide a sufficient discriminative signal for new snapshots, thereby limiting final ranking performance.

In contrast, $\gamma = 7$ achieves the highest average MRR, but its average ROD and average RTE are lower than those of $\gamma = 3$ and $\gamma = 5$, and positive ROD is observed on only five of the seven datasets. Compared with $\gamma = 7$, $\gamma = 5$ shows a slightly lower average MRR, but still maintains strong average ranking performance. It also records positive ROD on six of the seven datasets and positive RTE on all datasets. Thus, $\gamma = 5$ yields the fewest non-positive retention metric values among the three margin values and offers the most balanced trade-off between final performance and retention stability. Therefore, rather than maximizing final ranking performance alone, we selected $\gamma = 5$ as the default setting by considering the balance between adaptation to new knowledge and retention of past knowledge.

Table 9: Effect of the parameter regularization weight on TRM. MRR, ROD, and RTE are averaged over the seven datasets.

Weight	Avg. MRR	Avg. ROD	Avg. RTE	Pos. ROD	Pos. RTE
λ_{evo}=10^{-4}	0.3266	-0.0281	-0.0111	3/7	4/7
λ_{evo}=10^{-3}	**0.3383**	0.0003	0.0089	5/7	6/7
λ_{evo}=10^{-2}	0.3349	**0.0145**	**0.0163**	**6/7**	**7/7**
λ_{evo}=10^{-1}	0.3270	0.0107	0.0079	**6/7**	6/7

Parameter regularization weight. Table 9 summarizes the performance changes of TRM as a function of the parameter regularization weight λ_{evo} in terms of the average MRR, average ROD, and average RTE over the seven datasets, as well as the number of datasets in which each retention metric has a positive value. Overall, when λ_{evo} is too small, the model does not sufficiently constrain parameter changes from past snapshots, leading to unstable retention performance. In particular, $\lambda_{\text{evo}} = 10^{-4}$ records the lowest average ROD and average RTE, and ROD and RTE are positive on only three and four of the seven datasets, respectively. This indicates that excessively weak regularization can be unfavorable for preserving past knowledge.

As the regularization strength increases from $\lambda_{\text{evo}} = 10^{-4}$, the retention metrics generally improve, although the improvement is not monotonic. $\lambda_{\text{evo}} = 10^{-2}$ records the highest average ROD and average RTE, with positive ROD on six of the seven datasets and positive RTE on all datasets.

$\lambda_{\text{evo}} = 10^{-1}$ also maintains positive ROD and RTE on six of the seven datasets, but its average MRR decreases to 0.3270. This suggests that stronger regularization can help preserve past knowledge to some extent, but may suppress adaptation to new snapshots and reduce final ranking performance. Meanwhile, $\lambda_{\text{evo}} = 10^{-3}$ records the highest average MRR, suggesting that it favors adaptation to new knowledge. However, its average ROD and average RTE are lower than those of $\lambda_{\text{evo}} = 10^{-2}$, and positive RTE is observed on only six of the seven datasets. Therefore, we selected $\lambda_{\text{evo}} = 10^{-2}$ as the default setting, as it maintains relatively strong average MRR while providing the best average retention scores and the most stable RTE behavior across datasets.

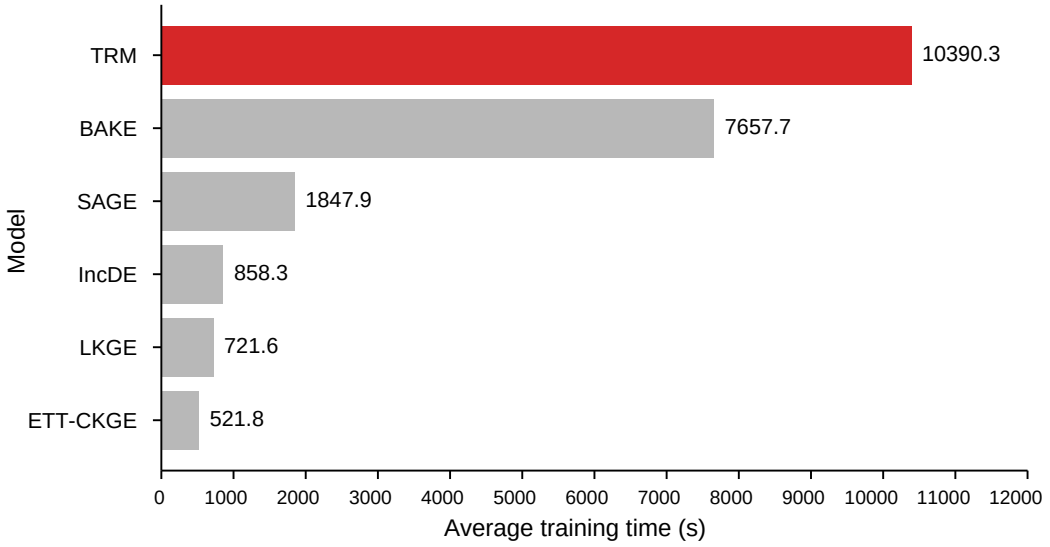

Figure 2: Average training time on seven CKGE benchmarks.

