IterDE: An Iterative Knowledge Distillation Framework for Knowledge Graph Embeddings

Jiajun Liu, Peng Wang*, Ziyu Shang, Chenxiao Wu

School of Computer Science and Engineering, Southeast University
{jiajliu, pwang, ziyus1999, chenxiaowu}@seu.edu.cn

Abstract

Knowledge distillation for knowledge graph embedding (KGE) aims to reduce the KGE model size to address the challenges of storage limitations and knowledge reasoning efficiency. However, current work still suffers from performance drops when compressing a high-dimensional original KGE model to a low-dimensional distillation KGE model. Moreover, most work focuses on the reduction of inference time but ignores the time-consuming training process of distilling KGE models. In this paper, we propose IterDE, a novel knowledge distillation framework for KGEs. First, IterDE introduces an iterative distillation way and enables a KGE model to alternately be a student model and a teacher model during the iterative distillation process. Consequently, knowledge can be transferred in a smooth manner between high-dimensional teacher models and low-dimensional student models, while preserving good KGE performances. Furthermore, in order to optimize the training process, we consider that different optimization objects between hard label loss and soft label loss can affect the efficiency of training, and then we propose a soft-label weighting dynamic adjustment mechanism that can balance the inconsistency of optimization direction between hard and soft label loss by gradually increasing the weighting of soft label loss. Our experimental results demonstrate that IterDE achieves a new state-of-the-art distillation performance for KGEs compared to strong baselines on the link prediction task. Significantly, IterDE can reduce the training time by 50% on average. Finally, more exploratory experiments show that the soft-label weighting dynamic adjustment mechanism and more fine-grained iterations can improve distillation performance.

Introduction

Knowledge graphs (KGs) describe concepts and facts in graph models (Dong et al. 2014), where knowledge is stored as triples. With the increasing sizes of KGs such as Wikipedia (Bizer et al. 2009) and Yago (Suchanek, Kasneci, and Weikum 2007), efficient knowledge graph embedding (KGE), which embeds triples into a continuous vector space, plays a pivotal role in downstream applications such as question answering (Bordes, Weston, and Usunier 2014), recommendation system (Zhang et al. 2016) and knowledge graph completion (Lin et al. 2015). Most KGE models such as TransE (Bordes et al. 2013), ComplEx (Trouillon et al. 2016), SimplE (Kazemi and Poole 2018), RotatE (Sun et al. 2019) have shown better performances with higher embedding dimensions and larger model sizes, however, that also leads to slower inference efficiency for practical applications. Specifically, the 512-dimensional KGE models have 7-15 times more embedding layer parameters and 2-6 times more inference time than the 32-dimensional KGE models (Zhu et al. 2022). Therefore, it is a nontrivial problem to compress KGEs from high-dimensional teacher models to low-dimensional student models while maintaining excellent performance. In realistic applications, KGE models are often required to simultaneously keep high performance and fast inference speed. For example, financial investors need to get accurate and fast market decision aids from financial KGs via edge devices. In this scenario, KGE models can be compressed by knowledge distillation and then deployed to edge devices to help financial investors make faster and more accurate decisions.

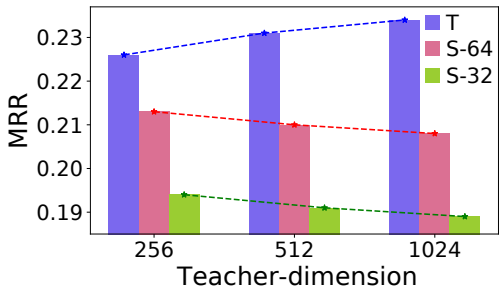

Figure 1: The phenomenon of *good teachers cannot always teach good students in KGEs based on KD*: with the increase of dimension (from 256 to 1024) of teacher T, the performances of students (S-64 denotes 64 dimensions and S-32 denotes 32 dimensions) drop. Results are obtained on WN18RR dataset with TransE.

Knowledge distillation (KD) is a popular technique for model compression, where a larger model serves as a teacher model, and a smaller model as a student model tries to mimic the output of the teacher model (Hinton et al. 2014). Recently, although several compression methods for KGEs

*Corresponding author.

based on KD have been proposed (Wang et al. 2021; Zhu et al. 2022), they still suffer from three drawbacks. First, existing work distills the high-dimensional teacher KGEs directly to the low-dimensional student KGEs, but better teachers cannot always teach better students. Clear evidence has shown that the gap in model size between a teacher and a student can affect the distillation performance (Mirzadeh et al. 2020). A similar phenomenon also holds for the KGEs based on KD, which is manifested by the gap between the embedded dimensions. More specifically, when distilling a high-dimensional teacher KGEs directly to a low-dimensional student KGEs, the dimension gap could cause better teachers not always teach better students. For example in Figure 1, the performance of distilled student KGEs is worse as the dimension gap between the teacher KGEs and the student KGEs increases, even though higher dimensional teachers have better performance. Such a dimension gap implies different distillation performance, which further leads to a significant difference in the output distribution between the teacher and student models, which makes the student models hard to learn. Second, another great challenge faced by existing methods is that different optimization directions between the distillation objective and the original task objective cause the models to be difficult to converge (Li, Zhang, and Xing 2022). Third, current work has predominantly concentrated on improving inference efficiency but suffers from the problem of the time-consuming training process. Especially, existing distillation methods for KGEs either use a multi-teacher distillation framework or have teacher models trained together, which makes the training time several times longer compared to direct distillation. Hence, reducing training costs can speed up the process of obtaining distilled students, which means less consumption of resources and time. In real-world applications, reducing the training cost is beneficial for updating and deploying models in time and better adapting to changes in data and requirements. Reducing the training time for distilling KGEs in the medical, financial, and military scenarios is essential. For example, a medical assist clinic system must frequently extract knowledge from new drug instructions, journals, and databases to add to the constructed medical KGs for decision aids. In this scenario, frequently updating the medical KGs would lead to retraining KGEs many times, and such training should be very efficient.

