TWO NEGATIVES MAKE A POSITIVE: DUAL NEGATIVE LEARNING FOR CONTINUAL KNOWLEDGE GRAPH EMBEDDING

Anonymous authors
Paper under double-blind review

ABSTRACT

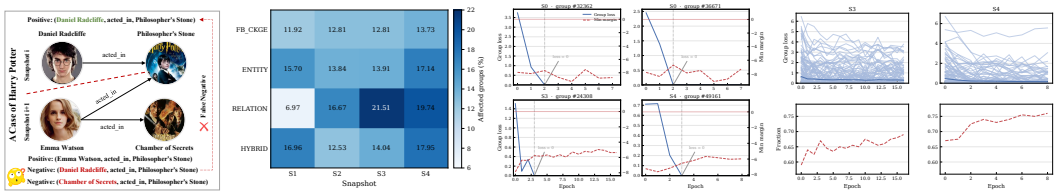
Continual Knowledge Graph Embedding (CKGE) aims to efficiently update Knowledge Graph (KG) representations to accommodate emerging facts while preserving previously learned knowledge. As new facts arrive, evolving graph structures and interactions between new and existing knowledge complicate continual adaptation. Existing CKGE methods primarily address these challenges by adapting representations, while the learning signals conveyed by negative samples remain insufficiently explored. Negative supervision in CKGE changes in two respects: accumulated facts alter the validity of negative samples, while model evolution affects their informativeness for training. To address these challenges, we propose **EvoNS** (**Evo**lution-Aware **N**egative **S**upervision), a framework that jointly adapts negative construction and utilization through dual negative learning. Specifically, EvoNS exploits accumulated observations to construct history-consistent negatives. It further reweights individual negatives according to their model-perceived difficulty, enabling more informative negative signals to guide continual optimization. Extensive experiments on seven CKGE benchmarks demonstrate that EvoNS consistently outperforms existing baselines, achieving up to 12.40% relative improvement in MRR and reducing training time by up to 18.92% in several settings. The results demonstrate the effectiveness and efficiency of EvoNS for CKGE.

1 INTRODUCTION

Knowledge graphs (KGs) organize relational knowledge for applications such as question answering and recommendation (Liang et al., 2024). Knowledge graph embedding (KGE) learns continuous entity and relation representations to infer missing facts. Real-world KGs, however, continually evolve with the addition of new entities, relations, and facts. Repeatedly retraining embeddings on the entire accumulated graph incurs growing computational costs, while learning only from new observations risks losing previously acquired knowledge. Continual KGE (CKGE) incrementally updates entity and relation representations across successive graph snapshots (Cui et al., 2023). Its central challenge is to incorporate emerging knowledge while preserving earlier knowledge at an affordable update cost.

Existing CKGE methods preserve historical knowledge through embedding transfer, regularization (Cui et al., 2023), and incremental distillation (Liu et al., 2024a), while using structural adaptation and efficient parameter updates to learn new knowledge (Liu et al., 2024b; Lee & Choi, 2026). However, improved representation updates alone do not ensure effective training signals. As knowledge accumulates, supervision may conflict with observed facts; as learning progresses, training signals should emphasize distinctions the model still finds difficult. Negative samples offer a direct way to address both issues: their construction determines which candidates receive negative supervision, while their utilization governs how individual ranking constraints contribute to optimization (Bordes et al., 2013; Yang et al., 2020). Existing CKGE approaches centered on representation updates give limited attention to the consistency of this supervision with accumulated knowledge and its informativeness for the evolving model. This motivates jointly examining negative construction and utilization, which we refer to as *negative learning* in this work (Figure 1).

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.



(a) Illustrative case (b) Collision heatmap (c) Example trajectories (d) Aggregate statistics

Figure 1: Motivating observations for CKGE. (a) An illustrative case of known-positive collisions and relation-incompatible corruptions. (b) Percentage of pre-filter groups containing cumulative training facts. (c) Example trajectories with zero group-level loss despite individual margin violations. (d) Group losses (top) and the fraction of groups exhibiting this discrepancy (bottom) on ENTITY.

New observations cover only part of the accumulated knowledge: a corrupted triple absent from the current update may already have been observed in an earlier snapshot. Treating it as negative therefore introduces a training target that conflicts with historical knowledge retention. Moreover, filtering known facts does not by itself prevent relation-incompatible corruptions, which can emphasize mismatched entity roles rather than distinctions among plausible alternatives. Figure 1a illustrates both issues: replacing Emma Watson in a positive at Snapshot $i+1$ can reproduce Daniel Radcliffe's appearance observed at Snapshot i or yield a film–`acted_in`–film triple. The pre-filter audit further shows that 6.97%–21.51% of candidate groups contain at least one cumulative training fact across the displayed dataset–snapshot pairs, with percentages averaged over epochs (Figure 1b). These considerations motivate using observed relation roles to guide candidate generation and accumulated training facts to identify and repair known-positive collisions.

Improved candidate construction alone does not ensure informative optimization, because loss aggregation can mask individual ranking constraints. Averaging negative distances before applying a hinge allows easy negatives to offset difficult ones, satisfying the group-level margin while individual margins remain violated. A zero group loss can thus suppress training signals for negatives that remain insufficiently separated from the positive. Figure 1c illustrates this discrepancy; post-hoc audits of 200 fixed groups per snapshot on ENTITY also show it in a substantial fraction of groups throughout the audited epochs of Snapshots 3 and 4 (Figure 1d). As the model evolves, which negatives remain difficult can also change. This motivates preserving individual ranking losses before aggregation and adapting their weights to emphasize distinctions the current model has yet to learn.

We propose EVONS (Evolution-Aware Negative Supervision), a dual negative learning framework that couples history-consistent negative construction with difficulty-aware negative utilization for CKGE. HCNS (History-Consistent Negative Sampling) mixes global and relation-aware proposals, then checks and repairs collisions with cumulative training facts. DANR (Difficulty-Aware Negative Reweighting) computes individual ranking losses before aggregation and weights them by model-perceived difficulty. Together, these two stages align negative candidates with accumulated knowledge and their contributions to optimization with the evolving model.

Our main contributions can be summarized as follows: ❶ **Problem.** We characterize conflicts with accumulated facts and the masking of individual ranking constraints in CKGE negative learning through illustrative cases and empirical diagnostics. ❷ **Method.** We develop EVONS, a dual negative learning framework coupling history-consistent, relation-aware sampling with difficulty-aware optimization of individual negative constraints. ❸ **Experiments.** We evaluate EVONS on seven CKGE benchmarks, covering link prediction performance, efficiency, snapshot-wise analysis, ablation, sensitivity and convergence.

2 RELATED WORK

Continual Knowledge Graph Embedding. CKGE learns from successive graph updates while retaining knowledge acquired from earlier snapshots (Daruna et al., 2021). Retention-oriented methods transfer and regularize embeddings (Cui et al., 2023) or distill historical representations

across snapshots (Liu et al., 2024a). More recent approaches account for evolving relational structure through relation-path transfer (Yang et al., 2025), structural alignment (Jian et al., 2026), and Bayesian updates with clustering constraints (Li et al., 2026). A complementary line adapts model capacity using incremental low-rank updates (Liu et al., 2024b) or scale-aware embedding expansion (Li et al., 2025). STARK combines structural capacity allocation with an adaptive training objective (Lee & Choi, 2026). We examine how accumulated training facts and the current model can jointly guide negative construction and utilization.

Negative Learning for Knowledge Graph Embedding. *Negative learning* covers the construction, selection, and weighting of negative examples for KGE. Candidate construction exploits relation-type constraints and graph neighborhoods (Krompaß et al., 2015; Ahrabian et al., 2020), whereas adversarial generation and caching prioritize informative corruptions (Cai & Wang, 2018; Zhang et al., 2019). Subsequent methods adapt negative generation to representation changes or training difficulty (Meng et al., 2024; Li & Huang, 2026). Hardness does not establish factual validity, motivating adaptive self-adversarial sampling and denoising mixup to reduce false-negative risk (Qin et al., 2021; Chen et al., 2023). RotatE assigns model-dependent weights to emphasize plausible negatives (Sun et al., 2019). Building on these principles, EvoNS uses cumulative training facts to constrain negative construction and model-dependent weights to optimize individual ranking losses. Appendix A provides a detailed discussion.

3 Preliminary

Continual Knowledge Graph. A continual knowledge graph (CKG) can be represented as a sequence of N_{snap} knowledge graph snapshots $\{\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_{N_{\text{snap}}-1}\}$, where $\mathcal{G}_i = (\mathcal{E}_i, \mathcal{R}_i, \mathcal{T}_i)$ denotes the knowledge graph at snapshot i. Here, $\mathcal{E}_i$, $\mathcal{R}_i$, and $\mathcal{T}_i$ denote the entity set, relation set, and factual triple set, respectively. Each fact is represented as a triple $(h, r, t) \in \mathcal{T}_i$, where $h, t \in \mathcal{E}_i$ and $r \in \mathcal{R}_i$. As the knowledge graph evolves, new entities, relations, and facts continually emerge in subsequent snapshots. We denote the newly introduced knowledge at each incremental snapshot $i \geq 1$ as

$$\Delta\mathcal{E}_i = \mathcal{E}_i \setminus \mathcal{E}_{i-1}, \qquad \Delta\mathcal{R}_i = \mathcal{R}_i \setminus \mathcal{R}_{i-1}, \qquad \Delta\mathcal{T}_i = \mathcal{T}_i \setminus \mathcal{T}_{i-1}.$$

Accordingly, as new knowledge arrives, the observed knowledge graph progressively accumulates the knowledge available up to the current snapshot.

Continual Knowledge Graph Embedding. Continual knowledge graph embedding (CKGE) learns entity and relation representations from successive graph snapshots. At snapshot i, the model M_i maps the entities and relations in $\mathcal{G}_i$ into a low-dimensional continuous vector space to capture their semantic and structural dependencies. As new knowledge arrives, M_{i-1} is updated to M_i to incorporate newly introduced entities, relations, and facts. The objective is to learn from these updates while retaining knowledge acquired from previous snapshots. Further details on continual link prediction are provided in Appendix B.1.

4 Methodology

We propose EvoNS (Evolution-Aware Negative Supervision), which adapts negative construction and utilization to accumulated knowledge and the evolving model (Figure 2). HCNS (History-Consistent Negative Sampling) constructs relation-aware candidates and corrects collisions with observed training facts. DANR (Difficulty-Aware Negative Reweighting) then weights individual ranking losses by model-perceived difficulty, preserving negative-level constraints before aggregation. Together, they provide *dual negative learning* for continual graph updates.

4.1 History-Consistent Negative Sampling

Relation-aware candidate construction. HCNS balances relation relevance and historical consistency through the construction and checking stages shown in the upper block of Figure 2. For a positive training fact $x^+ = (h, r, t)$ at snapshot i, it first constructs candidate negatives by replacing either the head or the tail entity. Uniform entity replacement provides diverse candidates, but many may be weakly related to the query relation. We therefore combine global proposals with candidates drawn from the entity roles that the relation has involved in observed training facts.