C.4 Computational cost analysis

Although TRM achieves strong ranking performance and stable knowledge retention, it entails a computational trade-off due to its BoxE-based architecture. In link prediction, conventional KGE backbones such as TransE and DistMult can compute scores for all candidate tails for a given query $(h, r, ?)$ using relatively simple vector operations or matrix multiplication. For example, DistMult can compute scores for all candidate entities at once in the form of $(\mathbf{h} \odot \mathbf{r})\mathbf{E}^\top$. In contrast, BoxE uses a cross-bump structure for a triple (h, r, t). Specifically, the head-role representation is constructed as $\mathbf{q}_h = \mathbf{p}_h + \mathbf{b}_t$ using the tail bump, while the tail-role representation is constructed as $\mathbf{q}_t = \mathbf{p}_t + \mathbf{b}_h$ using the head bump. Therefore, when evaluating the query $(h, r, ?)$, $\mathbf{q}_h$ also changes for each candidate tail t. In TRM, snapshot-wise residual memory is additionally incorporated, so for each candidate tail, $\tilde{\mathbf{b}}_t^s = \mathbf{b}_t + \boldsymbol{\rho}_t^s$ must first be constructed before computing the BoxE point-to-box distance. As a result, tail ranking in TRM involves candidate-wise cross-bump recomposition and box-distance scoring, and thus cannot be fully reduced to simple matrix multiplication.

The scoring cost of TRM therefore includes memory-augmented bump construction, cross-bump recomposition, and two role-specific point-to-box distance computations for each candidate entity. As in other KGE models that evaluate all candidate entities, the theoretical computational cost increases in proportion to the number of candidates. In practice, however, candidate-wise recomposition and box-distance computation introduce additional runtime overhead. In our implementation, both the candidate set and the query batch were processed in chunks to enable stable scoring over large candidate sets.

Fig. 2 compares the average training time across the seven CKGE benchmarks. ERPP and STCKGE were excluded from this comparison because their training times were not reported. The training time reported in this study includes periodic validation for early stopping. Therefore, the validation cost of computing ranking scores over all candidate entities is also reflected in the final training time.

TRM requires a longer training time than existing baselines, indicating a structural trade-off in which additional computational cost is incurred to achieve higher accuracy and more stable retention. Thus, although the memory reuse strategy of TRM is effective for mitigating forgetting, its BoxE-based scoring path increases the evaluation cost over large candidate entity sets. Future work should explore ways to reduce the cost of candidate-wise recomposition while preserving the expressive power of the cross-bump structure.

D TRM AGGREGATION ANALYSIS

Aggregation setup. In this study, we use the uniform aggregation of TRM as the reference setting and introduce five attention-based aggregation variants to analyze whether dynamically weighted aggregation provides additional benefits for snapshot residual memory. All variants were evaluated under the same training and evaluation settings as those used in the main text, and the comparison focuses on the aggregation rule itself. As in the main text, for an entity e at the current snapshot s, the valid memory set is defined as $\mathcal{V}_e^s = \{\tau \mid 1 \le \tau \le s,\ \tau \ge b(e)\}$, and the residual memory formed at snapshot τ is denoted by $\bar{\mathbf{m}}_e^\tau$. For notational simplicity in this section, we write the current trainable memory $\mathbf{m}_e^s$ as $\bar{\mathbf{m}}_e^s$ when it appears inside the aggregation formula. The original TRM aggregates past memory by uniform averaging, and the aggregated residual and final bump are defined as $\boldsymbol{\rho}_e^s$ and $\tilde{\mathbf{b}}_e^s = \mathbf{b}_e + \boldsymbol{\rho}_e^s$, respectively. The uniform and attention-based aggregation rules can be summarized as follows:

$$\boldsymbol{\rho}_e^s = \begin{cases} \dfrac{1}{|\mathcal{V}_e^s|} \displaystyle\sum_{\tau \in \mathcal{V}_e^s} \bar{\mathbf{m}}_e^\tau, & \text{(uniform aggregation)}, \\ \displaystyle\sum_{\tau \in \mathcal{V}_e^s} \alpha_{r,e}^\tau \, \bar{\mathbf{m}}_e^\tau, & \text{(attention-based aggregation)}. \end{cases} \tag{9}$$

Uniform aggregation averages all valid snapshot memories with equal weights, whereas attention-based aggregation adjusts the contribution of each snapshot memory through the relation-conditioned weight $\alpha_{r,e}^\tau$. The subsequent cross-bump scoring path of BoxE is the same as in the main text and is given by

$$\tilde{\mathbf{q}}_h^s = \mathbf{p}_h + \tilde{\mathbf{b}}_t^s, \qquad \tilde{\mathbf{q}}_t^s = \mathbf{p}_t + \tilde{\mathbf{b}}_h^s. \tag{10}$$

Therefore, this experiment does not modify the relation boxes or the point-to-box distance itself. It compares only the residual aggregation rule on the entity side.

Design axes of aggregation variants. The top three rows of Table 10 summarize the design axes of each variant. Along the query axis, U denotes the uniform baseline that does not use a query, while A1 integrates the relation-box geometry of the two roles into a single unified query. A2–A5 all use role-specific queries and compute snapshot attention while preserving role information. Along the key axis, A1–A3 use learnable snapshot keys, whereas A4 and A5 use snapshot-level memory keys. The mapping-role axis indicates how the role-specific query is connected to the current residual composition. Cross-role mapping reads head-side residual memory using the tail-role relation context and reads tail-side residual memory using the head-role relation context. Self-role mapping reads each residual memory using the relation context of the same role. Thus, this experiment is designed to compare not only the presence or absence of attention, but also how the role specificity of the query, the definition of the key, and cross-role or self-role mapping affect aggregation quality.

