

# Croppable Knowledge Graph Embedding

**Yushan Zhu<sup>♦◊</sup>, Wen Zhang<sup>♦◊</sup>, Zhiqiang Liu<sup>♦◊</sup>, Mingyang Chen<sup>♦</sup>,**  
**Lei Liang<sup>♣</sup>, Huajun Chen<sup>♣◊♡\*</sup>**

<sup>♦</sup> Zhejiang University <sup>♣</sup> Ant Group

<sup>◊</sup> Zhejiang University - Ant Group Joint Laboratory of Knowledge Graph

<sup>♡</sup> Zhejiang Key Laboratory of Big Data Intelligent Computing

{yushanzhu, zhang.wen, zhiqiangliu, mingyangchen, huajunsir}@zju.edu.cn

## Abstract

Knowledge Graph Embedding (KGE) is a common approach for Knowledge Graphs (KGs) in AI tasks. Embedding dimensions depend on application scenarios. Requiring a new dimension means training a new KGE model from scratch, increasing cost and limiting efficiency and flexibility. In this work, we propose a novel KGE training framework MED. It allows one training to obtain a croppable KGE model for multiple scenarios with different dimensional needs. Sub-models of required dimensions can be directly cropped and used without extra training. In MED, we propose a mutual learning mechanism to improve the low-dimensional sub-models and make high-dimensional sub-models retain the low-dimensional sub-models' capacity, an evolutionary improvement mechanism to promote the high-dimensional sub-models to master the triple that the low-dimensional sub-models can not, and a dynamic loss weight to adaptively balance the multiple losses. Experiments on 4 KGE models across 4 standard KG completion datasets, 3 real-world scenarios using a large-scale KG, and extending MED to the BERT language model demonstrate its effectiveness, high efficiency, and flexible extensibility.

## 1 Introduction

Knowledge Graphs (KGs) consist of triples in the form of (*head entity, relation, tail entity*), abbreviated as  $(h, r, t)$ . KGs are widely used in recommendation systems (Zhu et al., 2021), information extraction (Daiber et al., 2013), question answering (Diefenbach et al., 2018), etc. A common KG application way is knowledge graph embedding (KGE) (Bordes et al., 2013; Sun et al., 2019b), which maps KG entities and relations into continuous vector spaces for various tasks.

Higher-dimensional KGEs have more expressive power and better performance but come with more parameters, demanding more storage and computing resources (Zhu et al., 2022; Sachan, 2020). The

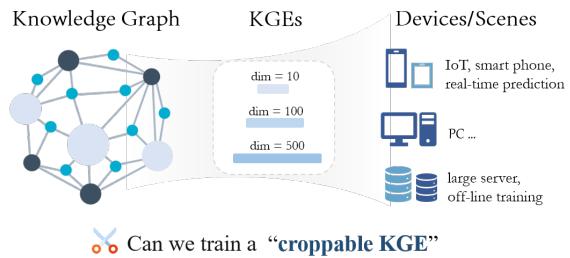


Figure 1: Diverse KGE dimensions for a KG.

suitable KGE dimensions vary by device or scenario. As Fig. 1 shows, large remote servers can support high-dimensional KGEs, while small-to-medium terminal devices like in-vehicle systems or smartphones can only handle low-dimensional ones due to resource limits. Thus, people prefer training high-quality KGEs with appropriate dimensions. However, a new KGE must be trained from scratch when a new dimension is needed. Especially for low-dimensional KGEs, model compression techniques like knowledge distillation (Hinton et al., 2015; Zhu et al., 2022) are required to ensure performance. This raises training costs and restricts KGE's efficiency and flexibility across scenarios.

Thus a new concept "croppable KGE" is proposed and we are interested in the research question that **is it possible to train a croppable KGE, with which KGEs of various required dimensions can be cropped out of it and directly used without extra training, and perform well?**

In this work, our croppable KGE learning idea is to train an entire KGE with multiple different-dimensional sub-models sharing embedding parameters and trained simultaneously. The aim is for low-dimensional sub-models to benefit from high-dimensional ones, and high-dimensional sub-models to retain low-dimensional ones' ability and learn what they can't. Based on this, we propose a croppable KGE training framework **MED**, with three main modules: **Mutual learning mechanism**, **Evolutionary improvement mechanism**, and **Dynamic loss weight**. Specifically, the **mutual learning mechanism**, based on knowledge distil-

lation, enables neighbor sub-models to learn from each other, improving low-dimensional sub-model performance and helping high-dimensional ones retain low-dimensional abilities. The **evolutionary improvement mechanism** helps high-dimensional focus on and master triples that low-dimensional ones can't correctly predict. The **dynamic loss weight** adaptively balances sub-models' losses according to dimension to enhance overall performance.

We evaluate the effectiveness of MED by implementing it on four typical KGE methods and four standard KG datasets. We prove its practical value via a real-world large-scale KG and downstream tasks, and demonstrate its extensibility on BERT (Devlin et al., 2019) and GLUE (Wang et al., 2019) benchmarks. Experimental results show that: (1) MED trains a croppable KGE for various dimensional needs, with high-performing, parameter-shared sub-models ready for direct use. (2) MED has far higher training efficiency than independent training or knowledge distillation. (3) MED can be extended to other neural networks and performs well. (4) The three modules in MED are crucial for optimal performance. In summary, our contributions are as follows:

- We introduce the research question and task of training croppable KGE, allowing direct use of differently-dimensional KGEs.
- We propose a novel framework MED, including a mutual learning mechanism, an evolutionary improvement mechanism, and a dynamic loss weight, to ensure the overall performance of all sub-models.
- We show that MED's sub-models perform well, with low-dimensional ones outperforming KGEs trained by SOTA distillation methods. MED also shows good performance and extensibility in real-world applications and other types of neural networks.

## 2 Related Work

This work is to achieve a croppable KGE for different dimensional needs. A common way to get a good-performance KGE of the target dimension is using knowledge distillation with a high-dimensional powerful teacher KGE. So, we focus on two relevant research areas: knowledge graph embedding and knowledge distillation.

### 2.1 Knowledge Graph Embedding

Knowledge graph embedding (KGE) maps KG entities and relations into continuous vector spaces for downstream tasks. TransE (Bordes et al., 2013), a representative translation-based KGE, treats relations as translations between entities. Its variants include TransH(Wang et al., 2014), TransR(Lin et al., 2015), TransD(Ji et al., 2015), etc. RESCAL(Nickel et al., 2011), based on vector decomposition, was followed by improvements like DistMult(Yang et al., 2015), ComplEx(Trouillon et al., 2016), and SimplE(Kazemi and Poole, 2018). RotatE(Sun et al., 2019b), a rotation-based method, views relations as rotations between entities, similar to QuatE(Zhang et al., 2019) and DihEdral(Xu and Li, 2019). PairRE(Chao et al., 2021) uses two relation vectors for complex pattern encoding. With neural network development, KGEs based on graph neural networks (GNNs) (Dettmers et al., 2018; Nguyen et al., 2018; Schlichtkrull et al., 2018; Vashisht et al., 2020) also emerged. While KGEs are simple and effective, a key challenge remains: Different scenarios demand different KGE dimensions based on device resources. Training a new KGE model from scratch for each dimension requirement hikes training costs and restricts flexibility in serving diverse scenarios.

### 2.2 Knowledge Distillation

High-dimensional KGEs possess strong expressive power thanks to numerous parameters. However, they demand substantial storage and computing resources, rendering them unsuitable for all scenarios, particularly those involving small devices. To address this, a prevalent approach is to compress a high-dimensional KGE into target dimension via knowledge distillation (Hinton et al., 2015; Mirzadeh et al., 2020) and quantization (Bai et al., 2021; Stock et al., 2021) technology.

Quantization replaces continuous vector representations with lower-dimensional discrete codes. TS-CL (Sachan, 2020) was the first to apply quantization in KGE compression. LightKG (Wang et al., 2021a) uses a residual module to create diversity in codebooks. Yet, as quantization doesn't boost inference speed, it's still not suitable for devices with limited computing resources.

Knowledge distillation (KD) is widely used in Computer Vision (Mirzadeh et al., 2020) and Natural Language Processing (Devlin et al., 2019; Sun et al., 2019a) to shrink model size and boost infer-

ence speed. Its core is using a large teacher model’s output to guide a small student model’s training. DualDE (Zhu et al., 2022) is a representative KD-based work to transfer knowledge from high- to low-dimensional KGE, considering teacher-student mutual influences. MulDE (Wang et al., 2021b) transfers knowledge from multiple hyperbolic KGE models to one student. ISD (Zhou et al., 2022b) improves low-dimensional KGE by playing teacher and student roles alternatively. IterDE (Liu et al., 2023) introduces iterative distillation for smooth knowledge transfer. Other KG-related distillation works include PMD (Fan et al., 2024), which applies distillation to pre-trained language models for KG completion, IncDE (Liu et al., 2024), using distillation between same-dimensional models at different times for incremental learning, and SKDE (Xu et al., 2024), which proposes self-knowledge distillation to avoid a complex teacher model. Among these, DualDE (Zhu et al., 2022) and IterDE (Liu et al., 2023) are most relevant to our work as they compress high-dimensional teacher into low-dimensional student. In this work, we propose a novel KD-based KGE training framework MED, one training can obtain a croppable KGE that meets multiple dimensional requirements.

### 3 Preliminary

KGE method	Scoring Function $f(\mathbf{h}, \mathbf{r}, \mathbf{t})$
TransE (Bordes et al., 2013)	$-\ \mathbf{h} + \mathbf{r} - \mathbf{t}\ $
SimplE (Kazemi and Poole, 2018)	$\frac{1}{2}(\langle \mathbf{h}^H, \mathbf{r}, \mathbf{t}^T \rangle + \langle \mathbf{t}^H, \mathbf{r}^{-1}, \mathbf{h}^T \rangle)$
RotatE (Sun et al., 2019b)	$-\ \mathbf{h} \circ \mathbf{r} - \mathbf{t}\ $
PairRE (Chao et al., 2021)	$-\ \mathbf{h} \circ \mathbf{r}^H - \mathbf{t} \circ \mathbf{r}^T\ $

Table 1: Score functions.

