Fast and Continual Knowledge Graph Embedding via Incremental LoRA

Jiajun Liu[1] , **Wenjun Ke**[1,2∗] , **Peng Wang**[1,2∗] , **Jiahao Wang**[1] , **Jinhua Gao**[3] , **Ziyu Shang**[1] , **Guozheng Li**[1] , **Zijie Xu**[1] , **Ke Ji**[1] and **Yining Li**[4]
[1]School of Computer Science and Engineering, Southeast University
[2]Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education
[3]Institute of Computing Technology, Chinese Academy of Sciences
[4]College of Software Engineering, Southeast University
{jiajliu, kewenjun, pwang, wang_jh, ziyus1999, gzli, zijiexu, keji, liyining}@seu.edu.cn, {gaojinhua}@ict.ac.cn

Abstract

Continual Knowledge Graph Embedding (CKGE) aims to efficiently learn new knowledge and simultaneously preserve old knowledge. Dominant approaches primarily focus on alleviating catastrophic forgetting of old knowledge but neglect efficient learning for the emergence of new knowledge. However, in real-world scenarios, knowledge graphs (KGs) are continuously growing, which brings a significant challenge to fine-tuning KGE models efficiently. To address this issue, we propose a fast CKGE framework (FastKGE), incorporating an incremental low-rank adapter (IncLoRA) mechanism to efficiently acquire new knowledge while preserving old knowledge. Specifically, to mitigate catastrophic forgetting, FastKGE isolates and allocates new knowledge to specific layers based on the fine-grained influence between old and new KGs. Subsequently, to accelerate fine-tuning, FastKGE devises an efficient IncLoRA mechanism, which embeds the specific layers into incremental low-rank adapters with fewer training parameters. Moreover, IncLoRA introduces adaptive rank allocation, which makes the LoRA aware of the importance of entities and adjusts its rank scale adaptively. We conduct experiments on four public datasets and two new datasets with a larger initial scale. Experimental results demonstrate that FastKGE can reduce training time by 34%-49% while still achieving competitive link prediction performance against state-of-the-art models on four public datasets (average MRR score of 21.0% vs. 21.1%). Meanwhile, on two newly constructed datasets, FastKGE saves 51%-68% training time and improves link prediction performance by 1.5%.

1 Introduction

Knowledge graph embedding (KGE) [Wang *et al.*, 2017; Rossi *et al.*, 2021] aims to embed entities and relations in

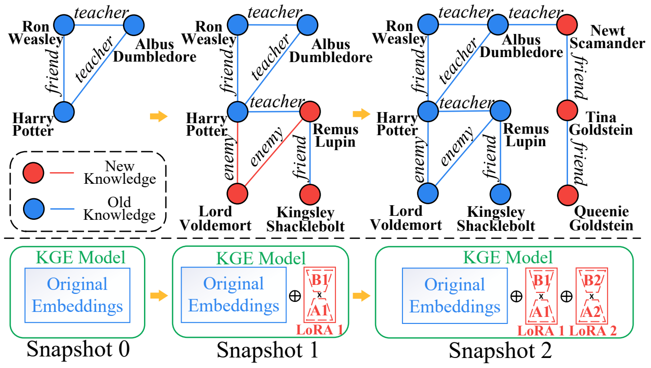

Figure 1: Illustration of IncLoRA for CKGE. The above is a growing KG about *Harry Potter* as the storyline unfolds, and below are the KGE models with incremental LoRAs in each snapshot.

knowledge graphs (KGs) [Dong *et al.*, 2014] into vectors, which is crucial for various downstream applications, such as question answering [Bordes *et al.*, 2014], recommendation systems [Zhang *et al.*, 2016], fact detecting [Shang *et al.*, 2024], and information extraction [Li *et al.*, 2022; Xu *et al.*, 2023; Li *et al.*, 2024]. Traditional KGE methods [Bordes *et al.*, 2013; Sun *et al.*, 2019; Liu *et al.*, 2020; Pan and Wang, 2021; Shang *et al.*, 2023; Liu *et al.*, 2023] primarily focus on dealing with static KGs. However, real-world KGs, such as Wikidata [Vrandečić and Krötzsch, 2014] and YAGO [Suchanek *et al.*, 2007], are dynamic and continuously evolving with emerging knowledge. For instance, Wikidata expanded from 16M entities to 46M between 2014 and 2018 [Wikidata, 2023]. A challenge with traditional KGE is that updating the embeddings of entities and relations requires retraining the entire KG, leading to heavy training costs in large-scale KGs. To address this issue, the continual knowledge graph embedding (CKGE) task has received growing attention to fine-tuning with only new knowledge [Song and Park, 2018; Daruna *et al.*, 2021].

The primary challenge in CKGE lies in alleviating catastrophic forgetting [Kirkpatrick *et al.*, 2017; Liu *et al.*, 2024] while simultaneously reducing training costs. One solution for CKGE, known as the full-parameter fine-tuning

∗Corresponding author.

paradigm, memorizes old knowledge by replaying a core old data set [Lopez-Paz and Ranzato, 2017; Wang *et al.*, 2019; Kou *et al.*, 2020] or introducing additional regularization constraints [Zenke *et al.*, 2017; Kirkpatrick *et al.*, 2017; Cui *et al.*, 2023]. Although this paradigm effectively mitigates catastrophic forgetting, it significantly increases training costs, especially when handling large-scale KGs. Another solution adopts the incremental-parameter fine-tuning paradigm, with only a few parameters to learn emerging knowledge [Rusu *et al.*, 2016; Lomonaco and Maltoni, 2017]. Despite eliminating explicit knowledge replay, the straightforward alignment of new and old parameter dimensions may still result in an unacceptable increase in parameters and training time. Meanwhile, in recent years, to efficiently reduce the training time of large language models (LLMs) [Radford *et al.*, 2019; Brown *et al.*, 2020] for downstream tasks, some work has employed low-rank adapters, such as LoRA [Hu *et al.*, 2022], to lower the parameter dimension and enable efficient parameter fine-tuning. Namely, LoRA freezes the pre-trained model weights and injects trainable rank decomposition matrices into original model architectures. In this paper, we try to innovatively adapt this mechanism to address continual learning problems, i.e., CKGE.