Query construction. Here, $\mathrm{concat}(\cdot)$ denotes the concatenation operation that joins multiple vectors into a single long vector. The unified query used in A1 is first constructed by compressing the relation-box geometry of the two roles into a single relation-level feature.

$$\mathbf{x}_r^{\mathrm{uni}} = \mathrm{concat}(\mathbf{c}_{r,h}, \log \boldsymbol{\omega}_{r,h}, \mathbf{c}_{r,t}, \log \boldsymbol{\omega}_{r,t}, \mathbf{c}_{r,t} - \mathbf{c}_{r,h}), \qquad \mathbf{q}_r^{\mathrm{uni}} = f_{\mathrm{uni}}\big(\mathbf{x}_r^{\mathrm{uni}}\big). \tag{11}$$

A1 therefore integrates the head-role box and the tail-role box into a single relation-level query, and this query induces a unified snapshot weighting mechanism for residual memory. A1 is the simplest way to introduce relation-box geometry into attention. At the same time, because it does not preserve role-specific structure, it serves as the first dynamic aggregation setting most directly contrasted with the uniform baseline.

In contrast, A2–A5 use role-specific queries to separate the geometry of the head role and the tail role. Each role query is defined as follows:

$$\begin{aligned} \mathbf{x}_{r,h} &= \mathrm{concat}(\mathbf{c}_{r,h}, \log \boldsymbol{\omega}_{r,h}), \qquad \mathbf{q}_{r,h} = f_h(\mathbf{x}_{r,h}), \\ \mathbf{x}_{r,t} &= \mathrm{concat}(\mathbf{c}_{r,t}, \log \boldsymbol{\omega}_{r,t}), \qquad \mathbf{q}_{r,t} = f_t(\mathbf{x}_{r,t}). \end{aligned} \tag{12}$$

Whereas A1 compresses the two role boxes into a single query, A2–A5 maintain separate role-specific queries and preserve asymmetric information in the relation geometry. Accordingly, these variants are used to evaluate whether preserving role information improves attention-based aggregation over the uniform baseline.

Key definition and attention weights. The definition of the key also clarifies the contrast with the uniform baseline. The learnable keys used in A1–A3 are parameters learned separately for each snapshot. In contrast, the memory keys used in A4 and A5 are defined by the memory itself, as introduced in the main text, for each snapshot. In other words, the key and value are identical. For each side, this can be written as

$$\mathbf{k}_h^{\tau,\mathrm{mem}} = \bar{\mathbf{m}}_h^{\tau}, \qquad \mathbf{k}_t^{\tau,\mathrm{mem}} = \bar{\mathbf{m}}_t^{\tau}. \tag{13}$$

Thus, A1–A3 can be interpreted as settings that examine how well the current relation query matches learned snapshot keys, whereas A4 and A5 can be interpreted as settings that examine how well the current relation query matches the residual memory from past snapshots. In other words, learnable keys and memory keys differ in whether snapshots are indexed by separately learned snapshot parameters or by the residual memory defined in the main text.

Based on this formulation, the attention weights are defined as follows. For A1, which uses the unified query and the global learnable key $\mathbf{k}^{\tau}$, the attention weight is given by

$$\alpha_{r,e}^{\tau,\mathrm{A1}} = \frac{\exp\Big((\mathbf{q}_r^{\mathrm{uni}})^{\top}\mathbf{k}^{\tau}/\sqrt{d}\Big)}{\sum_{\tau' \in \mathcal{V}_e^s} \exp\Big((\mathbf{q}_r^{\mathrm{uni}})^{\top}\mathbf{k}^{\tau'}/\sqrt{d}\Big)}. \tag{14}$$

Here, d denotes the dimensionality of the query and key vectors. In contrast, A2–A5 all use role-specific queries, so the general role-conditioned attention weight can be written as

$$\alpha_{r,u}^{\tau,(i;\kappa)} = \frac{\exp\Big(\mathbf{q}_{r,i}^{\top}\mathbf{k}_u^{\tau,(\kappa)}/\sqrt{d}\Big)}{\sum_{\tau' \in \mathcal{V}_u^s} \exp\Big(\mathbf{q}_{r,i}^{\top}\mathbf{k}_u^{\tau',(\kappa)}/\sqrt{d}\Big)}, \qquad u, i \in \{h, t\}, \quad \kappa \in \{\mathrm{learn}, \mathrm{mem}\}. \tag{15}$$

Here, i denotes the query role, and u denotes the target side whose residual memory is aggregated. In addition, $\kappa = \mathrm{learn}$ corresponds to A2 and A3, while $\kappa = \mathrm{mem}$ corresponds to A4 and A5. Therefore, the difference between the learnable-key variants A2 and A3 and the memory-key variants A4 and A5 lies not in the query, but in the source of the key.