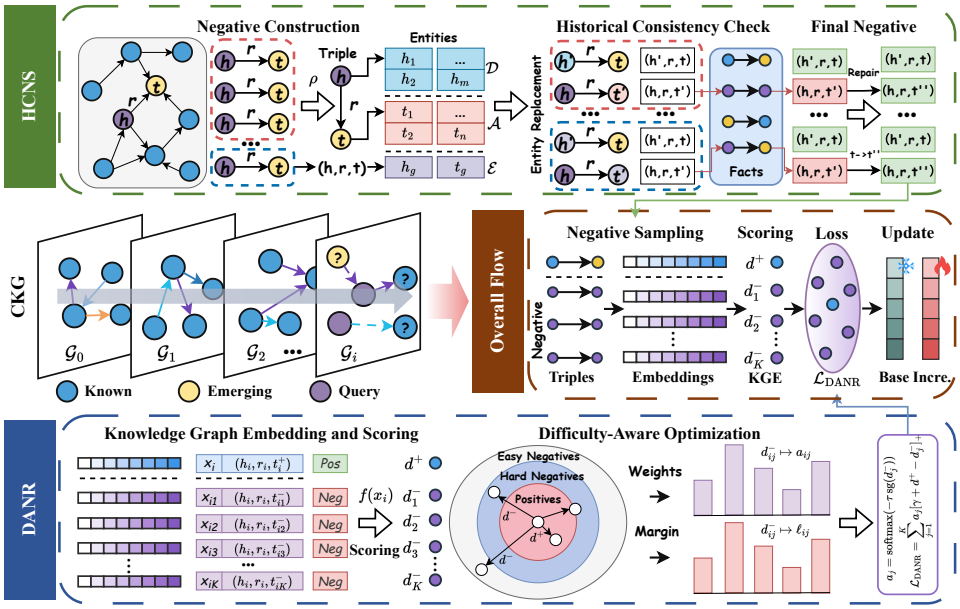


Figure 2: Overview of the EvoNS framework.

Let $\mathcal{T}_i^{\mathrm{tr}} \subseteq \mathcal{T}_i$ denote the cumulative set of training facts available up to and including snapshot i. For relation r, its historical head and tail entity sets are $\mathcal{D}_i(r) = \{e \in \mathcal{E}_i \mid \exists t' : (e, r, t') \in \mathcal{T}_i^{\mathrm{tr}}\}$ and $\mathcal{A}_i(r) = \{e \in \mathcal{E}_i \mid \exists h' : (h', r, e) \in \mathcal{T}_i^{\mathrm{tr}}\}$. These pools encode empirical head and tail roles observed in the training history, rather than explicit entity-type constraints. Let $c \in \{h, t\}$ indicate the entity position to corrupt. With probability $\rho \in [0, 1]$, a replacement is drawn uniformly from $\mathcal{D}_i(r)$ for head corruption or $\mathcal{A}_i(r)$ for tail corruption; otherwise, it is drawn uniformly from the global entity set $\mathcal{E}_i$. Replacing the selected entity produces a candidate triple $\widetilde{x}^-$, such as (h', r, t) or (h, r, t'). This mixture balances relation relevance and candidate diversity, but neither proposal source by itself prevents a candidate from reproducing an observed true fact.

Historical consistency checking and correction. Before a candidate is used as a negative, HCNS checks whether $\widetilde{x}^- \in \mathcal{T}_i^{\mathrm{tr}}$. A matching candidate would assign negative supervision to an observed positive fact; HCNS therefore resamples its replacement entity. To characterize the admissible replacements, we define the known positive tails for query $(h, r, ?)$ as $\mathcal{P}_i^t(h, r) = \{e \in \mathcal{E}_i \mid (h, r, e) \in \mathcal{T}_i^{\mathrm{tr}}\}$. The corresponding legal candidate set is $\mathcal{C}_i^t(h, r) = \mathcal{E}_i \setminus \mathcal{P}_i^t(h, r)$. Similarly, for head corruption, we define $\mathcal{P}_i^h(r, t) = \{e \in \mathcal{E}_i \mid (e, r, t) \in \mathcal{T}_i^{\mathrm{tr}}\}$ and $\mathcal{C}_i^h(r, t) = \mathcal{E}_i \setminus \mathcal{P}_i^h(r, t)$. Intersecting these sets with the historical relation pools gives $\mathcal{Q}_i^h(r, t) = \mathcal{C}_i^h(r, t) \cap \mathcal{D}_i(r)$ and $\mathcal{Q}_i^t(h, r) = \mathcal{C}_i^t(h, r) \cap \mathcal{A}_i(r)$.

Candidates that do not conflict with cumulative training facts are retained. For a conflicting candidate, the replacement entity is resampled uniformly from the legal subset of its original proposal source: $\mathcal{Q}_i^c(x^+)$ for the relation-conditioned branch or $\mathcal{C}_i^c(x^+)$ for the global branch. When the relation-conditioned legal set is empty, that branch falls back to the global legal set. Otherwise, preserving the proposal branch during correction yields the following *final* distribution over history-consistent replacement entities:

$$q_i^{\mathrm{HCNS}}(e \mid x^+, c) = (1 - \rho)\,\mathcal{U}\big(\mathcal{C}_i^c(x^+)\big) + \rho\,\mathcal{U}\big(\mathcal{Q}_i^c(x^+)\big)\,, \tag{1}$$

where $\mathcal{U}(\cdot)$ denotes the uniform distribution over a finite set. This distribution characterizes the output of candidate construction, consistency checking, and correction, rather than the unfiltered proposals.

The procedure yields the final negative set $\mathcal{N}_i(x^+) = \{x_1^-, x_2^-, \ldots, x_K^-\}$ for each positive fact. These negatives do not contradict the training facts observed up to snapshot i, while their candidate

pools evolve with the accumulated knowledge. Historical consistency does not, however, determine how informative each negative is to the current model, motivating adaptive negative utilization.

4.2 Difficulty-Aware Negative Reweighting

Knowledge graph embedding and scoring. As shown in the lower block of Figure 2, DANR takes a positive fact x^+ and the negative set $\mathcal{N}_i(x^+)$ produced by HCNS. The current KGE model M_i retrieves their entity and relation embeddings and computes the triple distances $d^+ = d_{M_i}(x^+)$ and $d_j^- = d_{M_i}(x_j^-)$, where a smaller distance indicates a more plausible triple. A negative with a smaller d_j^- is therefore harder for the model to distinguish from a true fact. As the model evolves, the relative difficulty of these negatives changes. We use their distances as continuous difficulty signals, without explicitly partitioning the negatives into easy and hard groups.

Difficulty-aware negative-level optimization. The scores support two complementary computations: a difficulty-aware weight that determines each negative's relative emphasis, and a pairwise ranking loss that measures its constraint violation. For the weighting branch, DANR assigns normalized weights to the negatives associated with the same positive fact:

$$a_j = \frac{\exp\big(-\tau \ \mathrm{sg}(d_j^-)\big)}{\sum_{k=1}^{K} \exp\big(-\tau \ \mathrm{sg}(d_k^-)\big)}, \tag{2}$$

where $\tau \geq 0$ controls the concentration of the weights and $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator. Smaller negative distances receive larger weights, emphasizing candidates that are harder for the current model. Detaching the distances in this branch prevents gradients from flowing through the weight computation; optimization proceeds through the individual ranking losses.

When $\tau = 0$, $a_j = 1/K$ and the weights are uniform. Increasing τ concentrates more weight on harder negatives, while the weights themselves adapt as the model's scores change.

In parallel, the loss branch preserves a separate ranking constraint for each positive-negative pair: $\ell_j = [\gamma + d^+ - d_j^-]_+$, where $[z]_+ = \max(0, z)$ and $\gamma > 0$ is a fixed margin. The loss is positive whenever the negative distance is less than the positive distance plus γ. Computing this hinge separately for every negative retains violations that could be concealed by applying a single hinge to an averaged negative distance.

The two branches are then combined by weighting each individual loss, giving the objective for a single positive fact:

$$\mathcal{L}_{\mathrm{DANR}}(x^+) = \sum_{j=1}^{K} a_j \left[\gamma + d^+ - d_j^-\right]_+ . \tag{3}$$

During training, consider a mini-batch containing B positive samples, and let b index the positive samples in the batch. We denote the distances of the b-th positive sample and its j-th negative by d_b^+ and $d_{b,j}^-$, respectively, and the corresponding difficulty weight by $a_{b,j}$. The final batch-level objective is

$$\mathcal{L}_{\mathrm{DANR}}^{(i)} = \frac{1}{B} \sum_{b=1}^{B} \sum_{j=1}^{K} a_{b,j} \left[\gamma + d_b^+ - d_{b,j}^-\right]_+ . \tag{4}$$

Thus, the individual losses preserve unresolved ranking constraints, while the difficulty weights control their relative contributions. Together, they turn the history-consistent negatives produced by HCNS into supervision that adapts to the evolving model without first collapsing their distances into a group-level score.

4.3 Overall Framework

After training the base KGE model on the initial snapshot, EvoNS applies HCNS and DANR during subsequent incremental updates. For each current positive fact, HCNS constructs relation-aware candidates, checks them against cumulative training facts, and corrects conflicting candidates to obtain the final negative set. The current model scores the positive and negative triples, after

which DANR computes difficulty weights and individual ranking losses and combines them into the training objective. This couples knowledge-consistent negative construction with model-adaptive negative utilization. During each incremental update, previously learned embeddings remain fixed while the incremental parameters are optimized.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

5.1.1 DATASETS AND METRICS

Datasets. We evaluate EVONS on seven FB15K-237-based benchmarks with five snapshots each: ENTITY, RELATION, and HYBRID (Cui et al., 2023); FB-CKGE (Liu et al., 2024b); and GRAPHLOWER, GRAPHEQUAL, and GRAPHHIGHER (Liu et al., 2024a). These benchmarks cover entity- and relation-centric growth, mixed growth, large initial graphs, and varying rates of fact arrival. Table 1 summarizes snapshot-wise statistics; detailed descriptions and data splits are provided in Appendix C.1.

Table 1: Snapshot-wise dataset statistics.

Dataset	Snapshot 0			Snapshot 1			Snapshot 2			Snapshot 3			Snapshot 4		
	#Ent.	#Rel.	#Fact	#Ent.	#Rel.	#Fact	#Ent.	#Rel.	#Fact	#Ent.	#Rel.	#Fact	#Ent.	#Rel.	#Fact
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
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
FB-CKGE	7,505	237	186,070	11,258	237	31,012	13,134	237	31,012	14,072	237	31,012	14,541	237	31,010
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
GRAPHEQUAL	2,908	226	57,636	5,816	235	62,023	8,724	237	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

Metrics. Following the standard CKGE protocol (Liu et al., 2024b; Lee & Choi, 2026), we evaluate filtered head and tail prediction using MRR and Hits@k ($k = 1, 3, 10$). Main results aggregate scores over all observed test sets after the final snapshot. We also report total training time across five snapshots, T (s). Higher ranking scores and lower T indicate better performance. Detailed protocols and metric definitions are provided in Appendix C.3.

5.1.2 BASELINES

We compare EVONS with nine baselines: six general continual-learning methods adapted to KGE—PNN (Rusu et al., 2016), CWR (Lomonaco & Maltoni, 2017), GEM (Lopez-Paz & Ranzato, 2017), EMR (Wang et al., 2019), EWC (Kirkpatrick et al., 2017), and SI (Zenke et al., 2017)—and three CKGE-specific methods: LKGE (Cui et al., 2023), FastKGE (Liu et al., 2024b), and STARK (Lee & Choi, 2026). Detailed baseline descriptions and implementation settings are provided in Appendices C.2 and C.4, respectively.

