AdaE: Knowledge Graph Embedding with Adaptive Embedding Sizes

Zhanpeng Guan, Fuwei Zhang, Zhao Zhang, Fuzhen Zhuang, Fei Wang, Zhulin An, Yongjun Xu

***Abstract*—Knowledge Graph Embedding (KGE) aims to learn dense embeddings as the representations for entities and relations in KGs. Indeed, the entities in existing KGs suffer from the data imbalance issue, i.e., there exists a substantial disparity in the occurrence frequencies among various entities. Existing KGE models pre-define a unified and fixed dimension size for all entity embeddings. However, embedding sizes of entities are highly desired for their frequencies, while a uniform embedding size may result in inadequate expression of entities, i.e., leading to overfitting for low-frequency entities and underfitting for high-frequency ones. A straight-forward idea is to set the embedding sizes for each entity before KGE training. However, manually selecting different embedding sizes is labor-intensive and time-consuming, posing challenges in real-world applications. To tackle this problem, we propose AdaE, which adaptively learns KG embeddings with different embedding sizes during training. In particular, AdaE is capable of selecting appropriate dimension sizes for each entity from a continuous integer space. To this end, we specially tailor bilevel optimization for the KGE task, which alternately learns representations and embedding sizes of entities. Our framework is general and flexible, fitting various existing KGE models. Extensive experiments demonstrate the effectiveness and compatibility of AdaE.**

***Index Terms*—Knowledge graph embedding, Data imbalance issue, Dimension selection**

I. Introduction

KNOWLEDGE Graphs (KGs) such as Wikidata [1], Freebase [2] and Yago [3] have been widely used in a number of downstream AI tasks, e.g., information retrieval [4], [5], question answering [6], [7] and recommendation systems [8], [9]. A knowledge graph is typically represented as a large-scale, directed graph, where nodes correspond to entities, and edges represent relations between these entities. In this structured format, real-world entities and their relations are captured in the form of knowledge triples, denoted as (s, r, o), where s and o are two entities representing the subject and object of the triple, while r signifies the relation connecting them. e.g., (*Paris, capitalOf, France*).

For a given KG, the key technique applied to downstream tasks is Knowledge Graph Embedding (KGE), which aims to map entities and relations into a low-dimensional space to obtain vector representations. Compared to one-hot vectors, embeddings reduce memory consumption and facilitate the prediction of potential triples. Given a (s, r, o) triple, existing KGE models typically take embeddings of entities and relation from the triple as input and output a score representing the plausibility of the triple [10]. These approaches are able to encode rich information of entities and relations, and are of great benefits for knowledge completion, fusion, and inference [11], [12].

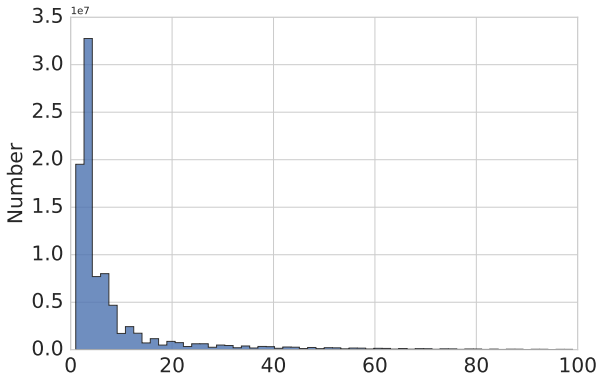


Fig. 1. Entity Frequency Histogram.

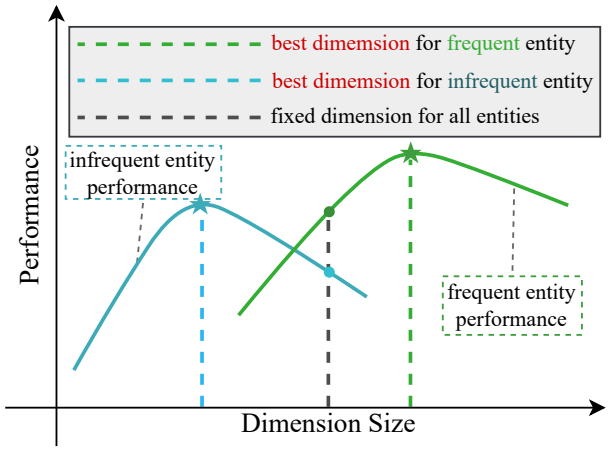


Fig. 2. Impact of embedding size.

However, real-world KGs suffer from the data imbalance issue, where various entities exhibit notable disparities in occurrence frequencies. Figure 1 shows the distribution histogram of the real-world KG Wikidata [1]. The horizontal axis represents the frequencies of entities, and the vertical

Zhanpeng Guan is with Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China and University of Chinese Academy of Sciences, Beijing 100049, China. E-mail: {guanzhanpeng22s@ict.ac.cn}

Zhao Zhang, Fei Wang, Zhulin An and Yongjun Xu are with Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China. E-mail: {zhangzhao2021, wangfei, anzhulin, xyj}@ict.ac.cn

Fuwei Zhang and Fuzhen Zhuang are with Institute of Artificial Intelligence, Beihang University, Beijing 100191, China. E-mail: {fuwei.zhang@foxmail.com, zhuangfuzhen@buaa.edu.cn}

This paper was produced by the IEEE Publication Technology Group. They are in Piscataway, NJ.