Role mapping. The substantive difference among the variants lies in the query role used to compute the attention weights and the target-side residual composition to which the resulting weights are applied. Cross-role mapping and self-role mapping are respectively given by

$$\boldsymbol{\rho}_h^{s,\mathrm{cross}}(r;\kappa) = \sum_{\tau \in \mathcal{V}_h^s} \alpha_{r,h}^{\tau,(t;\kappa)}\, \bar{\mathbf{m}}_h^{\tau}, \qquad \boldsymbol{\rho}_t^{s,\mathrm{cross}}(r;\kappa) = \sum_{\tau \in \mathcal{V}_t^s} \alpha_{r,t}^{\tau,(h;\kappa)}\, \bar{\mathbf{m}}_t^{\tau}, \tag{16}$$

$$\boldsymbol{\rho}_h^{s,\mathrm{self}}(r;\kappa) = \sum_{\tau \in \mathcal{V}_h^s} \alpha_{r,h}^{\tau,(h;\kappa)}\, \bar{\mathbf{m}}_h^{\tau}, \qquad \boldsymbol{\rho}_t^{s,\mathrm{self}}(r;\kappa) = \sum_{\tau \in \mathcal{V}_t^s} \alpha_{r,t}^{\tau,(t;\kappa)}\, \bar{\mathbf{m}}_t^{\tau}. \tag{17}$$

Thus, A2 can be understood as cross-role mapping with learnable keys, A3 as self-role mapping with learnable keys, A4 as cross-role mapping with memory keys, and A5 as self-role mapping with memory keys. In contrast to the uniform baseline U, A1 can be interpreted as the simplest attention variant that adds only a unified query, A2 and A3 as variants that preserve role-specific queries, and A4 and A5 as variants that further combine role-specific queries with memory keys.

Table 10: TRM aggregation variants across all CKGE benchmarks. Best results are shown in bold.

Dataset	Metric	U	A1	A2	A3	A4	A5
query		Uniform	Unified	Role	Role	Role	Role
key		–	Learn.	Learn.	Learn.	Mem.	Mem.
mapping role		–	–	Cross	Self	Cross	Self
ENTITY	MRR	**0.410**	0.405	0.407	0.407	0.404	0.404
	ROD	**0.025**	-0.005	0.001	-0.001	-0.006	-0.011
	RTE	**0.020**	-0.011	-0.002	0.000	-0.004	-0.013
RELATION	MRR	**0.334**	0.325	0.325	0.324	0.322	0.326
	ROD	**0.023**	0.007	0.000	-0.007	-0.008	0.006
	RTE	**0.019**	-0.001	-0.008	-0.008	-0.006	0.004
FACT	MRR	0.294	0.292	0.294	0.294	**0.300**	0.296
	ROD	-0.010	-0.005	0.005	0.004	**0.032**	0.020
	RTE	0.009	0.010	0.016	0.016	**0.034**	0.028
HYBRID	MRR	**0.337**	0.327	0.326	0.329	0.328	0.318
	ROD	**0.002**	-0.017	-0.018	-0.013	-0.007	-0.048
	RTE	**0.003**	-0.047	-0.018	-0.018	-0.003	-0.035
GraphEqual	MRR	**0.339**	0.328	0.320	0.324	0.326	0.329
	ROD	**0.033**	0.031	0.003	0.006	0.016	0.032
	RTE	0.028	0.019	0.007	0.014	0.021	**0.038**
GraphHigher	MRR	**0.323**	0.312	0.312	0.312	0.316	0.317
	ROD	**0.020**	-0.025	-0.022	-0.017	0.004	0.003
	RTE	**0.018**	-0.026	-0.021	-0.017	0.004	0.003
GraphLower	MRR	**0.307**	0.303	0.304	0.303	0.305	0.305
	ROD	0.009	0.022	0.021	0.021	0.022	**0.029**
	RTE	0.016	0.022	0.022	0.022	0.024	**0.030**

Results. As shown in Table 10, uniform aggregation is the most robust default strategy across the benchmarks. It achieves the best MRR on five of the seven datasets and maintains positive RTE on all datasets, while the attention-based variants improve only on a subset of benchmarks. This suggests that, in CKGE, assigning equal weight to all valid snapshot memories can act as a useful inductive bias. It avoids over-selecting a particular past snapshot and instead preserves a stable mixture of entity-side residual information.

The attention-based variants show that dynamic weighting is not universally beneficial, although some variants provide selective gains on specific benchmarks. The simplest attention variant A1, which compresses the two role-specific relation boxes into a unified query, generally performs worse than uniform aggregation. This indicates that adding attention alone is insufficient, and that overly compressed relation context may not provide a reliable signal for selecting useful snapshot memories. A2 and A3 preserve role-specific queries, which makes them more expressive than A1, but their gains remain inconsistent. Thus, role-specific querying can make the attention signal more structured, but it does not by itself overcome the stability advantage of uniform aggregation.

The variants that use residual memory itself as the key, such as A4 and A5, show selective advantages in several settings. A4, which combines memory keys with cross-role mapping, achieves the best MRR, ROD, and RTE on FACT. A5, which combines memory keys with self-role mapping, achieves competitive retention performance on GraphEqual and GraphLower. These results indicate that directly using stored residual memories as retrieval keys can be beneficial in some cases, but its advantage remains setting-dependent and is not consistent enough to replace uniform aggregation.

Taken together, these results indicate that dynamic aggregation should be viewed as a conditional refinement rather than a replacement for uniform aggregation. Uniform aggregation provides the most reliable default for TRM because it consistently preserves positive retention while maintaining strong ranking performance. Attention-based aggregation can provide additional benefits in some cases, but its effectiveness is sensitive to the choice of keys and role mapping used for memory retrieval.