To address the above issues, we propose IterDE, a novel iterative distillation framework for KGEs. To the best of our knowledge, different from the traditional distillation frameworks that directly compress a high-dimensional teacher model into a low-dimensional student model, IterDE first utilizes an iterative distillation method that gradually reduces the model size to close the dimension gap between the teacher model and the student model. Specifically, given a high-dimensional KGE model, it is first used as the teacher model and is compressed into a smaller model by the knowledge distillation method according to a specific compression ratio. Then this process will be iterated, and the student model obtained in the previous iteration will be used as the teacher model for the next iteration to instruct the smaller student model for training. Finally, the iteration stops when the dimension of the student model is equal to the desired dimension. As a result, the dimension gap between the teacher model and the student model is reduced by iterative distillation. In other words, IterDE allows knowledge transfer from high-dimensional teachers to low-dimensional students smoothly, meanwhile making the student model of the last iteration inherit well from the teacher model of the first iteration.

For achieving better distillation performance, we propose a new soft-label weighting dynamic adjustment mechanism. At the early stage of training, we give immense weight to the original hard label loss and a smaller weight to the soft label loss so that the model mainly optimizes toward the original hard loss. As the training proceeds, the weight of soft label loss gradually increases, making the soft label loss dominate the optimization of the model later in training. This mechanism could avoid the problem that the model cannot converge well due to different optimization directions with fixed weights.

In order to accelerate the training process, we use a single teacher for distillation and abandon updating the parameters of the teacher model during the training process. In IterDE, we fix the teacher model parameters unchanged and train only the student model during distillation, so we save a lot of training time while maintaining excellent distillation results.

We verify IterDE for typical KGEs on two widely-used datasets, FB15K-237 and WN18RR. The experimental results show that IterDE outperforms other distillation methods and can reduce training time by 50% compared to the previous best distillation methods. At the same time, we also verify the effectiveness of the soft-label weighting dynamic adjustment mechanism and each layer iteration by ablation experiments. Last but not least, we discover that the more fine-grained iterations, the better the distillation performance by additional experiments.

Our key contributions are summarized as follows:

- We propose an efficient knowledge distillation framework IterDE for knowledge graph embedding, which reduces the model sizes and inference time, while the low-dimensional student model can maintain the performance of the high-dimensional teacher model very well.
- We design a new soft-label weighting dynamic change mechanism in IterDE to balance the difference between different optimization objectives.
- Experimental results on public datasets and typical KGE models for the link prediction task demonstrate that IterDE achieves state-of-the-art performances compared with other distillation methods for KGEs. In addition, IterDE can save 50% training time on average, and more fine-grained iterations can improve the distillation performance.

Related Work

Our work is mainly related two research branches: knowledge graph embedding and knowledge distillation. We will first analyze these two topics one by one, then discuss the KGE work based on KD.

Knowledge Graph Embedding

Translation-based methods are the most typical KGE methods, such as TransE (Bordes et al. 2013), TransH (Wang et al. 2014) and TransD (Ji et al. 2015). They treat relationships as the distances between head and tail entities as the optimization objective of embedding. Compared with translation-based methods, bilinear-based methods can better increase the confidence in the latent semantic information of entities and relations in vector space. The classical bilinear model RESCAL (Nickel, Tresp, and Kriegel 2011) calculates the credibility of latent semantics of entities and relations in vector space. Based on RESCAL, DisMult (Yang et al. 2015) relaxes the constraint on the relation matrix and solves the problem of easy overfitting of RESCAL. Then, ComplEx (Trouillon et al. 2016) extends DisMult to complex space representation to have more powerful representation capability. Moreover, SimplE (Kazemi and Poole 2018) addresses the problem of head-tail irrelevance in training. Different from translation-based models and bilinear-based methods, rotation-based methods, such as RotatE (Sun et al. 2019), QuatE (Zhang et al. 2019), and DihEdral (Xu and Li 2019), treat relations as rotations between head and tail entities to handle the symmetric, anti-symmetric, inversion, and composition relations in the KGs. With the development of neural networks, some neural network-based embedding methods such as ConvE (Dettmers et al. 2018), AprilE (Liu et al. 2020) have also been proposed.

Knowledge Distillation

Knowledge distillation (KD) aims to improve the performance of student models by using larger teacher models to guide the training of smaller student models. KD is first introduced by Hinton et al. (2014), and since then, it has been widely adopted for compressing deep learning models. Knowledge distillation has been used in natural language processing to compress the pre-trained language models, e.g. Bert (Devlin et al. 2019). To the best of our knowledge, Tang et al. (2019) is the first to distill the knowledge of Bert into BiLSTM. DistillBert (Sanh et al. 2019) reduces the number of layers in the original Bert and adds more loss function computations to ensure the model performance. TinyBert (Jiao et al. 2019) divides knowledge distillation into two stages, pre-training and fine-tuning, and then performs knowledge distillation in both stages. Existing studies also show that the gaps in model sizes between teachers and students usually leads to a decrease in distillation (Mirzadeh et al. 2020). In computer vision, some studies try to mitigate the gap using teacher assistants (Gao et al. 2020; Mirzadeh et al. 2020), but far too little attention has been paid to it in the field of natural language processing.

Knowledge Distillation for Knowledge Graph Embedding

Most embedding methods tend to perform better with higher embedding dimensions. However, high-dimensional KGEs suffer from the drawbacks of long inference time and large model sizes. Few studies have attempted to compress the embedding models using knowledge distillation methods. MulDE (Wang et al. 2021) tries to use multi-teacher distillation methods to enhance the effectiveness of low-dimensional models but ignores the dimension gap which affects distillation performance. DualDE (Zhu et al. 2022) is the first single-teacher distillation method for KGEs, which uses two-stage distillation, where the teacher model fixes in the first stage and the teacher model is involved in training in the second stage. Although two-stage distillation converges the teacher and student distributions, this degrades the performance of the teacher model and thus reduces the upper bound on the performance of the student model. Therefore, DualDE cannot handle the dimension gap problem between the teacher model and the student model. In addition, multi-teacher distillation implies more training cost than using a single teacher, and two-stage distillation increases the training time significantly for bringing the teacher model into the training.