5.2 MAIN RESULTS

5.2.1 MAIN RESULTS AND ANALYSIS

Overall performance. Table 2 shows that EVONS achieves the best MRR, Hits@1, and Hits@3 on all four datasets, leading all 12 reported link-prediction comparisons. The largest MRR margin occurs on ENTITY, where the graph evolves primarily through new entities; EVONS improves MRR over the strongest baseline by 0.030 (12.4%). It retains MRR gains of 0.011, 0.010, and 0.007 (3.1%–6.3%) on RELATION, HYBRID, and FB-CKGE, respectively, showing that its advantage extends to relation-dominated and mixed evolution as well as the large-initial-graph setting. Its gains in MRR, Hits@1, and Hits@3 indicate more accurate placement of the correct entity near the top of the ranking.

Table 2: Main results across four continual knowledge graph settings. Best results are in **bold**; second-best results are underlined.

Method	ENTITY				RELATION				HYBRID				FB-CKGE			
	MRR	Hits@1	Hits@3	T(s)	MRR	Hits@1	Hits@3	T(s)	MRR	Hits@1	Hits@3	T(s)	MRR	Hits@1	Hits@3	T(s)
PNN	0.230	0.131	0.265	870	0.165	0.094	0.191	944	0.186	0.102	0.217	935	0.215	0.122	0.246	631
CWR	0.088	0.027	0.114	1091	0.022	0.010	0.025	756	0.038	0.015	0.047	1097	0.074	0.010	0.105	957
GEM	0.168	0.088	0.190	932	0.091	0.037	0.105	637	0.134	0.068	0.149	810	0.186	0.101	0.211	458
EMR	0.176	0.095	0.199	1558	0.110	0.050	0.126	1171	0.142	0.075	0.158	1465	0.179	0.096	0.203	645
EWC	0.231	0.132	0.265	919	0.154	0.083	0.179	916	0.189	0.105	0.217	819	0.215	0.121	0.245	538
SI	0.152	0.070	0.177	691	0.109	0.049	0.127	600	0.086	0.035	0.096	781	0.185	0.099	0.210	472
LKGE	0.242	0.143	0.278	1001	0.173	0.091	0.199	830	0.199	0.111	0.229	846	0.220	0.125	0.250	530
FastKGE	0.232	0.142	0.261	406	0.168	0.099	0.197	296	0.194	0.114	0.226	**326**	0.223	0.131	0.256	**190**
STARK	0.232	0.141	0.264	582	0.174	0.103	0.203	387	0.198	0.118	0.230	387	0.224	0.132	0.259	245
EVONS (ours)	**0.272**	**0.167**	**0.317**	**352**	**0.185**	**0.110**	**0.219**	**240**	**0.209**	**0.125**	**0.247**	354	**0.231**	**0.137**	**0.268**	197
Improve	12.40%	16.78%	14.03%	13.30%	6.32%	6.80%	7.88%	18.92%	5.03%	5.93%	7.39%	-	3.13%	3.79%	3.47%	-

Table 3: Main results on GRAPHLOWER, GRAPHEQUAL, and GRAPHHIGHER.

Method	GRAPHLOWER					GRAPHEQUAL					GRAPHHIGHER				
	MRR	Hits@1	Hits@3	Hits@10	T(s)	MRR	Hits@1	Hits@3	Hits@10	T(s)	MRR	Hits@1	Hits@3	Hits@10	T(s)
PNN	0.214	0.120	0.243	0.407	480	0.215	0.120	0.245	0.409	712	0.184	0.095	0.212	0.363	643
CWR	0.031	0.005	0.042	0.076	734	0.121	0.040	0.151	0.275	1127	0.191	0.097	0.226	0.376	978
GEM	0.187	0.099	0.213	0.366	400	0.181	0.094	0.206	0.357	591	0.196	0.107	0.223	0.373	561
EMR	0.185	0.099	0.209	0.361	522	0.184	0.098	0.207	0.357	1007	0.201	0.112	0.229	0.378	1342
EWC	0.210	0.115	0.240	0.404	425	0.208	0.114	0.237	0.401	632	0.196	0.104	0.225	0.380	487
SI	0.185	0.098	0.211	0.365	400	0.179	0.092	0.204	0.353	547	0.189	0.099	0.218	0.369	521
LKGE	0.213	0.118	0.243	0.408	443	0.223	0.127	0.255	0.418	676	0.219	0.126	0.251	0.404	531
FastKGE	0.216	0.125	0.249	0.400	**165**	0.211	0.123	0.239	0.387	**299**	0.194	0.113	0.215	0.353	428
STARK	0.214	0.123	0.244	0.397	328	0.213	0.124	0.243	0.391	423	0.190	0.110	0.209	0.348	408
EVONS (ours)	**0.226**	**0.131**	**0.261**	**0.413**	229	**0.242**	**0.141**	**0.281**	**0.440**	392	**0.240**	**0.143**	**0.276**	**0.432**	**388**
Improve	4.63%	4.80%	4.82%	1.23%	-	8.52%	11.02%	10.20%	5.26%	-	9.59%	13.49%	9.96%	6.93%	4.90%

Training efficiency. Compared with FastKGE, the fastest baseline on all four datasets, EVONS reduces total training time by 13.3% and 18.9% on ENTITY and RELATION, respectively, but takes 28 and 7 seconds longer on HYBRID and FB-CKGE. Across these benchmarks, its mean total training time is 285.8 seconds versus 304.5 for FastKGE (a 6.2% reduction), while achieving higher scores on all reported ranking metrics.

Performance under varying graph growth. Table 3 covers decreasing, approximately constant, and increasing fact-arrival rates. EVONS leads all four ranking metrics, with relative MRR gains over the strongest baseline of 4.63%, 8.52%, and 9.59% on GRAPHLOWER, GRAPHEQUAL, and GRAPHHIGHER, respectively. Training efficiency varies by dataset: EVONS is second fastest behind FastKGE on the first two benchmarks, but fastest on GRAPHHIGHER, requiring 388 seconds versus 408 for STARK (a 4.90% reduction).

5.2.2 SNAPSHOT-WISE PERFORMANCE

Figure 3 compares EVONS with STARK and LKGE across five snapshots on FB-CKGE and GRAPHEQUAL.

On FB-CKGE, EVONS matches or exceeds both baselines on all four metrics at every snapshot, with widening MRR, Hits@1, and Hits@3 margins over STARK. At Snapshot 4, its MRR reaches 0.231, exceeding STARK by 0.007 (Table 2). On GRAPHEQUAL, EVONS performs similarly to STARK initially but leads all four metrics from Snapshot 1 onward, including Hits@10, where LKGE is the stronger baseline in later snapshots. These results indicate sustained ranking advantages across updates in both settings. The declining aggregate scores do not isolate forgetting, because the evaluated test sets and candidate entity vocabulary also change across snapshots.

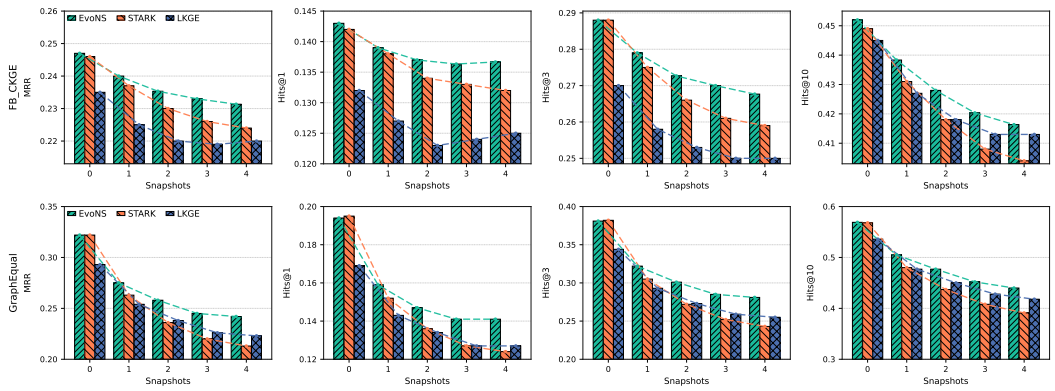


Figure 3: Snapshot-wise link-prediction performance of EvoNS, STARK, and LKGE.

5.3 Ablation Study

The *w/o* DANR and *w/o* HCNS variants remove the corresponding module entirely while retaining the other and keeping all remaining settings unchanged. Table 4 reports results on Entity, Relation, and Hybrid; results on FB-CKGE and the three graph-growth benchmarks are provided in Appendix D.1.

Table 4: Module-wise ablation of HCNS and DANR.

Method	Entity					Relation					Hybrid				
	MRR	Hits@1	Hits@3	Hits@10	T(s)	MRR	Hits@1	Hits@3	Hits@10	T(s)	MRR	Hits@1	Hits@3	Hits@10	T(s)
EvoNS	0.272	0.167	0.317	0.471	352	0.185	0.110	0.219	0.327	240	0.209	0.125	0.247	0.368	354
w/o DANR	0.239	0.147	0.272	0.419	383	0.176	0.106	0.206	0.308	416	0.201	0.120	0.235	0.352	435
w/o HCNS	0.261	0.159	0.303	0.456	323	0.180	0.106	0.211	0.316	275	0.202	0.120	0.238	0.356	367

The complete model leads all reported ranking metrics. Removing DANR degrades all 12 accuracy entries, with mean relative decreases of 6.94% in MRR and 7.07% in Hits@10 across the three datasets. The largest losses occur on Entity, where MRR drops from 0.272 to 0.239 and Hits@10 from 0.471 to 0.419. Removing DANR also increases training time in every setting, most notably on Relation (from 240 to 416 seconds). These results indicate that DANR is the primary contributor to predictive performance and training efficiency in these ablations.

Removing HCNS produces smaller but consistent losses across all 12 accuracy entries, with mean relative decreases of 3.37% in MRR and 3.27% in Hits@10. Its runtime effect is dataset-dependent: removal reduces training time on Entity from 352 to 323 seconds but increases it on the other two datasets. Overall, HCNS provides complementary ranking gains without a systematic efficiency penalty.

5.4 Sensitivity Analysis

We examine how the temperature and relation-mix hyperparameters jointly affect EvoNS's link-prediction performance. Figure 4 reports MRR and Hits@10 on all four benchmarks over a 5×5 grid, with temperatures in $\{0, 0.25, 0.5, 1, 2\}$ and relation-mix values in $\{0, 0.1, 0.25, 0.5, 1\}$.

MRR and Hits@10 exhibit broadly consistent trends within each dataset, while the effect of each hyperparameter depends on the other parameter and the benchmark. On Entity, nonzero relation mixing generally improves both metrics relative to zero mixing. The response to temperature is non-monotonic, with strong results at an intermediate temperature and moderate-to-high relation-mix values. On Relation and Hybrid, increasing temperature is particularly detrimental at high relation-mix values. The combination of temperature 2 and relation mix 1 produces the lowest scores in both metrics, whereas lower temperatures with moderate mixing yield substantially better results.