Manuscript received April 19, 2021; revised August 16, 2021.

axis is the number of entities with a certain frequency. Here, the "frequency" of an entity refers to the times the entity appears in the dataset. We find that a small number of entities appear frequently, while most entities have a fewer number of occurrences. Therefore, there exists a significant imbalance in real-world KGs. In real-world KGs, the number of entities is often much larger than that of relations, and the imbalance issue of entities becomes more severe than relations. Thus in this paper, we focus on addressing the data imbalance issue for entities.

Current KGE models utilize a uniform embedding size, which limits the model's expressive capacity for handling data with varying frequencies, i.e., leading to overfitting for low-frequency entities and underfitting for high-frequency ones. This naturally raises the question - do we need different embedding dimensions for different entities? To address this question, we conduct experiments using the ComplEx [13] model on the FB15k-237 [14] benchmark. We focus on the entity $Albert\ Einstein$. For this entity, the performance changes with different dimension sizes. When size is 64, the average MRR score of queries containing $Albert\ Einstein$ is 0.297. When size is 256, the MRR score reaches the largest value 0.643. When the embedding size continues to increase to 1024, the score drops to 0.527. This result shows that there is an optimal dimension size for each entity. In cases where the size is either excessively large or small, there will be a noticeable decline in performance.

We show our motivation in Figure 2. The horizontal axis is dimension size, and the vertical axis is performance (the quality of the learned embeddings). We provide the following three explanations: (1) for each entity, the performance will first rise and then fall down as the dimension size increases. (2) Existing KGE models use a fixed dimension size to represent all entities, which is denoted by the black dotted line in the figure, limiting the expressive ability of entities with different frequencies. Specifically, this would lead to overfitting for infrequent entities and underfitting for frequent ones. (3) Our idea is to enable all entities to match the optimal dimension size, which is expressed by the colored dotted line in the figure. This facilitates each entity to be fully expressed.

Previous works attempt to tackle the problem of data imbalance in two ways. The first is to utilize the reweighting technique that assigns different weights to different entities [15] during training. Specifically, they endow lower weights to high-frequency entities, and higher weights to low-frequency ones. The second way is to employ data augmentation to increase the occurrences of low-frequency entities. However, these methods still have the following drawbacks: (1) Due to the infrequent occurrences of low-frequency entities, adjusting weights may exacerbate overfitting. (2) Data augmentation might introduce noise through the creation of pseudo-triples that harm the performance.

In contrast to prior approaches, we propose AdaE, an adaptive embedding size selection method that differentially emphasizes entities to address the issue of data imbalance. Particularly, AdaE is capable of adaptively searching embedding sizes from a continuous integer space for each entity. Specifically, during the training process, AdaE is able to adaptively allocates smaller embedding sizes to infrequent entities, while assigning larger sizes to frequent ones. To this end, we first devise a picker module to generate dimension selection probabilities with regard to entity frequencies. Then, we design an alignment module to align various dimension sizes to the same size for subsequent KGE models. Since simultaneously learning the embedding sizes and the representations of entities may hinder stable convergence for both objectives, we draw inspiration from DARTS [16] to design a bilevel optimization process to alternately optimize different modules, mitigating the complexity brought by discrete search. Note that our method is different from neural architecture search. The latter searches a unified and fixed embedding for all entities, while our method is capable of adaptively searching appropriate embedding sizes for each individual entity.

It is worth noting that AdaE is versatile and flexible, and is applicable to numerous existing KGE models, such as TransE [17], DistMult [18], ConvE [19], and ComplEx [13]. Our contributions are as follows:

- In this paper, we propose an adaptive model AdaE, which allocates different embedding sizes to various entities to alleviate data imbalance issue in KGE. To the best of our knowledge, our approach is the first work on dimension search for the link prediction task.
- The dimension search framework is general and flexible, which is applicable to numerous existing KGE models.
- Extensive empirical analyses show AdaE outperforms state-of-the-art competitors.

II. Related Work

A. Knowledge Graph Embedding

Knowledge graph embedding aims to map entities and relations in a knowledge graph to a low-dimensional vector space. Currently, there are three main categories of knowledge graph embedding methods. In this section, we take a further step to provide a detailed review of the methods in each category.

Distance-based models. Distance-based models are the most classic type of KGE models, with TransE being the pioneer and followed by various variants. The fundamental concept of these models measure plausibility of fact triples as distance between entities. TransE originally proposes that the head entity, relation, and tail entity in the knowledge graph should follow the principle of $head + relation \approx tail$. Subsequent variants such as TranH [20], TransR [21], TransD [22] and TransA [23] enhance this concept with different projection strategies. RotatE [24] further advances this notion by conceptualizing relations as rotations from the head entity to the tail entity via rotational operations within the complex space, utilizing a single rotating surface. By introducing adaptive sparsity, TranSparse [25] addresses the imbalance issue in data distribution within the embedding model. In general, distance-based models have achieved superior performance compared to previous models by employing simple yet effective modeling methods.