To make the low-dimensional student models better maintain the performances of the high-dimensional teacher models, in this paper, we focus on design a novel iterative distillation framework for KGEs. Moreover, we design a soft-label weighting dynamic change mechanism to optimize the distillation process for better distillation performance.

Methodology

Preliminaries

Knowledge Graph Embedding Given a set $\mathcal{E}$ of entities and a set $\mathcal{R}$ of relations, a KG $\mathcal{G} \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a set of fact triples. Generally, we use (*h, r, t*) to denote (*head, relation, tail*). KGEs take origin *h*, *r*, and *t* as positive triples T^+ and randomly replace *h* or *t* as negative triples T^-. Then, KGEs set a scoring function *S* on the vector representation of the triple. Different KGEs have different scoring functions. Usually, the binary cross-entropy (BCE) loss can be used to optimize the score function which can be calculated as follow:

$$\mathcal{L} = -\sum_{(h,r,t)\in T} [p \log \sigma(S) + (1-p) log(1-\sigma(S))] \quad (1)$$

where $T \in T^+ \cup T^-$, $p = 1$ for positive triples and $p = 0$ for negative triples. σ is the Softmax function.

Knowledge Distillation for KGEs The core of the knowledge distillation for KGEs is to make the low-dimensional student KGEs imitate the output of the high-dimensional teacher KGEs to learn knowledge. Specifically, given triples (h, r, t), we firstly feed them into the pre-trained teacher model and the untrained student model. Then, we define the scoring of the teacher model as S_{Tea} and the scoring of the student model as S_{Stu}. The distillation loss, always called the soft label loss, is Huber loss which is defined as Equation 2:

$$L_{Soft} = \begin{cases} \frac{1}{2}(S_{Tea} - S_{Stu})^2, & |S_{Tea} - S_{Stu}| \leq 1 \\ |S_{Tea} - S_{Stu}| - \frac{1}{2}, & |S_{Tea} - S_{Stu}| > 1 \end{cases} \quad (2)$$

Then we define L_{Hard} as equal to the original loss function of KGEs $\mathcal{L}$. Finally, we define the total loss L of student model training as shown in Equation 3:

$$L = L_{Hard} + \lambda L_{Soft} \quad (3)$$

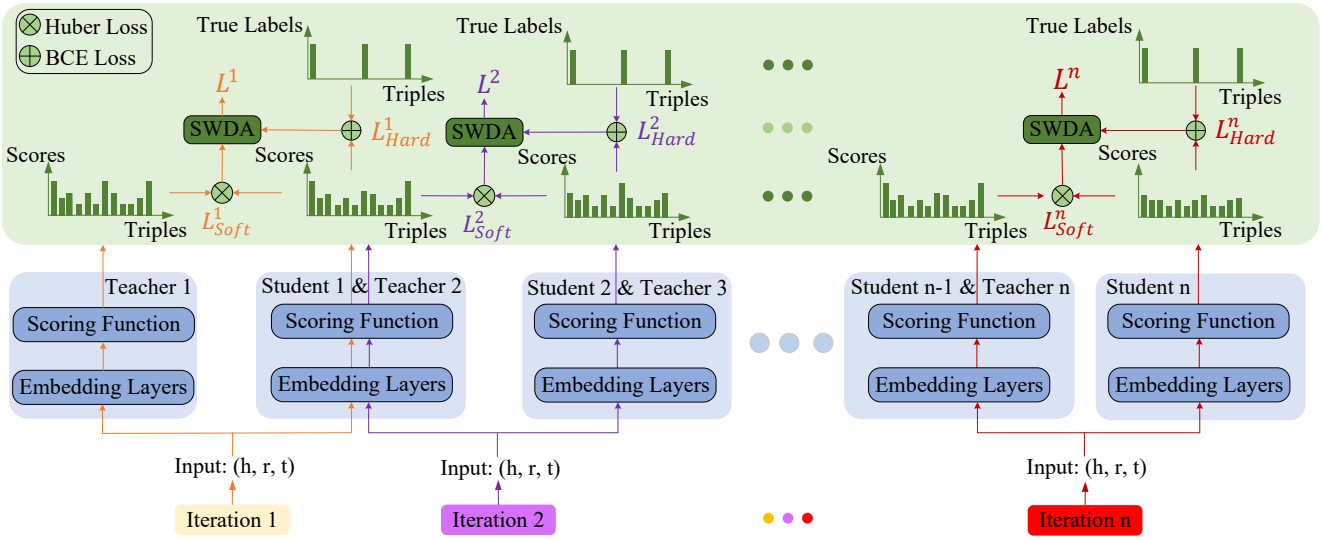

Figure 2: The framework of IterDE. SWDA is the soft-label weighting dynamic adjustment mechanism. In the k_{th} iteration, triples (h, r, t) are fed into both Teacher k and Student k. Then the L^k_{Soft} between Teacher k and Student k and the L^k_{Hard} of Student k are calculated. Finally, the two losses are combined as the final loss L^k by SWDA.

where λ is the weighting of soft label loss to balance L_{Hard} and L_{Soft}. Note that the teacher model parameters are fixed during the training, and only the student model is trained.

Framework Overview

The framework of IterDE is depicted in Figure 2. Similar with traditional distillation methods, we choose a higher dimensional KGE model as the teacher model and a lower dimensional KGE model as the student model. Then the student model mimics the output of the teacher model in scoring the triples during the training process. For hard label loss, we utilize the BCE loss to calculate the difference between true labels and the student model prediction. While the traditional distillation loss uses KL loss as the soft label loss, IterDE utilizes the more effective Huber loss, which can avoid the problem of asymmetry in KL loss.