E EXTENSION TO OTHER BACKBONES DETAILS

TRM is built on the point-bump structure of BoxE, and our main setting defines TRM as injecting snapshot-wise residual memory through the BoxE bump path. To further examine whether the proposed memory reuse idea is specific to the BoxE structure or can also contribute to forgetting mitigation in other KGE backbones, we implemented variants that extend this idea to TransE and DistMult. The purpose of this experiment is not to propose TransE or DistMult as new main backbones, but to examine whether the principle of reusing entity-side residual memory formed in past snapshots remains effective across different KGE scoring functions. In addition, to examine how this tendency changes depending on the loss function, we compared margin ranking loss and negative sampling loss for both TRM and the backbone extensions.

Implementation on TransE. The memory extension to TransE was implemented to follow the TransE-based continual backbone setting used in LKGE Cui et al. (2023). Each entity and relation is represented as an embedding in $\mathbb{R}^d$, and L_2 normalization is applied to entity and relation representations before scoring. To apply the core idea of TRM, we assign snapshot-wise residual memory $\mathbf{m}_e^\tau$ to each entity e and compute $\boldsymbol{\rho}_e^s$ through birth-aware uniform aggregation, as in the main text. The entity representation used at snapshot s is then defined as $\tilde{\mathbf{e}}_e^s = \mathbf{e}_e + \boldsymbol{\rho}_e^s$. The TransE distance is computed as:

$$D^s_{\mathrm{TransE}}(h,r,t) = \left\| \mathrm{norm}(\tilde{\mathbf{e}}_h^s) + \mathrm{norm}(\mathbf{r}) - \mathrm{norm}(\tilde{\mathbf{e}}_t^s) \right\|_1 . \tag{18}$$

Unlike TRM, this extension does not use a separate bump or cross-bump structure. Instead, it augments the entity embedding itself with residual memory and then feeds the augmented representation into the original TransE scoring path.

Implementation on DistMult. Although DistMult has not been widely used as a backbone in existing CKGE baselines, it is a representative bilinear scoring model in semantic-matching-based KGE. We therefore used it as a comparison target to examine the extensibility of the memory reuse idea. As in TransE, each entity representation in DistMult is augmented with snapshot-wise residual memory and is defined as $\tilde{\mathbf{e}}_e^s = \mathbf{e}_e + \boldsymbol{\rho}_e^s$. The triple score is then computed according to the standard DistMult scoring function as:

$$f^s_{\mathrm{DistMult}}(h,r,t) = \langle \tilde{\mathbf{e}}_h^s, \mathbf{r}, \tilde{\mathbf{e}}_t^s \rangle . \tag{19}$$

In this implementation, to unify the loss interface of all backbones in the form of distance minimization, we used $D^s_{\mathrm{DistMult}}(h,r,t) = -f^s_{\mathrm{DistMult}}(h,r,t)$.

Loss functions. In the backbone extension experiments, we compared both margin ranking loss and negative sampling loss. Margin ranking loss trains the model so that the distance of a positive triple becomes smaller than that of a corrupted triple by a margin, and is defined as:

$$\mathcal{L}_{\mathrm{margin}} = \sum_i \left[\gamma + d_i^+ - \frac{1}{n} \sum_{j=1}^{n} d_{i,j}^- \right]_+ . \tag{20}$$

Here, d_i^+ is the distance of a positive triple, and $d_{i,j}^-$ is the distance of the j-th corrupted triple. Negative sampling loss trains the model so that the distance of a positive triple becomes smaller than the margin and the distance of a negative triple becomes larger than the margin, and is defined as

$$\mathcal{L}_{\mathrm{neg}} = -\sum_i \log \sigma(\gamma - d_i^+) - \sum_i \frac{1}{n} \sum_{j=1}^{n} \log \sigma(d_{i,j}^- - \gamma). \tag{21}$$

Both losses were compared using the same memory aggregation method and the same snapshot-wise training protocol within each backbone.

Loss sensitivity and reporting choice. Table 11 and Table 12 compare margin ranking loss and negative sampling loss for BoxE, TransE, and DistMult. This comparison is intended to examine two aspects of the proposed memory reuse idea: whether entity-side residual memory can be extended beyond the TRM architecture, and how its effect changes under different optimization objectives. For BoxE, 'TRM w/o Memory' denotes the setting in which snapshot-wise residual memory is

Table 11: Backbone and loss comparison on ENTITY, RELATION, FACT, and HYBRID.