Inspired by low-rank adaptation [Hu *et al.*, 2022] in parameter fine-tuning for LLMs, we are among the first to store new knowledge in KGs via low-rank adapters (LoRAs) to reduce training costs. As shown in Figure 1, in contrast to the old entity *Ron Weasley*, new incremental LoRAs should be assigned to the new entity *Remus Lupin*. Moreover, recent work has discovered that new parameters of different layers should be adaptively assigned distinct dimensions [Ansuini *et al.*, 2019], which can also be applied to emerging knowledge with different structural features in CKGE. As depicted in Figure 1, *Remus Lupin* connects with more entities than *Lord Voldemort*. Consequently, it retains a broader influence, requiring more parameters for effective learning. In light of this consideration, we propose a fast CKGE framework (FastKGE), incorporating a novel incremental low-rank adapter (IncLoRA) mechanism, to learn new knowledge both effectively and efficiently. To alleviate catastrophic forgetting, FastKGE isolates new knowledge to specific layers. Concretely, FastKGE sorts and divides the new entity representations into explicit layers according to their distance from the old KG and the degree centrality. To reduce training costs, IncLoRA embeds the specific layers into incremental low-rank adapters with fewer training parameters. Specifically, the rank scales of each layer are allocated adaptively with the awareness of the KG structure.

To make up for the deficiency of small initial KG size in current CKGE datasets [Hamaguchi *et al.*, 2017; Kou *et al.*, 2020; Daruna *et al.*, 2021; Cui *et al.*, 2023], we construct two new CKGE datasets: FB-CKGE and WN-CKGE based on FB15K-237 [Dettmers *et al.*, 2018] and WN-18RR [Toutanova *et al.*, 2015]. FB-CKGE and WN-CKGE allocate 60% of total triples to generate the larger initial KG. Results on four traditional datasets show that FastKGE reduces training time by 34%-49% while still achieving competitive MRR scores in link prediction tasks against SOTAs (average in 21.0% vs. 21.1%). Meanwhile, on the new datasets FB-CKGE and WN-CKGE, FastKGE reduces 51%-68% training time while improving link prediction performance by 1.5% in MRR on average.

To sum up, the contributions of this paper are three-fold:

- To the best of our knowledge, we are among the first to introduce low-rank adapters to CKGE, namely, emerging knowledge can be stored in low-rank adapters to reduce training costs and preserve old knowledge well.
- We devise a fast CKGE framework (FastKGE), which isolates knowledge into specific layers to alleviate catastrophic forgetting and utilizes an incremental low-rank adapter (IncLoRA) mechanism to reduce training costs.
- Experimental results demonstrate that FastKGE significantly reduces training time with competitive performance in link prediction tasks compared with SOTAs. Additionally, two new open CKGE datasets with large-scale initial KGs are released.

2 Related Work

In contrast to traditional knowledge graph embedding (KGE) approaches [Bordes *et al.*, 2013; Trouillon *et al.*, 2016; Kazemi and Poole, 2018], continual knowledge graph embedding (CKGE) methods [Song and Park, 2018; Daruna *et al.*, 2021] enable KGE models to acquire new knowledge while retaining previously learned knowledge. Existing CKGE methods can be categorized into two main groups. First, full-parameter fine-tuning methods preserve learned knowledge by replaying old data [Lopez-Paz and Ranzato, 2017; Wang *et al.*, 2019; Kou *et al.*, 2020], or introducing constraints on weight updates in neural networks [Zenke *et al.*, 2017; Kirkpatrick *et al.*, 2017; Cui *et al.*, 2023]. Second, incremental-parameter fine-tuning methods [Rusu *et al.*, 2016; Lomonaco and Maltoni, 2017] adaptively adjust architectural properties to accommodate new information while preserving old parameters, thus facilitating continual learning. However, these methods only focus on preserving knowledge while ignoring training efficiency when KGs evolve.

In the field of large language models (LLMs), some work tries to utilize low-rank adapters (LoRAs) to fine-tune LLMs efficiently [Hu *et al.*, 2022; Zhang *et al.*, 2023]. Based on it, recent work tries to utilize LoRAs to separately store knowledge to alleviate catastrophic forgetting in continual learning [Wang *et al.*, 2023]. In the field of KGE, recent work tries to learn a small set of reserved entities to represent all entities for parameter-efficient learning [Chen *et al.*, 2023]. However, few works focus on efficient fine-tuning for CKGE.

3 Methodology

3.1 Preliminary and Problem Statement

Growing Knowledge Graph. A growing knowledge graph (KG) is denoted as snapshot sequences, i.e., $\mathcal{G} = \{\mathcal{S}_0, \mathcal{S}_1, ..., \mathcal{S}_n\}$. Snapshot $\mathcal{S}_i$ is a triplet $(\mathcal{E}_i, \mathcal{R}_i, \mathcal{T}_i)$, where $\mathcal{E}_i$, $\mathcal{R}_i$ and $\mathcal{T}_i$ denote the set of entities, relations, and triples at time i, respectively. Furthermore, we denote $\Delta\mathcal{T}_i = \mathcal{T}_i - \mathcal{T}_{i-1}$, $\Delta\mathcal{E}_i = \mathcal{E}_i - \mathcal{E}_{i-1}$ and $\Delta\mathcal{R}_i = \mathcal{R}_i - \mathcal{R}_{i-1}$ as new triples, entities, and relations, respectively.