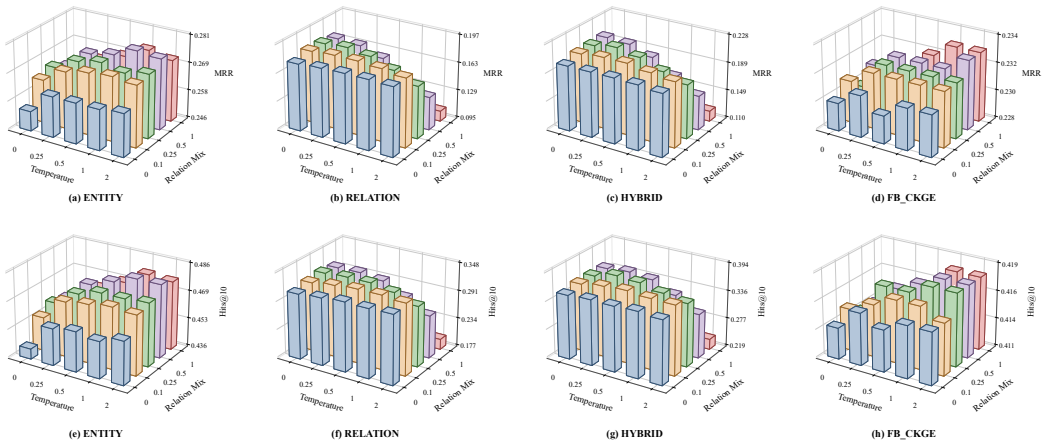


Figure 4: Joint sensitivity of EvoNS to temperature and relation mix.

FB-CKGE exhibits much smaller absolute variation across the tested grid, as reflected by its narrower vertical-axis ranges. Its higher-scoring configurations tend to occur at temperatures of 1–2 and relation-mix values of 0.5–1, although the gains are modest. Taken together, these results show that increasing temperature and relation mixing is not uniformly beneficial across graph-evolution settings, supporting joint hyperparameter selection based on validation performance for each benchmark. Additional metrics and results on the three graph-growth benchmarks are provided in Appendix D.2.

5.5 Convergence Analysis

Figure 5 compares training loss on ENTITY over four incremental snapshots under matched dataset settings and identical early-stopping patience. EvoNS's loss drops sharply in the first few epochs and then changes little. Its runs generally terminate earlier than the baselines, although STARK has comparable run lengths in Snapshots 1 and 3. In Snapshot 4, EvoNS stops before epoch 10, compared with approximately epoch 20 for STARK. Since the methods optimize different objectives, their absolute loss values are not directly comparable and do not alone establish predictive quality. Additional results are provided in Appendix D.3.

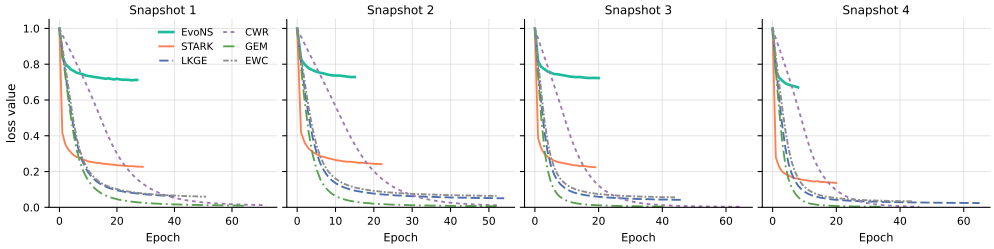


Figure 5: Convergence on ENTITY.

6 Conclusion

We introduced EvoNS, a dual negative learning framework for CKGE. HCNS constructs relation-aware negatives consistent with accumulated training facts, while DANR preserves individual ranking constraints through difficulty-aware weighting. Experiments on seven benchmarks show improved link prediction with competitive training efficiency, highlighting the value of adapting negative construction and utilization to graph and model evolution.

AI USE STATEMENT

In this work, we used generative AI tools for manuscript drafting. Additionally, we used generative AI tools for grammar checking and improving the clarity and concision of the manuscript. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Kian Ahrabian, Aarash Feizi, Yasmin Salehi, William L Hamilton, and Avishek Joey Bose. Structure aware negative sampling in knowledge graphs. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 6093–6101, 2020.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

Liwei Cai and William Yang Wang. Kbgan: Adversarial learning for knowledge graph embeddings. In *Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long papers)*, pp. 1470–1480, 2018.

Xiangnan Chen, Wen Zhang, Zhen Yao, Mingyang Chen, and Siliang Tang. Negative sampling with adaptive denoising mixup for knowledge graph embedding. In *International semantic web conference*, pp. 253–270. Springer, 2023.

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 AAAI conference on artificial intelligence*, volume 37, pp. 4217–4224, 2023.

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.

Yue Jian, Lin Li, Kaize Shi, Junwei Zhou, and Yu Yang. Paal: Pattern-anchor alignment for continual knowledge graph embedding under structural distribution shift. In Diego Calvanese (ed.), *Proceedings of the Thirty-Fifth International Joint Conference on Artificial Intelligence, IJCAI-26*, pp. 3890–3898. International Joint Conferences on Artificial Intelligence Organization, 8 2026. doi: 10.24963/ijcai.2026/433. URL https://doi.org/10.24963/ijcai.2026/433. Main Track.

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.

Xiaoyu Kou, Yankai Lin, Shaobo Liu, Peng Li, Jie Zhou, and Yan Zhang. Disentangle-based continual graph representation learning. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 2961–2972, 2020.

Denis Krompaß, Stephan Baier, and Volker Tresp. Type-constrained representation learning in knowledge graphs. In *International semantic web conference*, pp. 640–655. Springer, 2015.

Kyung-Hwan Lee and Dong-Wan Choi. Stark: Structure-aware and adaptive representation learning for continual knowledge graph embedding. In *Proceedings of the ACM Web Conference 2026*, pp. 3941–3951, 2026.

Haoning Li and Qinghua Huang. Dans-kgc: Diffusion based adaptive negative sampling for knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 22913–22921, 2026.

Linyu Li, Zhi Jin, Yuanpeng He, Dongming Jin, Yichi Zhang, Haoran Duan, Xuan Zhang, Zhengwei Tao, and Tashi Nyima. Learning to evolve: Bayesian-guided continual knowledge graph embedding. In *Proceedings of the ACM Web Conference 2026*, pp. 4372–4383, 2026.

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*, pp. 1600–1611, 2025.

Ke Liang, Lingyuan Meng, Meng Liu, Yue Liu, Wenxuan Tu, Siwei Wang, Sihang Zhou, Xinwang Liu, Fuchun Sun, and Kunlun He. A survey of knowledge graph reasoning on graph types: Static, dynamic, and multi-modal. *IEEE transactions on pattern analysis and machine intelligence*, 46 (12):9456–9478, 2024.

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 AAAI conference on artificial intelligence*, volume 38, pp. 8759–8768, 2024a.

Jiajun Liu, Wenjun Ke, Peng Wang, Jiahao Wang, Jinhua Gao, Ziyu Shang, Guozheng Li, Zijie Xu, Ke Ji, and Yining Li. Fast and continual knowledge graph embedding via incremental lora. In Kate Larson (ed.), *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pp. 2198–2206. International Joint Conferences on Artificial Intelligence Organization, 8 2024b. doi: 10.24963/ijcai.2024/243. URL https://doi.org/10.24963/ijcai.2024/243. Main Track.

Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on robot learning*, pp. 17–26. PMLR, 2017.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Lingyuan Meng, Ke Liang, Hao Yu, Yue Liu, Sihang Zhou, Meng Liu, and Xinwang Liu. Fedean: Entity-aware adversarial negative sampling for federated knowledge graph reasoning. *IEEE Transactions on Knowledge and Data Engineering*, 36(12):8206–8219, 2024.

Lingyuan Meng, Ke Liang, Zeyu Zhu, Xinwang Liu, and Wenpeng Lu. Fs-ken: Few-shot knowledge graph reasoning by adversarial negative enhancing. In *IJCAI*, pp. 4597–4605, 2025.

Trung-Kien Nguyen and Yuan Fang. Diffusion-based negative sampling on graphs for link prediction. In *Proceedings of the ACM Web Conference 2024*, pp. 948–958, 2024.

Jing Qi, Yuxiang Wang, Zhiyuan Yu, Xiaoliang Xu, Yuanshi Zheng, and Tianxing Wu. Multi-faceted continual knowledge graph embedding for semantic-aware link prediction. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '26, pp. 1462–1472, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400725999. doi: 10.1145/3805712.3809683. URL https://doi.org/10.1145/3805712.3809683.

Xiao Qin, Nasrullah Sheikh, Berthold Reinwald, and Lingfei Wu. Relation-aware graph attention model with adaptive self-adversarial training. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 9368–9376, 2021.

Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592*, 2020.

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.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

Hong Wang, Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Shiyu Chang, and William Yang Wang. Sentence embedding alignment for lifelong relation extraction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 796–806, 2019.

Peifeng Wang, Shuangyin Li, and Rong Pan. Incorporating gan for negative sampling in knowledge representation learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28, 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*, pp. 1197–1206, 2025.

Zhen Yang, Ming Ding, Chang Zhou, Hongxia Yang, Jingren Zhou, and Jie Tang. Understanding negative sampling in graph representation learning. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1666–1676, 2020.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pp. 3987–3995. Pmlr, 2017.

Yongqi Zhang, Quanming Yao, Yingxia Shao, and Lei Chen. Nscaching: simple and efficient negative sampling for knowledge graph embedding. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 614–625. IEEE, 2019.

Tianzhe Zhao, Jiaoyan Chen, Yanchi Ru, Qika Lin, Yuxia Geng, Haiping Zhu, Yudai Pan, and Jun Liu. Rethinking continual knowledge graph embedding: Benchmarks and analysis. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 138–147, 2025.

Appendix

CONTENTS

A Comprehensive Related Work **14**

B Extended Preliminaries **16**

B.1 Continual Link Prediction 16

C Detailed Experimental Settings **17**

C.1 Datasets and Snapshot Construction 17

C.2 Baseline Methods 18

C.3 Evaluation Protocol and Metrics 18

C.4 Implementation Details 19

D Additional Experimental Results **19**

D.1 Extended Ablation Studies 19

D.2 Hyperparameter Sensitivity across Datasets 20

D.3 Additional Convergence Results 22

D.4 Case Study 24

E Additional Methodological Details **25**

E.1 Notation Summary 25

E.2 Overall Training Algorithm 26

E.3 Detailed HCNS Procedure 27

E.4 Detailed DANR Procedure 28

F Discussion and Limitations **29**

F.1 Scope and Assumptions 29

F.2 Limitations and Future Directions 29

A Comprehensive Related Work

Continual Knowledge Graph Embedding. CKGE incrementally acquires new facts while retaining previously learned knowledge (Daruna et al., 2021). General continual-learning strategies provide several foundations. PNN preserves earlier network components while adding capacity (Rusu et al., 2016), and CWR separates batch-specific learning from weight consolidation (Lomonaco & Maltoni, 2017). Memory-based approaches constrain updates using episodic gradients (Lopez-Paz & Ranzato, 2017) or combine replay with representation alignment (Wang et al., 2019). EWC and SI instead regularize important parameters, estimating importance through Fisher information and optimization trajectories, respectively (Kirkpatrick et al., 2017; Zenke et al., 2017). These strategies differ in whether historical knowledge is retained through model components, examples, or constraints on parameter updates.

KG-specific methods further exploit relational dependencies. DiCGRL uses disentangled representations for continual multi-relational learning (Kou et al., 2020). LKGE combines a masked KG autoencoder with embedding transfer and regularization to support both new-knowledge acquisition and historical retention (Cui et al., 2023). IncDE organizes incoming triples hierarchically and couples structure-aware incremental distillation with two-stage training (Liu et al., 2024a). These approaches incorporate graph structure into representation learning, transfer, and the ordering of continual updates. MF-CKGE separates old and new knowledge into distinct embedding spaces and aggregates their predictions using query-dependent semantic importance weights (Qi et al., 2026).