Different from traditional one-time distillation methods, IterDE takes an iterative way to distillation. Specifically, we first choose a well-trained high-dimensional KGE model as Teacher 1 and an untrained homogeneous low-dimensional KGE model as Student 1. We input the triples *(h,r,t)* to both Student 1 and the Teacher 1 for scoring. Then we use the original loss function of Student 1 as the hard label loss L^1_{Hard}, and the difference between the scoring distribution of Teacher 1 and the scoring distribution of Student 1 as the soft label loss L^1_{Soft}. Therefore, the total loss L^1 is the weighted sum of the two formulated as Equation 4:

$$L^1 = L^1_{Hard} + \lambda L^1_{Soft} \tag{4}$$

where λ is the weighting of L^1_{Soft}. Subsequently, we let the well-trained Student 1 serve as the Teacher 2 for the next iteration and then use it to guide the training of the Student 2 with smaller dimensions. After *n* iterations, the final compressed model can be obtained in the last iteration. Similarly, the total loss for the last iteration can be formulated as Equation 5:

$$L^n = L^n_{Hard} + \lambda L^n_{Soft} \tag{5}$$

Especially, IterDE introduces a new soft-label weighting dynamic adjustment mechanism (SWDA) for improving efficiency in the distillation process. During the training process, we dynamically change the weight λ of the soft labels. In other words, λ increases gradually with training to balance the difference in optimization direction between hard label loss and soft label loss. Meanwhile, we fix the teacher model parameters during training and only train the student model, so our method has a significant advantage over previous distillation methods for KGEs in terms of training time.

Iterative Distillation for KGEs

IterDE introduces a novel iterative knowledge distillation way for KGEs. We first define the ratio of the teacher model dimension Dim^k_{Tea} to the student model dimension Dim^k_{Stu} in k_{th} iteration as the compression ratio α as Equation 6:

$$\alpha = \frac{Dim^k_{Tea}}{Dim^k_{Stu}} \tag{6}$$

Subsequently, we compress the model one at a time with this fixed α. We use the pre-trained teacher in the first iteration to train the first student. In the k_{th} iteration, we train the student k using the student $(k-1)$ generated in the $(k-1)_{th}$ iteration as the teacher k. According to Equation 2, the soft label loss of k_{th} iteration L^k_{Soft} can be defined as Equation 7:

$$L^k_{Soft} = \begin{cases} \frac{1}{2}(S^k_{Tea} - S^k_{Stu})^2, & |S^k_{Tea} - S^k_{Stu}| \le 1 \\ |S^k_{Tea} - S^k_{Stu}| - \frac{1}{2}, & |S^k_{Tea} - S^k_{Stu}| > 1 \end{cases} \tag{7}$$

where S^k_{Tea} defines the scoring of teacher model of k_{th} iteration, S^k_{Stu} defines the scoring of student model of k_{th}

iteration. The total loss L^k is computed as Equation 8:

$$L^k = L^k_{Hard} + \lambda L^k_{Soft} \tag{8}$$

where L^k_{Hard} defines the hard label loss of k_{th} iteration. We define the total number of iterations as N. Then the final compression ratio A is shown in Equation 9:

$$A = \underbrace{\alpha \cdots \alpha}_{N} \tag{9}$$

The iteration stops when $Dim^N_{Stu} \cdot A = Dim^1_{Tea}$. We use the student model obtained in the last iteration as the final compression result. We finally transfer the knowledge of the teacher model to the student model with compression ratio *A*.

Soft-Label Weighting Dynamic Adjustment Mechanism

In order to balance the weight between hard label loss and soft label loss during distillation, we dynamically adjust the weight of soft labels λ during the training process. At the beginning of training, we assign smaller initial values of soft label weights. Then, as the training progresses, the soft label weights are increased and finally fixed. Specifically, we divide the complete training process into two phases. In the first stage, hard label loss dominates, and soft label loss weights are assigned smaller initial values and gradually increase. In the second stage, the soft label loss weights are fixed. Let the total number of epochs be *M*. Then the soft label loss weight λ at the m_{th} epoch is as shown in Equation 10:

$$\lambda = \begin{cases} \lambda_0 \cdot \frac{k}{L_{Hard}+L_{Soft}} & 0 \le m \le M/p \\ \lambda_0 & M/p < m \le M \end{cases} \tag{10}$$

where k ensures that $\frac{k}{L_{Hard}+L_{Soft}}$ is in the range of $[0, 1]$, and *p* is the hyperparameter used to control the time of soft label weighting adjustment. In this way we avoid the difficulty of converging the model in the correct direction due to the difference in the direction of hard label loss optimization and soft label loss optimization at the early stage of training.

Experiments

This section reports the experimental results. First, we address the experimental setup, including datasets, model selection, baselines, metrics and settings. Then, we present the experimental results on two datasets, including comparisons for performance and training time. Meanwhile, we conduct the ablation experiments and detailed analysis of the effectiveness of each layer of iteration. Furthermore, we perform the study of distillation performance with different iteration ratios. Finally, we provide intensive discussions of the methods and experimental results. The code of IterDE and the datasets can be accessed via https://github.com/seukgcode/IterDE.

Experimental Setup

Datasets We use the two popular and open datasets in KGEs: FB15K-237 (Dettmers et al. 2018) and WN18RR (Toutanova et al. 2015). FB15K-237 is a subset of Freebase (Bordes et al. 2013), and WN18RR is a subset of WordNet (Bordes et al. 2013). FB15K-237 and WN18RR are the most commonly used datasets for evaluating KGE models. The detailed statistical information is shown in Table 1.

Datasets	N_e	N_r	N_{Train}	N_{Valid}	N_{Test}
FB15K-237	14,541	237	272,115	17,535	20,466
WN18RR	40,943	11	86,835	3,034	3,134

Table 1: The statistics of datasets.