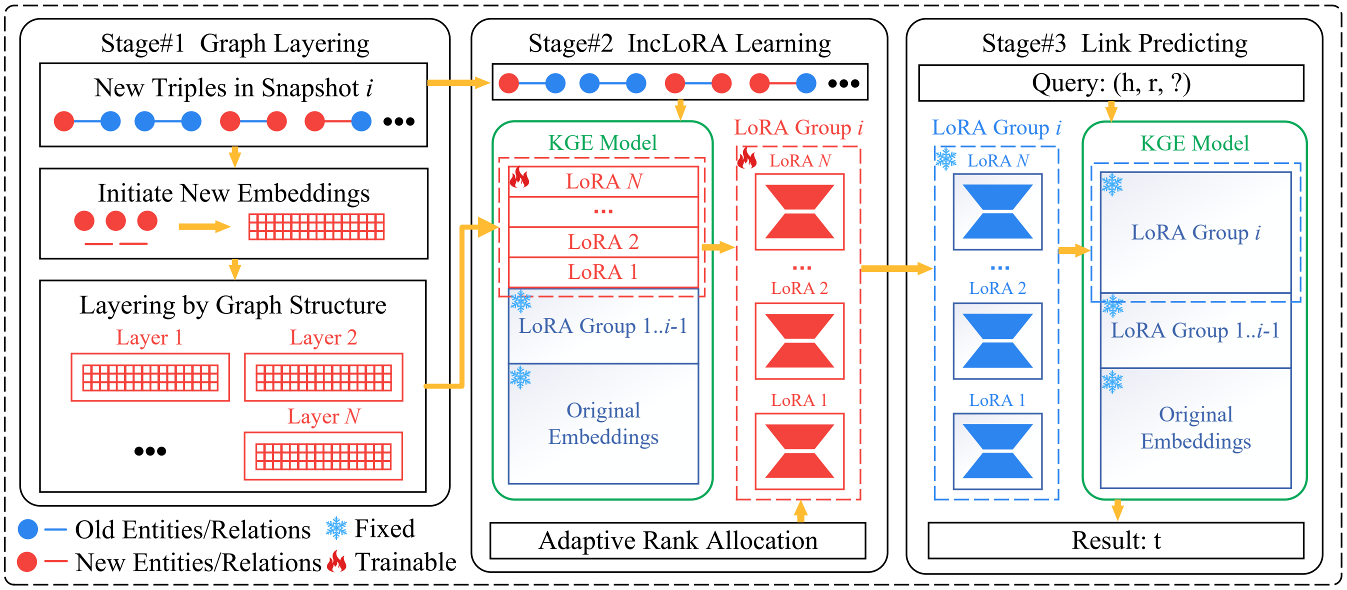

Figure 2: An overview of FastKGE framework. LoRA group i denotes the set for all LoRAs in a snapshot i.

Continual Knowledge Graph Embedding. Continual knowledge graph embedding (CKGE) aims to embed entities and relations into vectors in the growing KG $\mathcal{G} = \{\mathcal{S}_0, \mathcal{S}_1, ..., \mathcal{S}_n\}$. Specifically, when new triples $\Delta\mathcal{T}_i$ emerge in time i, CKGE learns the representations of new entities $\Delta\mathcal{E}_i$ and relations $\Delta\mathcal{R}_i$, and updates the representations of old entities $\mathcal{E}_{i-1}$ and relations $\mathcal{R}_{i-1}$ to adapt $\Delta\mathcal{T}_i$. Finally, all representations of entities $\mathcal{E}_i$ and relations $\mathcal{R}_i$ are obtained.

3.2 Framework

The framework of FastKGE is illustrated in Figure 2. Overall, as a KG grows in each snapshot, we utilize incremental low-rank adapters (LoRAs) for different KG layers to learn and preserve new entities and relations. First, during the knowledge graph layering stage, we divide new entities and relations into several layers based on their distances from the old graph and node degrees. Second, in the IncLoRA learning stage, embeddings of entities and relations in each layer are represented by incremental LoRAs with adaptive rank allocation. Finally, in the link predicting stage, we compose all new LoRAs into a LoRA group and concat all LoRA groups and initial embeddings for inference.

3.3 Graph Layering

In order to achieve separate storage of KGs and assign different ranks to LoRAs of different layers, we divide new knowledge into several layers according to a graph structure. For new triples $\Delta\mathcal{T}_i$ emerging in a new snapshot $i > 0$, we firstly get new entities $\Delta\mathcal{E}_i$ and relations $\Delta\mathcal{R}_i$ and initiate embeddings of them as shown in Stage 1 of Figure 2. Then, to sort and divide embeddings of $\Delta\mathcal{E}_i$ in snapshot i, we calculate the importance of $\Delta\mathcal{E}_i$ by the distance from the old graph and degree centrality. Specifically, we use the breadth-first search (BFS) algorithm to progressively expand $\Delta\mathcal{E}_i$ in snapshot i from S_{i-1}. Then, we get the sorted entity sequence $\mathbf{s}_{entity}$ as follows:

$$\mathbf{s}_{entity} = [e_1, e_2, ..., e_{|\Delta\mathcal{E}_i|}] \tag{1}$$

where for $e_j, e_k \in \mathbf{s}_{entity}$, if $j \leq k$, the distance of e_j from the old graph is closer than e_k. To further sort entities with the same distance, we denote $f_{dc}(e)$ as the degree centrality of e in the new graph composed of new triples $\Delta\mathcal{T}_i$ as follows:

$$f_{dc}(e) = \frac{f_{neighbor}(e)}{|\Delta\mathcal{E}_i| - 1} \tag{2}$$

where $f_{neighbor}(e)$ denotes the number of the neighbors of e in $\Delta\mathcal{T}_i$. For entities with the same distance from the old graph in $\mathbf{s}_{entity}$, we use f_{dc} for further sorting. Then, we divide the entities in $\mathbf{s}_{entitiy}$, previously sorted with importance, equally into N distinct layers $\overline{E} = \{E_1, E_2, ..., E_N\}$, where E_k denotes the k-th layer of entities, and N is a hyper-parameter. For relations, we only put all new relations into a layer $\overline{R}$ rather than layering. That is because the number of entities is increasing more significantly than relations in the dynamic evolution of KGs, and the number of total embedding parameters linearly to the number of entities in practice [Chen *et al.*, 2023]. Therefore, we focus on storage and training optimization for new entities. Finally, we get the layering entities and relations $\overline{E}$ and $\overline{R}$.