Knowledge graph embedding (KGE) methods use a scoring function  $f$  to represent entities and relations in a continuous vector space. Given a KG  $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$  where  $\mathcal{E}$ ,  $\mathcal{R}$  and  $\mathcal{T}$  are sets of entities, relations and observed triples, for a triple  $(h, r, t)$  ( $h \in \mathcal{E}$ ,  $r \in \mathcal{R}$ ,  $t \in \mathcal{E}$ ), we use the triple scoring function  $s_{(h,r,t)} = f(\mathbf{h}, \mathbf{r}, \mathbf{t})$  (with entity and relation embeddings as input) to measure triple plausibility in the embedding space. Table 1 summarizes scoring functions of some popular KGE methods, where  $\circ$  is the Hadamard product,  $\langle x^1, \dots, x^k \rangle = \sum_i x_i^1 \dots x_i^k$  is the generalized dot product. Higher triple scores mean the model is more likely to consider triples true. The optimiza-

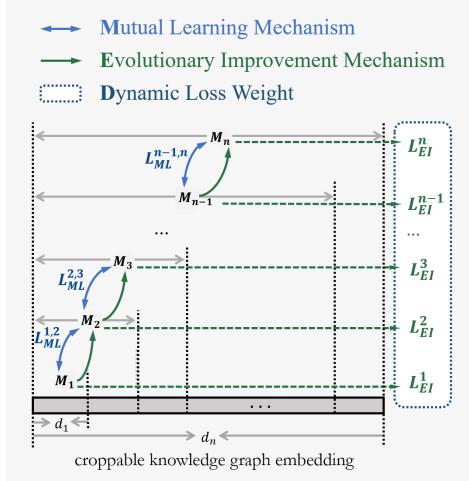


Figure 2: Overview of MED.

tion objective of the KGE model is

$$L_{KGE} = - \sum_{(h,r,t) \in \mathcal{T} \cup \mathcal{T}^-} y \log \sigma(s_{(h,r,t)}) + (1 - y) \log(1 - \sigma(s_{(h,r,t)})), \quad (1)$$

where  $\mathcal{T}^- = \mathcal{E} \times \mathcal{R} \times \mathcal{E} \setminus \mathcal{T}$  is negative triple set,  $\sigma$  is Sigmoid function,  $y$  is ground-truth label of  $(h, r, t)$ ,  $y = 1$  for positive and 0 for negative one.

### 4 MED Framework

As in Fig. 2, our croppable KGE framework MED contains  $n$  sub-models of different dimensions,  $M_i (i = 1, 2, \dots, n)$  with dimension of  $d_i$ . Each  $M_i$  consists of the first  $d_i$  dimensions of the full embedding. The triple  $(h, r, t)$  score from  $M_i$  is  $s_{(h,r,t)}^i = f(\mathbf{h}[0:d_i], \mathbf{r}[0:d_i], \mathbf{t}[0:d_i])$ , where  $\mathbf{h}[0:d_i]$  are the first  $d_i$  elements of vector  $\mathbf{h}$ . The parameters of sub-model  $M_i$  are shared by higher-dimensional  $M_j (i < j \leq n)$ . The number of sub-models  $n$  and each sub-model’s dimension  $d_i$  can be set per actual needs. For low-dimensional sub-models, we aim to maximize performance. High-dimensional sub-models should not only replicate the capabilities of low-dimensional ones but also learn what low-dimensional ones can’t, that is, correctly predicting triples mispredicted by low-dimensional sub-models. MED is based on knowledge distillation (Hinton et al., 2015; Tang et al., 2019; Devlin et al., 2019) technique, where student fits hard (ground-truth) label and soft label from teacher simultaneously. In MED, we first introduce a *mutual learning mechanism*. This helps low-dimensional sub-models learn from high-dimensional ones for better performance, and vice versa, so high-dimensional sub-models retain the

capabilities of low-dimensional ones. Next, we present an *evolutionary improvement mechanism* to let high-dimensional sub-models acquire knowledge that low-dimensional ones can't learn well. Finally, we train MED with *dynamic loss weight* to adaptively balance multiple optimization goals of sub-models.

#### 4.1 Mutual Learning Mechanism

We consider each sub-model  $M_i$  as the student of its higher-dimensional neighbor  $M_{i+1}$  for better performance, as high-dimensional KGEs with more parameters are more expressive (Sachan, 2020; Zhu et al., 2022). We also make  $M_i$  the student of its lower-dimensional neighbor  $M_{i-1}$ , enabling the higher-dimensional sub-model to review and retain the lower-dimensional one's capabilities. So, pairwise neighbor sub-models are both teacher and student, learning from each other. Mutual learning loss between each pair of neighbor sub-models is

$$L_{ML}^{i-1,i} = \sum_{(h,r,t) \in \mathcal{T} \cup \mathcal{T}^-} d_\delta \left( s_{(h,r,t)}^{i-1}, s_{(h,r,t)}^i \right), \quad 1 < i \leq n, \quad (2)$$

where  $s_{(h,r,t)}^i$  is the score of triple  $(h, r, t)$  from sub-model  $M_i$ , indicating the triple's existence likelihood.  $d_\delta$  is Huber loss (Huber and Peter, 1964) with  $\delta = 1$ , often used in KGE knowledge distillation (Zhu et al., 2022). In MED, each sub-model learns only from its neighbor sub-models. This not only cuts down training computational complexity but also keeps the dimension gap between each teacher-student pair relatively small. A large dimension gap between them can undermine the distillation effect (Mirzadeh et al., 2020; Zhu et al., 2022), making our approach crucial and effective.

#### 4.2 Evolutionary Improvement Mechanism

In knowledge distillation (Hinton et al., 2015), the hard (ground-truth) label is a key supervision signal during training. High-dimensional sub-models need to master triples that low-dimensional sub-models can not, that is correctly predicting positive (negative) triples mispredicted as negative (positive) by low-dimensional sub-models. In MED, for a triple  $(h, r, t)$ , sub-model  $M_i$ 's optimization weight for it depends on triple score from  $M_{i-1}$ .

For a positive triple,  $M_i$ 's optimization weight is negatively correlated with the score from  $M_{i-1}$ . If  $M_{i-1}$  gives a high score (correctly identifying it as positive),  $M_i$ 's optimization weight for it is low. If  $M_{i-1}$  gives a low score (misidentifying

it as negative),  $M_i$ 's optimization weight is high as  $M_{i-1}$  can't predict it well. The optimization weight of  $M_i$  for the positive triple is

$$pos_{h,r,t}^i = \begin{cases} 1/|T_{batch}|, & i = 1 \\ \frac{\exp w_1 / s_{(h,r,t)}^{i-1}}{\sum_{(h,r,t) \in \mathcal{T}_b} \exp w_1 / s_{(h,r,t)}^{i-1}}, & 1 < i \leq n, \end{cases} \quad (3)$$

where  $s_{(h,r,t)}^{i-1}$  is the score for triple  $(h, r, t)$  from  $M_{i-1}$ ,  $\mathcal{T}_b$  is positive triple set in a batch, and  $w_1$  is a learnable scaling parameter. Conversely, for a negative triple,  $M_i$ 's optimization weight is positively correlated with the score from  $M_{i-1}$ , as also used in (Sun et al., 2019b). The optimization weight of  $M_i$  for negative triple is

$$neg_{h,r,t}^i = \begin{cases} 1/|T_{batch}^-|, & i = 1 \\ \frac{\exp w_2 \cdot s_{(h,r,t)}^{i-1}}{\sum_{(h,r,t) \in \mathcal{T}_b^-} \exp w_2 \cdot s_{(h,r,t)}^{i-1}}, & 1 < i \leq n, \end{cases} \quad (4)$$

where  $\mathcal{T}_b^-$  is negative triple set in a batch, and  $w_2$  is a learnable scaling parameter. The evolutionary improvement loss of the sub-model  $M_i$  is

$$L_{EI}^i = - \sum_{(h,r,t) \in \mathcal{T} \cup \mathcal{T}^-} pos_{h,r,t}^i \cdot y \log \sigma(s_{(h,r,t)}^i) + neg_{h,r,t}^i \cdot (1 - y) \log(1 - \sigma(s_{(h,r,t)}^i)), \quad (5)$$

In each sub-model, different hard-label loss weights are set for different triples, and high-dimensional sub-models focus more on triples low-dimensional ones can't learn well.

#### 4.3 Dynamic Loss Weight

As MED optimizes multiple sub-models, we use dynamic loss weights during training. At first, low-dimensional sub-models prioritize learning from high-dimensional ones to improve performance, relying more on soft label information. Thus, for them, evolutionary improvement loss should be less than mutual learning loss. Conversely, high-dimensional sub-models should focus more on what low-dimensional ones can't correctly predict and reduce the impact of low-quality low-dimensional outputs. They rely more on hard label information, so their evolutionary improvement loss should exceed mutual learning loss. For a teacher-student pair, the mutual learning loss affects both models equally. We set different evolutionary improvement loss weights for sub-models,

and the final training loss function of MED is

$$L = \sum_{i=2}^n L_{ML}^{i-1,i} + \sum_{i=1}^n \exp\left(\frac{w_3 \cdot d_i}{d_n}\right) \cdot L_{EI}^i, \quad (6)$$

where  $w_3$  is a learnable scaling parameter, and  $d_i$  is the dimension of the  $i$ th sub-model.

## 5 Experiment

We evaluate MED on typical KGE and GLUE benchmarks and address these research questions: **(RQ1)** Can MED train a croppable KGE at once, with multiple differently-dimensional sub-models cropped from it, all performing well? **(RQ2)** Can MED yield parameter-efficient KGE models? **(RQ3)** Does MED work in real-world applications? **(RQ4)** Can MED be extended to non-KGE neural networks?

### 5.1 Experiment Setting

#### 5.1.1 Dataset and KGE methods

MED is universal and applicable to any KGE method with a triple score function. We take 4 common KGE methods in Table 1 as examples: TransE (Bordes et al., 2013), Simple (Kazemi and Poole, 2018), RotatE (Sun et al., 2019b) and PairRE (Chao et al., 2021).

Dataset	#Ent.	#Rel.	#Train	#Valid	#Test
WN18RR	40,943	11	86,835	3,034	3,134
FB15K237	14,541	237	272,115	17,535	20,466
CoDEx-L	77,951	69	551,193	30,622	30,622
YAGO3-10	123,143	37	1,079,040	4,978	4,982
SKG	6,974,959	15	50,775,620	-	-

Table 2: Statistics of datasets.

We conduct comparison experiments on four KG completion benchmark datasets: two common ones, WN18RR (Toutanova et al., 2015) and FB15K237 (Dettmers et al., 2018), and two larger-scale ones, CoDEx-L (Safavi and Koutra, 2020) and YAGO3-10 (Mahdisoltani et al., 2015). In real-world scenarios, we apply MED to a large-scale e-commerce social knowledge graph (SKG) from Taobao, which has over 50 million social record triples from about 7 million users. Table 2 presents dataset statistics.

#### 5.1.2 Evaluation Metric

For link prediction task, we adopt standard metrics MRR and Hit@ $k$  ( $k = 1, 3, 10$ ) with filter setting (Bordes et al., 2013). For a test triple  $(h, r, t)$ , we construct candidate triples by replacing  $h$  with all entities and calculate the triple score rank of

$(h, r, t)$  among candidate triples as the head prediction rank  $rank_h$ . Likewise, we get the tail prediction rank  $rank_t$ . We average  $rank_h$  and  $rank_t$  as  $(h, r, t)$ 's final rank. MRR is the mean reciprocal rank of all test triples, Hit@ $k$  is the percentage of test triples with rank  $\leq k$ . We use  $Effi=MRR/\#P$  ( $\#P$  is the parameter number) (Chen et al., 2023) to measure models' parameter efficiency. We use f1-score and accuracy for user labeling task, and normalized discounted cumulative gain ndcg@ $k$  ( $k = 5, 10$ ) for product recommendation task.