Models	Scoring Function for $(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t})$
TransE	$-\left\|\boldsymbol{h}+\boldsymbol{r}-\boldsymbol{t}\right\|_{L1/L2}$
ComplEx	$Re(\boldsymbol{h}^\top diag(\boldsymbol{r})\bar{\boldsymbol{t}})$
SimplE	$\left(\langle \boldsymbol{h_H}, \boldsymbol{r}, \boldsymbol{t_T}\rangle + \langle \boldsymbol{t_H}, \boldsymbol{r^{-1}}, \boldsymbol{h_T}\rangle\right)/2$
RotatE	$-\left\|\boldsymbol{h} \odot \boldsymbol{r} - \boldsymbol{t}\right\|$

Table 2: The scoring function of KGEs. $Re(\boldsymbol{x})$ denotes a real vector component of $\boldsymbol{x}$. $\bar{\boldsymbol{x}}$ denotes the conjugate of a complex number $\boldsymbol{x}$. $< \boldsymbol{v}, \boldsymbol{w}, \boldsymbol{x} > \doteq (\boldsymbol{v} \odot \boldsymbol{w}) \cdot \boldsymbol{x}$, where $\cdot$ denotes dot product and $\odot$ denotes the Hadamard product. $\boldsymbol{r}^{-1}$ denotes the inverse relationship of $\boldsymbol{r}$.

Model Selection The KGEs we choose are TransE (Bordes et al. 2013), ComplEx (Trouillon et al. 2016), SimplE (Kazemi and Poole 2018) and RotatE (Sun et al. 2019). The scoring functions of these four KGEs are shown in Table 2. All these models are typical KGEs, and they have better performances with higher the embedding dimensions. Since most hyperbolic models suffer from the phenomenon that the higher the embedding dimension is, the worse the performance is (Wang et al. 2021; Pan and Wang 2021), we do not use hyperbolic models for distillation. In order to fully reveal the advantage that IterDE can improve the performance of the student on the situation of large dimensional gap between the teacher and the student, we set the dimension of the teacher to 512 and the dimension of the student to 32. Meanwhile, we try to choose the teacher with the best performance. Experientially, the 512-dimensional embedding ensures that a KGE model obtains excellent performance, and as the embedding dimension increases, the performance will not improve significantly. As mentioned above, to demonstrate the advantages of IterDE, we choose the 32-dimensional student, which has as large a dimensional gap as possible with the teacher.

Baselines We choose strong distillation methods as baselines:

- **BKD** (Hinton et al. 2014): The primary distillation method uses the traditional KL divergence to calculate the scores between the teacher model and the student model.
- **RKD** (Park et al. 2019): The structural distillation method by calculating the distance and angle of the embedding vector between the teacher model and the student model as distillation loss.
- **TA** (Mirzadeh et al. 2020): The method of distillation through teacher assistants selects a medium model whose

Dataset	Method	TransE			ComplEx			SimplE			RotatE		
		MRR	Hits@10	Hits@1	MRR	Hits@10	Hits@1	MRR	Hits@10	Hits@1	MRR	Hits@10	Hits@1
FB15K-237	$\mathrm{No-KD}^{*}_{\mathrm{Teacher}}$	0.286	0.481	0.185	0.298	0.472	0.213	0.283	0.454	0.199	0.326	0.520	0.229
	$\mathrm{No-KD}^{*}_{\mathrm{Student}}$	0.183	0.317	0.115	0.162	0.317	0.096	0.155	0.314	0.075	0.285	0.461	0.195
	$\mathrm{BKD}^{*}_{\mathrm{Student}}$	0.223	0.378	0.146	0.239	0.394	0.161	0.187	0.331	0.112	0.294	0.457	0.214
	$\mathrm{RKD}^{*}_{\mathrm{Student}}$	0.246	0.410	0.162	0.270	0.440	0.185	0.213	0.367	0.125	0.303	0.486	0.210
	$\mathrm{TA}^{*}_{\mathrm{Student}}$	0.242	0.405	0.158	0.259	0.423	0.177	0.208	0.362	0.119	0.296	0.475	0.206
	$\mathrm{MulDE}_{\mathrm{Student}}$	0.238	0.417	0.149	0.183	0.301	0.124	0.190	0.320	0.126	0.300	0.477	0.211
	$\mathrm{DualDE}^{*}_{\mathrm{Student}}$	0.254	0.418	0.173	0.274	0.444	0.189	0.236	0.407	0.149	0.306	0.489	0.216
	IterDE$_{\mathrm{Student}}$	**0.266**	**0.443**	**0.175**	**0.277**	**0.454**	**0.192**	**0.244**	**0.410**	**0.162**	**0.310**	**0.492**	**0.225**
WN18RR	$\mathrm{No-KD}^{*}_{\mathrm{Teacher}}$	0.232	0.531	0.025	0.433	0.515	0.387	0.421	0.485	0.389	0.477	0.575	0.427
	$\mathrm{No-KD}^{*}_{\mathrm{Student}}$	0.164	0.410	0.023	0.268	0.366	0.216	0.273	0.396	0.186	0.421	0.453	0.401
	$\mathrm{BKD}^{*}_{\mathrm{Student}}$	0.184	0.442	0.026	0.343	0.406	0.302	0.321	0.452	0.259	0.441	0.497	0.412
	$\mathrm{RKD}^{*}_{\mathrm{Student}}$	0.194	0.454	0.028	0.368	0.456	0.322	0.372	0.475	0.297	0.455	0.529	0.416
	$\mathrm{TA}^{*}_{\mathrm{Student}}$	0.189	0.458	0.032	0.372	0.464	0.315	0.359	0.476	0.283	0.452	0.519	0.416
	$\mathrm{MulDE}_{\mathrm{Student}}$	0.209	0.499	0.022	0.362	0.444	0.313	0.333	0.450	0.267	0.451	0.536	0.409
	$\mathrm{DualDE}^{*}_{\mathrm{Student}}$	0.210	0.484	0.035	0.397	0.473	0.352	0.384	0.479	0.311	0.468	**0.560**	0.419
	IterDE$_{\mathrm{Student}}$	**0.218**	**0.505**	**0.037**	**0.400**	**0.484**	**0.374**	**0.387**	**0.482**	**0.324**	**0.471**	0.558	**0.421**

Table 3: Distillation results on FB15K-237 and WN18RR. All results of * are retrieved from (Zhu et al. 2022). Results of MulDE are obtained from our reproduced version of MulDE. Results of IterDE are the average of random five running times.

performance is closest to the mean of the performance of the teacher model and the student model as the teacher assistant.