3.4 IncLoRA Learning

Incremental Low-Rank Decomposition

To accelerate learning, we propose an incremental low-rank adapter learning mechanism (IncLoRA) to reduce training costs. Specifically, we obtain entity and relation layers $\overline{E}$ and $\overline{R}$ from graph layering stage. Then, for each layer in $\overline{E}$ and $\overline{R}$, we use an incremental low-rank adapter to learn and store knowledge. Take $\overline{E}$ as example, the embeddings of k-th layer E_k in $\overline{E}$ can be denoted as $\mathbf{E}_k$. For learning the embeddings $\mathbf{E}_k \in \mathbb{R}^{n\times d}$, we learn matrices $\mathbf{A}_k \in \mathbb{R}^{n\times r}$ and $\mathbf{B}_k \in \mathbb{R}^{r\times d}$, making it meet the following condition:

$$\mathbf{E}_k = \mathbf{A}_k\mathbf{B}_k \quad (3)$$

where n denotes the number of entities in a layer E_k, d stands for embedding dimension and r refers to the rank of $(\mathbf{A}_k, \mathbf{B}_k)$. We denote $(\mathbf{A}_k, \mathbf{B}_k)$ as an incremental LoRA. To make sure that parameters of low-rank learning is fewer than normal training for faster learning, the decomposition should meet the following condition:

$$r \leq \frac{n \times d}{n + d} \quad (4)$$

By this way, we compose $\mathbf{E}_k$ to an incremental LoRA $(\mathbf{A}_k, \mathbf{B}_k)$. Finally, we get a LoRA group $\mathbf{G}_i^{\mathbf{E}}$ for all embeddings of $\Delta\mathcal{E}_i$ as follows:

$$\mathbf{G}_i^{\mathbf{E}} = \text{concat}(\{(\mathbf{A}_k \times \mathbf{B}_k)|1 \leq k \leq N\}) \quad (5)$$

where concat$(\cdot)$ denotes the concatenation of several matrices. We can also get the LoRA group $\mathbf{G}_i^{\mathbf{R}}$ for new relations by the same decomposition.

Adaptive Rank Allocation

In order to preserve more information for more important entities, we utilize an adaptive rank allocation strategy to assign different ranks to LoRAs in different layers. Specifically, instead of assigning a fixed base rank r_{base} to all incremental LoRAs in $\mathbf{G}_i^{\mathbf{E}}$, we assign more ranks to more important low-rank LoRAs with higher f_{dc}. Firstly, we denote the total sum of degree centrality in the k-th layer Sum_k and the average degree centrality Avg_{dc} in all layers as follows:

$$Sum_k = \sum_{e \in E_k} f_{dc}(e) \quad (6)$$

$$Avg_{dc} = \frac{\sum_{k \in N} Sum_k}{N} \quad (7)$$

Then, the r_k of the k-th LoRA in $\mathbf{G}_i^{\mathbf{E}}$ is denoted as follows:

$$r_k = \frac{r_{base} \cdot Sum_k}{Avg_{dc}} \quad (8)$$

Finally, the rank of all adapters in $\mathbf{G}_i^{\mathbf{E}}$ is determined. Since there is only one layer in $\overline{R}$, we do not utilize adaptive rank allocation for $\mathbf{G}_i^{\mathbf{R}}$.

IncLoRA Training

From above, we obtain the LoRA group $\mathbf{G}_i^{E}$ and $\mathbf{G}_i^{R}$, and adaptively determine the rank of each LoRA. Then, the embeddings of all entities $\mathbf{E}_{all}$ and all relations $\mathbf{R}_{all}$ can be denoted as follows:

$$\mathbf{E}_{all} = \text{concat}(\mathbf{E}_{origin}, \{\mathbf{G}_j^{\mathbf{E}}|1 \leq j \leq i\}) \quad (9)$$

$$\mathbf{R}_{all} = \text{concat}(\mathbf{R}_{origin}, \{\mathbf{G}_j^{\mathbf{R}}|1 \leq j \leq i\}) \quad (10)$$

where $\mathbf{E}_{origin}$ and $\mathbf{R}_{origin}$ denote the origin embeddings of entities and relations in snapshot 0, respectively. Finally, we train all LoRAs in $\mathbf{G}_i^{\mathbf{E}}$ and $\mathbf{G}_i^{\mathbf{R}}$ with new triples $\Delta\mathcal{T}$. We take TransE [Bordes *et al.*, 2013] as the base KGE model, and the loss can be denoted as follows:

$$\mathcal{L} = \sum_{(h,r,t) \in \Delta\mathcal{T}_i} max(0, f(h,r,t) - f(h',r,t') + \gamma) \quad (11)$$

where (h', r, t') is the negative triple of $(h, r, t) \in \Delta\mathcal{T}_i$, and $f(h, r, t) = |\mathbf{h} + \mathbf{r} - \mathbf{t}|_{L1/L2}$ is the score function of TransE. $\mathbf{h} \in \mathbf{E}_{all}$, $\mathbf{r} \in \mathbf{R}_{all}$, and $\mathbf{t} \in \mathbf{E}_{all}$ denote embeddings of h, r, and t, respectively. We only train the parameters in $\mathbf{G}_i^{\mathbf{E}}$ and $\mathbf{G}_i^{\mathbf{R}}$, and fix parameters in all other LoRA groups and origin embeddings. Finally, all representations of entities and relations, i.e., $\mathbf{E}_{all}$ and $\mathbf{R}_{all}$ are obtained.

3.5 Link Predicting

In the stages of link predicting, we compose all LoRA groups as Equation 9 and 10. Taking link prediction as an example, we freeze all parameters of LoRA groups and initial embeddings. For a given query $(h, r, ?)$, we calculate the tail entity t that gives the highest triple score of TransE as the prediction result. Notably, FastKGE only involves assembling LoRAs into a comprehensive embedding module before the inference stage without requiring additional operations, resulting in no additional time consumption in inference.