Tensor decomposition models. This type of model defines a product-based functions over the entity and relation

embeddings. This line of thought was initially introduced by RESCAL [26], and after that DistMult [18] simplifies RESCAL by constraining the relation matrix to be a diagonal matrix. ComplEx [13] further extends the space of embedding values from real numbers to the complex numbers, which enhances the expressive abilities of entities and relations. SimplE [27] extends CP decomposition [28] with the inverse of relations to address the independence of the two vectors for each entity. Furthermore, SimplE proves that with a sufficient number of parameters, tensor decomposition models posses the capability of full expressivity.

Neural network models. These models leverage the advantages of deep learning, which learn entity and relation embeddings from the perspective of neural networks. For instance, R-GCN [29] represents various relations in knowledge graphs with Graph Convolutional Networks, ConvE [19] and ConvKB [30] construct features and learn from the perspective of convolutional networks, which has achieved superior results.

B. Data Imbalance Issue in KGE

The data imbalance issue often arises due to the inherent nature of knowledge graphs, where certain relations and entities have sparse occurrences. Thus addressing data imbalance is crucial for robust KGE models. Previous studies have predominantly focused on class imbalance in traditional machine learning [31]–[35], and limited attention has been given to the unique challenges posed by imbalanced entities in KGE. A recent line of work has explored sampling strategies, adaptive loss functions, and meta-learning techniques to mitigate this issue. WeightE [15] attempts to reweight the loss from a perspective of weight allocation, aiming to enhance the model's focus on low-frequency entities and relations. LSU [36] stands apart from conventional methodologies, by sharing latent semantic units, this approach transfers knowledge from frequently occurring entities and relations to less frequent ones. However, comprehensive solutions tailored to the intricacies of KGE data imbalance remain an open challenge, demanding further research to enhance model generalization and performance on real-world KGs.

C. Embedding Dimension Search

The Embedding Dimension Search (EDS) problem has gained increasing attention with the rapid development of neural architecture search (NAS) [16], [37]–[39]. Efforts in addressing this problem span across various domains, including computer vision [40], [41], text generation [42], recommender systems [43]–[46], etc.

Many approaches conceptualize the EDS problem as a hyperparameter optimization task, where NAS stands out as a key method for the automatic dimension search from a candidate dimension set. RDTDF [42] uses feature search strategy to search the optimal feature subset, reducing the dimension of the features while maintaining the predictive accuracy of a text classifier. DMaskingNAS [40] proposes a masking method for feature map reuse, supporting searches over spatial and channel dimensions. In recent years, some studies have proposed to utilize the technique of AutoML [43], [44], [47] to enhance search efficiency. For example, DNIS [47] employs a soft layer to learn optimal vocabulary sizes for categorical features and prunes the unimportant components after training. AutoEmb [43] and AutoDim [44] utilize AutoML-based framework to search embedding dimensions from a candidate set for recommendation. CIESS [46] presents a RL-based embedding size search algorithm for recommendation to select tailored embedding sizes from a continuous interval for each user/item. However, the aforementioned methods are designed to address memory consumption or architecture search, overlooking the significant impact of dimension size on embedding expressive ability. In this paper, we for the first time learns adaptive embedding sizes in the KGE task.

III. PRELIMINARIES

In this section, we provide some basic definitions used in this paper.

Definition 1: **Knowledge Graph.** A knowledge graph is viewed as a graph $\mathcal{G} = \{(s, r, o)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $\mathcal{E}$ and $\mathcal{R}$ are the entity (node) set and relation (edge) set, respectively.

Definition 2: **Frequent/Infrequent Entities.** In a KG, the entities with top 20% frequencies are named as frequent/high-frequency entities, while the remaining 80% entities are infrequent/low-frequency entities.

Definition 3: **Frequency-aware Triples.** For a (s, r, o) triple, if s and o are all frequent entities, it is named a frequent triple. Conversely, if s and o are all infrequent entities, it is named an infrequent triple. Frequent triples and infrequent triples are named as frequency-aware triples.

Since there are many symbols in this paper, for better understanding, we draw the Table I for clear understanding.

TABLE I
DESCRIPTIONS OF MATHEMATICAL SYMBOLS

Symbol Name	Description
$\mathbf{e}_i$	the embedding of the i-th entity e_i
$\hat{\mathbf{a}}_i$	the dimension selection vector of the input entity e_i
$\mathbf{a}_i$	the pre-processed probability output of the input entity e_i
$\mathbf{e}_i^*$	the final embedding of the entity e_i
$\mathbf{M}$	the mask matrix (a lower triangular matrix)
$\mathbf{A}_b$	dimension selection vector of batch b
$\mathbf{E}_b$	the embedding of batch b
$\cdot$	matrix multiplication
$\odot$	hadamard product
D	search space size
$\mathbf{\Theta}$	the parameters of the KGE module
$\mathbf{\Lambda}$	the parameters of the Picker module
$\phi(\cdot)$	the score function
$\mathrm{Re}(\cdot)$	the real part of a complex vector
ξ	learning rate

IV. METHODOLOGY

In this section, we present the details of the proposed KGE model with **Ada**ptive **E**mbedding Sizes (AdaE). Specifically, we first give an overview of the dimension search framework. Following that, we sequentially introduce the Picker module, Alignment module, and the optimization method. Finally, we delve into the application of these three components on KGE models and propose AdaE.