#### 5.1.3 Implementation

For the link prediction task, we set  $d_n = 640$  for the highest-dimensional sub-model  $M_n$  and  $d_1 = 10$  for the lowest-dimensional sub-model  $M_1$ . We set  $n = 64$  and the dimension gap 10 for every pair of neighbor sub-models. There are a total of 64 available sub-models of different dimensions from 10 to 640 in our croppable KGE model. The dimension of  $M_i$  ( $i = 1, 2, \dots, 64$ ) is  $10 \times i$ . For the user labeling and product recommendation task, we set  $n = 3$  and train the croppable KGE containing 3 sub-models:  $M_1$  with  $d_1 = 10$  for mobile phone (MB) terminals that are limited by storage and computing resources,  $M_2$  with  $d_2 = 100$  for the personal computer (PC), and  $M_3$  with  $d_3 = 500$  for the platform's servers. We initialize the learnable scaling parameters  $w_1, w_2$  and  $w_3$  in (3), (4) and (6) to 1. We implement MED by extending OpenKE (Han et al., 2018), an open-source KGE framework based on PyTorch. We set the batch size to 1024 and the maximum training epoch to 3000 with early stopping. For each positive triple, we generate 64 negative triples by randomly replacing its head or tail entity with another entity. We use Adam (Kingma and Ba, 2015) optimizer with a linear decay learning rate scheduler and perform a search on the initial learning rate in  $\{0.0001, 0.0005, 0.001, 0.01\}$ . We train all sub-models simultaneously by optimizing the uniformly sampled sub-models from the full Cropable model in each step.

#### 5.1.4 Baselines

For each required dimension  $d_r$ , we extract the first  $d_r$  dimensions from our croppable KGE as the target model and compare it to the KGE models obtained by 8 baselines of the following 3 types:

- Directly training the target KGE model of requirement dimension  $d_r$ , referred to as **1) DT**.

The directly trained highest-dimensional KGE model ( $d_r = d_n$ ) is denoted as  $M_{max}^{DT}$ .

- Extracting the first  $d_r$  dimensions from  $M_{max}^{DT}$  as the target model, called **2) Ext**. We update  $M_{max}^{DT}$  by arranging 640 dimensions in descending order based on their importance before extracting as (Molchanov et al., 2017; Voita et al., 2019): **3) Ext-L**, the importance for each dimension of  $M_{max}^{DT}$  is the variation of KGE loss on validation set after removing it; and **4) Ext-V**, the importance for each dimension is the average absolute of parameter weights of all entities and relations.
- Distilling the target KGE by KD methods: **5) BKD** (Hinton et al., 2015) is the most basic one by minimizing the KL divergence of the output distributions of teacher and student; **6) TA** (Mirzadeh et al., 2020) uses a medium-size teaching assistant (TA) model as a bridge for size gap, where TA model has the same dimension as the directly trained one whose MRR is closest to the average MRR of teacher and student. We also compare with two KD methods proposed for KGE, which have similar configurations to ours, i.e. compressing high-dimensional teacher into low-dimensional student: **7) DualDE** (Zhu et al., 2022) considers the mutual influences between teacher and student and optimizes them simultaneously; **8) IterDE** (Liu et al., 2023) enables the KGE model to alternately act as student and teacher so that knowledge can be transferred smoothly between high-dimensional teacher and low-dimensional student. In these baselines,  $M_{max}^{DT}$  is the teacher, and other settings including hyperparameters are the same as their original papers.

## 5.2 Performance Comparison

We report the link prediction results of some representative dimensions in Table 3, more results of other dimensions and metrics are in Appendix A and the ablation studies are in Appendix B.

MED outperforms baselines in most settings, especially at extremely low dimensions. On WN18RR with  $d=10$ , MED achieves an improvement of **14.9%** and **15.1%** on TransE, **8.4%** and **6.6%** on RotatE, **29.4%** and **10.6%** on PairRE than the best MRR and Hit@10 of baselines. A similar trend is seen on FB15K237. This success stems from MED’s rich knowledge sources for low-dimensional models: For sub-model  $M_i$ ,

$M_{i+1}$  is a direct teacher, and  $M_{i+2}$  can indirectly influence  $M_i$  via  $M_{i+1}$ . In theory, all higher-dimensional sub-models can transfer knowledge to lower-dimensional ones through stepwise propagation. Although this propagation may harm high-dimensional models with low-quality knowledge from low-dimensional ones, MED’s evolutionary improvement mechanism mitigates the damage, allowing high-dimensional models to compete with directly trained KGEs (Fig. 3). We also note that Ext-based methods are highly unstable, performing worse than DT in most cases, suggesting that dimension importance alone doesn’t ensure sub-model performance.

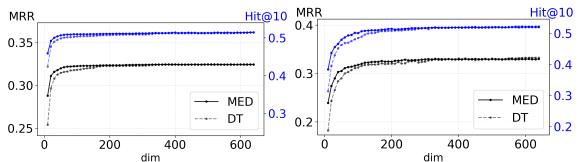


Figure 3: Results of different dimensions for PairRE on WN18RR (left) and FB15K237 (right).

## 5.3 Parameter efficiency of MED

In Table 4, we compare our sub-models of suitable low dimensions to parameter-efficient KGEs especially proposed for large-scale KGs including NodePiece (Galkin et al., 2022) and EARL (Chen et al., 2023). With a similar number of model parameters, MED’s sub-models outperform specialized parameter-efficient KGE methods, showing their parameter efficiency. More crucially, it can offer different-sized parameter-efficient models for applications.

## 5.4 MED in real applications

We apply the trained cropable KGE with TransE on SKG to three real-world applications: user labeling on servers and product recommendation on PCs and mobile phones. Table 5 shows that our cropable user embeddings outperform all baselines, including directly trained (DT), the best baseline DualDE, and a common dimension reduction method in industry principal components analysis (PCA) on  $M_{max}^{DT}$ . Notably, the strong performance in the mobile phone task (constrained by storage and computing resources to a max embedding dimension of 10) highlights the great practical value of MED. More application details are in Appendix C.

## 5.5 Extend MED to Neural Networks

To test our method’s extensibility to other neural networks, we use BERT (Devlin et al., 2019) as

KGE	Method	WN18RR								FB15K237							
		10d		40d		160d		640d		10d		40d		160d		640d	
		MRR	H10														
TransE	DT	0.121	0.287	0.214	0.496	0.233	0.531	0.237	0.537	0.150	0.235	0.299	0.477	0.315	0.499	0.322	<b>0.508</b>
	Ext	0.125	0.298	0.199	0.468	0.225	0.515	0.237	0.537	0.115	0.211	0.236	0.392	0.286	0.462	0.322	0.508
	Ext-L	0.139	0.315	0.224	0.497	0.236	<b>0.534</b>	0.237	0.537	0.109	0.194	0.232	0.381	0.285	0.462	0.322	0.508
	Ext-V	0.139	0.309	0.222	0.494	0.236	0.532	0.237	0.537	0.139	0.256	0.237	0.396	0.293	0.466	0.322	0.508
	BKD	0.141	0.323	0.226	0.513	0.233	0.531	-	-	0.176	0.293	0.303	0.480	0.315	0.501	-	-
	TA	0.144	0.335	0.226	0.512	0.234	0.533	-	-	0.175	0.246	0.303	0.484	0.319	0.504	-	-
	DualDE	0.148	0.337	0.225	0.514	0.235	0.533	-	-	0.179	0.301	0.306	0.483	0.319	0.505	-	-
Simple	IterDE	0.143	0.332	0.224	0.511	0.236	0.531	-	-	0.176	0.285	0.307	0.482	0.317	0.505	-	-
	MED	<b>0.170</b>	<b>0.388</b>	<b>0.232</b>	<b>0.518</b>	<b>0.236</b>	<b>0.529</b>	<b>0.237</b>	<b>0.537</b>	<b>0.196</b>	<b>0.341</b>	<b>0.308</b>	<b>0.486</b>	<b>0.320</b>	<b>0.505</b>	<b>0.322</b>	0.507
Simple	DT	0.061	0.126	0.316	0.389	0.409	0.459	0.421	0.481	0.097	0.179	0.236	0.390	0.285	0.458	<b>0.295</b>	<b>0.472</b>
	Ext	0.004	0.007	0.160	0.249	0.357	0.401	0.421	0.481	0.037	0.068	0.090	0.144	0.229	0.372	0.295	0.472
	Ext-L	0.005	0.006	0.169	0.244	0.398	0.454	0.421	0.481	0.045	0.059	0.083	0.146	0.196	0.316	0.295	0.472
	Ext-V	0.004	0.006	0.246	0.317	0.398	0.461	0.421	0.481	0.049	0.069	0.105	0.149	0.224	0.369	0.295	0.472
	BKD	0.075	0.156	0.343	0.399	0.414	0.468	-	-	0.113	0.204	0.244	0.412	0.287	0.463	-	-
	TA	0.089	0.189	0.368	0.418	0.415	0.472	-	-	0.124	0.221	0.254	0.416	0.290	0.465	-	-
	DualDE	0.083	0.175	<b>0.386</b>	<b>0.423</b>	<b>0.419</b>	<b>0.475</b>	-	-	0.120	0.213	0.258	<b>0.429</b>	0.293	0.466	-	-
RotatE	IterDE	0.077	0.162	0.375	0.419	0.416	0.469	-	-	0.120	0.215	0.257	0.427	<b>0.293</b>	0.465	-	-
	MED	<b>0.111</b>	<b>0.224</b>	0.385	<b>0.431</b>	0.418	<b>0.477</b>	<b>0.421</b>	<b>0.482</b>	<b>0.143</b>	<b>0.267</b>	<b>0.261</b>	0.427	0.291	<b>0.466</b>	0.294	0.470
PairRE	DT	0.172	0.418	0.456	0.556	0.471	0.567	0.476	0.575	0.254	0.424	0.312	0.495	0.322	0.506	<b>0.325</b>	<b>0.515</b>
	Ext	0.299	0.378	0.437	0.516	0.467	0.549	0.476	<b>0.575</b>	0.138	0.245	0.251	0.410	0.291	0.465	0.325	0.515
	Ext-L	0.206	0.277	0.399	0.487	0.445	0.541	0.476	0.575	0.135	0.243	0.221	0.365	0.280	0.453	0.325	0.515
	Ext-V	0.261	0.377	0.337	0.471	0.416	0.532	0.476	0.575	0.160	0.281	0.238	0.393	0.288	0.458	0.325	0.515
	BKD	0.175	0.434	0.457	0.556	0.472	0.570	-	-	0.277	0.442	0.314	0.503	0.322	0.510	-	-
	TA	0.177	0.438	0.459	0.558	0.473	0.572	-	-	0.280	0.447	0.313	0.501	0.323	0.510	-	-
	DualDE	0.179	0.440	0.462	0.559	<b>0.473</b>	0.573	-	-	0.282	0.449	0.315	0.502	0.322	<b>0.512</b>	-	-
PairRE	IterDE	0.176	0.436	0.459	0.560	0.471	0.569	-	-	0.276	0.445	0.317	0.504	0.323	0.512	-	-
	MED	<b>0.324</b>	<b>0.469</b>	<b>0.461</b>	<b>0.561</b>	0.471	<b>0.574</b>	<b>0.476</b>	0.574	<b>0.288</b>	<b>0.459</b>	<b>0.318</b>	<b>0.504</b>	<b>0.323</b>	0.510	0.324	0.514

Table 3: MRR and Hit@10 (H10) of some dimensions on WN18RR (WN) and FB15K237 (FB).