4 Experiments

4.1 Experimental Setup

Datasets

We use six datasets in the main experiments: ENTITY, RELATION, FACT, HYBRID, FB-CKGE, and WN-CKGE. ENTITY, RELATION, FACT, and HYBRID are traditional datasets for CKGE [Cui *et al.*, 2023], in which the number of entities, relations, triples and mix of them are growing equally in each snapshot, respectively. However, real-world KGs are typically built on a substantial foundational graph, to which a small increment of new knowledge merges in each snapshot. To make up for the deficiency of small initial KGs in current CKGE datasets [Hamaguchi *et al.*, 2017; Kou *et al.*, 2020; Daruna *et al.*, 2021; Cui *et al.*, 2023], we construct two new datasets for CKGE: FB-CKGE and WN-CKGE, which are based on two widely-used KGE datasets FB15K-237 [Dettmers *et al.*, 2018] and WN18RR [Toutanova *et al.*, 2015]. In FB-CKGE and WN-CKGE, we assign 60% of the total triples to the initial snapshot, and 10% of the total triples to each next snapshot. Compared to traditional datasets ENTITY, RELATION, FACT, and HYBRID based on FB15K-237, FB-CKGE has 2 to 4 times triples in base KGs in snapshot 0. We set the snapshots for all datasets to 5. The ratio of train, valid, and test sets for all datasets is 3:1:1. The details of the datasets are shown in Table 1. The datasets are available at https://github.com/seukgcode/FastKGE.

Baselines

We choose two standard baselines: incremental-parameter fine-tuning methods and full-parameter fine-tuning methods. For incremental-parameter fine-tuning methods, we choose PNN [Rusu *et al.*, 2016] and CWR [Lomonaco and Maltoni, 2017]. For full-parameter fine-tuning methods, we choose replay-based methods GEM [Lopez-Paz and Ranzato, 2017], EMR [Wang *et al.*, 2019], DiCGRL [Kou *et al.*, 2020], and regularization-based methods SI [Zenke *et al.*, 2017], EWC [Kirkpatrick *et al.*, 2017], LKGE [Cui *et al.*, 2023].

	Snapshot 0			Snapshot 1			Snapshot 2			Snapshot 3			Snapshot 4		
Dataset	N_E	N_R	N_T	N_E	N_R	N_T	N_E	N_R	N_T	N_E	N_R	N_T	N_E	N_R	N_T
ENTITY	2,909	233	46,388	5,817	236	72,111	8,275	236	73,785	11633	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
FB-CKGE	7505	237	186,070	11,258	237	31,012	13,134	237	31,012	14,072	237	31012	14,541	237	31010
WN-CKGE	24,567	11	55,801	28,660	11	9,300	32,754	11	9,300	36,848	11	9,300	40,943	11	9,302

Table 1: The statistics of datasets. N_E, N_R and N_T denote the number of cumulative entities, cumulative relations and current triples in each snapshot i.

Metrics

We assess the performance of our model in the link prediction task. Specifically, we substitute the head or tail entity of the triples in the test set with all other entities and then calculate and rank the scores for each triple. Subsequently, we measure the Mean Reciprocal Rank (MRR), Hits@1, Hits@3, and Hits@10 as evaluation metrics. The higher MRR, Hits@1, Hits@3, and Hits@10 indicate better model performance. For each snapshot i, we compute the average of the above metrics evaluated on all the test sets of current and previous snapshots as the final metric. The main results are derived from the model generated at the final time. In addition, we calculate the total training time of all snapshots to evaluate the time efficiency.

Settings

All experiments are conducted on the NVIDIA RTX 3090Ti GPU. The codes of the experiments are supported by PyTorch [Paszke *et al.*, 2019]. The number of snapshots is set to 5. We choose the batch size from [258, 512, 1024] and the learning rate from [1e-1, 2e-1, 3e-1]. We choose the Adam as the optimizer. In our experiments, we set the entity base rank of LoRA from the range [10, 50, 100, 150, 200] and the relation base rank to 20. Also, we set the number of LoRA layers N from the range [2, 5, 10, 20]. We set the embedding size for all methods to 200. To fairness, all the results are from the average of random five running times, and the patience of early stopping is 3 for all methods to compare time efficiency.

4.2 Results

Main Results

The main results are shown in Table 2 and Table 3. Overall, our method FastKGE achieves competitive performance compared to the state-of-the-art methods on all datasets. Furthermore, it outperforms all other baselines regarding time efficiency, highlighting its superior training speed.

Firstly, FastKGE outperforms all other baselines across all datasets regarding training time efficiency. Specifically, on the four traditional datasets ENTITY, RELATION, FACT, and HYBRID, FastKGE can save 34%-49% training time compared to the fastest baselines. Notably, on the two newly constructed datasets FB-CKGE and WN-CKGE, FastKGE can save 51%-68% training time compared to the fastest baselines. This demonstrates that our method with low-rank adapters will be more efficient in larger initial graphs.

Secondly, FastKGE has strong competitiveness in performance compared to the best baselines. Specifically, FastKGE achieves the best performance on two traditional datasets (ENTITY and HYBRID), and two newly constructed datasets (FB-CKGE, and WN-CKGE). Compared to the best baselines, FastKGE achieves 0.4%-1.5%, 0.7%-1.8%, 0.2%-2.8%, and 0.2%-0.9% higher in MRR, Hits@1, Hits@3, and Hits@10, respectively. Notably, on two newly constructed datasets (FB-CKGE and WN-CKGE) with extensive initial triples, FastKGE can significantly improve performance by 1.5%, 1.3%, 2.4%, and 0.6% in MRR, Hits@1, Hits@3, and Hits@10 on average compared to the best methods, respectively. This proves that FastKGE performs better on KGs with larger initial scales, which is more in line with the changes in real life. Besides, FastKGE also demonstrates strong competitiveness on two other traditional datasets (RELATION and FACT), with only 0.7%, 0.2%, 0.6%, and 0.5% decrease compared to the optimal baselines in MRR, Hits@1, Hits@3, and Hits@10 on average, respectively. This performance decline on RELATION and FACT can be attributed to the limited number of entities they contain. Since our method addresses this issue by employing low-rank decomposition and knowledge storage for representing new entities. Therefore, the small changes in entity number on RELATION and FACT pose a challenge in showcasing the benefits of our method.