		ENTITY			RELATION			FACT			HYBRID		
Loss	Model	MRR	ROD	RTE	MRR	ROD	RTE	MRR	ROD	RTE	MRR	ROD	RTE
Rank.	TransE	0.175	-0.103	-0.112	0.141	-0.173	-0.135	0.147	-0.092	-0.003	0.155	-0.092	-0.066
	TransE + Memory	0.179	-0.050	-0.015	0.164	-0.018	0.007	0.158	-0.005	0.039	0.168	0.033	0.033
	Δ	**+0.004**	**+0.053**	**+0.097**	**+0.023**	**+0.155**	**+0.142**	**+0.011**	**+0.087**	**+0.042**	**+0.013**	**+0.126**	**+0.099**
	DistMult	0.181	0.006	0.009	0.172	-0.090	-0.059	0.197	0.017	0.017	0.192	-0.015	-0.005
	DistMult + Memory	0.199	-0.002	-0.001	0.193	-0.066	-0.041	0.231	-0.003	-0.002	0.210	-0.004	-0.001
	Δ	**+0.018**	-0.008	-0.010	**+0.021**	**+0.024**	**+0.018**	**+0.034**	-0.020	-0.019	**+0.018**	**+0.011**	**+0.004**
	TRM w/o Memory	0.393	0.016	0.017	0.316	0.019	0.023	0.277	-0.065	-0.027	0.323	0.003	0.009
	TRM	0.410	0.025	0.020	0.334	0.023	0.019	0.294	-0.010	0.009	0.337	0.002	0.003
	Δ	**+0.017**	**+0.009**	**+0.003**	**+0.018**	**+0.004**	-0.004	**+0.017**	**+0.055**	**+0.036**	**+0.014**	-0.001	-0.006
Neg.	TransE	0.181	-0.464	-0.402	0.123	-0.552	-0.491	0.153	-0.354	-0.266	0.143	-0.454	-0.358
	TransE + Memory	0.248	-0.185	-0.116	0.157	-0.400	-0.325	0.193	-0.155	-0.101	0.193	-0.190	-0.142
	Δ	**+0.067**	**+0.279**	**+0.286**	**+0.034**	**+0.153**	**+0.166**	**+0.040**	**+0.200**	**+0.165**	**+0.050**	**+0.264**	**+0.216**
	DistMult	0.287	-0.089	-0.069	0.213	-0.047	-0.030	0.187	0.079	0.099	0.230	-0.012	-0.019
	DistMult + Memory	0.321	-0.017	-0.011	0.247	0.057	0.042	0.218	0.069	0.064	0.250	0.021	0.019
	Δ	**+0.034**	**+0.072**	**+0.058**	**+0.034**	**+0.104**	**+0.072**	**+0.031**	-0.010	-0.035	**+0.020**	**+0.034**	**+0.038**
	TRM w/o Memory	0.363	-0.105	-0.076	0.287	-0.053	-0.025	0.268	-0.103	-0.061	0.299	-0.084	-0.065
	TRM	0.403	-0.029	-0.020	0.313	-0.003	0.008	0.292	-0.040	-0.020	0.329	-0.012	-0.010
	Δ	**+0.039**	**+0.076**	**+0.056**	**+0.026**	**+0.050**	**+0.033**	**+0.024**	**+0.063**	**+0.041**	**+0.030**	**+0.072**	**+0.054**

Δ denotes the difference between each memory-extended model and its corresponding base model. Rank. and Neg. denote margin ranking loss and negative sampling loss, respectively.

Table 12: Backbone and loss comparison on GraphEqual, GraphHigher, and GraphLower.

		GraphEqual			GraphHigher			GraphLower		
Loss	Model	MRR	ROD	RTE	MRR	ROD	RTE	MRR	ROD	RTE
Rank.	TransE	0.184	0.032	-0.003	0.208	0.153	0.100	0.181	0.041	0.023
	TransE + Memory	0.191	0.143	0.101	0.205	0.217	0.165	0.183	0.066	0.045
	Δ	**+0.007**	**+0.112**	**+0.104**	-0.003	**+0.065**	**+0.065**	**+0.002**	**+0.025**	**+0.022**
	DistMult	0.179	0.008	0.012	0.189	0.100	0.088	0.194	-0.010	-0.007
	DistMult + Memory	0.183	0.012	0.009	0.187	0.043	0.004	0.198	0.001	0.001
	Δ	**+0.004**	**+0.005**	-0.003	-0.002	-0.057	-0.084	**+0.004**	**+0.012**	**+0.008**
	TRM w/o Memory	0.329	0.020	0.019	0.321	0.032	0.030	0.301	-0.024	-0.004
	TRM	0.339	0.033	0.028	0.323	0.020	0.018	0.307	0.009	0.016
	Δ	**+0.010**	**+0.013**	**+0.009**	**+0.002**	-0.012	-0.012	**+0.006**	**+0.033**	**+0.020**
Neg.	TransE	0.189	-0.181	-0.205	0.205	-0.228	-0.218	0.185	-0.163	-0.171
	TransE + Memory	0.212	-0.078	-0.046	0.219	-0.127	-0.104	0.217	-0.042	-0.033
	Δ	**+0.023**	**+0.103**	**+0.159**	**+0.014**	**+0.101**	**+0.114**	**+0.032**	**+0.120**	**+0.138**
	DistMult	0.260	0.046	0.015	0.232	-0.027	-0.023	0.227	-0.008	-0.008
	DistMult + Memory	0.275	0.096	0.064	0.251	0.005	0.004	0.241	0.062	0.048
	Δ	**+0.015**	**+0.050**	**+0.049**	**+0.019**	**+0.032**	**+0.027**	**+0.014**	**+0.069**	**+0.056**
	TRM w/o Memory	0.314	-0.009	-0.009	0.304	-0.019	-0.015	0.281	-0.033	-0.028
	TRM	0.326	0.054	0.046	0.308	0.000	-0.002	0.290	0.001	0.010
	Δ	**+0.012**	**+0.063**	**+0.055**	**+0.004**	**+0.019**	**+0.013**	**+0.009**	**+0.034**	**+0.038**

Δ denotes the difference between each memory-extended model and its corresponding base model. Rank. and Neg. denote margin ranking loss and negative sampling loss, respectively.

removed from TRM, while 'TRM' denotes the memory-based model proposed in this study. For TransE and DistMult, we compare the original backbone with the '+ Memory' variant, which extends the core memory reuse idea of TRM to the corresponding scoring function. The Δ rows report within-backbone changes induced by memory reuse.