	FB15k-237				WN18RR				CoDeEx-L				YAGO3-10							
	Dim	#P(M)	MRR	Hit@10	Efфи	Dim	#P(M)	MRR	Hit@10	Efфи	Dim	#P(M)	MRR	Hit@10	Efфи	Dim	#P(M)	MRR	Hit@10	Efфи
RotatE	1000	29.3	0.336	0.532	0.011	500	40.6	0.508	0.612	0.013	500	78	0.258	0.387	0.003	500	123.2	0.495	0.670	0.004
RotatE	100	2.9	0.296	0.473	0.102	50	4.1	0.411	0.429	0.100	25	3.8	0.196	0.322	0.052	20	4.8	0.121	0.262	0.025
+ NodePiece	100	3.2	0.256	0.420	0.080	100	4.4	0.403	0.515	0.092	100	3.6	0.190	0.313	0.053	100	4.1	0.247	0.488	0.060
+ EARL	150	1.8	0.310	0.501	0.172	200	3.8	0.440	0.527	0.116	100	2.1	0.238	<b>0.390</b>	<b>0.113</b>	100	3	0.302	0.498	<b>0.101</b>
+ MED	40	1.2	<b>0.318</b>	<b>0.504</b>	<b>0.265</b>	40	3.2	<b>0.466</b>	<b>0.561</b>	<b>0.146</b>	20	3.1	<b>0.243</b>	0.385	0.078	20	4.9	<b>0.313</b>	<b>0.528</b>	0.064

Table 4: Link prediction results on WN18RR, FB15K237, CoDeEx-L and YAGO3-10.

Method	train time	User Labeling		Product Recommendation	
		server (500d)	PC terminal (100d)	PC terminal (10d)	MP terminal (10d)
DT	103h	0.889	0.874	0.411	0.441
PCA	-	-	-	0.417	0.447
DualDE	195h	-	-	0.423	0.456
MED	53h	<b>0.893</b>	<b>0.879</b>	<b>0.431</b>	<b>0.465</b>
				<b>0.422</b>	<b>0.451</b>

Table 5: Results on SKG.

an example. We use distillation methods based on Hugging Face Transformers (Wolf et al., 2020) as baselines. As in previous works(Sun et al., 2019a; Tang et al., 2019; Jung et al., 2023; Zhou et al., 2022a), we perform distillation at the fine-tuning stage. See Appendix D for more details. Table 6 shows the results on the GLUE development set(Wang et al., 2019). We compare MED with other KD models of similar speedup or parameter number. MED performs competitively on most tasks against BERT-specialized KD methods. Compared to HAT (Wang et al., 2020a), which has a similar architecture to ours, MED’s sub-models outperform HAT across three parameter levels. Sub-models with 54M, 17.5M, and 6.36M parameters show average improvements of 16.3%, 21.7% and 19.7% respectively.

## 5.6 Analysis of MED

### 5.6.1 Training efficiency

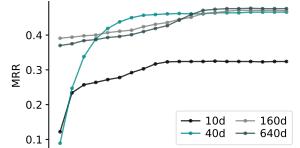


Figure 4: Sub-models’ MRR during training on WN18RR with RotatE.

Table 7 reports the training time of getting 64 models of various sizes ( $d=10, 20, \dots, 640$ ). Figure 4 shows MRR change of MED’s sub-models during training. For fair comparison, all training is done on a single NVIDIA Tesla A100 40GB GPU. For DT, the training time is the sum of directly training 64 KGE models sequentially. Ext-based baselines have the same training time, equal to that of training a  $d_n$ -dimensional KGE model as dimension arrangement time is negligible. KD-based baselines’ training time is the sum of training a  $d_n$ -dimensional teacher model and distilling 63 student models. On FB15K237, we don’t train all 63 sizes of student models for TA, DualDE, and IterDE, estimated to take over 400 hours per KGE

Method	#P(M)	Speedup	MNLI-m acc.	MNLI-mm acc.	MRPC fl/acc.	QNLI acc.	QQP fl/acc.	RTE acc.	STS-2 acc.	STS-B pear./spear.
BERT <sub>Base</sub> <sup>†</sup>	110	1.0×	84.4	85.3	88.6/84.1	89.7	89.6/91.1	67.5	92.5	88.8/88.5
BERT <sub>6</sub> -BKD	66	2.0×	82.2	82.9	86.2/80.8	88.5	88.0/91.0	65.4	90.9	88.2/87.8
BERT <sub>6</sub> -PKD	66	2.0×	82.3	82.6	86.4/81.0	88.6	87.9/91.0	63.9	90.8	88.5/88.1
BERT <sub>6</sub> -MiniLM	66	2.0×	82.2	82.6	84.6/78.1	<b>89.5</b>	87.2/90.5	61.5	90.2	87.8/87.5
BERT <sub>6</sub> -RKD	66	2.0×	82.4	82.9	86.9/81.8	88.9	88.1/ <b>91.2</b>	65.2	91.0	88.4/88.1
BERT <sub>6</sub> -FSD	66	2.0×	82.4	83.0	87.1/82.2	89.0	88.1/ <b>91.2</b>	66.6	91.0	<b>88.7/88.3</b>
BERT <sub>4</sub> -BKD	55	2.9×	80.5	80.9	87.2/83.1	87.5	86.6/90.4	65.2	90.2	84.5/84.2
BERT <sub>4</sub> -PKD	55	2.9×	80.9	81.3	87.0/82.9	87.7	86.8/90.5	66.1	90.5	84.3/84.0
BERT <sub>4</sub> -MetaDistil	55	2.9×	82.4	82.7	<b>88.4/84.2</b>	88.6	87.8/90.8	<b>67.8</b>	91.8	86.3/86.0
BERT-HAT <sup>†</sup>	54	2.0×	70.8	71.6	81.2/74.8	65.3	76.1/80.4	52.7	84.3	79.6/80.1
BERT-MED	54	2.0×	<b>82.7</b>	<b>83.3</b>	88.0/84.0	86.8	<b>89.1/90.7</b>	67.2	<b>91.9</b>	87.6/87.2
BERT-HAT <sup>†</sup>	17.5	4.7×	63.6	64.2	68.4/78.4	61.1	69.0/79.7	47.2	82.9	74.1/75.8
BERT-MED	17.5	4.7×	81.2	82.4	86.1/82.0	86.4	83.8/86.2	64.6	88.2	86.1/86.4
BERT-HAT <sup>†</sup>	6.36	5.2×	59.9	60.0	66.5/77.3	60.1	66.5/77.1	46.2	81.7	71.9/70.4
BERT-MED	6.36	5.2×	72.6	73.7	84.1/78.1	86.0	79.6/82.7	61.7	86.9	82.8/81.6

Table 6: Results on the dev set of GLUE. The results of knowledge distillation methods for BERT<sub>4</sub> and BERT<sub>6</sub> are reported by (Jung et al., 2023; Zhou et al., 2022a) and the <sup>†</sup> results reported by us.

	TransE	Simple	RotatE	PairRE
WN TA	DT 74.0 (9.49×)	68.0 (12.14×)	141.0 (11.10×)	67.4 (10.06×)
	Ext-based 1.5 (0.19×)	1.3 (0.23×)	2.5 (0.20×)	1.6 (0.24×)
	BKD 91.5 (11.73×)	72.0 (12.86×)	163.0 (12.83×)	87.5 (13.06×)
	DualDE 172.0 (22.05×)	142.0 (25.36×)	272.0 (21.42×)	166.0 (24.78×)
	IterDE 151.0 (19.36×)	133.0 (23.75×)	240.0 (18.90×)	133.0 (19.85×)
	MED 7.8 (1.00×)	5.6 (1.00×)	12.7 (1.00×)	6.7 (1.00×)
FB	DT 218.0 (10.23×)	179.0 (10.65×)	381.0 (10.73×)	179.0 (9.37×)
	Ext-based 4.7 (0.22×)	5.1 (0.30×)	9.5 (0.27×)	3.7 (0.19×)
	BKD 248.0 (11.64×)	227.0 (13.51×)	443.0 (12.48×)	231.0 (12.09×)
	MED 21.3 (1.00×)	16.8 (1.00×)	35.5 (1.00×)	19.1 (1.00×)

Table 7: Training time (hours).

n	train time	10d		40d		160d		640d	
		MRR	H10	MRR	H10	MRR	H10	MRR	H10
64	12.7h	0.324	0.469	0.466	0.561	0.471	0.574	0.476	0.574
16	6.2h	0.322	0.467	0.465	0.561	0.473	0.575	0.477	0.576
4	3.3h	0.319	0.463	0.463	0.561	0.475	0.577	0.480	0.578

Table 8: Results of different n.

method. Compared to DT, MED speeds up by up to 10× for 4 KGE methods. Ext-based baselines have the shortest training time but poor performance and low practical value. Except for BKD, other KD-based methods need to optimize both student and teacher, increasing train cost.

### 5.6.2 Effect of the number of sub-models

We set the number of different sub-models ( $n=64, 16, 4$ ) for RotatE on WN18RR. Table 8 shows that reducing the number of sub - models boosts high-dimensional ( $d=160$  and  $640$ ) model performance but lowers that of low-dimensional ( $d=10$  and  $40$ ) models. Training efficiency is nearly linearly related to the number of models.

### 5.6.3 Whether high-dimensional sub-models cover low-dimensional ones' capabilities

A high-dimensional model retaining lower-dimensional ones' ability should correctly predict all triples the latter can. We calculate the percentage of test-set triples where if the smallest sub-model correctly predicting a triple is  $M_i$ , all higher-dimensional ones ( $M_{i+1}, M_{i+2}, \dots, M_n$ ) do too.

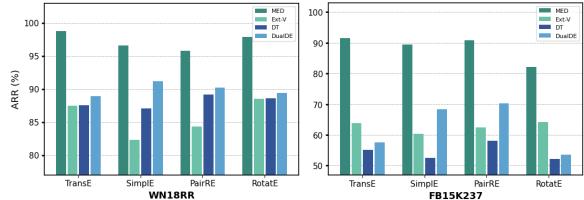


Figure 5: The ability retention ratio (ARR).

This result is the ability retention ratio (ARR). We use Hit@10 to judge correct prediction:  $M_i$  predicts a triple correctly if it scores in the top 10 among candidates. Fig. 5 shows that MED's ARR is is much higher than baselines, especially on FB15K237. This means MED's high-dimensional sub-models cover low-dimensional ones' power, thanks to the mutual learning mechanism for knowledge review. This advantage of MED provides a simple way to judge triple difficulty for KGE methods: triples low-dimensional sub-models predict may be easy as high-dimensional ones can too, while those only predicted by high-dimensional sub-models are difficult.

## 6 Conclusion

In this work, we propose a novel KGE training framework, MED, which trains a croppable KGE in one go. Sub-models of various dimensions can be cropped from it and used directly without extra training. In MED, we introduce the mutual learning mechanism to improve low-dimensional sub-models and make the high-dimensional sub-models retain low-dimensional ones' ability, the evolutionary improvement mechanism to prompt high-dimensional sub-models to learn what low-dimensional ones can't, and dynamic loss weights to balance multiple losses adaptively. Experimental results demonstrate MED's effectiveness, high efficiency, and flexible extensibility.

## 7 Limitations

As a research paper, we are unable to conduct experiments on all knowledge graph embedding (KGE) methods. To demonstrate the generality of the method proposed in this paper, we selected representative models from different types of KGE methods, including distance-based TransE, rotation-based RotatE, semantic matching-based SimpleE, and subrelation encoding-based PairRE. In fact, there are many KGE methods that outperform those used in our experiments. However, the method proposed in this paper is applicable to any KGE method with a triple scoring function.