- **MulDE** (Wang et al. 2021): It is a multi-teacher distillation framework for KGEs that uses four high-dimensional models as teachers to guide the student training. In our experiments, we choose high-dimensional TransE, ComplEx, SimplE, and RotatE as teachers to train single low-dimensional KGEs.
- **DualDE** (Zhu et al. 2022): The previous best distillation framework for KGEs, which uses a two-stage distillation combining with the structural distillation method. The teacher model is engaged in the second training stage to improve the distillation performance of the student model.

Metrics We evaluate the performance of the distilled KGEs on the link prediction task. Given the triples *(h,r,t)* in the dataset, we replace *h* with all other entities and then score and rank all the generated triples to get $rank^h$. Similarly, we can also get $rank^t$ and then average them to get *rank*. Then, we calculate MRR, Hits@1 and Hits@10 of *rank* as evaluation metrics. The higher MRR, Hits@1, and Hits@10, the better performance a KGE has.

Settings All experiments are implemented on GPU GeForce RTX 2080 Ti. The experiments are extended from OpenKE (Han et al. 2018), an open source library based on PyTorch (Paszke et al. 2019), with CUDA version 10.2.89. In all experiments, we set the teacher model dimension to 512 and the student model dimension to 32. In distillation, we set the compression ratio of each layer α to 2 and the number of iterations *N* to 4. We set the value of hyperparameter *p* to 2, while 5 and 10 have the similar results. We set the batch size to 1024 and the epoch for each iteration to a maximum of 1000. We use Adagrad as the optimizer, and the learning rate is chosen among [0.5, 0.1, 0.01]. The initial soft label weight λ_0 is chosen in the range [1, 0.1, 0.01]. Different learning rates and initial soft label weight have tiny differences in results, and we report the best results.

Dataset	Method	TransE	ComplEx	SimplE	RotatE
FB15K-237	DualDE	111	347	201	296
	IterDE	**57**	**167**	**105**	**134**
WN18RR	DualDE	86	162	124	300
	IterDE	**41**	**99**	**57**	**111**

Table 4: Comparison of training time (min) between different distillation methods for KGEs.

Results

Comparison for Performance Main experimental results on FB15K-237 and WN18RR are shown in Table 3. The best results are in bold. The second best results are indicated by underlining. *No-kd* denotes the results of models without distillation. Firstly, we observe a significant performance drop between the high-dimensional *No-kd* teacher model (512 dimensions) and the low-dimensional *No-kd* student model (32 dimensions). By using IterDE for distillation, there is a noticeable improvement in the performances of the low-dimensional student models. Compared to the *No-kd* student models, the MRR of TransE, ComplEx, SimplE, and RotatE can be improved by 45% (from 0.183 to 0.266), 71% (from 0.162 to 0.277), 57% (from 0.155 to 0.244), and 9% (from 0.285 to 0.310), respectively, after IterDE distillation on FB15K-237. On WN18RR, performances of these four models can be improved by 33%, 49%, 42%, and 11% compared to no distillation. Moreover, the performances of the low-dimensional student models after distillation by IterDE are very close to the performance of the teacher model. On

Dataset	Method	TransE			ComplEx			SimplE			RotatE		
		MRR	Hits@10	Hits@1	MRR	Hits@10	Hits@1	MRR	Hits@10	Hits@1	MRR	Hits@10	Hits@1
FB15K-237	IterDE	**0.266**	**0.443**	**0.175**	**0.277**	**0.454**	**0.192**	**0.244**	**0.410**	**0.162**	**0.310**	**0.492**	**0.225**
	IterDE w/o SWDA	0.264	0.440	0.174	0.264	0.425	0.182	0.233	0.403	0.146	0.288	0.476	0.195
	IterDE w/o Iter	0.259	0.439	0.169	0.246	0.409	0.164	0.235	0.394	0.156	0.280	0.465	0.188
WN18RR	IterDE	**0.218**	**0.505**	**0.037**	**0.400**	**0.484**	**0.374**	**0.387**	**0.482**	**0.324**	**0.471**	**0.558**	**0.421**
	IterDE w/o SWDA	0.215	0.502	0.022	0.396	0.453	0.360	0.347	0.430	0.298	0.429	0.484	0.397
	IterDE w/o Iter	0.214	0.503	0.020	0.391	0.455	0.363	0.344	0.427	0.293	0.448	0.534	0.406

Table 5: Ablation experimental results.

Dataset	Ratio α	TransE			ComplEx			SimplE			RotatE		
		MRR	Hits@10	Hits@1	MRR	Hits@10	Hits@1	MRR	Hits@10	Hits@1	MRR	Hits@10	Hits@1
FB15K-237	$\alpha = 16$	0.259	0.439	0.169	0.246	0.409	0.164	0.235	0.394	0.156	0.280	0.465	0.188
	$\alpha = 4$	0.264	0.440	0.172	0.260	0.426	0.178	0.211	0.384	0.124	0.282	0.473	0.185
	$\alpha = 2$	**0.266**	**0.443**	**0.175**	**0.277**	**0.454**	**0.192**	**0.244**	**0.410**	**0.162**	**0.310**	**0.492**	**0.225**
WN18RR	$\alpha = 16$	0.215	0.503	0.020	0.391	0.455	0.363	0.344	0.427	0.293	0.448	0.534	0.406
	$\alpha = 4$	0.217	0.504	0.023	0.400	0.456	0.367	0.351	0.440	0.281	0.447	0.531	0.405
	$\alpha = 2$	**0.218**	**0.505**	**0.037**	**0.400**	**0.484**	**0.374**	**0.387**	**0.482**	**0.324**	**0.471**	**0.558**	**0.421**

Table 6: Distillation performance with different iteration ratios.

FB15K-237, the MRR of the four KGEs can achieve 93%, 93%, 86%, and 95% of homogeneous teacher model performance. On WN18RR, these four models can achieve 94%, 92%, 92%, and 98% of homogeneous teacher model performance. These results demonstrate that IterDE achieves excellent distillation results on various models and datasets.