Beyond graph size, the PS-CKGE benchmarks identify relational pattern shifts as a source of degradation in retention and adaptation (Zhao et al., 2025). ERPP transfers relational knowledge through snapshot-conditioned path passing and relation inheritance and evolution (Yang et al., 2025). PAAL uses relation-level structural anchors to modulate historical constraints according to distributional alignment (Jian et al., 2026). BAKE formulates updates as sequential Bayesian inference and constrains representation drift through continual clustering (Li et al., 2026). These methods adapt the form or strength of historical knowledge transfer to evolving graph structure.

Capacity and efficiency constitute another line of research. FastKGE learns incremental low-rank adapters with adaptive rank allocation (Liu et al., 2024b). SAGE adjusts embedding dimensions to update scale, combining expansion with limited replay and dynamic distillation (Li et al., 2025). STARK combines structural capacity allocation with an adaptive TransE objective that accounts for tail cardinality and graph growth (Lee & Choi, 2026). These designs connect representation capacity and optimization to the characteristics of incoming knowledge.

Negative Learning for Knowledge Graph Embedding. Negative examples in KGE are commonly constructed by corrupting the head or tail of an observed triple (Bordes et al., 2013). TransH uses relation-dependent Bernoulli replacement to reduce false negatives arising from non-one-to-one relations (Wang et al., 2014). Type-constrained learning restricts candidate entities through relation domain and range information, including constraints inferred from observed triples (Krompaß et al., 2015). SANS instead draws candidates from multi-hop neighborhoods (Ahrabian et al., 2020). These approaches exploit relation mapping properties, semantic compatibility, or graph locality to refine the corruption process.

Model-guided approaches prioritize negatives that remain informative during training. KBGAN (Cai & Wang, 2018) and the adversarial framework of Wang et al. (2018) learn generators to propose useful corruptions. NSCaching maintains a compact, refreshed cache of high-scoring negatives (Zhang et al., 2019), offering an alternative to generator optimization. More recently, DANS-KGC combines semantic and structural difficulty assessment with conditional diffusion and a training-dependent hardness distribution (Li & Huang, 2026). DMNS similarly generates negatives at multiple difficulty levels for general graph link prediction (Nguyen & Fang, 2024).

Specialized learning settings impose additional requirements on negative generation. FedEAN combines adversarial generation with entity-aware adaptation based on representation changes before and after server aggregation in federated KG reasoning (Meng et al., 2024). FS-KEN uses policy-gradient adversarial learning to augment negative examples in few-shot KG reasoning (Meng et al., 2025). Both address negative-example quality under restricted data availability, although their learning protocols differ from continual KGE.

Hardness alone does not establish negative-label reliability in an incomplete KG. The adaptive self-adversarial sampler introduced with RelGNN uses positive relationships to guide selection and reduce false-negative risk (Qin et al., 2021). DeMix addresses potential label noise through adaptive denoising mixup (Chen et al., 2023). These approaches target uncertainty about unobserved triples, which is distinct from filtering candidates already known to be positive.

Negative utilization also depends on the learning objective. RotatE assigns self-adversarial weights from current model scores and evaluates this strategy with both its negative-sampling loss and a margin-based ranking criterion (Sun et al., 2019). Broader analysis connects negative-sampling distributions to the objective and estimation variance in graph representation learning (Yang et al., 2020), while hardness-controlled sampling has also been studied in unsupervised contrastive learning (Robinson et al., 2020). Together, these studies highlight candidate selection, label reliability, and optimization weighting as interrelated aspects of learning from negatives.

B Extended Preliminaries

B.1 Continual Link Prediction

Continual link prediction (CLP) evaluates the model's ability to infer missing facts throughout the evolution of a knowledge graph. Given a query triple with either the head or tail entity masked, e.g., $(?, r, t)$ or $(h, r, ?)$, a CKGE model scores candidate entities and ranks them according to their compatibility with the query. For example, in tail prediction, the model ranks each candidate entity $e \in \mathcal{E}_i$ based on the plausibility of (h, r, e), while head prediction analogously ranks (e, r, t). The ranking performance across successive snapshots reflects the model's ability to acquire emerging knowledge while retaining previously learned knowledge.

C Detailed Experimental Settings

C.1 Datasets and Snapshot Construction

All seven benchmarks are constructed from FB15K-237 and contain five snapshots, indexed from 0 to 4. Each sequence ultimately covers 14,541 entities, 237 relations, and 310,116 facts, but exposes them in a different order. Newly introduced facts at each snapshot are divided into training, validation, and test sets in an approximately 3:1:1 ratio (Cui et al., 2023; Liu et al., 2024b;a).

Entity. This benchmark follows entity-centric growth (Cui et al., 2023), progressively expanding the set of observed entities and their associated facts. It emphasizes learning representations for newly arriving entities while retaining knowledge about previously observed ones.

Relation. The relation-centric benchmark (Cui et al., 2023) progressively introduces relation types and their associated facts. It evaluates adaptation to an expanding relational schema rather than predominantly to new entities.

Hybrid. This benchmark interleaves entity, relation, and fact expansion (Cui et al., 2023). Its uneven snapshot sizes combine changes in graph structure and relation vocabulary, yielding a mixed-growth setting.

FB-CKGE. Introduced by Liu et al. (2024b), this benchmark assigns approximately 60% of all facts to the initial snapshot and 10% to each subsequent snapshot. It evaluates incremental learning from a substantial initial graph, in contrast to settings that distribute growth more evenly.

GraphLower, GraphEqual, and GraphHigher. These benchmarks vary the volume of incoming facts (Liu et al., 2024a): the increments decrease, remain approximately constant, or increase, respectively. Their construction also permits new facts involving only new entities. They therefore test adaptation to different update sizes without requiring every new fact to connect directly to an existing entity.

Tables 5, 6, and 7 provide the full snapshot-wise statistics underlying Table 1. Entity and relation counts are cumulative, whereas training, validation, and test counts refer only to facts introduced at the corresponding snapshot. Relation counts exclude the inverse relations used internally for head prediction.

Table 5: Statistics of the Entity, Relation, and Hybrid datasets across five snapshots. #Ent. and #Rel. denote cumulative entities and relations; #Train, #Valid, and #Test denote triples in the current training, validation, and test sets.

Snapshot	Entity					Relation					Hybrid				
	#Ent.	#Rel.	#Train	#Valid	#Test	#Ent.	#Rel.	#Train	#Valid	#Test	#Ent.	#Rel.	#Train	#Valid	#Test
Snapshot 0	2,909	233	27,832	9,278	9,278	11,560	48	59,291	19,764	19,764	8,628	86	34,535	11,513	11,513
Snapshot 1	5,817	236	43,265	14,423	14,423	13,343	96	56,119	18,708	18,708	10,040	102	12,523	4,175	4,175
Snapshot 2	8,275	236	44,269	14,758	14,758	13,754	143	39,680	13,228	13,228	12,779	151	52,809	17,604	17,604
Snapshot 3	11,633	237	42,302	14,102	14,102	14,387	190	18,018	6,007	6,007	14,393	209	62,003	20,668	20,668
Snapshot 4	14,541	237	28,394	9,466	9,466	14,541	237	12,956	4,319	4,319	14,541	237	24,194	8,066	8,066

Table 6: Statistics of the three graph-evolution datasets across five snapshots.

Snapshot	GraphLower					GraphEqual					GraphHigher				
	#Ent.	#Rel.	#Train	#Valid	#Test	#Ent.	#Rel.	#Train	#Valid	#Test	#Ent.	#Rel.	#Train	#Valid	#Test
Snapshot 0	7,505	237	96,000	32,000	32,000	2,908	226	34,581	11,527	11,528	900	197	6,000	2,000	2,000
Snapshot 1	11,258	237	48,000	16,000	16,000	5,816	235	37,213	12,404	12,406	1,838	221	12,000	4,000	4,000
Snapshot 2	13,134	237	24,000	8,000	8,000	8,724	237	37,213	12,404	12,406	3,714	234	24,000	8,000	8,000
Snapshot 3	14,072	237	12,000	4,000	4,000	11,632	237	37,213	12,404	12,406	7,467	237	48,000	16,000	16,000
Snapshot 4	14,541	237	6,069	2,023	2,024	14,541	237	39,846	13,282	13,283	14,541	237	96,069	32,023	32,024

Table 7: Statistics of FB-CKGE across five snapshots.

Snapshot	FB-CKGE				
	#Ent.	#Rel.	#Train	#Valid	#Test
Snapshot 0	7,505	237	111,642	37,214	37,214
Snapshot 1	11,258	237	18,607	6,202	6,203
Snapshot 2	13,134	237	18,607	6,202	6,203
Snapshot 3	14,072	237	18,607	6,202	6,203
Snapshot 4	14,541	237	18,606	6,202	6,202

C.2 Baseline Methods

We compare against six general continual-learning baselines adapted to KGE and three CKGE-specific methods. The following descriptions summarize the core mechanism of each baseline.

PNN (Rusu et al., 2016) expands model capacity for new tasks while freezing previously learned components. New components can reuse historical representations without modifying earlier parameters, providing a parameter-isolation approach to retaining acquired knowledge.

CWR (Lomonaco & Maltoni, 2017) maintains temporary and consolidated parameters. Temporary parameters are re-initialized for each new training stage, and the learned weights are then consolidated with stored weights. This separates learning from incoming data from the long-term storage of previously acquired knowledge.

GEM (Lopez-Paz & Ranzato, 2017) stores examples from earlier tasks and uses their gradients to constrain current updates. When gradients conflict, it projects the current gradient onto a feasible region designed to avoid increasing the memory losses under a first-order approximation.

EMR (Wang et al., 2019) retains a memory of past training examples and replays sampled examples alongside new data. Unlike GEM, it directly optimizes the combined training objective without gradient projection, preserving historical knowledge through repeated exposure to earlier facts.

EWC (Kirkpatrick et al., 2017) estimates parameter importance using the diagonal Fisher information. A weighted quadratic penalty discourages changes to parameters important for previous tasks while allowing less constrained parameters to adapt to new knowledge.

SI (Zenke et al., 2017) estimates parameter importance from each parameter's contribution to loss reduction along the optimization trajectory. It accumulates these importance estimates across tasks and regularizes subsequent changes, retaining previous knowledge without requiring an episodic replay memory.

LKGE (Cui et al., 2023) combines a masked KG autoencoder, embedding transfer, and embedding regularization. The autoencoder connects new and historical graph information, transfer initializes newly introduced entities and relations from learned knowledge, and regularization limits disruptive changes to existing embeddings.

FastKGE (Liu et al., 2024b) represents newly introduced entities and relations with incremental low-rank adapters while retaining earlier embeddings. It organizes new entities into graph layers and allocates adapter ranks according to structural importance, reducing the number of trainable parameters during incremental updates.

STARK (Lee & Choi, 2026) combines structural novelty prioritization with an adaptive TransE objective. It uses PageRank and betweenness centrality to guide representational capacity allocation, and adjusts embedding constraints according to the tail cardinality of head–relation queries, adapting both capacity and optimization to graph structure.