## 8 Acknowledgements

This work is founded by National Natural Science Foundation of China (NSFCU23B2055/NSFCU19B2027/NSFC62306276), Zhejiang Provincial Natural Science Foundation of China (No. LQ23F020017), Yongjiang Talent Introduction Programme (2022A-238-G), and Fundamental Research Funds for the Central Universities (226-2023-00138). This work was supported by AntGroup.

## References

- Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jin Jin, Xin Jiang, Qun Liu, Michael R. Lyu, and Irwin King. 2021. Binarybert: Pushing the limit of BERT quantization. In *ACL/IJCNLP (1)*, pages 4334–4348. Association for Computational Linguistics.
- Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795.
- Linlin Chao, Jianshan He, Taifeng Wang, and Wei Chu. 2021. Pairre: Knowledge graph embeddings via paired relation vectors. In *ACL/IJCNLP (1)*, pages 4360–4369. Association for Computational Linguistics.
- Mingyang Chen, Wen Zhang, Zhen Yao, Yushan Zhu, Yang Gao, Jeff Z. Pan, and Huajun Chen. 2023. Entity-agnostic representation learning for parameter-efficient knowledge graph embedding. In *AAAI*, pages 4182–4190. AAAI Press.
- Alexis Conneau and Douwe Kiela. 2018. *Senteval: An evaluation toolkit for universal sentence representations*. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA).
- Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *I-SEMANTICS*, pages 121–124. ACM.
- Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*, pages 1811–1818. AAAI Press.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics.
- Dennis Diefenbach, Kamal Deep Singh, and Pierre Maret. 2018. Wdaqua-core1: A question answering service for RDF knowledge bases. In *WWW (Companion Volume)*, pages 1087–1091. ACM.
- William B. Dolan and Chris Brockett. 2005. *Automatically constructing a corpus of sentential paraphrases*. In *Proceedings of the Third International Workshop on Paraphrasing, IWP@IJCNLP 2005, Jeju Island, Korea, October 2005, 2005*. Asian Federation of Natural Language Processing.
- Cunhang Fan, Yujie Chen, Jun Xue, Yonghui Kong, Jianhua Tao, and Zhao Lv. 2024. Progressive distillation based on masked generation feature method for knowledge graph completion. In *AAAI*, pages 8380–8388. AAAI Press.
- Mikhail Galkin, Etienne G. Denis, Jiapeng Wu, and William L. Hamilton. 2022. Nodepiece: Compositional and parameter-efficient representations of large knowledge graphs. In *ICLR*. OpenReview.net.
- Xu Han, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. 2018. Openke: An open toolkit for knowledge embedding. In *EMNLP (Demonstration)*, pages 139–144. Association for Computational Linguistics.
- Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*, pages 173–182. ACM.
- Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.
- Huber and J. Peter. 1964. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101.
- Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL (1)*, pages 687–696. The Association for Computer Linguistics.
- Hee-Jun Jung, Doyeon Kim, Seung-Hoon Na, and Kangil Kim. 2023. Feature structure distillation with centered kernel alignment in BERT transferring. *Expert Syst. Appl.*, 234:120980.

- Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *NeurIPS*, pages 4289–4300.
- Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.
- Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187. AAAI Press.
- Jiajun Liu, Wenjun Ke, Peng Wang, Ziyu Shang, Jinhua Gao, Guozheng Li, Ke Ji, and Yanhe Liu. 2024. Towards continual knowledge graph embedding via incremental distillation. In *AAAI*, pages 8759–8768. AAAI Press.
- Jiajun Liu, Peng Wang, Ziyu Shang, and Chenxiao Wu. 2023. Iterde: An iterative knowledge distillation framework for knowledge graph embeddings. In *AAAI*, pages 4488–4496. AAAI Press.
- Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. 2015. YAGO3: A knowledge base from multilingual wikipedias. In *CIDR*. www.cidrdb.org.
- Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. 2020. Improved knowledge distillation via teacher assistant. In *AAAI*, pages 5191–5198. AAAI Press.
- Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2017. Pruning convolutional neural networks for resource efficient inference. In *ICLR (Poster)*. OpenReview.net.
- Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Q. Phung. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In *NAACL-HLT (2)*, pages 327–333. Association for Computational Linguistics.
- Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*, pages 809–816. Omnipress.
- Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. 2019. Relational knowledge distillation. In *CVPR*, pages 3967–3976. Computer Vision Foundation / IEEE.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. **Squad: 100, 000+ questions for machine comprehension of text**. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.
- Mrinmaya Sachan. 2020. Knowledge graph embedding compression. In *ACL*, pages 2681–2691. Association for Computational Linguistics.
- Tara Safavi and Danai Koutra. 2020. Codex: A comprehensive knowledge graph completion benchmark. In *EMNLP (1)*, pages 8328–8350. Association for Computational Linguistics.
- Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. **Recursive deep models for semantic compositionality over a sentiment treebank**. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1631–1642. ACL.
- Pierre Stock, Angela Fan, Benjamin Graham, Edouard Grave, Rémi Gribonval, Hervé Jégou, and Armand Joulin. 2021. Training with quantization noise for extreme model compression. In *ICLR*. OpenReview.net.
- Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019a. Patient knowledge distillation for BERT model compression. In *EMNLP/IJCNLP (1)*, pages 4322–4331. Association for Computational Linguistics.
- Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019b. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR (Poster)*. OpenReview.net.
- Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from BERT into simple neural networks. *CoRR*, abs/1903.12136.
- Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, pages 1499–1509. The Association for Computational Linguistics.
- Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2071–2080. JMLR.org.
- Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. 2020. Composition-based multi-relational graph convolutional networks. In *ICLR*. OpenReview.net.
- Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *ACL (1)*, pages 5797–5808. Association for Computational Linguistics.

- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. *GLUE: A multi-task benchmark and analysis platform for natural language understanding*. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.
- Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. 2020a. HAT: hardware-aware transformers for efficient natural language processing. In *ACL*, pages 7675–7688. Association for Computational Linguistics.
- Haoyu Wang, Yaqing Wang, Defu Lian, and Jing Gao. 2021a. A lightweight knowledge graph embedding framework for efficient inference and storage. In *CIKM*, pages 1909–1918. ACM.
- Kai Wang, Yu Liu, Qian Ma, and Quan Z. Sheng. 2021b. Mulde: Multi-teacher knowledge distillation for low-dimensional knowledge graph embeddings. In *WWW*, pages 1716–1726. ACM / IW3C2.
- Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020b. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.
- Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119. AAAI Press.
- Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierrick Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. *Transformers: State-of-the-art natural language processing*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45. Association for Computational Linguistics.
- Canran Xu and Ruijiang Li. 2019. Relation embedding with dihedral group in knowledge graph. In *ACL (1)*, pages 263–272. Association for Computational Linguistics.
- Haotian Xu, Yuhua Wang, and Jiahui Fan. 2024. Self-knowledge distillation for knowledge graph embedding. In *LREC/COLING*, pages 14595–14605. ELRA and ICCL.
- Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR (Poster)*.
- Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embeddings. In *NeurIPS*, pages 2731–2741.
- Wen Zhang, Chi Man Wong, Ganqiang Ye, Bo Wen, Wei Zhang, and Huajun Chen. 2021. Billion-scale pre-trained e-commerce product knowledge graph model. In *ICDE*, pages 2476–2487. IEEE.
- Wangchunshu Zhou, Canwen Xu, and Julian J. McAuley. 2022a. *BERT learns to teach: Knowledge distillation with meta learning*. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 7037–7049. Association for Computational Linguistics.
- Zhehui Zhou, Defang Chen, Can Wang, Yan Feng, and Chun Chen. 2022b. Improving knowledge graph embedding via iterative self-semantic knowledge distillation. *CoRR*, abs/2206.02963.
- Yushan Zhu, Wen Zhang, Mingyang Chen, Hui Chen, Xu Cheng, Wei Zhang, and Huajun Chen. 2022. Du-alde: Dually distilling knowledge graph embedding for faster and cheaper reasoning. In *WSDM*, pages 1516–1524. ACM.
- Yushan Zhu, Huaixiao Zhao, Wen Zhang, Ganqiang Ye, Hui Chen, Ningyu Zhang, and Huajun Chen. 2021. Knowledge perceived multi-modal pretraining in e-commerce. In *ACM Multimedia*, pages 2744–2752. ACM.

## A More Results of link prediction

More results of link prediction are shown in Table 9 and Table 10 for WN18RR, and Table 11 and Table 12 for FB15K237. All comparison results of sub-models of MED to the directly trained KGEs (DT) of 10- to 640-dimension are shown in Fig. 6.

## B Ablation Study

We conduct ablation studies to evaluate the effect of three modules in MED: the mutual learning mechanism (MLM), the evolutionary improvement mechanism (EIM), and the dynamic loss weight (DLW). Table 13 shows the MRR and Hit@ $k$  ( $k = 1, 3, 10$ ) of MED removing these modules respectively on WN18RR and TransE.