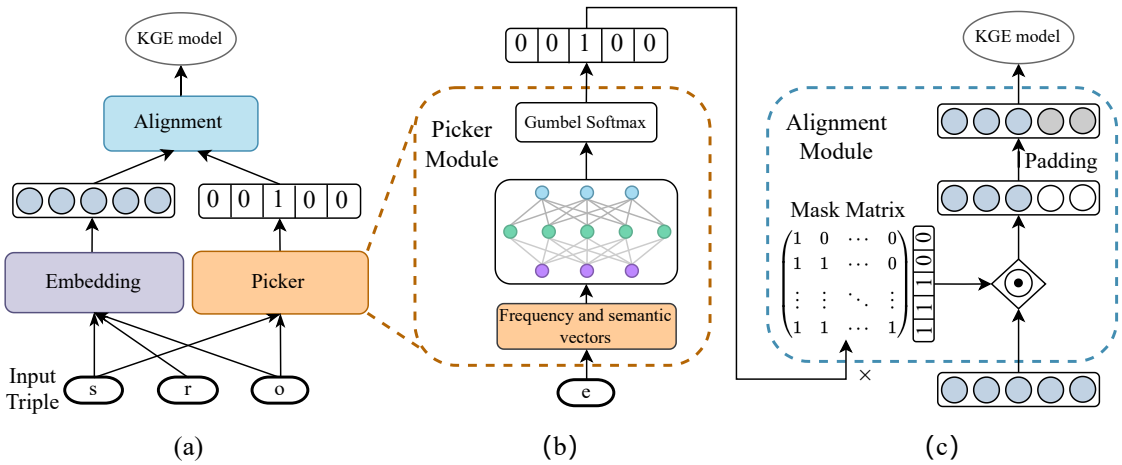


Fig. 3. Dimension Search Framework on KGE models. (a) Overview of the entire framework. Input triples: (s, r, o). Embedding Module: Lookup table for entities and relations. Picker Module and Alignment Module are detailed in (b) and (c). (b) Picker Module generates dimension selection vectors. (c) Alignment Module utilizes Mask Matrix Mechanism and an alignment method to select dimension sizes.

A. Overview

The illustration of the proposed framework is shown in Figure 3 (a). First, triples (s, r, o) are fed into the embedding module and picker module, separately. The embedding module functions as a lookup table, which contains the embeddings of entities or relations. Picker is an architecture search module that produces dimension selection vectors for entities. Then, the alignment module selects the embedding of the corresponding dimension size based on the dimension selection vector, and aligns the embeddings of different dimensions to those of the same dimension. Then, the embeddings of the same dimension are input to the KGE model. In particular, we devise a bilevel optimization process to ensure a more stable update when training different modules within the entire model. In the following, we introduce the details of the aforementioned modules.

B. Picker Module

As mentioned in Section I, smaller dimension sizes, due to their lower parameter count, exhibit better generalization performance for infrequency entities. As the frequency increases, small sizes constrain the representational capacity of these entities, necessitating larger sizes. Motivated by this phenomenon, we design Picker, an architecture search module to generate appropriate dimension size selection probabilities for different entities.

1) Input and Output.: **Input:** Picker takes frequency and semantic features as input, both highly relevant to capturing dimension size. Specifically, the input contains two parts, frequency bucket embedding and entity embedding. For bucket embedding, we divide the frequencies into several buckets, and set an embedding for each bucket. For entity embedding, we use the last selected embedding $\mathbf{e}_i$ of entity e_i. Particularly, we concatenate the two components as the input for Picker.

Output: The output of Picker is a one-hot vector (called dimension selection vector) $\hat{\mathbf{a}}_i \in \mathbb{R}^D$, where D denotes the size of search space. We select embedding sizes from {1, 2, ..., D}. The embedding space is selected based on the highest entry from a pre-processed probability $\mathbf{a}_i$, which is the output of the MLP network in Figure 3(b):

$$\mathbf{e}_i^* = \mathbf{e}_i^k, \quad \text{where } k = \arg\max_{d \in [1,D]} \mathbf{a}_i^d. \tag{1}$$

$\mathbf{e}_i^*$ is the selected embedding of entity e_i. $\{\mathbf{e}_i^1, \mathbf{e}_i^2, \ldots, \mathbf{e}_i^j, \ldots, \mathbf{e}_i^D\}$ represents the embeddings of entity e_i in different spaces, where the superscript denotes the embedding size. It is noteworthy that the framework we propose involves a search across a continuous integer space. This implies that D spaces are continuous, where d can take values continually from 1 to D in integer space.

2) Hard Selection.: Architecturally, we employ a multilayer perceptron (MLP) to capture pertinent features related to entity frequency and semantics, generating corresponding selection probabilities. However, directly adopting the dimension size with the maximum probability (as in Eq.(1)) would render the entire architecture learning process non-differentiable. To address this problem, we introduce the straight-through Gumbel softmax operation to emulate hard selection. Specifically, the output of MLP can be approximated as:

$$\hat{\mathbf{a}}_i^d = \frac{\exp\left(\left(\log\left(\mathbf{a}_i^d\right) + g_d\right)/\tau\right)}{\sum_{k=1}^{D} \exp\left(\left(\log\left(\mathbf{a}_i^k\right) + g_k\right)/\tau\right)}, \quad \forall d \in [1, D], \tag{2}$$