C.3 Evaluation Protocol and Metrics

Evaluation protocol. After learning snapshot i, the selected model M_i is evaluated on the test facts introduced at every snapshot $j \leq i$. Each test triple yields one head-prediction query and one tail-prediction query. For a test set from snapshot j, candidate entities are drawn from $\mathcal{E}_j$, the

entities available at that snapshot. We use filtered ranking: alternative correct entities identified from training, validation, and test facts observed through snapshot j are removed from the ranking, while the target entity is retained. These evaluation filters are separate from HCNS, whose training-time filter uses only observed training facts.

Ranking metrics. Let $\mathcal{Q}_j^{\text{test}}$ denote the head- and tail-prediction queries derived from the test triples of snapshot j, and let $\text{rank}_{i,j}(q) \geq 1$ be the filtered rank of the correct entity for query q under M_i. Mean reciprocal rank and Hits@k are defined as

$$\text{MRR}_{i,j} = \frac{1}{|\mathcal{Q}_j^{\text{test}}|} \sum_{q \in \mathcal{Q}_j^{\text{test}}} \frac{1}{\text{rank}_{i,j}(q)}, \qquad \text{Hits@}\,k_{i,j} = \frac{1}{|\mathcal{Q}_j^{\text{test}}|} \sum_{q \in \mathcal{Q}_j^{\text{test}}} \mathbf{1}[\text{rank}_{i,j}(q) \leq k], \quad (5)$$

where $k \in \{1, 3, 10\}$ and $\mathbf{1}[\cdot]$ is the indicator function. MRR emphasizes the position of the correct entity throughout the ranked list, whereas Hits@k measures its frequency within the top-k candidates.

Aggregation across snapshots. For either metric m, scores after learning snapshot i are aggregated as

$$m_i = \frac{\sum_{j=0}^{i} |\mathcal{Q}_j^{\text{test}}|\, m_{i,j}}{\sum_{j=0}^{i} |\mathcal{Q}_j^{\text{test}}|}. \quad (6)$$

Because every test triple contributes two queries, this is equivalent to weighting snapshot-level scores by their numbers of test triples, rather than assigning equal weight to snapshots of different sizes. The main tables report m_4, after all five snapshots have been learned, while the snapshot-wise results report m_i at each stage.

Training time. We report $T = \sum_{i=0}^{4} T_i$ in seconds, where T_i is the measured wall-clock duration of the training loop for snapshot i. This includes per-epoch validation and checkpoint saving but excludes post-training test evaluation. Higher MRR and Hits@k and lower T indicate better predictive performance and training efficiency, respectively.

C.4 Implementation Details

We implement EvoNS in PyTorch 2.1.2 with CUDA 12.1 and conduct experiments on an NVIDIA GeForce RTX 4080 GPU with driver version 535.183.06. The model uses the TransE scoring function with ℓ_1 distance and is optimized with Adam. Each dataset comprises five snapshots, processed sequentially. The default batch size and learning rate are 1024 and 10^{-4}, respectively, and the negative-to-positive sampling ratio is 10:1. We evaluate validation MRR after each epoch and stop training after three consecutive epochs without improvement, restoring the best checkpoint for evaluation and subsequent snapshot updates. For incremental snapshots, HCNS constructs its filtering index and relation-aware candidate pools using only cumulative training facts; validation and test facts are excluded from negative construction. Convergence comparisons use identical dataset settings and early-stopping patience across methods.

D Additional Experimental Results

D.1 Extended Ablation Studies

Table 8 extends the main-text module-wise ablations to FB-CKGE under the same protocol. Removing DANR reduces MRR from 0.231 to 0.226 and Hits@10 from 0.416 to 0.408, while increasing training time from 197 to 203 seconds. Removing HCNS leaves MRR and Hits@10 unchanged at the reported precision, but slightly reduces Hits@1 and Hits@3 and increases training time to 210 seconds. Thus, the accuracy contribution of HCNS is smaller in this large-initial-graph setting.

Table 8: Module-wise ablation of HCNS and DANR on FB-CKGE. Higher MRR and Hits@k and lower T (s) indicate better performance.

Method	MRR	Hits@1	Hits@3	Hits@10	T(s)
EvoNS	0.231	0.137	0.268	0.416	197
w/o DANR	0.226	0.133	0.260	0.408	203
w/o HCNS	0.231	0.136	0.267	0.416	210

Table 9: MRR ablations on the graph-growth benchmarks. Best results are in bold.

Method	GraphLower	GraphEqual	GraphHigher
EvoNS	**0.226**	**0.242**	**0.240**
w/o DANR	0.217	0.217	0.196
w/o HCNS	0.224	0.237	0.227

Table 9 compares the complete model with its two single-module ablations. The complete model achieves the highest MRR on each dataset, with larger decreases when DANR is removed than when HCNS is removed.

D.2 Hyperparameter Sensitivity across Datasets

We extend the sensitivity analysis with additional ranking metrics and graph-growth benchmarks. All plots use the same grid as Figure 4: temperatures in $\{0, 0.25, 0.5, 1, 2\}$ and relation-mix values in $\{0, 0.1, 0.25, 0.5, 1\}$. Vertical axes use panel-specific ranges and do not start at zero.

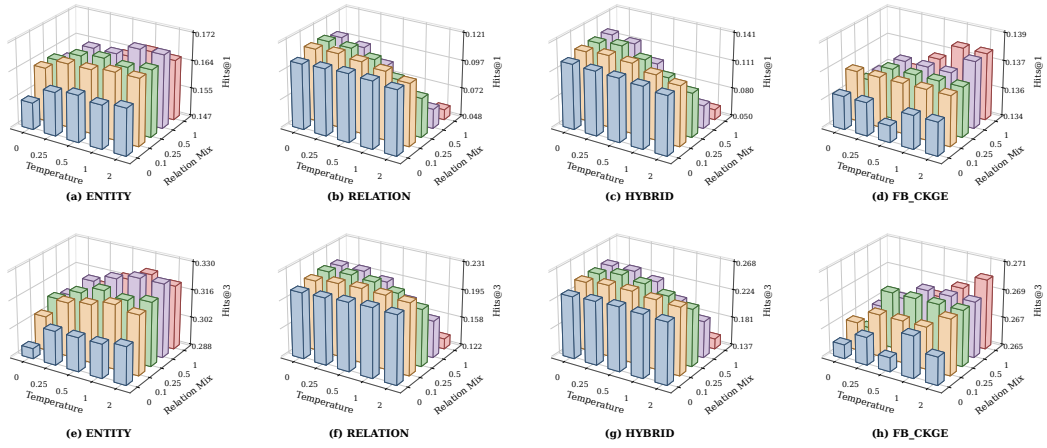


Figure 6: Additional sensitivity metrics for EvoNS. The top and bottom rows report Hits@1 and Hits@3, respectively, on Entity, Relation, Hybrid, and FB-CKGE from left to right. The grid matches Figure 4; vertical axes use panel-specific ranges and do not start at zero.

Additional metrics on the main benchmarks. Figure 6 complements the main-text MRR and Hits@10 results with Hits@1 and Hits@3. On Entity, nonzero relation mixing generally improves both metrics, while the response to temperature is non-monotonic. On Relation and Hybrid, combining high temperature with strong relation mixing substantially reduces both metrics. FB-CKGE exhibits a narrower range of variation and metric-dependent local fluctuations. These observations extend the main-text trends to the stricter top-ranked predictions.

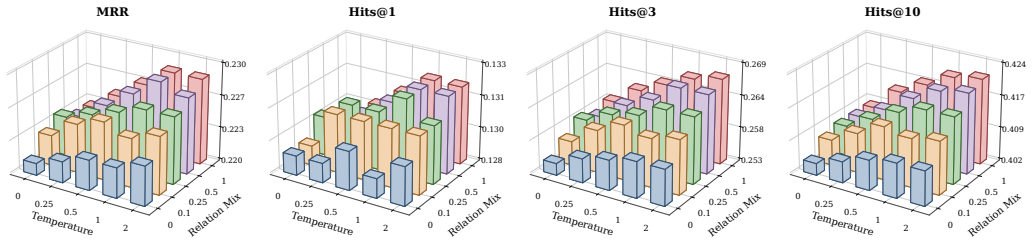


Figure 7: Sensitivity to temperature and relation mix on GRAPHLOWER. Panels report MRR, Hits@1, Hits@3, and Hits@10 from left to right.

GRAPHLOWER. Figure 7 shows generally higher scores with stronger relation mixing across all four metrics. Temperature has a smaller, non-monotonic effect, and the preferred value varies with the mixing ratio and metric. For this decreasing-growth sequence, relation-aware construction remains useful across the tested temperatures, without requiring the strongest concentration on difficult negatives.

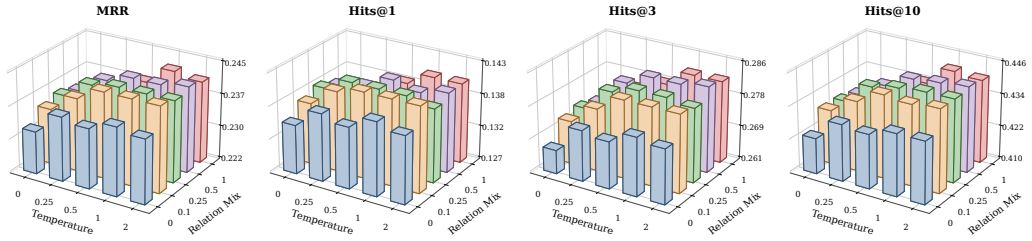


Figure 8: Sensitivity to temperature and relation mix on GRAPHEQUAL. Panels report MRR, Hits@1, Hits@3, and Hits@10 from left to right.

GRAPHEQUAL. Figure 8 also favors nonzero relation mixing over global-only proposals. Improvements occur across MRR and the three Hits metrics, while temperature produces local fluctuations rather than a uniform increase or decrease. Thus, for approximately constant snapshot sizes, performance benefits from relation-aware candidates across several temperature settings rather than only at an isolated combination.

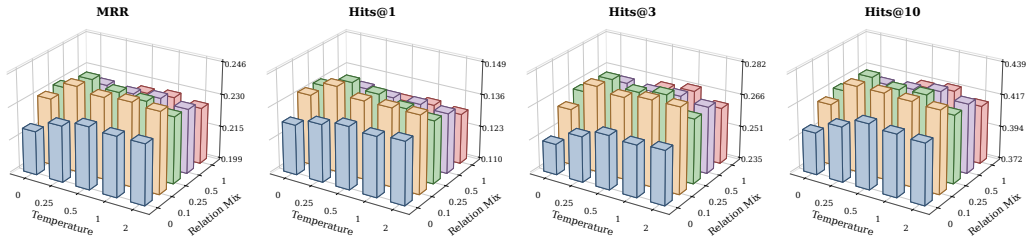


Figure 9: Sensitivity to temperature and relation mix on GRAPHHIGHER. Panels report MRR, Hits@1, Hits@3, and Hits@10 from left to right.

GRAPHHIGHER. Figure 9 shows a different response: moderate relation mixing and lower-to-intermediate temperatures generally yield higher scores. Increasing both parameters does not improve performance further, and strong mixing is less favorable than on GRAPHLOWER. This suggests that more concentrated negative construction and weighting are not uniformly beneficial as the number of incoming facts grows.

Summary. The supplementary results show that the interaction between temperature and relation mix extends across ranking metrics, but varies by dataset. Even the three graph-growth benchmarks, which share the same final graph, favor different parameter regions. The two hyperparameters should therefore be selected jointly using validation performance rather than increased independently. These plots describe empirical responses over the tested grid; their panel-specific scales preclude direct comparisons of bar heights across datasets.