For TRM, the effect of memory reuse differs depending on the loss function. Under negative sampling loss, TRM improves MRR over 'TRM w/o Memory' on all seven datasets and jointly improves MRR, ROD, and RTE on all datasets. Under margin ranking loss, TRM improves MRR on all seven datasets, ROD on five datasets, and RTE on four datasets. This indicates that the memory component consistently improves final ranking performance in most settings, while its retention effect is more sensitive to the loss function.

A loss-dependent difference is also observed when memory is extended to TransE and DistMult. Under margin ranking loss, the TransE memory extension improves RTE on all datasets, whereas the DistMult memory extension shows decreased RTE on some datasets. In contrast, under negative sampling loss, the TransE memory extension improves MRR, ROD, and RTE on all seven datasets, and the DistMult memory extension improves MRR on all seven datasets and RTE on six datasets, with FACT being the exception.

For TRM, margin ranking loss was used as the main setting because it maintains strong final performance and positive RTE across all seven datasets in the main configuration. In addition, as the representative results for backbone extension, we reported the experiments based on negative sampling loss, as this setting showed the most consistent effect of residual memory reuse across TRM, TransE, and DistMult. The complete results under both loss functions are also reported in this appendix to show the sensitivity to the loss function.

Analysis. The overall results lead to two observations. First, entity-side residual memory reuse, which is the core idea of TRM, is not limited to the BoxE structure and can be extended to different KGE scoring functions such as TransE and DistMult. This is supported by the positive within-backbone changes observed after adding memory, particularly in MRR across both TransE and DistMult and in retention metrics for TransE. Second, the stability of this effect depends on the combination of the loss function and the backbone. Under margin ranking loss, the pattern of retention improvement varies depending on the backbone and dataset, whereas under negative sampling loss, more consistent improvements are observed, particularly for MRR and for retention in TransE and TRM. Therefore, the backbone extension results support the interpretation that, under an appropriate optimization setting, reusing entity-side residual memory can serve as an effective forgetting mitigation strategy beyond a single backbone. This does not mean that the effect of TRM is invariant to the loss function. Rather, the memory reuse mechanism operates together with the scoring geometry of the backbone and the optimization objective, suggesting that the appropriate loss setting may differ across backbones and scoring geometries.

F Entity and relation memory scope analysis

Experimental setup. TRM does not apply snapshot-wise memory to relation-side representations, but manages snapshot-wise entity residual memory. This is because relation-side structure tends to maintain relatively stable structural semantics across snapshots, whereas the semantic position and context of entities can change more directly as new facts and connections are added. To empirically examine this design choice, we compare three settings for TransE, DistMult, and the proposed TRM, namely the setting without memory, the setting in which memory is applied to entity-side representations, and the setting in which memory is applied to relation-side representations. For TransE and DistMult, we use the negative sampling loss based extension setting analyzed in Appendix E, while for TRM, we use the setting as in the main text. The purpose of this analysis is not to compare absolute performance across different backbones, but to compare the effect of memory scope on performance and retention within each backbone.

Relation-side memory formulation. Specifically, the relation memory settings for TransE and DistMult are variants that apply the entity-side memory integration described in Appendix E to relation representations rather than entity representations. For TRM, the relation memory setting applies residual memory to the role-specific relation box center rather than injecting entity residual memory through the BoxE bump path. For each relation r, snapshot τ, and role $i \in \{h, t\}$, we define the relation-side residual memory as $\mathbf{m}^{\tau}_{r,i}$, and define the valid memory set available at the current snapshot s as

$$\mathcal{V}^{s}_{r} = \{\tau \mid 1 \le \tau \le s,\ \tau \ge b(r)\}. \tag{22}$$

Here, $b(r)$ denotes the snapshot in which relation r first appears. As in the main text, memory from past snapshots is kept frozen and is denoted by $\bar{\mathbf{m}}^{\tau}_{r,i}$. The relation-side residual memory is uniformly aggregated as:

$$\boldsymbol{\rho}^{s}_{r,i} = \frac{1}{|\mathcal{V}^{s}_{r}|}\left(\sum_{\tau \in \mathcal{V}^{s}_{r},\ \tau < s} \bar{\mathbf{m}}^{\tau}_{r,i} + \mathbf{m}^{s}_{r,i}\right), \qquad i \in \{h, t\}. \tag{23}$$

Table 13: Memory scope comparison on ENTITY, RELATION, FACT, and HYBRID.

		ENTITY			RELATION			FACT			HYBRID		
Backbone	Scope	MRR	ROD	RTE	MRR	ROD	RTE	MRR	ROD	RTE	MRR	ROD	RTE
TransE	None	0.181	-0.464	-0.402	0.123	-0.552	-0.491	0.153	-0.354	-0.266	0.143	-0.454	-0.358
	Entity	**+0.067**	**+0.279**	**+0.286**	**+0.034**	**+0.152**	**+0.166**	**+0.040**	**+0.199**	**+0.165**	**+0.050**	**+0.264**	**+0.216**
	Relation	+0.001	+0.000	-0.003	+0.001	+0.008	+0.010	+0.006	+0.014	+0.012	-0.001	+0.000	-0.032
DistMult	None	0.287	-0.089	-0.069	0.213	-0.047	-0.030	0.187	0.079	0.099	0.230	-0.012	-0.019
	Entity	**+0.034**	**+0.072**	**+0.058**	**+0.034**	**+0.104**	**+0.072**	**+0.031**	**-0.010**	**-0.035**	**+0.020**	**+0.033**	**+0.038**
	Relation	+0.016	+0.027	+0.020	+0.007	+0.022	+0.023	+0.008	-0.055	-0.048	+0.000	-0.005	+0.006
TRM	None	0.393	0.016	0.017	0.316	0.019	0.023	0.277	-0.065	-0.027	0.323	0.003	0.009
	Entity	**+0.017**	**+0.009**	**+0.003**	**+0.018**	**+0.004**	**-0.004**	**+0.017**	**+0.055**	**+0.036**	**+0.014**	**-0.001**	**-0.006**
	Relation	+0.004	-0.011	-0.012	+0.002	-0.006	-0.005	+0.001	-0.002	-0.007	+0.007	-0.011	-0.014