where $\mathbf{a}_i^d$ is the d-th entry of $\mathbf{a}_i$, and it is the value before Gumbel Softmax. $g_k = -\log(-\log(z_k))$ and $z_k \sim \text{Uniform}(0, 1)$ denote the Gumbel noise designed to prevent overfitting in the early stages of training. τ is the temperature hyperparameter to regulate the output. Specifically, when τ approaches 0, the output tends to be close to one-hot vector. Then, Eq.(1) can be reformulated in a differentiable form as Eq.(3) when $\tau \to 0$:

$$\mathbf{e}_i^* = \sum_{k=1}^{D} \hat{\mathbf{a}}_i^k \mathbf{e}_i^k. \tag{3}$$

With Gumbel-softmax, we can perform an operation akin to hard selection to select the optimal embedding sizes. The complete Picker module is depicted in Figure 3 (b).

C. Alignment Module

This module aims to align different embedding sizes into the same one, and feed the aligned embeddings to the KGE model. In this subsection, we delve further into the details of the Alignment module: Mask Matrix Mechanism and Alignment Method.

1) Mask Matrix Mechanism: Existing dimension search methods use multiple lookup tables with different sizes as embeddings in different spaces [44], [47], which is memory-unfriendly. Assuming we select the embedding size from {1,2, ..., D}, existing methods need to store D lookup tables, resulting in a space complexity $O(n+2n+...+D\cdot n)=O(nD^2)$, where n is the number of entities. Such a high complexity makes it difficult for such methods to perform continues integer search in real-world scenarios. Indeed, they always predefine a candidate set for embedding sizes, which limits the size of the search space. In this paper, we design the mask matrix mechanism to perform continuous integer search.

To this end, we devise a shared embedding matrix $\mathbf{E}\in\mathbb{R}^{n\times D}$. Different from existing approaches that set multiple embedding matrices with different sizes, in our method, $\mathbf{E}$ is shared by different embedding sizes. Specifically, for entity e_i, when the embedding size is set to d, our method uses the first d entries of $\mathbf{E}_i$ as the embedding for e_i, where $\mathbf{E}_i$ indicates the i-th row of $\mathbf{E}$. In this manner, the space complexity falls down from $O(nD^2)$ to $O(nD)$.

Specifically, we crafte a mask matrix, denoted as $\mathbf{M}\in\{0,1\}^{D\times D}$, The i-th line represents the mask vector $\mathbf{m}_i\in\{0,1\}^D$ corresponding to size i. In this setup, $\mathbf{m}_i$ is used to mask shared embedding to fit different sizes.

$$m_{i,j}=\begin{cases}1 & j\le i\\ 0 & i<j<D\end{cases},\quad i\in[0,D-1]. \tag{4}$$

This matrix is a lower triangular unit, as is shown in Figure 3 (c). It is worth mentioning that this mechanism is tensor-friendly, which can be accomplished by efficient tensor multiplications. Particularly, assuming that the size of the batch size is b, for a batch of embeddings $\mathbf{E}_b\in\mathbb{R}^{b\times D}$, Picker generates $\mathbf{A}_b\in\{0,1\}^{b\times D}$ as dimension selection vectors, each row of $\mathbf{A}_b$ is a D-dimensional one-hot vector indicating the selected embedding size. When $\mathbf{A}_b$ is input to the alignment module, the masked embeddings of the batch can be easily obtained by

$$\hat{\mathbf{E}}_b=(\mathbf{A}_b\cdot\mathbf{M})\odot\mathbf{E}_b, \tag{5}$$

where $\cdot$ is matrix multiplication and $\odot$ is the Hadamard product.

2) Alignment Method.: The alignment method aims to align entities with different embedding sizes to the same size. Previous dimension search methods [43], [44], [47] utilize transformation matrices to project different embedding sizes to the same one. However, this method inevitably brings about an increase in parameters. In this paper, we design a padding technique for alignment. Specifically, when d is selected as the embedding size for entity e_i, we align the embedding of e_i to a D-dimensional vector, in which the first d entries is the same as $\mathbf{E}_i$, and the last $D-d$ entries are padded with the average value of the initial embedding vector. It is worth noting we tried different padding value, and empirically find this setting achieved the best performance.

D. Optimization Method

In this subsection, we propose an optimization method for the dimension search framework. As the proposed framework is end-to-end differentiable, drawing inspiration from DARTS, we introduce a bilevel optimization process in pursuit of a more stable learning process of the representations and the dimension selection results.

This approach optimizes the KG embeddings ($\mathbf{\Theta}$) and Picker parameters ($\mathbf{\Lambda}$) alternately. The nested optimization problem, as formulated in Eq. (6), is challenging to solve directly. To address this, we adopt alternating optimization [48], breaking it down into two simpler optimizations. Essentially, we iteratively execute the following two steps:

- $\mathbf{\Theta}$ Update. In this step, we fix $\mathbf{\Lambda}$ while $\mathbf{\Theta}$ is optimized using triples (s,r,o) sampled from the inner set S_I.
- $\mathbf{\Lambda}$ Update. In this step, we fix $\mathbf{\Theta}$ while $\mathbf{\Lambda}$ is optimized using triples (s,r,o) sampled from the outer set S_O.