	Method	10d		20d		40d		80d		160d		320d		640d	
		MRR	Hit@10												
TransE	DT	0.121	0.287	0.176	0.453	0.214	0.496	0.227	0.524	0.233	0.531	0.235	0.534	<b>0.237</b>	<b>0.537</b>
	Ext	0.125	0.298	0.172	0.423	0.199	0.468	0.213	0.495	0.225	0.515	0.226	0.521	0.237	0.537
	Ext-L	0.139	0.315	0.196	0.461	0.224	0.497	0.232	0.516	0.236	<b>0.534</b>	0.236	0.535	0.237	0.537
	Ext-V	0.139	0.309	0.198	0.458	0.222	0.494	0.234	0.525	0.236	0.532	0.236	0.536	0.237	0.537
	BKD	0.141	0.323	0.207	0.480	0.226	0.513	0.232	0.527	0.233	0.531	0.236	0.533	-	-
	TA	0.144	0.335	0.211	0.483	0.226	0.512	0.233	0.527	0.234	0.533	0.236	0.535	-	-
	DualDE	0.148	0.337	0.213	0.488	0.225	0.514	<b>0.234</b>	<b>0.530</b>	0.235	0.533	<b>0.238</b>	0.535	-	-
	IterDE	0.143	0.332	0.211	0.484	0.224	0.511	0.232	0.528	0.236	0.531	0.237	0.533	-	-
SimpleE	MED	<b>0.170</b>	<b>0.388</b>	<b>0.219</b>	<b>0.491</b>	<b>0.232</b>	<b>0.518</b>	0.232	0.523	<b>0.236</b>	0.529	0.237	<b>0.536</b>	0.237	0.537
	Method	MRR	Hit@10												
	DT	0.061	0.126	0.257	0.372	0.316	0.389	0.382	0.446	0.409	0.459	0.417	0.474	<b>0.421</b>	0.481
	Ext	0.004	0.007	0.051	0.107	0.160	0.249	0.219	0.314	0.357	0.401	0.407	0.451	0.421	0.481
	Ext-L	0.005	0.006	0.048	0.078	0.169	0.244	0.369	0.435	0.398	0.454	0.417	0.481	0.421	0.481
	Ext-V	0.004	0.006	0.047	0.076	0.246	0.317	0.368	0.402	0.398	0.461	0.413	0.472	0.421	0.481
	BKD	0.075	0.156	0.285	0.381	0.343	0.399	0.394	0.450	0.414	0.468	0.418	0.475	-	-
	TA	0.089	0.189	0.316	0.386	0.368	0.418	0.405	0.456	0.415	0.472	0.421	0.481	-	-
RotatE	DualDE	0.083	0.175	0.328	0.388	<b>0.386</b>	0.423	0.407	0.454	<b>0.419</b>	0.475	<b>0.422</b>	<b>0.482</b>	-	-
	IterDE	0.077	0.162	0.321	0.378	0.375	0.419	0.404	0.452	0.416	0.469	0.421	0.482	-	-
	MED	<b>0.111</b>	<b>0.224</b>	<b>0.335</b>	<b>0.395</b>	0.385	<b>0.431</b>	<b>0.407</b>	<b>0.457</b>	0.418	<b>0.477</b>	0.421	0.481	0.421	<b>0.482</b>
	Method	MRR	Hit@10												
	DT	0.172	0.418	0.409	0.504	0.456	0.556	0.465	0.564	0.471	0.567	0.474	0.573	<b>0.476</b>	<b>0.575</b>
	Ext	0.299	0.378	0.379	0.464	0.437	0.516	0.458	0.544	0.467	0.549	0.471	0.552	0.476	0.575
	Ext-L	0.206	0.277	0.336	0.424	0.399	0.487	0.423	0.515	0.445	<b>0.541</b>	0.466	0.564	0.476	0.575
	Ext-V	0.261	0.377	0.304	0.433	0.337	0.471	0.366	0.497	0.416	0.532	0.451	0.561	0.476	0.575
PairRE	BKD	0.175	0.434	0.424	0.540	0.457	0.556	0.471	0.565	0.472	0.570	0.474	0.572	-	-
	TA	0.177	0.438	0.424	0.542	0.459	0.558	0.470	0.567	0.473	0.572	0.474	0.572	-	-
	DualDE	0.179	0.440	0.425	0.542	0.462	0.559	<b>0.471</b>	0.567	<b>0.473</b>	0.573	0.475	0.573	-	-
	IterDE	0.176	0.436	0.421	0.538	0.459	0.560	0.470	0.567	0.471	0.569	0.474	0.572	-	-
	MED	<b>0.324</b>	<b>0.469</b>	<b>0.456</b>	<b>0.543</b>	<b>0.466</b>	<b>0.561</b>	0.471	<b>0.568</b>	0.471	<b>0.574</b>	<b>0.476</b>	<b>0.573</b>	0.476	0.574

Table 9: MRR and Hit@10 of some representative dimensions on WN18RR.

		10d		20d		40d		80d		160d		320d		640d	
	Method	Hit@3	Hit@1												
<b>TransE</b>	DT	0.202	0.011	0.291	0.016	0.385	0.018	0.401	0.025	0.403	0.027	0.407	0.033	0.412	<b>0.034</b>
	Ext	0.201	0.016	0.285	0.023	0.338	0.023	0.364	0.028	0.384	0.033	0.388	0.028	0.412	0.034
	Ext-L	0.218	0.029	0.317	0.025	0.361	0.039	0.403	0.046	0.405	0.036	0.408	0.033	0.412	0.034
	Ext-V	0.218	0.029	0.314	<b>0.045</b>	0.391	<b>0.051</b>	0.407	<b>0.047</b>	0.408	0.036	0.411	0.027	0.412	0.034
	BKD	0.216	0.035	0.331	0.040	0.392	0.033	0.401	0.031	0.404	0.030	0.407	0.032	-	-
	TA	0.224	0.040	0.343	0.043	0.395	0.037	0.408	0.030	0.407	0.030	0.410	0.034	-	-
	DualDE	0.226	0.037	0.346	0.043	0.394	0.037	<b>0.408</b>	0.031	<b>0.408</b>	0.031	<b>0.411</b>	<b>0.034</b>	-	-
	IterDE	0.217	0.032	0.345	0.044	0.392	0.036	0.407	0.030	0.408	0.031	0.407	0.033	-	-
	MED	<b>0.269</b>	<b>0.040</b>	<b>0.369</b>	<b>0.045</b>	<b>0.399</b>	0.038	0.404	0.042	0.407	<b>0.037</b>	0.410	0.033	<b>0.412</b>	0.031
<b>Simple</b>	Method	Hit@3	Hit@1												
	DT	0.061	0.028	0.297	0.193	0.361	0.289	0.406	0.343	0.420	0.382	0.428	0.386	0.433	<b>0.391</b>
	Ext	0.003	0.001	0.055	0.023	0.181	0.114	0.249	0.168	0.377	0.329	0.422	0.381	0.433	0.391
	Ext-L	0.004	0.003	0.051	0.031	0.187	0.128	0.389	0.333	0.413	0.365	0.429	0.384	0.433	0.391
	Ext-V	0.004	0.002	0.050	0.029	0.269	0.205	0.378	0.349	0.409	0.372	0.426	0.382	0.433	0.391
	BKD	0.077	0.034	0.331	0.225	0.384	0.311	0.415	0.358	0.426	0.371	0.431	0.385	-	-
	TA	0.093	0.042	0.349	0.269	0.375	0.349	0.412	<b>0.384</b>	0.425	0.388	0.431	0.389	-	-
	DualDE	0.086	0.038	0.361	0.285	0.391	<b>0.368</b>	0.416	0.383	0.427	<b>0.389</b>	0.434	<b>0.392</b>	-	-
	IterDE	0.079	0.033	0.355	0.279	0.382	0.356	0.415	0.379	0.424	0.383	0.433	0.389	-	-
	MED	<b>0.119</b>	<b>0.048</b>	<b>0.366</b>	<b>0.292</b>	<b>0.395</b>	0.359	<b>0.419</b>	0.380	<b>0.429</b>	0.389	<b>0.435</b>	0.391	<b>0.434</b>	0.390
<b>RotatE</b>	Method	Hit@3	Hit@1												
	DT	0.304	0.005	0.436	0.357	0.475	0.393	0.487	0.420	0.489	0.423	0.491	0.428	0.493	<b>0.429</b>
	Ext	0.315	0.257	0.399	0.335	0.452	0.395	0.472	0.415	0.480	0.413	0.470	0.418	0.493	0.429
	Ext-L	0.224	0.166	0.359	0.288	0.420	0.352	0.441	0.373	0.461	0.396	0.481	0.417	0.493	0.429
	Ext-V	0.289	0.197	0.336	0.234	0.377	0.263	0.402	0.293	0.442	0.357	0.467	0.397	0.493	0.429
	BKD	0.312	0.009	0.452	0.361	0.479	0.403	0.487	0.421	0.490	0.424	0.492	0.425	-	-
	TA	0.314	0.010	0.452	0.363	0.481	0.408	0.489	0.420	0.488	0.422	0.492	0.425	-	-
	DualDE	0.320	0.011	0.452	0.364	0.483	0.412	0.489	<b>0.423</b>	0.488	<b>0.426</b>	0.491	0.425	-	-
	IterDE	0.311	0.013	0.439	0.356	0.479	0.407	0.484	0.423	0.488	0.425	0.493	0.424	-	-
	MED	<b>0.354</b>	<b>0.277</b>	<b>0.476</b>	<b>0.409</b>	<b>0.486</b>	<b>0.418</b>	<b>0.490</b>	0.422	<b>0.492</b>	0.424	<b>0.493</b>	<b>0.427</b>	<b>0.495</b>	0.428
<b>PairRE</b>	Method	Hit@3	Hit@1												
	DT	0.271	0.174	0.368	0.313	0.428	0.384	0.450	0.399	0.463	0.405	0.462	0.406	<b>0.464</b>	<b>0.407</b>
	Ext	0.163	0.120	0.292	0.198	0.366	0.267	0.398	0.314	0.437	0.364	0.452	0.388	0.464	0.407
	Ext-L	0.175	0.129	0.302	0.237	0.383	0.319	0.431	0.377	0.450	0.395	0.455	0.400	0.464	0.407
	Ext-V	0.192	0.124	0.323	0.269	0.407	0.352	0.435	0.379	0.452	0.398	0.458	0.400	0.464	0.407
	BKD	0.279	0.184	0.388	0.334	0.435	0.372	0.452	0.405	0.460	0.405	0.463	0.407	-	-
	TA	0.293	0.197	0.387	0.332	0.437	0.380	0.460	0.404	0.462	0.409	0.463	0.408	-	-
	DualDE	0.281	0.175	0.389	0.330	0.437	0.381	<b>0.463</b>	<b>0.409</b>	0.463	0.410	0.465	<b>0.410</b>	-	-
	IterDE	0.285	0.172	0.390	0.331	0.435	0.377	0.461	0.405	0.463	<b>0.411</b>	0.464	0.410	-	-
	MED	<b>0.314</b>	<b>0.259</b>	<b>0.426</b>	<b>0.367</b>	<b>0.443</b>	<b>0.392</b>	0.462	0.405	<b>0.464</b>	0.406	<b>0.465</b>	0.407	0.464	0.406

Table 10: Hit@3 and Hit@1 of some representative dimensions on WN18RR.