Values in Entity and Relation rows report differences from the None scope. **Bold** marks the more favorable change between Entity and Relation for each metric, excluding ties.

Table 14: Memory scope comparison on GraphEqual, GraphHigher, and GraphLower.

		GraphEqual			GraphHigher			GraphLower		
Backbone	Scope	MRR	ROD	RTE	MRR	ROD	RTE	MRR	ROD	RTE
TransE	None	0.189	-0.181	-0.205	0.205	-0.228	-0.218	0.185	-0.163	-0.171
	Entity	**+0.023**	**+0.103**	**+0.159**	**+0.014**	**+0.101**	**+0.114**	**+0.032**	**+0.121**	**+0.138**
	Relation	-0.003	-0.031	-0.016	-0.004	-0.044	-0.038	-0.004	-0.014	-0.007
DistMult	None	0.260	0.046	0.015	0.232	-0.027	-0.023	0.227	-0.008	-0.008
	Entity	**+0.015**	**+0.050**	**+0.049**	**+0.019**	**+0.032**	**+0.027**	**+0.014**	**+0.070**	**+0.056**
	Relation	+0.005	-0.004	+0.007	+0.007	-0.013	-0.009	-0.006	-0.025	-0.016
TRM	None	0.329	0.020	0.019	0.321	0.032	0.030	0.301	-0.024	-0.004
	Entity	**+0.010**	**+0.013**	**+0.009**	+0.002	-0.012	-0.012	**+0.006**	**+0.033**	**+0.020**
	Relation	+0.005	-0.011	-0.012	**+0.004**	**-0.011**	-0.012	+0.002	+0.002	-0.005

Values in Entity and Relation rows report differences from the None scope. **Bold** marks the more favorable change between Entity and Relation for each metric, excluding ties.

The relation box center used at snapshot s is then defined as $\tilde{\mathbf{c}}^s_{r,i} = \mathbf{c}_{r,i} + \boldsymbol{\rho}^s_{r,i}, i \in \{h, t\}$. In this setting, entity-side residual memory is not used, and the entity bump of BoxE remains unchanged as $\mathbf{b}_e$. Therefore, this setting uses the same aggregation rule as the default TRM, but can be viewed as a variant that changes the target of memory application from the entity-side bump to the role-specific relation box center.

Analysis. Table 13 and Table 14 compare the changes induced by different memory scopes. Here, 'None' denotes the performance of the original backbone without residual memory, while the Entity and Relation rows report the changes obtained by applying memory to entity representations and relation representations, respectively, relative to the 'None' row for the same backbone and dataset. In TransE and DistMult, entity-side residual memory shows a generally consistent advantage. In TransE, entity memory jointly improves MRR, ROD, and RTE on all datasets, whereas relation memory shows smaller improvements overall and even negative changes on some datasets. In DistMult, entity memory also yields larger improvements than relation memory for most datasets and metrics, while relation memory provides only limited gains in MRR or retention metrics on some datasets. These results indicate that applying residual memory to entity representations, rather than directly to relation representations, can induce more stable improvements across TransE and DistMult.

A similar tendency is also observed in the proposed TRM, although the gain from entity memory is not uniform across all growth patterns. Entity memory improves MRR on all seven datasets

Table 15: Average comparison between entity memory and relation memory in TRM.

Scope	Avg. MRR	Avg. ROD	Avg. RTE	Pos. ROD	Pos. RTE
Entity	**0.334857**	**0.014520**	**0.016286**	**6/7**	**7/7**
Relation	0.326386	-0.006804	0.000302	4/7	4/7

and yields positive RTE on all datasets, whereas relation memory often decreases retention metrics despite small MRR gains in some cases. GraphHigher is the only dataset where relation memory is more favorable on some metrics, but both entity and relation memory still reduce retention metrics. Table 15 summarizes this overall trend. Entity memory achieves higher average MRR, average ROD, and average RTE than relation memory, and it also produces positive retention metrics on a larger number of datasets.

These results support the rationale behind the design of TRM, which prioritizes snapshot-wise entity residual memory rather than applying memory directly to relation-side representations. In CKGE, the context and position of entities can change more directly as new facts and connections accumulate, and the experimental results show that tracking and reusing such entity-side changes generally leads to more stable performance and retention. By contrast, relation-side memory does not consistently mitigate forgetting and often degrades retention metrics, supporting our choice to keep the backbone relation-side scoring structure unchanged. Therefore, the entity-side memory design of TRM should be interpreted as a targeted design choice for reusing temporal information where changes are more directly reflected in CKGE.