S_I and S_O are the splits of the training set S_T used to train the inner and outer loops, respectively. i.e., $S_I\cup S_O=S_T$. It is imperative to emphasize that these two module are intricately interdependent. As a result, simultaneously updating (without employing bilevel optimization) could potentially yield suboptimal performance. Therefore, the application of a bilevel optimization algorithm is necessary. We present the general form of bilevel optimization for our framework:

$$\begin{aligned}&\min_{\mathbf{\Lambda}}\mathcal{L}_{\text{outer}}\left(\mathbf{\Theta}^*(\mathbf{\Lambda}),\mathbf{\Lambda}\right)\\&\text{s.t. }\mathbf{\Theta}^*(\mathbf{\Lambda})=\arg\min_{\mathbf{\Theta}}\mathcal{L}_{\text{inner}}\left(\mathbf{\Theta},\mathbf{\Lambda}^*\right),\end{aligned} \tag{6}$$

where $\mathbf{\Lambda}$ is the outer level variable and $\mathbf{\Theta}$ is the inner level variable. The outer-level objective of the bilevel optimization is to assign suitable embedding sizes for each entity, while the inner-level objective focuses on learning high-quality embedding representations for entities. The optimization process of $\mathbf{\Lambda}$ relies on the optimal $\mathbf{\Theta}$ parameters. Hence, we utilize an approximation scheme:

$$\begin{aligned}&\nabla_{\mathbf{\Lambda}}\mathcal{L}_{\text{outer}}\left(\mathbf{\Theta}^*(\mathbf{\Lambda}),\mathbf{\Lambda}\right)\\\approx&\nabla_{\mathbf{\Lambda}}\mathcal{L}_{\text{outer}}\left(\mathbf{\Theta}-\xi\nabla_{\mathbf{\Theta}}\mathcal{L}_{\text{inner}}\left(\mathbf{\Theta},\mathbf{\Lambda}\right),\mathbf{\Lambda}\right),\end{aligned} \tag{7}$$

where ξ is the learning rate of parameters in Picker module. The approximate scheme estimates $\mathbf{\Theta}^*(\mathbf{\Lambda})$ by updating $\mathbf{\Theta}$ incrementally, avoiding a complete optimization of optimal parameters $\mathbf{\Theta}^*(\mathbf{\Lambda})=\arg\min_{\mathbf{\Theta}}\mathcal{L}_{\text{train}}\left(\mathbf{\Theta},\mathbf{\Lambda}^*\right)$ to achieve convergence.

E. Application on KGE models

1) Basic KGE model: In the subsection before, we present the dimension search framework and a bilevel optimization process. In this paper, we propose AdaE, which is built on the basis of the classic KGE model ComplEx [13] due to its simplicity and effectiveness. Indeed, this technique is general and flexible, making it applicable to a variety of existing

KGE models. In particular, we conduct a flexibility analysis in Subsection V-E by performing experiments on other KGE models to showcase its compatibility.

We provide a brief introduction to ComplEx, which is a representative tensor decomposition model. For a given triple (s, r, o), ComplEx maps entities s, o and relation r to complex space. In comparison to earlier models, ComplEx can handle asymmetry because of the complex conjugate of embedding vectors. Given a triple (s, r, o), the score function is defined as:

$$\phi(s, r, o; \mathbf{\Theta}) = \mathrm{Re}\left(< \mathbf{e}_s, \mathbf{w}_r, \bar{\mathbf{e}}_o >\right), \tag{8}$$

where $\mathbf{e}_s$, $\mathbf{w}_r$, and $\mathbf{e}_o$ are embeddings of s, r and o, respectively. $\mathbf{\Theta}$ denotes the parameters of ComplEx. $< \cdot, \cdot, \cdot >$ denotes the inner product of vectors and $\mathrm{Re}(\cdot)$ denotes the real part of the vector. A larger score indicates the triple is more likely to be true.

The loss function of ComplEx is

$$\min_{\Theta} \sum_{(s,r,o)\in\Omega} \log\left(1 + \exp\left(-\mathbf{Y}_{sro}\phi(s, r, o; \Theta)\right)\right) + \lambda\|\mathbf{\Theta}\|_2^2, \tag{9}$$

Θ corresponds to the embeddings of e_s, w_r, e_o, where $\lambda\|\Theta\|_2^2$ is the L_2 regularization part. $\mathbf{Y}_{sro} \in \{0, 1\}^{\Omega}$ denotes whether it is a truth of the triple and Ω is the training set. Substituting Eq. (9) into Eq. (6), we obtain specific inner and outer losses in the bilevel optimization. Subsequently, we outline the detailed CustomizE procedure.