Secondly, Table 3 shows that IterDE can achieve better distillation results compared to traditional distillation methods. Compared to BKD, RKD, and TA, which are compression methods that directly apply traditional distillation methods to KGE, IterDE significantly improves the performance. Specifically, for the four distilled KGEs, IterDE can improve the MRR by 19%, 16%, 30%, and 5%, respectively, compared to the traditional distillation methods on FB15K-237. On WN18RR, IterDE can improve the MRR at most 18%, 17%, 21%, and 7% compared with the conventional distillation method. This indicates that the well-designed IterDE is more effective than directly applying the existing distillation strategy to KGEs.

Thirdly, compared with MulDE and DualDE, two distillation methods for KGEs, IterDE outperforms them on all datasets and models. On FB15K-237, the MRR of IterDE increases up to 12%, 51%, 28%, and 3% compared to MulDE and DualDE. On WN18RR, IterDE can improve the MRR up to 4%, 10%, 16%, and 4% compared to MulDE and DualDE. These evidences indicate that IterDE outperforms all currently available distillation-based KGE compression methods. Compared to strong baselines such as DualDE, IterDE does not have a very significant performance improvement. The reason is that the students of IterDE and DualDE both achieve excellent performance close to the teacher models, which is the upper boundary performance of student models. Moreover, compared to DualDE, considering that the main metric is MRR, the improvement (0.01-0.02) of IterDE is non-trivial. Compared with other distillation methods, the MRR of IterDE is increased by 5%-30%,

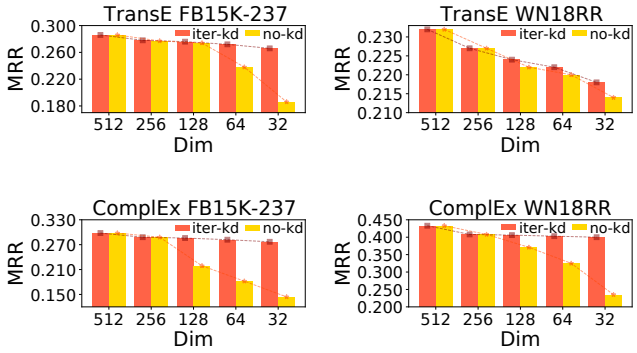

Figure 3: Effectiveness of each layer of iteration with TransE and ComplEx. The red denotes the MRR generated by *iter-kd* (iterative distillation from 512 to 32 dimension) and the yellow denotes the MRR generated by *no-kd* (without distillation).

which is a significant improvement.

Comparison for Training Time In order to compare the training time of IterDE and previous best baseline DualDE, we implement DualDE and run it in same experimental settings of IterDE. The training time results are shown in Table 4. It can be seen that IterDE can save 50% of time on average compared to DualDE. Especially, IterDE can reduce the training time up to 63% for RotatE. For one thing, the reason why IterDE can save training time is that the amount of parameters in IterDE that need to be updated is much less compared with DualDE. For another thing, DualDE incorporates a complex structural distillation method, which makes the vector operations in higher dimensions time-consuming.

Ablation Experiments We investigate the effect of whether using the iterative strategy (Iter) and the soft-label weighting dynamic adjustment mechanism (SWDA) through ablation experiments. Table 5 reports the results of ablation experiments on FB15K-237 and WN18RR. First, after removing the iterative strategy, i.e. distilling directly from the 512-dimensional teacher model to the 32-dimensional student model, the MRR of four distilled KGEs on FB15K-237 drops by 2.6%, 11.2%, 3.7% and 9.7%, respectively, and the MRR on WN18RR drops by 1.8%, 2.3%, 11.1% and 4.9%, respectively. This means that iterative strategy can not only transfer knowledge smoothly from the high-dimensional KGEs to the low-dimensional KGEs, but also close the dimension gap efficiently. Second, after removing the soft-label weighting dynamic adjustment mechanism, the MRR of four distilled KGEs on FB15K-237 drops by 0.75%, 0.68%, 4.5% and 7.1%, respectively, and the MRR on WN18RR drops by 1.4%, 0.6%, 10.3% and 8.9%, respectively. This means that the soft-label weighting dynamic adjustment mechanism is efficient.

Effectiveness of Each Layer of Iteration Figure 3 shows the performances of the intermediate models generated by IterDE in each iteration of TransE and ComplEx on FB15K-237 and WN18RR. It can be seen that the performances of the intermediate KGEs generated by IterDE are superior compared to the KGEs with the same dimension without distillation. Specifically, the MRR of distilling to 128, 64, and 32 dimensions shows a significant improvement compared to no distillation, especially when distilling from 64 to 32 dimensions. However, the distillation from 512 to 256 dimensions shows a slight improvement. From another perspective, the performance of iterative distilling from 512 to 32 dimensions is equal to iterative distilling from 256 to 32 dimensions, which means it can not only further reduce the train time but also maintain the performance of the final student. In addition, we can also find that the performance of each iteration of distillation is similar to the results of the previous iteration of distillation, which indicates that each iteration of the student KGEs can inherit the knowledge from the teacher KGEs very well.

Distillation Performance with Different Iteration Ratios This experiment investigates the impact of the compression ratio α on the distillation performance for each iteration. We set the teacher model to 512 dimensions and the final student model to 32 dimensions. Table 6 shows the performance of distillation with α = 2, 4, and 16, respectively. It is obvious that the distillation effect is the best with a compression ratio of 2 for each layer and the worst with a compression ratio of 16 for each layer. From this, we can find that the closer the performance of the adjacent teacher and student models during the iteration, the less knowledge loss is transmitted and the better the final student model performs. Therefore, our experimental results reveal an important fact, namely, more grained-fined distillation can achieve better distillation results.

Discussion

Different Choices of the Dimension for Student KGEs The dimensions of the student KGEs can be chosen in different ways, such as 32, 64, and 128. We report the main results of 32-dimensional student KGEs, because the improvement of 32-dimension student KGEs is more significant compared to others. It can be seen in Figure 3 that the lower the dimension of the student KGE, the more evident the effect of IterDE. In practical applications, we can choose the dimensions of the final generated student KGE according to the requirements. The smaller the dimension of the generated student KGE, the faster the inference speed, but the more significant the performance loss.