D.3 Additional Convergence Results

We extend Section 5.5 to HYBRID, FB-CKGE, and the three graph-growth benchmarks. Each figure compares EVONS with STARK, LKGE, CWR, GEM, and EWC over incremental Snapshots 1–4. Curves show training loss against epochs, using matched dataset settings and an early-stopping patience of three epochs; horizontal-axis ranges vary by panel.

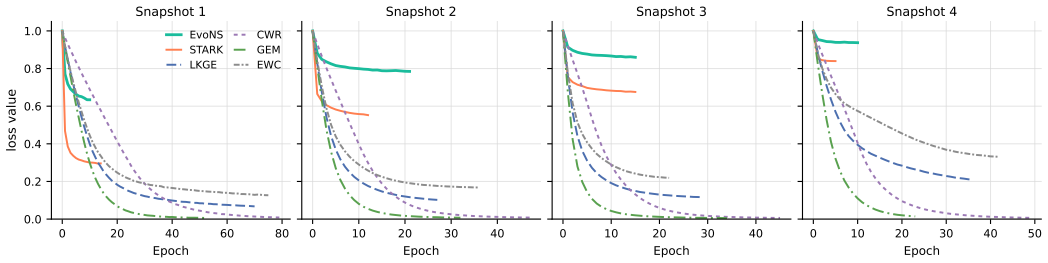


Figure 10: Training-loss trajectories on HYBRID. Panels correspond to incremental Snapshots 1–4 from left to right.

HYBRID. Figure 10 shows an initial loss decrease for EVONS followed by smaller changes, with higher plateaus in later snapshots. Its runs are shorter than CWR's throughout the sequence, but not uniformly shorter than STARK's: STARK stops earlier in Snapshots 2 and 4. The curves therefore show snapshot-dependent optimization behavior rather than a fixed ordering of all methods by stopping epoch.

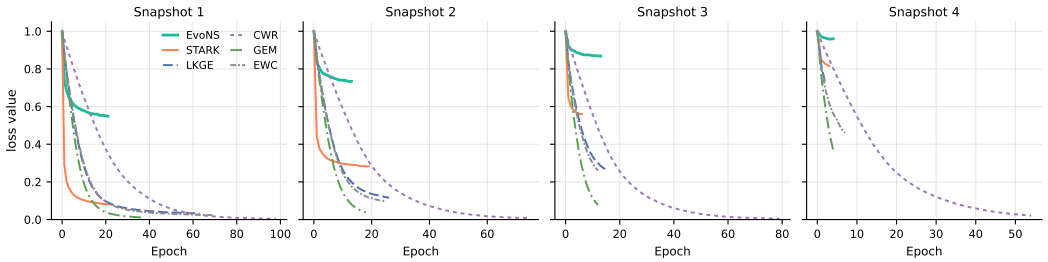


Figure 11: Training-loss trajectories on FB-CKGE. Panels correspond to incremental Snapshots 1–4 from left to right.

FB-CKGE. In Figure 11, EVONS's runs become shorter from Snapshot 1 to Snapshot 4, while the loss changes become smaller in later snapshots. CWR continues for substantially more epochs in each panel. Other baselines can stop earlier than EVONS, as STARK does in Snapshot 3, so the advantage in run length is comparator-dependent. The high final-snapshot plateau alone does not indicate worse link prediction.

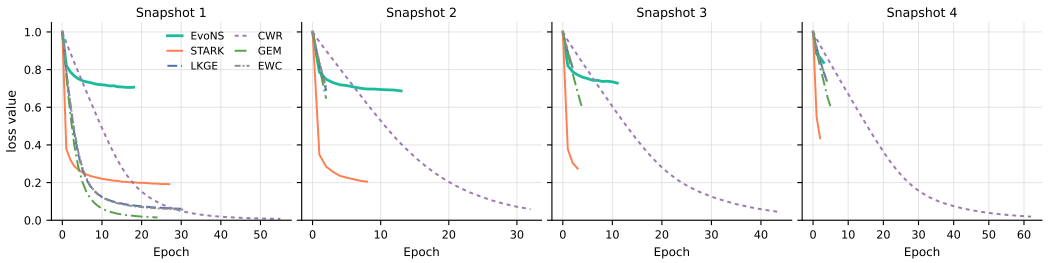


Figure 12: Training-loss trajectories on GRAPHLOWER. Panels correspond to incremental Snapshots 1–4 from left to right.

GRAPHLOWER. Figure 12 shows that EVONS's loss changes are concentrated near the beginning of training. Its runs shorten across the sequence as the incoming snapshots become smaller, ending within the first few epochs in Snapshot 4. CWR continues much longer, whereas several other baselines also terminate early in the later snapshots. Decreasing snapshot size and changing validation behavior accompany this trend, so it does not isolate an effect of the learning objective.

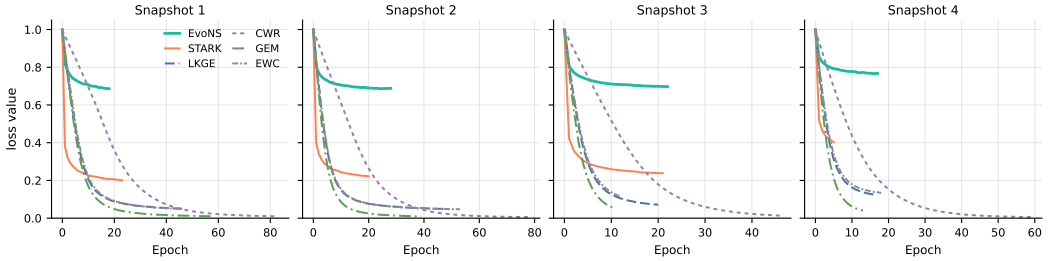


Figure 13: Training-loss trajectories on GRAPHEQUAL. Panels correspond to incremental Snapshots 1–4 from left to right.

GRAPHEQUAL. Figure 13 exhibits rapid initial changes followed by relatively flat trajectories for EVONS. Its runs finish well before CWR's across all four snapshots. However, GEM stops earlier in Snapshot 3, and STARK stops earlier in Snapshot 4. Even with approximately constant snapshot sizes, the stopping order thus varies with the newly observed graph content.

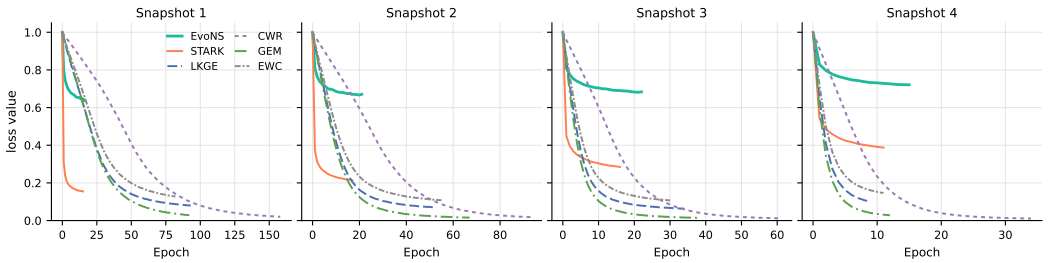


Figure 14: Training-loss trajectories on GRAPHHIGHER. Panels correspond to incremental Snapshots 1–4 from left to right.

GRAPHHIGHER. Figure 14 shows the largest separation in run length at Snapshot 1: EVONS and STARK stop within approximately 20 epochs, while the other methods continue for around 90 epochs or more. This gap narrows in later snapshots. At Snapshot 4, several baselines terminate earlier than EVONS, whose loss still changes most strongly near the start. The advantage in early-snapshot run length does not persist uniformly as the graph grows.

Summary. Across datasets, EvoNS typically exhibits its largest training-loss changes in the initial epochs, but relative stopping times vary by snapshot and baseline. Early stopping is governed by validation MRR, not the plotted loss, and different objectives make absolute loss levels incomparable as measures of predictive quality. Moreover, an epoch contains different numbers of updates across snapshots. These curves therefore complement the ranking results and measured wall-clock times, without establishing that shorter trajectories or lower plateaus necessarily imply better performance.

D.4 Case Study

Table 10 presents three selected tail-prediction cases from Entity at Snapshot 4, using each method's final checkpoint. Evaluation ranks all entities observed through Snapshot 4 and filters cumulative known true tails other than the target.

Table 10: Illustrative tail-prediction cases on Entity. Rank denotes the target's filtered rank (lower is better). Candidates are listed from rank 1 to 5, with the target in bold.

Method	Rank	Top-5 candidates
High-frequency relation: (Quiz Show, `film_release_region`, ?)		
Target: South Korea		
FastKGE	11	Venezuela; Iceland; Switzerland; Belgium; Colombia
EvoNS w/o DANR	8	Italy; Venezuela; Panama; Belize; Belgium
EvoNS w/o HCNS	9	Belgium; Iceland; Colombia; Switzerland; Italy
EvoNS	**2**	Belgium; **South Korea**; Italy; Mexico; New Zealand
Medium-frequency relation: (Sophie Thompson, `place_of_birth`, ?)		
Target: London		
FastKGE	26	Lewisham; Shropshire; Southport; Santa Barbara; City of Preston, Lancashire
EvoNS w/o DANR	14	Shropshire; Hammersmith; Lewisham; Ealing; Essex
EvoNS w/o HCNS	5	Lewisham; Salisbury; New York City; Hammersmith; **London**
EvoNS	**1**	**London**; Hammersmith; Essex; Fulham; York
Multi-tail query: (Skate punk, `artists`, ?)		
Target: NOFX		
FastKGE	295	Gorillaz; Limp Bizkit; No Doubt; Grateful Dead; Aerosmith
EvoNS w/o DANR	135	No Doubt; Gorillaz; Soundgarden; Limp Bizkit; Red Hot Chili Peppers
EvoNS w/o HCNS	36	Sparks; Linkin Park; The B-52's; Pink Floyd; The Black Keys
EvoNS	**2**	Sonic Youth; **NOFX**; Weezer; Faith No More; Van Halen

For the release-region query about *Quiz Show*, EvoNS ranks South Korea second, compared with ranks 8–11 for the three alternatives. For Sophie Thompson's birthplace, it ranks London first, ahead of both ablations (ranks 14 and 5) and FastKGE (rank 26). The multi-tail query shows a larger separation: EvoNS places NOFX second, whereas the ablations rank it 135th and 36th, and FastKGE ranks it 295th. In these three examples, combining HCNS and DANR yields better target rankings than either ablated variant.

E Additional Methodological Details

This section summarizes the notation and expands the training procedure in Section 4. The algorithms describe the learning steps independently of data structures and batching optimizations.

E.1 Notation Summary

Table 11 collects the main symbols. For consistency with the experiments, the pseudocode indexes the initial snapshot by 0 and subsequent updates by $1, \ldots, N_{\text{snap}} - 1$. We write $\Delta\mathcal{T}_i^{\text{tr}}$ for the training facts introduced at snapshot i and $\mathcal{T}_i^{\text{tr}} = \bigcup_{s=0}^{i} \Delta\mathcal{T}_s^{\text{tr}}$ for their cumulative union. The former supplies positive training examples, while the latter supports history-consistent negative construction.

Table 11: Notation used in the methodological description and algorithms.