Ablation Results

This section investigates the effectiveness of incremental low-rank adapter (IncLoRA) learning and graph layering strategy (GL). The results are shown in Table 4 and Table 5. Firstly, we find that if we remove IncLoRA and fine-tune the KGE model directly, the model performance will significantly decrease by 2.8%-9.1%, 1.3%-6.8%, 3.3%-9.4%, and 2.5%-13.8% in MRR, Hits@1, Hits@3, and Hits@10 on all datasets, respectively. This proves that IncLoRA can effectively preserve learned knowledge with incremental low-rank adapters. Meanwhile, with IncLoRA the training time will decrease obviously by 36%-65% on all datasets. This proves that using IncLoRA can significantly improve training efficiency with low-rank composition compared to direct fine-tuning. Secondly, if we remove GL and use one LoRA for all entities, the performance will decrease slightly by 0.4%-3.1% on all datasets. This proves that our multi-layer LoRAs are more effective than one LoRA with adaptive rank allocation. Also, the training time will decrease by 3%-18%, indicating that training and assembling multiple LoRAs require additional time. Notably, the additional training time only makes the 3%-9% time of the total training time on FB-CKGE and WN-CKGE with more initial triples, where the

	ENTITY					RELATION					FACT				
Method	MRR	H@1	H@3	H@10	T(s)	MRR	H@1	H@3	H@10	T(s)	MRR	H@1	H@3	H@10	T(s)
PNN	0.229	0.130	0.265	0.425	1,979	0.167	0.096	0.191	0.305	2,186	0.157	0.084	0.188	0.290	1,511
CWR	0.088	0.028	0.114	0.202	2,218	0.021	0.010	0.024	0.043	1,828	0.083	0.030	0.095	0.192	2,807
GEM	0.165	0.085	0.188	0.321	1,990	0.093	0.040	0.106	0.196	1,410	0.175	0.092	0.196	0.345	1,154
EMR	0.171	0.090	0.195	0.330	3,341	0.111	0.052	0.126	0.225	2,696	0.171	0.090	0.191	0.337	1,619
DiCGRL	0.107	0.057	0.110	0.211	2,416	0.133	0.079	0.147	0.241	2,481	0.162	0.084	0.189	0.320	2,204
SI	0.154	0.072	0.179	0.311	1,443	0.113	0.055	0.131	0.224	1,540	0.172	0.088	0.194	0.343	1,165
EWC	0.229	0.130	0.264	0.423	2,119	0.165	0.093	0.190	0.306	1,957	0.201	0.113	0.229	0.382	1,168
LKGE	0.234	0.136	0.269	0.425	1,515	**0.192**	0.106	**0.219**	**0.366**	1,242	**0.210**	**0.122**	**0.238**	**0.387**	958
FastKGE	**0.239**	**0.146**	**0.271**	**0.427**	**918**	0.185	**0.107**	0.213	0.359	**634**	0.203	0.117	0.231	0.384	**630**

Table 2: Main experimental results on ENTITY, RELATION and FACT. T denotes the total training time for all snapshots. The bold scores indicate the best results and underlined scores indicate the second best results.

	HYBRID					FB-CKGE					WN-CKGE				
Method	MRR	H@1	H@3	H@10	T(s)	MRR	H@1	H@3	H@10	T(s)	MRR	H@1	H@3	H@10	T(s)
PNN	0.185	0.101	0.216	0.349	2,098	0.215	0.122	0.245	0.403	1,209	0.134	0.002	0.241	0.342	1,291
CWR	0.037	0.015	0.044	0.077	2,030	0.075	0.011	0.105	0.192	2,046	0.005	0.002	0.007	0.012	1,185
GEM	0.136	0.070	0.152	0.263	2,022	0.188	0.103	0.212	0.359	1,031	0.119	0.002	0.215	0.297	951
EMR	0.141	0.073	0.157	0.267	2,976	0.180	0.097	0.204	0.346	1,500	0.114	0.002	0.205	0.286	1,153
DiCGRL	0.149	0.083	0.168	0.277	2,231	0.149	0.091	0.160	0.261	1,742	0.057	0.001	0.155	0.166	1,012
SI	0.111	0.049	0.126	0.229	1,300	0.187	0.102	0.211	0.359	1,088	0.115	0.001	0.209	0.289	1,077
EWC	0.186	0.102	0.214	0.350	1,535	0.218	0.124	0.247	0.410	1,196	0.136	0.003	0.248	0.338	1,013
LKGE	0.207	0.121	0.235	0.379	1,083	0.208	0.113	0.238	0.403	819	0.144	0.007	0.259	0.347	1,073
FastKGE	**0.211**	**0.128**	**0.241**	**0.382**	**700**	**0.223**	**0.131**	**0.257**	**0.405**	**405**	**0.159**	**0.015**	**0.287**	**0.356**	**342**

Table 3: Main experimental results on HYBRID, FB-CKGE and WN-CKGE.

	ENTITY					RELATION					FACT				
Method	MRR	H@1	H@3	H@10	T(s)	MRR	H@1	H@3	H@10	T(s)	MRR	H@1	H@3	H@10	T(s)
FastKGE	0.239	0.146	0.271	0.427	918	0.185	0.107	0.213	0.359	634	0.203	0.117	0.231	0.384	630
w/o IncLoRA	0.166	0.086	0.187	0.323	1,918	0.094	0.039	0.109	0.201	1,208	0.175	0.091	0.198	0.345	979
w/o GL	0.235	0.144	0.267	0.411	755	0.168	0.099	0.196	0.295	541	0.172	0.107	0.196	0.292	578

Table 4: Ablation results on ENTITY, RELATION and FACT. GL denotes graph layering.

	HYBRID					FB-CKGE					WN-CKGE				
Method	MRR	H@1	H@3	H@10	T(s)	MRR	H@1	H@3	H@10	T(s)	MRR	H@1	H@3	H@10	T(s)
FastKGE	0.211	0.128	0.241	0.382	700	0.223	0.131	0.257	0.405	405	0.159	0.015	0.287	0.356	342
w/o IncLoRA	0.139	0.072	0.155	0.267	1,828	0.185	0.101	0.210	0.353	1,168	0.119	0.002	0.215	0.298	959
w/o GL	0.198	0.118	0.231	0.349	674	0.218	0.126	0.252	0.308	367	0.152	0.012	0.281	0.349	333

Table 5: Ablation results on HYBRID, FB-CKGE and WN-CKGE.

additional time is much smaller than the whole training time.