Algorithm 1 Training procedure of AdaE

Input: $\mathbf{\Lambda}, \mathbf{\Theta}$: the Picker and embedding module parameters.
Input: S_I, S_O : the inner set and outer set.
Input: ξ : learning rate.
1: Initialize $\mathbf{\Theta}, \mathbf{\Lambda}$
2: **while** *not coveraged* **do**
3: Sample a mini-batch from S_O
4: Update the Picker module parameters $\mathbf{\Lambda}$ by descending $\nabla_{\mathbf{\Lambda}}\mathcal{L}_{\text{outer}}\left(\mathbf{\Theta} - \xi\nabla_{\mathbf{\Theta}}\mathcal{L}_{\text{inner}}\left(\mathbf{\Theta}, \mathbf{\Lambda}\right), \mathbf{\Lambda}\right)$ ($\xi = 0$ if using first-order approximation)
5: Sample a mini-batch of S_I
6: Update the embedding module parameters $\mathbf{\Theta}$ by descending $\nabla_{\mathbf{\Theta}}\mathcal{L}_{\text{inner}}\left(\mathbf{\Theta}, \mathbf{\Lambda}\right)$
7: Generate **a** via Picker with fixed $\mathbf{\Lambda}$
8: Generate **e** from embedding module and select dimension size according to **a**
9: Evaluation and learning rate update
10: **end while**

2) AdaE: Building upon the preceding section, we introduce AdaE, an adaptive embedding sizes selection framework with bilevel optimization. Subsequently, we delve into the details of its training process.

As shown in Algorithm 1, which outlines the step-by-step computation flow of AdaE. One iteration of the AdaE is divided into two significant phases. The first phase is the embedding update phase, whose goal is to update the embeddings based on the inner set and learn high-quality representations of entities and relations. The second phase is the Picker update phase, where the goal is to search for the most suitable embedding sizes through bilevel optimization over the outer set.

To provide a more concrete overview, we present a schematic representation of this process: (1) Firstly, we initialize embeddings and Picker module parameters (line 1). Then, we calculate the gradient of the loss function with respect to $\mathbf{\Lambda}$ on the inner set, AdaE utilize this gradient to update Picker parameters $\mathbf{\Lambda}$ (line 3-4). It is essential to note that the gradient with respect to $\mathbf{\Theta}$ within this formula must be computed on a copied model to prevent interference with the original model parameters. Next, we calculate the gradient of the loss function with respect to $\mathbf{\Theta}$ on the training batches, and employ this gradient to update embedding parameters $\mathbf{\Theta}$(line 5-6). Finally, with all parameters fixed, we generate **e** and evaluate the performance of AdaE on validation set, and adjust the learning rate with an exponential decay pattern. (line 7-9). The above process guarantees a stable learning stage of AdaE.

V. Experiment

In this section, we present our experimental analysis on the performance of AdaE. We evaluate our model on the link prediction task (a.k.a. knowledge graph completion task). Our objective is to investigate the following questions:

- **RQ 1**: Does AdaE perform better than other state-of-the-art KGE models?
- **RQ 2**: How does AdaE learn the embedding sizes for entities with different frequencies in a KG?
- **RQ 3**: Are the hyperparameter settings of AdaE reasonable?
- **RQ 4**: Can the dimension search framework be applied to other KGE models?
- **RQ 5**: How is the time efficiency and memory cost of AdaE compared with baselines?
- **RQ 6**: How does AdaE perform on extreme cases?

A. Experimental Setup

1) Datasets.: We conduct our experiments on three widely used benchmarks: WN18RR [19], FB15k-237 [14] and YAGO3-10 [19]. WN18RR is a subset of WordNet [49], which contains a collection of knowledge triples. FB15k-237 is a subset of Freebase [2], which provides general facts of the world. YAGO3-10 is a dataset sampled from YAGO3 [50], which describes the attributes of persons such as gender, citizenship, and profession. These three datasets are widely used benchmarks in the field of knowledge graph, and they all exhibit imbalanced data distributions. The statistics of three datasets are presented in Table II. Following the Pareto principle (the 80/20 rule)[1], we select the top 20% entities as "frequent entities" with a descending order of frequency, while the remaining 80% as "infrequent entities". Thus, we calculate their average frequency and the results are summarized in Table III.

[1]https://en.wikipedia.org/wiki/Pareto_principle

TABLE II
DATASET STATISTICS. $|\mathcal{E}|$ AND $|\mathcal{R}|$ ARE THE NUMBER OF ENTITY SET AND RELATION SET. $|\mathcal{G}_{train}|$, $|\mathcal{G}_{valid}|$, AND $|\mathcal{G}_{test}|$ ARE THE NUMBER OF TRIPLES IN THE TRAINING, VALID, AND TEST SET, RESPECTIVELY.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{G}_{train}\rvert$	$\lvert\mathcal{G}_{valid}\rvert$	$\lvert\mathcal{G}_{test}\rvert$
WN18RR	40,943	11	86,835	3,034	3,134
FB15k-237	14,541	237	272,115	17,535	20,466
YAGO3-10	123,182	37	1,079,040	5,000	5,000

2) Evaluation Metrics: We conduct our experiments on the link prediction task, which aims to predict the missing part of a triple. In this paper, we utilize two metrics to compare the performance of AdaE with other competitors: (i) Mean Reciprocal Rank (MRR, the mean of all the reciprocals of predicted ranks); (ii) Hits@k (H@k, the proportion of ranks not larger than k). All the results are reported in the "filtered" setting [17].

TABLE III
IMBALANCED DATA ANALYSIS. "AVG. FREQ." IS SHORT FOR "AVERAGE FREQUENCY". FREQUENT AVG. FREQ. AND INFREQUENT AVG. FREQ. REPRESENT THE AVERAGE FREQUENCY OF FREQUENT ENTITIES AND INFREQUENT ENTITIES, RESPECTIVELY.