Symbol	Description
N_{snap}, i	Number of snapshots and current snapshot index.
$\mathcal{E}_i$, $\mathcal{R}_i$	Entities and relations observed through snapshot i.
$\Delta\mathcal{T}_i^{\text{tr}}$	Newly available training facts at snapshot i.
$\mathcal{T}_i^{\text{tr}}$	Cumulative training facts available through snapshot i.
M_i	Model at snapshot i.
$x^+ = (h, r, t)$, c	Positive fact and the head or tail position selected for corruption.
$\mathcal{D}_i(r)$, $\mathcal{A}_i(r)$	Historical head and tail entity pools of relation r.
$\mathcal{P}_i^c(x^+)$	Replacements that reconstruct an observed training fact.
$\mathcal{C}_i^c(x^+)$	Global replacements excluding $\mathcal{P}_i^c(x^+)$.
$\mathcal{Q}_i^c(x^+)$	History-consistent replacements within the relation-specific pool.
K, $\mathcal{N}_i(x^+)$	Number and collection of negatives associated with a positive fact.
ρ, τ, γ	Relation-mix probability, difficulty temperature, and ranking margin.
d^+, d_j^-	Model distances for the positive fact and its j-th negative.
a_j, ℓ_j	Difficulty weight and individual margin violation of negative j.
$\mathcal{B}$, B	Mini-batch of positive facts and its size.
$\mathcal{U}(\cdot)$, $\text{sg}(\cdot)$	Uniform sampling and the stop-gradient operator.

E.2 Overall Training Algorithm

Algorithm 1 separates initial representation learning from continual adaptation. The initial model is trained with uniform entity corruption and a margin-based KGE objective. At each subsequent snapshot, the model retains previously learned representations and introduces trainable incremental parameters for newly observed entities and relations. These parameters use a low-rank representation in our implementation; HCNS and DANR determine the negative examples and their training signals.

Algorithm 1 Overall training procedure of EvoNS

Input: Snapshot training sets $\{\Delta\mathcal{T}_i^{\mathrm{tr}}\}_{i=0}^{N_{\mathrm{snap}}-1}$ and validation sets; negative count K, relation mix ρ, temperature τ, margin γ; batch size B, learning rate η, epoch limit L, patience p
Output: Snapshot models $M_0, \ldots, M_{N_{\mathrm{snap}}-1}$

Train M_0 on $\Delta\mathcal{T}_0^{\mathrm{tr}}$ and retain its best validation checkpoint
$\mathcal{T}_0^{\mathrm{tr}} \leftarrow \Delta\mathcal{T}_0^{\mathrm{tr}}$
for $i = 1, \ldots, N_{\mathrm{snap}} - 1$ **do**
    $\mathcal{T}_i^{\mathrm{tr}} \leftarrow \mathcal{T}_{i-1}^{\mathrm{tr}} \cup \Delta\mathcal{T}_i^{\mathrm{tr}}$
    Update the observed vocabulary and training-fact and relation-pool indices
    Initialize M_i from M_{i-1} for the current snapshot
    Initialize the best validation score and non-improvement counter
    for epoch $= 1, \ldots, L$ **do**
        for each positive mini-batch $\mathcal{B}$ from $\Delta\mathcal{T}_i^{\mathrm{tr}}$ **do**
            for each $x^+ \in \mathcal{B}$ **do**
                $\mathcal{N}_i(x^+) \leftarrow \mathrm{HCNS}(x^+, i, K, \rho)$ ▷ Algorithm 2
            end for
            $\mathcal{L} \leftarrow \mathrm{DANR}(M_i, \mathcal{B}, \{\mathcal{N}_i(x^+)\}, \tau, \gamma)$ ▷ Algorithm 3
            Optimize the trainable parameters of M_i with Adam on $\mathcal{L}$ using learning rate η
        end for
        Evaluate validation MRR; retain the best model and update the counter
        if validation MRR has not improved for p consecutive epochs **then**
            break
        end if
    end for
    Restore the best M_i and retain its learned representations for the next snapshot
end for
return $M_0, \ldots, M_{N_{\mathrm{snap}}-1}$

The cumulative training history is used to identify conflicting negatives and relation-specific candidate pools, rather than replayed as positive training data. Within each mini-batch, HCNS supplies K negatives for each current positive, and DANR converts their scores into the optimization objective. Historical parameters remain fixed during this update. Validation MRR selects the snapshot checkpoint, whereas validation and test facts are excluded from the HCNS indices. Thus, training-time consistency checking and evaluation-time filtering serve distinct purposes.

E.3 Detailed HCNS Procedure

Algorithm 2 expands the candidate-construction, checking, and correction stages in Section 4.1. For notational convenience, let $\mathcal{S}_i^h(r) = \mathcal{D}_i(r)$ and $\mathcal{S}_i^t(r) = \mathcal{A}_i(r)$ denote the relation pool for the chosen corruption side. The operation $\mathrm{Corrupt}(x^+, c, e)$ replaces that side of x^+ with entity e, leaving its other components unchanged. We assume that a history-consistent global replacement exists for each query side used in sampling.

Algorithm 2 History-Consistent Negative Sampling (HCNS)

Input: Positive fact $x^+ = (h, r, t)$, snapshot i, negative count K, relation mix ρ; observed entities $\mathcal{E}_i$, cumulative training facts $\mathcal{T}_i^{\mathrm{tr}}$, relation pools
Output: Collection $\mathcal{N}_i(x^+)$ of K history-consistent negatives

for $j = 1, \ldots, K$ **do**
  Select $c \in \{h, t\}$ with equal probability
  Draw $z \sim \mathrm{Bernoulli}(\rho)$
  if $z = 1$ and $\mathcal{Q}_i^c(x^+) \neq \emptyset$ **then**
    $\mathcal{S} \leftarrow \mathcal{S}_i^c(r)$ ▷ Relation-conditioned proposal source
  else
    $\mathcal{S} \leftarrow \mathcal{E}_i$ ▷ Global source, including fallback
  end if
  Draw $e \sim \mathcal{U}(\mathcal{S})$ and construct $\widetilde{x}^- \leftarrow \mathrm{Corrupt}(x^+, c, e)$
  if $\widetilde{x}^- \in \mathcal{T}_i^{\mathrm{tr}}$ **then**
    Draw $e \sim \mathcal{U}(\mathcal{S} \cap \mathcal{C}_i^c(x^+))$ ▷ Correct within the chosen source
  end if
  $x_j^- \leftarrow \mathrm{Corrupt}(x^+, c, e)$
end for
return $\mathcal{N}_i(x^+) = (x_1^-, \ldots, x_K^-)$

The proposal source is chosen before the sampled triple is checked against training history. A non-conflicting candidate is retained; a conflicting candidate is replaced by a direct draw from the legal portion of the same source. Thus, correction preserves the relation/global mixture instead of repeatedly changing the source after rejection. If a relation pool contains no legal replacement, the global branch is used. This availability check avoids proposing from an exhausted pool and is equivalent to the fallback described in the main text.

The sets in the pseudocode specify sampling supports; they need not be enumerated afresh for every draw. Query-specific positive indices and historical relation pools provide the required membership information. Sampling is with replacement, so K negatives need not be distinct. The result excludes observed training positives, including the original fact, but does not certify that every retained triple is false in the underlying world. This distinction is important when the graph is incomplete.

E.4 Detailed DANR Procedure

Algorithm 3 operates on each positive fact together with its own negative collection. The model first computes distances, with lower values indicating more plausible triples. These distances feed two parallel computations: a normalized difficulty weight and a separate margin violation for each negative. Their outputs are combined only after the individual violations have been evaluated.

Algorithm 3 Difficulty-Aware Negative Reweighting (DANR)

Input: Current model M_i, positive mini-batch $\mathcal{B}$ of size B; negative collections $\{\mathcal{N}_i(x_b^+)\}_{b=1}^B$, temperature τ, margin γ

Output: Mini-batch objective $\mathcal{L}_{\mathrm{DANR}}^{(i)}$

for $b = 1, \ldots, B$ **do**
  Compute $d_b^+ \leftarrow d_{M_i}(x_b^+)$ and $d_{b,j}^- \leftarrow d_{M_i}(x_{b,j}^-)$ for $j = 1, \ldots, K$
  $u_{b,j} \leftarrow -\tau \ \mathrm{sg}(d_{b,j}^-)$ for $j = 1, \ldots, K$
  $a_{b,j} \leftarrow \exp(u_{b,j}) / \sum_{k=1}^K \exp(u_{b,k})$ for $j = 1, \ldots, K$
  for $j = 1, \ldots, K$ **do**
    $\ell_{b,j} \leftarrow \max(0, \gamma + d_b^+ - d_{b,j}^-)$
  end for
  $\mathcal{L}_b \leftarrow \sum_{j=1}^K a_{b,j} \ell_{b,j}$
end for
return $\mathcal{L}_{\mathrm{DANR}}^{(i)} = B^{-1} \sum_{b=1}^B \mathcal{L}_b$

Weights are normalized within each positive's negative collection, not across the entire mini-batch. The stop-gradient operation applies only to the weighting branch: distances in the individual hinge losses remain differentiable, allowing gradients to reach the trainable incremental parameters. Although weights are held fixed during each backward pass, they are recomputed from the current model at every forward pass.

At $\tau = 0$, this procedure reduces to the mean of the individual hinge losses, not a hinge applied after averaging negative distances. It therefore retains negative-level constraints even without difficulty reweighting. For $\tau > 0$, lower-distance negatives receive greater relative emphasis, while a negative already satisfying the margin contributes zero loss. The algorithm uses the positive-wise mean convention of the main text; the implementation additionally divides by $K + 1$ when normalizing over all sampled triples. For fixed K, this is a uniform loss scale and does not change the relative contributions of negatives.

F Discussion and Limitations

F.1 Scope and Assumptions

The method targets snapshot-based, append-only knowledge graph growth: entities, relations, and training facts accumulate over time. Historical training facts are assumed to remain valid and accessible for consistency checking. Negative construction uses only the information available through the current snapshot, without consulting future, validation, or test facts. Although earlier facts are not replayed as positive training examples, their cumulative indices are retained; the approach is therefore not historical-memory-free.

History consistency means avoiding conflicts with observed training facts, not establishing that a candidate is globally false. Likewise, historical head and tail pools capture empirical relation roles rather than formal entity types or ontological constraints. The sampling procedure assumes that each selected query side has at least one legal replacement. Our experiments instantiate the method with a distance-based TransE model and fixed historical representations, while learning incremental parameters for newly observed entities and relations.

F.2 Limitations and Future Directions

Incomplete graphs can contain unobserved true facts among the retained negatives. Relation-aware proposals may select such facts, and difficulty weighting may emphasize them when the model considers them plausible. Neither component resolves this ambiguity by itself. Incorporating uncertainty estimates or additional evidence about candidate validity is a direction for improving robustness to missing or noisy observations.

The cumulative filtering indices and relation pools grow with the observed graph, introducing a storage cost beyond the trainable parameters. The current design also assumes persistent facts and does not explicitly handle deletions or corrections. Moreover, freezing historical representations can limit adaptation when new evidence changes relationships among existing entities. Bounded-memory indexing and selective revision of historical representations are possible extensions, but their consistency–adaptation trade-offs remain to be evaluated.

Empirical evidence is limited to the seven evaluated benchmarks, five-snapshot sequences, and the chosen embedding model. The sensitivity results also show that a single temperature and relation-mix setting is not uniformly preferable. Longer and larger-scale streams, alternative scoring models, and adaptive hyperparameter selection would help assess the broader applicability of the approach. The present convergence plots characterize observed training behavior rather than provide optimization guarantees.