Effectiveness of Base Ranks

This section investigates the effectiveness of different base ranks on the model performance and newly added parameters. Since the number of new entities is much more significant than new relations, the number of total embedding parameters is linear to the number of entities. Therefore, we only analyze the rank r_{base} for entities. We conduct our experiments on ENTITY, HYBRID, FB-CKGE, and WN-CKGE. The results are shown in Figure 3. Firstly, we find a significant increase in model performance from 1.5% to 5.2% in MRR when the rank changes from 10 to 150 in all datasets. However, when the rank size exceeds 150, there is only a slight improvement from 0.1% to 0.2% in MRR on ENTITY, FB-CKGE, and WN-CKGE. Notably, on HYBRID, there is a slight decrease of 0.1% in MRR when the rank changes from 150 to 200. This demonstrates that when the base rank is small, the larger the base rank, the better the performance. When the size of the base rank reaches an upper limit, the model performance hardly improves. Secondly, we find that the number of newly added parameters in the model increases continuously with the increase of rank. Therefore, the model performance does not continually improve with the increase of parameter quantity, but there is a specific upper limit. Combining Equation 4, we find that when the number of new entities is much

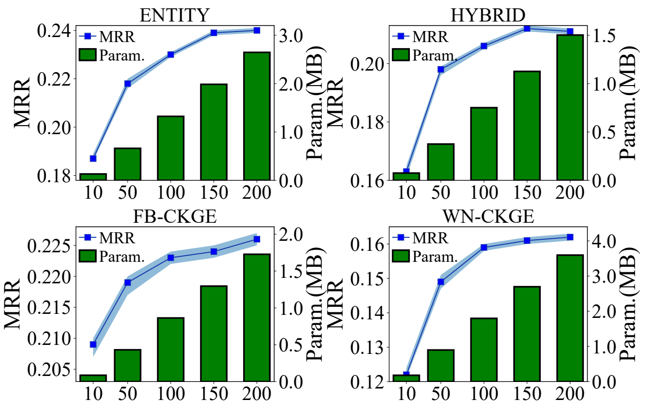

Figure 3: Effectiveness of different base ranks from 10 to 200.

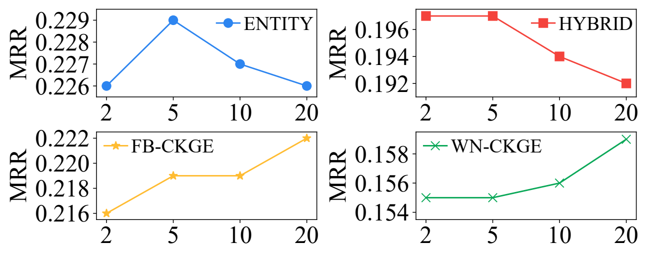

Figure 4: Effectiveness of different layer numbers. The horizontal axis represents different layer numbers from 2 to 20.

greater than the dimension, the upper-rank limit is approximately equal to the dimension, which is consistent with the upper limit of the rank 200 shown in the experimental results.

Effectiveness of Layer Numbers

In this section, we investigate the effectiveness of different layer numbers N on model performance. We conduct experiments on two traditional datasets (ENTITY and HYBRID) and two new datasets (FB-CKGE and WN-CKGE), as shown in Figure 4. Firstly, on ENTITY and HYBRID, the model performance peaks when layer number equals five and then decreases by 0.4%-0.5% in MRR from 5 to 20 layers. This indicates that the layer should be set to a small number for the datasets with small initial graphs. Secondly, we observe that on FB-CKGE and WN-CKGE, the model performance increases by 0.4%-0.6% in MRR when the layer number ranges from 2 to 20. It proves that graph layering can achieve more significant results with larger initial KGs.

Different Base Models

In this paper, although we choose the most typical model, TransE, as the base KGE model, our method can still be extended easily to other base KGE models. To verify the scalability of our method FastKGE, we extend FastKGE to two other different types of KGE models, a standard bilinear-based model ComplEx [Trouillon *et al.*, 2016] and a typical roto-translation-based Model RotatE [Sun *et al.*, 2019]. We conduct the experiments on FB-CKGE, and the results are shown in Figure 5. Firstly, we observe that FastKGE outperforms direct fine-tuning by 2.6%-3.1% in MRR on ComplEx and RotatE, respectively. Meanwhile, for KGE models with better performance, applying our method will also lead to better performance. This indicates that FastKGE can effectively store the learned knowledge for different KGEs, alleviating catastrophic forgetting in continual learning. Secondly, our method can significantly shorten the time from 61.3% to 70.1% for different KGEs. This demonstrates that FastKGE has robust scalability for different KGE models to accelerate the training process.

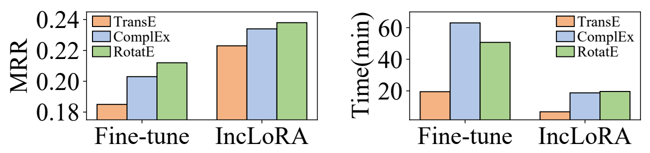

Figure 5: Effectiveness of different base KGE models in FB-CKGE.

New entity: *Sedona*	New entity: *Robert Morse*
(f_{dc}=4.7e-4, dis=1)	(f_{dc}=1.18e-3, dis=1)
Queries	
(Michelle Branch, location, ?)	(Mad Man, award_winner, ?)
Prediction Results of FastKGE	
$1st$: *Sedona* $2nd$: Orlando $3rd$: Detroit	$1st$: *Robert Morse* $2nd$: Bryan Batt $3rd$: Jared Harris
Prediction Results of FastKGE without graph layering	
$1st$: *Sedona* $2nd$: Wiltshire $3rd$: Cornwall	$1st$: Maria Jacquemetton $2nd$: Mathew Weiner $3rd$: *Robert Morse*

Table 6: Case study for the graph layering strategy.