Differences between IterDE and Other Strong Baselines Although IterDE and other strong baselines try to apply knowledge distillation to compress KGEs, IterDE is not an expanded work based on other strong baselines. IterDE and other strong baselines are different in the following aspects. First, the motivations are different. Other strong baselines try to integrate the knowledge of multiple teacher KGEs into student KGEs or utilize the dual influence of student KGEs and teacher KGEs to make the output distributions as similar as possible. IterDE focuses on reducing the dimension gap between student KGEs and teacher KGEs to transfer knowledge more smoothly. In addition, IterDE focuses on improving training efficiency. Second, the processes of distillation are different. Other strong baselines use a multi-teacher strategy or a two-stage distillation strategy. However, IterDE proposes a novel framework of iterative distillation to improve the performance of student KGEs significantly. Besides, the proposed soft label loss mechanism in IterDE further improves the training efficiency. Third, the performance of distillation is different. IterDE achieves the best distillation performance compared to other strong baselines. Also, IterDE works better for student KGEs with lower dimensions.

Conclusion

This paper proposes a novel distillation framework IterDE for KGEs, with promising performance and less training cost. IterDE first introduces iterative knowledge distillation way to gradually reduce the model sizes, which can alleviate the drawback of dimension gaps between the teacher models and the student models. Besides, a soft-label weighting dynamic adjustment mechanism is designed to make the student model converge to a better degree. In the future, we will explore how to compress KGEs based on neural networks or more complex pre-trained language models with iterative distillation, which need faster training and inference time.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. The work was supported in part by the 13th Five-Year All-Army Common Information System Equipment Pre-Research Project (Grant Nos. 31514020501, 31514020503). All opinions are of the authors and do not reflect the view of sponsors.

References

Bizer, C.; Lehmann, J.; Kobilarov, G.; Auer, S.; Becker, C.; Cyganiak, R.; and Hellmann, S. 2009. Dbpedia-a crystallization point for the web of data. *Journal of web semantics*, 7(3): 154–165.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Bordes, A.; Weston, J.; and Usunier, N. 2014. Open question answering with weakly supervised embedding models. In *ECML-PKDD*.

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.

Dong, X.; Gabrilovich, E.; Heitz, G.; Horn, W.; Lao, N.; Murphy, K.; Strohmann, T.; Sun, S.; and Zhang, W. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *SIGKDD*.

Gao, M.; Shen, Y.; Li, Q.; and Loy, C. C. 2020. Residual knowledge distillation. *arXiv preprint arXiv:2002.09168*.

Han, X.; Cao, S.; Xin, L.; Lin, Y.; Liu, Z.; Sun, M.; and Li, J. 2018. OpenKE: An open toolkit for knowledge embedding. In *EMNLP*.

Hinton, G.; Vinyals, O.; Dean, J.; et al. 2014. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Ji, G.; He, S.; Xu, L.; Liu, K.; and Zhao, J. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL*.

Jiao, X.; Yin, Y.; Shang, L.; Jiang, X.; Chen, X.; Li, L.; Wang, F.; and Liu, Q. 2019. Tinybert: Distilling bert for natural language understanding. In *EMNLP*.

Kazemi, S. M.; and Poole, D. 2018. Simple embedding for link prediction in knowledge graphs. *NeurIPS*.

Li, X.; Zhang, Y.; and Xing, C. 2022. Jointly Learning Knowledge Embedding and Neighborhood Consensus with Relational Knowledge Distillation for Entity Alignment. *arXiv preprint arXiv:2201.11249*.

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*.

Liu, Y.; Wang, P.; Li, Y.; Shao, Y.; and Xu, Z. 2020. AprilE: Attention with pseudo residual connection for knowledge graph embedding. In *COLING*.

Mirzadeh, S. I.; Farajtabar, M.; Li, A.; Levine, N.; Matsukawa, A.; and Ghasemzadeh, H. 2020. Improved knowledge distillation via teacher assistant. In *AAAI*.

Nickel, M.; Tresp, V.; and Kriegel, H.-P. 2011. A three-way model for collective learning on multi-relational data. In *ICML*.

Pan, Z.; and Wang, P. 2021. Hyperbolic hierarchy-aware knowledge graph embedding for link prediction. In *EMNLP*.

Park, W.; Kim, D.; Lu, Y.; and Cho, M. 2019. Relational knowledge distillation. In *CVPR*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; and et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *NeurIPS*.

Suchanek, F. M.; Kasneci, G.; and Weikum, G. 2007. Yago: A core of semantic knowledge. In *WWW*.

Sun, Z.; Deng, Z.-H.; Nie, J.-Y.; and Tang, J. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *ICLR*.

Tang, R.; Lu, Y.; Liu, L.; Mou, L.; Vechtomova, O.; and Lin, J. 2019. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*.

Toutanova, K.; Chen, D.; Pantel, P.; Poon, H.; Choudhury, P.; and Gamon, M. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*.

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; and Bouchard, G. 2016. Complex embeddings for simple link prediction. In *ICML*.

Wang, K.; Liu, Y.; Ma, Q.; and Sheng, Q. Z. 2021. Mulde: Multi-teacher knowledge distillation for low-dimensional knowledge graph embeddings. In *WWW*.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*.

Xu, C.; and Li, R. 2019. Relation Embedding with Dihedral Group in Knowledge Graph. In *ACL*.

Yang, B.; Yih, S. W.-t.; He, X.; Gao, J.; and Deng, L. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *ICLR*.

Zhang, F.; Yuan, N. J.; Lian, D.; Xie, X.; and Ma, W.-Y. 2016. Collaborative knowledge base embedding for recommender systems. In *SIGKDD*.

Zhang, S.; Tay, Y.; Yao, L.; and Liu, Q. 2019. Quaternion knowledge graph embeddings. In *NeurIPS*.

Zhu, Y.; Zhang, W.; Chen, M.; Chen, H.; Cheng, X.; Zhang, W.; and Chen, H. 2022. DualDE: Dually distilling knowledge graph embedding for faster and cheaper reasoning. In *WSDM*.