		10d		20d		40d		80d		160d		320d		640d	
	Method	MRR	Hit@10												
TransE	DT	0.150	0.235	0.277	0.440	0.299	0.477	0.313	0.484	0.315	0.499	0.318	0.501	0.322	<b>0.508</b>
	Ext	0.115	0.211	0.191	0.324	0.236	0.392	0.266	0.436	0.286	0.462	0.299	0.479	0.322	0.508
	Ext-L	0.109	0.194	0.175	0.293	0.232	0.381	0.263	0.424	0.285	<b>0.462</b>	0.301	0.484	0.322	0.508
	Ext-V	0.139	0.256	0.200	0.348	0.237	0.396	0.270	0.437	0.293	0.466	0.308	0.488	0.322	0.508
	BKD	0.176	0.293	0.279	0.446	0.303	0.480	0.315	0.500	0.315	0.501	0.320	0.502	-	-
	TA	0.175	0.246	0.281	0.441	0.303	0.484	0.314	0.498	0.319	0.504	0.321	0.504	-	-
	DualDE	0.179	0.301	0.281	0.443	0.306	0.483	0.316	0.502	0.319	0.505	<b>0.322</b>	<b>0.508</b>	-	-
	IterDE	0.176	0.285	0.276	0.446	0.307	0.482	0.315	<b>0.503</b>	0.317	0.505	0.319	0.505	-	-
SimpleE	MED	<b>0.196</b>	<b>0.341</b>	<b>0.290</b>	<b>0.472</b>	<b>0.308</b>	<b>0.486</b>	<b>0.317</b>	0.502	<b>0.320</b>	<b>0.505</b>	0.321	0.507	<b>0.322</b>	0.507
	DT	0.097	0.179	0.176	0.321	0.236	0.390	0.271	0.431	0.285	0.458	0.291	0.467	<b>0.295</b>	<b>0.472</b>
	Ext	0.037	0.068	0.069	0.107	0.090	0.144	0.159	0.258	0.229	0.372	0.269	0.432	0.295	0.472
	Ext-L	0.045	0.059	0.056	0.062	0.083	0.146	0.114	0.205	0.196	0.316	0.258	0.421	0.295	0.472
	Ext-V	0.049	0.069	0.066	0.101	0.105	0.149	0.138	0.224	0.224	0.369	0.261	0.414	0.295	0.472
	BKD	0.113	0.204	0.182	0.315	0.244	0.412	0.275	0.439	0.287	0.463	0.293	0.470	-	-
	TA	0.124	0.221	0.192	0.329	0.254	0.416	0.276	0.448	0.290	0.465	0.295	0.471	-	-
	DualDE	0.120	0.213	0.195	0.346	0.258	<b>0.429</b>	0.279	0.443	<b>0.293</b>	0.466	<b>0.296</b>	<b>0.468</b>	-	-
RotatE	IterDE	0.120	0.215	0.193	0.338	0.257	0.427	0.281	0.440	0.293	0.465	0.297	0.468	-	-
	MED	<b>0.143</b>	<b>0.267</b>	<b>0.233</b>	<b>0.384</b>	<b>0.261</b>	0.427	<b>0.279</b>	<b>0.448</b>	0.291	<b>0.466</b>	0.293	0.468	0.294	0.470
	DT	0.254	0.424	0.297	0.477	0.312	0.495	0.317	0.502	0.322	0.506	0.323	0.510	<b>0.325</b>	<b>0.515</b>
	Ext	0.138	0.245	0.203	0.340	0.251	0.410	0.276	0.443	0.291	0.465	0.305	0.485	0.325	0.515
	Ext-L	0.135	0.243	0.188	0.319	0.221	0.365	0.246	0.402	0.280	0.453	0.299	0.477	0.325	0.515
	Ext-V	0.160	0.281	0.198	0.340	0.238	0.393	0.265	0.427	0.288	0.458	0.302	0.478	0.325	0.515
	BKD	0.277	0.442	0.305	0.485	0.314	0.503	0.321	0.508	0.322	0.510	0.323	0.509	-	-
	TA	0.280	0.447	0.306	0.485	0.313	0.501	0.319	0.507	0.323	0.510	0.323	0.509	-	-
PairRE	DualDE	0.282	0.449	0.307	0.486	0.315	0.502	0.318	0.507	0.322	<b>0.512</b>	0.324	<b>0.514</b>	-	-
	IterDE	0.276	0.445	0.306	0.482	0.317	0.504	0.319	0.508	0.323	0.512	0.324	0.513	-	-
	MED	<b>0.288</b>	<b>0.459</b>	<b>0.311</b>	<b>0.492</b>	<b>0.318</b>	<b>0.504</b>	<b>0.322</b>	<b>0.509</b>	<b>0.323</b>	0.510	<b>0.324</b>	0.512	0.324	0.514

Table 11: MRR and Hit@10 of some representative dimensions on FB15K237.

		10d		20d		40d		80d		160d		320d		640d	
	Method	Hit@3	Hit@1												
TransE	DT	0.169	0.102	0.301	0.190	0.327	0.212	0.340	0.218	0.348	0.222	0.353	0.224	<b>0.358</b>	<b>0.228</b>
	Ext	0.123	0.065	0.211	0.122	0.264	0.156	0.296	0.180	0.320	0.197	0.331	0.208	0.358	0.228
	Ext-L	0.118	0.065	0.192	0.115	0.256	0.157	0.292	0.180	0.316	0.198	0.333	0.210	0.358	0.228
	Ext-V	0.150	0.081	0.222	0.126	0.265	0.156	0.301	0.185	0.325	0.205	0.341	0.217	0.358	0.228
	BKD	0.178	0.106	0.308	0.198	0.336	0.208	0.349	0.222	0.349	0.223	0.354	0.226	-	-
	TA	0.188	0.112	0.307	0.200	0.336	0.212	0.348	0.220	0.353	0.225	0.355	0.223	-	-
	DualDE	0.193	0.115	0.307	<b>0.201</b>	0.337	0.216	<b>0.351</b>	0.223	<b>0.354</b>	0.226	0.356	0.227	-	-
	IterDE	0.187	0.112	0.299	0.185	0.333	0.214	0.351	0.222	0.353	0.223	0.354	0.224	-	-
SimplE	MED	<b>0.215</b>	<b>0.122</b>	<b>0.321</b>	0.199	<b>0.338</b>	<b>0.218</b>	0.347	<b>0.223</b>	0.351	<b>0.226</b>	<b>0.356</b>	<b>0.227</b>	0.358	0.227
	DT	0.103	0.055	0.193	0.105	0.256	0.161	0.297	0.191	0.314	0.197	0.323	0.208	<b>0.324</b>	<b>0.211</b>
	Ext	0.039	0.019	0.071	0.047	0.091	0.057	0.171	0.109	0.251	0.159	0.294	0.187	0.324	0.211
	Ext-L	0.043	0.035	0.048	0.037	0.111	0.040	0.131	0.093	0.216	0.134	0.281	0.177	0.324	0.211
	Ext-V	0.047	0.036	0.074	0.043	0.097	0.077	0.145	0.109	0.248	0.156	0.289	0.189	0.324	0.211
	BKD	0.123	0.064	0.201	0.115	0.261	0.164	0.299	0.191	0.308	0.202	0.318	<b>0.213</b>	-	-
	TA	0.133	0.073	0.210	0.123	0.276	0.175	0.302	0.195	0.318	0.203	0.323	0.211	-	-
	DualDE	0.130	0.071	0.224	0.115	0.279	0.175	0.305	0.196	0.324	<b>0.208</b>	<b>0.326</b>	0.211	-	-
RotatE	IterDE	0.132	0.069	0.217	0.118	0.276	0.174	0.303	0.192	<b>0.326</b>	0.204	0.324	0.212	-	-
	MED	<b>0.164</b>	<b>0.073</b>	<b>0.254</b>	<b>0.158</b>	<b>0.288</b>	<b>0.177</b>	<b>0.305</b>	<b>0.196</b>	0.319	0.205	0.318	0.209	0.322	0.209
	DT	0.284	0.168	0.330	0.207	0.346	0.223	0.352	0.224	0.353	0.229	0.357	0.230	<b>0.363</b>	<b>0.234</b>
	Ext	0.152	0.080	0.225	0.129	0.278	0.170	0.304	0.190	0.322	0.203	0.335	0.217	0.363	0.234
	Ext-L	0.147	0.078	0.209	0.121	0.247	0.146	0.275	0.166	0.312	0.193	0.333	0.209	0.363	0.234
	Ext-V	0.174	0.097	0.218	0.126	0.264	0.159	0.293	0.182	0.319	0.201	0.336	0.213	0.363	0.234
	BKD	0.306	0.193	0.338	0.214	0.352	0.224	0.354	0.230	0.356	0.230	0.358	0.231	-	-
	TA	0.308	0.196	0.339	0.216	0.353	0.225	0.358	0.229	0.359	0.229	0.358	0.231	-	-
PairRE	DualDE	0.311	0.197	0.341	0.216	0.353	<b>0.227</b>	<b>0.360</b>	0.230	0.361	0.232	0.361	0.233	-	-
	IterDE	0.307	0.195	0.342	0.215	0.355	0.225	0.359	<b>0.232</b>	<b>0.363</b>	0.233	0.362	<b>0.234</b>	-	-
	MED	<b>0.324</b>	<b>0.201</b>	<b>0.344</b>	<b>0.216</b>	<b>0.355</b>	0.225	0.357	0.231	0.358	<b>0.233</b>	<b>0.362</b>	0.233	0.362	0.232

Table 12: Hit@3 and Hit@1 of some representative dimensions on FB15K237.

dim	MED				MED w/o MLM				MED w/o EIM				MED w/o DLW			
	MRR	Hit@10	Hit@3	Hit@1	MRR	Hit@10	Hit@3	Hit@1	MRR	Hit@10	Hit@3	Hit@1	MRR	Hit@10	Hit@3	Hit@1
10	.170	<b>.388</b>	<b>.269</b>	.036	.149	.335	.234	.032	.169	.388	.267	<b>.037</b>	<b>.171</b>	.387	.268	.035
20	<b>.219</b>	<b>.491</b>	<b>.369</b>	.042	.197	.437	.323	.032	.217	.488	.366	<b>.044</b>	.218	.487	.367	.039
40	<b>.232</b>	<b>.518</b>	.399	<b>.048</b>	.224	.496	.379	.029	.232	.517	<b>.403</b>	.042	.232	.517	.402	.037
80	.232	.523	.404	<b>.042</b>	.228	.521	.399	.033	<b>.235</b>	<b>.529</b>	.408	.037	.234	.523	<b>.410</b>	.041
160	<b>.236</b>	<b>.529</b>	<b>.407</b>	<b>.037</b>	.234	.525	.406	.034	.234	.527	.405	.032	.235	.527	.405	.032
320	<b>.237</b>	<b>.536</b>	<b>.410</b>	.033	.236	.532	.409	<b>.035</b>	.233	.530	.398	.031	.234	.533	.405	.029
640	.237	.537	.412	.031	<b>.238</b>	.535	.412	<b>.042</b>	.232	.528	.402	.029	.233	.530	.396	.025

Table 13: Ablation study on WN18RR with TransE.

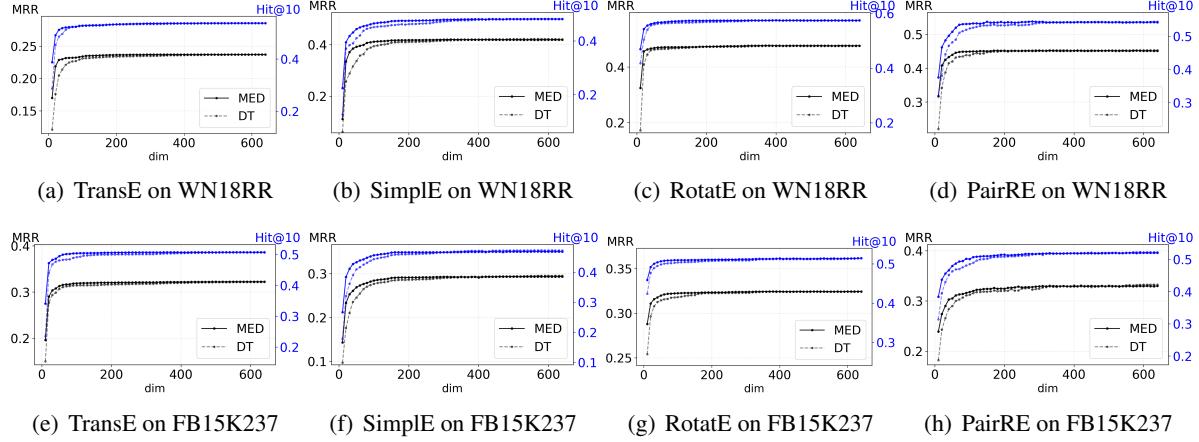


Figure 6: Performance of sub-models of MED and the directly trained (DT) KGEs of dimensions from 10 to 640.