4.3 Case Study

We conduct a case study to verify that FastKGE learns more critical entities well by graph layering with dynamic rank allocation. As shown in Table 6, we select two new entities with different importance, namely, with the same distance from the old graph (dis=1) but different f_{dc}, and test the prediction results with layering and non-layering strategies. Results show that for entity *Sedona* with small $f_{dc} = 4.7e - 4$, both strategies can rank it first. However, for entity *Robert Morse* with larger $f_{dc} = 1.18e - 3$, the rank of the correct answer will increase after layering. It proves that graph layering learns critical new entities effectively without affecting others.

5 Conclusion

This paper proposes a novel fast learning framework for CKGE, FastKGE, which utilizes an IncLoRA mechanism to preserve learned knowledge well and accelerate fine-tuning. Firstly, to alleviate catastrophic forgetting, we conduct graph layering for new knowledge to achieve separate storage of old and new knowledge. Moreover, to reduce training costs, we propose incremental low-rank adapter learning to learn new knowledge efficiently with adaptive rank allocation. In the future, we will explore how to conduct CKGE learning when the knowledge of growing KGs is forgotten or modified.

Acknowledgments

We thank the reviewers for their insightful comments. This work was supported by National Science Foundation of China (Grant Nos.62376057) and the Start-up Research Fund of Southeast University (RF1028623234). All opinions are of the authors and do not reflect the view of sponsors.

References

[Ansuini *et al.*, 2019] Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. In *NeurIPS*, 2019.

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

[Bordes *et al.*, 2014] Antoine Bordes, Jason Weston, and Nicolas Usunier. Open question answering with weakly supervised embedding models. In *ECML-PKDD*, 2014.

[Brown *et al.*, 2020] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurlPS*, 2020.

[Chen *et al.*, 2023] Mingyang Chen, Wen Zhang, Zhen Yao, Yushan Zhu, Yang Gao, Jeff Z Pan, and Huajun Chen. Entity-agnostic representation learning for parameter-efficient knowledge graph embedding. In *AAAI*, 2023.

[Cui *et al.*, 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 *AAAI*, 2023.

[Daruna *et al.*, 2021] 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.

[Dettmers *et al.*, 2018] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI*, 2018.

[Dong *et al.*, 2014] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *SIGKDD*, 2014.

[Hamaguchi *et al.*, 2017] Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. Knowledge transfer for out-of-knowledge-base entities: a graph neural network approach. In *IJCAI*, 2017.

[Hu *et al.*, 2022] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

[Kazemi and Poole, 2018] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In *NeurIPS*, 2018.

[Kirkpatrick *et al.*, 2017] 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.

[Kou *et al.*, 2020] Xiaoyu Kou, Yankai Lin, Shaobo Liu, Peng Li, Jie Zhou, and Yan Zhang. Disentangle-based continual graph representation learning. In *EMNLP*, 2020.

[Li *et al.*, 2022] Guozheng Li, Xu Chen, Peng Wang, Jiafeng Xie, and Qiqing Luo. Fastre: Towards fast relation extraction with convolutional encoder and improved cascade binary tagging framework. In *IJCAI*, 2022.

[Li *et al.*, 2024] Guozheng Li, Wenjun Ke, Peng Wang, Zijie Xu, Ke Ji, Jiajun Liu, Ziyu Shang, and Qiqing Luo. Unlocking instructive in-context learning with tabular prompting for relational triple extraction. *arXiv preprint arXiv:2402.13741*, 2024.

[Liu *et al.*, 2020] Yuzhang Liu, Peng Wang, Yingtai Li, Yizhan Shao, and Zhongkai Xu. Aprile: Attention with pseudo residual connection for knowledge graph embedding. In *COLING*, 2020.

[Liu *et al.*, 2023] Jiajun Liu, Peng Wang, Ziyu Shang, and Chenxiao Wu. Iterde: an iterative knowledge distillation framework for knowledge graph embeddings. In *AAAI*, 2023.

[Liu *et al.*, 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 *AAAI*, 2024.

[Lomonaco and Maltoni, 2017] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *CoRL*, 2017.

[Lopez-Paz and Ranzato, 2017] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *NIPS*, 2017.

[Pan and Wang, 2021] Zhe Pan and Peng Wang. Hyperbolic hierarchy-aware knowledge graph embedding for link prediction. In *EMNLP*, 2021.

[Paszke *et al.*, 2019] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, and et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[Radford *et al.*, 2019] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[Rossi *et al.*, 2021] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. Knowledge graph embedding for link prediction: A comparative analysis. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(2):1–49, 2021.

[Rusu *et al.*, 2016] 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.

[Shang *et al.*, 2023] Ziyu Shang, Peng Wang, Yuzhang Liu, Jiajun Liu, and Wenjun Ke. Askrl: An aligned-spatial knowledge representation learning framework for open-world knowledge graph. In *ISWC*, 2023.

[Shang *et al.*, 2024] Ziyu Shang, Wenjun Ke, Nana Xiu, Peng Wang, Jiajun Liu, Yanhui Li, Zhizhao Luo, and Ke Ji. Ontofact: Unveiling fantastic fact-skeleton of llms via ontology-driven reinforcement learning. In *AAAI*, 2024.

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

[Suchanek *et al.*, 2007] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *WWW*, 2007.

[Sun *et al.*, 2019] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*, 2019.

[Toutanova *et al.*, 2015] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, 2015.

[Trouillon *et al.*, 2016] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *ICML*, 2016.

[Vrandečić and Krötzsch, 2014] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[Wang *et al.*, 2017] 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.

[Wang *et al.*, 2019] Hong Wang, Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Shiyu Chang, and William Yang Wang. Sentence embedding alignment for lifelong relation extraction. In *NAACL*, 2019.

[Wang *et al.*, 2023] Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. *arXiv preprint arXiv:2310.14152*, 2023.

[Wikidata, 2023] Wikidata. Welcome to wikidata. https://www.wikidata.org/wiki/Wikidata:Main_Page/, 2023. Accessed: 2023-10-01.

[Xu *et al.*, 2023] Zijie Xu, Peng Wang, Ziyu Shang, and Jiajun Liu. A two-stage label rectification framework for noisy event extraction. In *DASFAA*, 2023.

[Zenke *et al.*, 2017] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017.

[Zhang *et al.*, 2016] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *SIGKDD*, 2016.

[Zhang *et al.*, 2023] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *ICLR*, 2023.