### B.1 Mutual Learning Mechanism (MLM)

We remove the mutual learning mechanism from MED and keep the other parts unchanged, where (6) is rewritten as

$$L = \sum_{i=1}^n \exp\left(\frac{w_3 \cdot d_i}{d_n}\right) \cdot L_{EI}^i. \quad (7)$$

From the result of “MED w/o MLM” in Table 13, we find that after removing the mutual learning mechanism, the performance of low-dimensional sub-models deteriorates seriously since the low-dimensional sub-models can not learn from the high-dimensional sub-models. For example, the MRR of the 10-dimensional sub-model decreased by 12.4%, and the MRR of the 20-dimensional sub-model decreased by 10%. While the performance degradation of the high-dimensional sub-model is not particularly obvious, and the MRR of the highest-dimensional sub-model ( $dim = 640$ ) is not worse than that of MED, which is because to a certain degree, removing the mutual learning mechanism also avoids the negative influence to high-dimensional sub-models from low-dimensional sub-models. On the whole, this mechanism greatly improves the performance of low-dimensional sub-models.

### B.2 Evolutionary Improvement Mechanism (EIM)

In this part, we replace evolutionary improvement loss  $L_{EI}^i$  in (6) with the regular KGE loss  $L_{KGE}^i$ :

$$\begin{aligned} L_{KGE}^i &= \sum_{(h,r,t) \in \mathcal{T} \cup \mathcal{T}^-} y \log \sigma(s_{(h,r,t)}^i) \\ &\quad + (1 - y) \log(1 - \sigma(s_{(h,r,t)}^i)). \end{aligned} \quad (8)$$

From the result of “MED w/o EIM” in Table 13, we find that removing the evolutionary improvement mechanism mainly degrades the performance of high-dimensional sub-models. While due to the existence of the mutual learning mechanism, the low-dimensional sub-model can still learn from the high-dimensional sub-model, so as to ensure the certain performance of the low-dimensional sub-model. In addition, we also find that as the dimension increases to a certain extent, the performance of the sub-model does not improve, and even begins to decline. We guess that this is because the mutual learning mechanism makes every pair of neighbor sub-models learn from each other, resulting in some low-quality or wrong knowledge gradually transferring from the low-dimensional sub-models to the high-dimensional sub-models, and when the evolutionary improvement mechanism is removed, the high-dimensional sub-models can no longer correct the wrong information from the low-dimensional sub-models. The higher the dimension of the sub-model, the more the accumulated error, so the performance of the high-dimensional sub-models is seriously damaged. On the whole, this mechanism mainly helps to improve the effect of high-dimensional sub-models.

### B.3 Dynamic Loss Weight (DLW)

To study the effect of the dynamic loss weight, we fix the ratio of all mutual learning losses to all evolutionary improvement losses as 1 : 1, and (6) is rewritten as

$$L = \sum_{i=2}^n L_{ML}^{i-1,i} + \sum_{i=1}^n L_{EI}^i. \quad (9)$$

According to the result of “MED w/o DLW” in

Table 13, the overall results of “MED w/o DLW” are in the middle of the results of “MED w/o MLM” and “MED w/o EIM”: the performance of the low-dimensional sub-model is better than that of “MED w/o MLM”, and the performance of the high-dimensional sub-model is better than that of “MED w/o EIM”. On the whole, its results are more similar to “MED w/o EIM”, that is, the performance of the low-dimensional sub-model does not change much, while the performance of the high-dimensional sub-model decreases more significantly. We believe that for the high-dimensional sub-model, the proportion of mutual learning loss is still too large, which makes it more negatively affected by the low-dimensional sub-model. This result indicates that the dynamic loss weight plays a role in adaptively balancing multiple losses and contributes to improving overall performance.

## C Details of applying the trained KGE by MED to real applications

The SKG is used in many tasks related to users, and injecting user embeddings trained over SKG into downstream task models is a common and practical way.

User labeling is one of the common user management tasks that e-commerce platforms run on backend servers. We model user labeling as a multiclass classification task for user embeddings with a 2-layer MLP:

$$\mathcal{L} = -\frac{1}{|\mathcal{U}|} \sum_{i=1}^{|\mathcal{U}|} \sum_{j=1}^{|\mathcal{CLS}|} y_{ij} \log(\text{MLP}(u_i)), \quad (10)$$

where  $u_i$  is the  $i$ -th user’s embedding, the label  $y_{ij} = 1$  if user  $u_i$  belongs to class  $\text{cls}_j$ , otherwise  $y_{ij} = 0$ .

The product recommendation task is to properly recommend items to users that users will interact with a high probability and it often runs on terminal devices. Following PKGM (Zhang et al., 2021), which recommends items to users using the neural collaborative filtering (NCF) (He et al., 2017) framework with the help of pre-trained user embeddings as service vectors, we add trained user embeddings over SKG as service vectors to NCF. In NCF, the MLP layer is used to learn item-user interactions based on the latent feature of the user and item, that is, for a given user-item pair  $user_i - item_j$ , the interaction function is

$$\phi_1^{MLP}(p_i, q_j) = \text{MLP}([p_i; q_j]), \quad (11)$$

where  $p_i$  and  $q_j$  are latent feature vectors of user and item learned in NCF. We add the trained user embedding  $u_i$  to NCF’s MLP layer and rewrite Equation (11) as

$$\phi_1^{MLP}(p_i, q_j, u_i) = \text{MLP}([p_i; q_j; u_i]), \quad (12)$$

and the other parts of NCF stay the same as in PKGM (Zhang et al., 2021).

We train entity and relation embeddings for SKG based on TransE (Bordes et al., 2013) and input the trained entity (user) embedding into Equation (10) and Equation (12).

## D Details of extending MED to language model BERT-base

### D.1 Dataset and Evaluation Metric

For the experiments extending MED to BERT, we adopt the common GLUE (Wang et al., 2019) benchmark for evaluation. To be specific, we use the development set of the GLUE benchmark which includes four tasks: Paraphrase Similarity Matching, Sentiment Classification, Natural Language Inference, and Linguistic Acceptability. For Paraphrase Similarity Matching, we use MRPC (Dolan and Brockett, 2005), QQP and STS-B (Conneau and Kiela, 2018) for evaluation. For Sentiment Classification, we use SST-2 (Socher et al., 2013). For Natural Language Inference, we use MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), and RTE for evaluation. In terms of evaluation metrics, we follow previous work (Devlin et al., 2019; Sun et al., 2019a). For MRPC and QQP, we report F1 and accuracy. For STS-B, we consider Pearson and Spearman correlation as our metrics. The other tasks use accuracy as the metric. For MNLI, the results of MNLI-m and MNLI-mm are both reported separately.

### D.2 Baselines

For comparison, we choose Knowledge Distillation (KD) models and Hardware-Aware Transformers (Wang et al., 2020a) (HAT) customized for transformers as baselines. For the KD models, we compare MED with Basic KD (BKD) (Hinton et al., 2015), Patient KD (PKD) (Sun et al., 2019a), Relational Knowledge Distillation (RKD) (Park et al., 2019), Deep Self-attention Distillation (MiniLM) (Wang et al., 2020b), Meta Learning-based KD (MetaDistill) (Zhou et al., 2022a) and Feature Structure Distillation (FSD) (Jung et al., 2023). For the comparability of the results, we

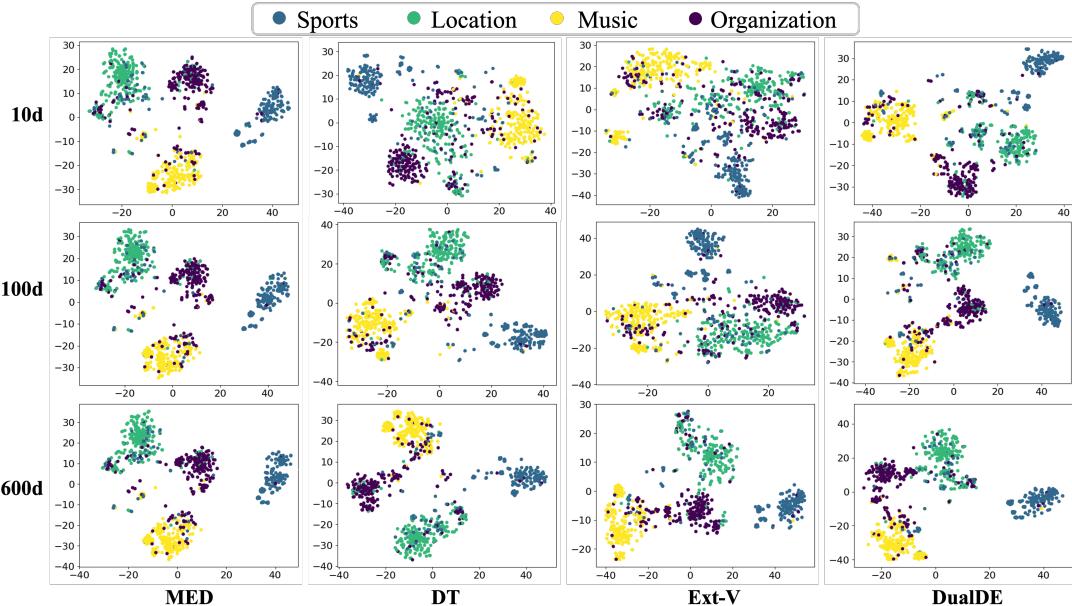


Figure 7: Clustering on FB15K237 with RotatE.

choose 4-layer BERT ( $BERT_4$ ) or 6-layer BERT ( $BERT_6$ ) as the student model architectures, which guarantees that the number of model parameters ( $\#P(M)$ ) or *speedup* is comparable. For HAT, we use the same model architecture as our MED for training and show the results of sub-models with three parameter scales.

### D.3 Implementation

To implement MED on BERT, for the word embedding layer, all sub-models share the front portion of embedding parameters in the same way as in KGE, and for the transformer layer, all sub-models share the front portion of weight parameters as in HAT (Wang et al., 2020a). Specifically, assuming that the embedding dimension of the largest BERT model  $B_n$  is  $d_n$ , and the embedding dimension of the sub-model  $B_i$  is  $d_i$ , for any parameter matrix with the shape  $x \times y$  in  $B_n$ , the front portion sub-matrix of it with the shape  $\frac{d_i}{d_n}x \times \frac{d_i}{d_n}y$  is the parameter matrix of the corresponding position in  $B_i$ . Finally, it just need to replace the triple score  $s_{(h,r,t)}$  in Equation (2), Equation (3), Equation (4), and Equation (5) with the logits output for the corresponding category of the classifier in the classification task.

We set  $n = 4$  for BERT applying MED, and 4 sub-models have the following settings: [768, 512, 256, 128] for embedding dim and [768, 512, 256, 128] for hidden dim, [12, 12, 6, 6] for the head number in attention modules, 12 for encoder layer

number.

## E Visual analysis of embedding

We select four primary entity categories ('organization', 'sports', 'location', and 'music') that contain more than 300 entities in FB15K237, and randomly select 250 entities for each. We cluster these entities' embeddings of 3 different dimensions ( $d=10, 100, 600$ ) by the t-SNE algorithm, and the clustering results are visualized in Fig. 7. Under the same dimension, the clustering result of MED is always the best, followed by DualDE, while the result of Ext-V is generally poor, which is consistent with the conclusion in Section 5.2. We also find some special phenomenons for MED when dimension increases: 1) the nodes of the 'sports' gradually become two clusters meaning MED learns more fine-grained category information as dimension increases, and 2) the relative distribution among different categories hardly changes and shows a trend of "inheritance" and "improvement". This further proves MED achieves our expectation that high-dimensional sub-models retain the ability of low-dimensional sub-models, and can learn more knowledge than low-dimensional sub-models.