Dataset	avg. freq.	Frequent avg. freq.	Infrequent avg. freq.
WN18RR	4	11	3
FB15k-237	37	111	19
YAGO3-10	17	53	9

3) Baselines: We compare AdaE with the following baselines.

- Distance-based models, including TransE [17], RotatE [24], MuRP [51], MuRE [51] and MDE [52].
- Tensor decomposition models, including DistMult [18], ComplEx [13], QutaE [53], BoxE [54] and NagE [55].
- Neural network methods, including ConvE [19], ConvKB [30], HypER [56], KMAE [57], MFAE [58], LKE [59] and SDFormer [60].
- ComplEx$_{Mixup}$ [61] and LSU [36] are two models that address the data imbalance issue with data augmentation methods and latent semantic units, respectively.
- AdaE-rule, a variant that allocates dimension sizes based on the frequencies of entities. Specifically, higher frequencies are assigned larger sizes.
- AdaE-sim, a variant of AdaE which abandons the bilevel training procedure. For AdaE-sim, we train the picker module and the embeddings of entities simultaneously.
- AdaE-iter, a variant which alternately trains the embedding module and picker module without the sophisticated gradient update method in bilevel optimization.

4) Implementation Details: In the training phase, all experiments are conducted as follows: CPU: Intel(R) Xeon(R) Gold 6330 CPU @2.00GHz, memory: 256GB DDR4; GPU: 1× NVIDIA GeForce RTX 4090. We adopt Adagrad [62] as the optimizer to update all parameters. During the training process, before starting each epoch, we randomly split 80% of training set as the inner set S_I, and 20% as outer set S_O. For the parameters of the picker module, we set the learning rate to 0.1 and initialize the MLP with Xavier initialization [63]. The MLP in Picker module uses a three-layer fully connected network with 512 hidden layer neurons. The activation function uses the Tanh$(\cdot)$ function. For the parameters of the embedding module, we set the learning rate to 0.1 and initialize the embeddings with uniform distribution $\mathrm{Uniform}(0, 1)$. The batch size is 1024 and the dropout rate of MLP is 0.2. For Gumbel-softmax [64], we use an annealing scheme for temperature $\tau = \max(0.01, 1 - 5e^{-2} \cdot \text{epoch})$.

B. Overall Performance (RQ1)

0.371 0.278 0.406 0.562 0.491 0.455 0.508 0.585 0.564 0.476 0.582 0.712

To demonstrate the performance of AdaE on link prediction task, we perform a comparative analysis with state-of-the-art methods, as shown in Table IV. From this table, we make the following observations: (1) In the aggregate, AdaE exhibits superior performance compared to all baseline methods on all datasets, demonstrating the effectiveness of our proposed method. (2) Compared to baseline methods, AdaE achieves greater improvements on WN18RR and YAGO3-10 than on FB15k-237. As shown in Table III, the average frequency (degree) of the former two datasets is smaller than that of FB15k-237. The above results indicate the proposed AdaE is more suitable for sparser datasets. We conjecture the reason lies in that sparser datasets have more entities with very few occurrences. Such entities only need small dimension sizes. Compared to baseline methods that use a uniform large dimension size, choosing small dimension sizes for entities is exactly what AdaE good at doing. As a result, AdaE shows more significant improvements on WN18RR and YAGO3-10. (3) AdaE significantly outperforms its backbone model ComplEx, indicating that our proposed method with adaptive embedding sizes greatly enhances the representation ability of the backbone model. (4) AdaE-rule achieves improvements over the backbone model, suggesting the feasibility of allocating entities with different frequencies to distinct dimension sizes. However, AdaE-rule still falls short of AdaE, which shows the effectiveness of the dimension search framework. This discrepancy arises as rule-based partitioning solely considers frequency as the determining factor, lacking the capacity to capture potential non-linearity, interactions, or complex data patterns. (5) AdaE-sim outperforms AdaE-rule, confirming the superiority of automatically learned dimension sizes over the rule-based method. This highlights the importance of automatically allocating dimension sizes. The performance of AdaE-sim is still poorer than AdaE, indicating the importance of bilevel optimization. (6) We find the performance of AdaE-iter surpasses that of AdaE-sim. The reason lies as separating the picker module and embedding module in AdaE-iter reduces the interdependence of each other, thereby enhancing the performance. Nevertheless, AdaE-iter still falls short of AdaE, we conjecture the reason lies in that with bilevel optimization, AdaE employs additional information, such as the second order derivative in dimension search, which is beneficial to achieve a more stable and effective learning process.

TABLE IV
EXPERIMENTAL RESULTS ON WN18RR, FB15K-237 AND YAGO3-10. SUPERSCRIPTS †, ‡, ♯, AND § INDICATE THE RESULTS ARE TAKEN FROM [65], [66], [67], AND THE ORIGINAL PAPER, RESPECTIVELY. ♢ INDICATES THE RESULTS ARE OBTAINED BY OURSELVES. ∗ DENOTES THE IMPROVEMENT OF ADAE IS STATISTICALLY SIGNIFICANT COMPARED WITH THE BEST BASELINE AT P-VALUE < 0.05 OVER PAIRED T-TEST. TEXT IN **BOLD** FONTS AND UNDERLINED FONTS INDICATES THE BEST AND SECOND-BEST RESULTS FOR EACH DATASET, RESPECTIVELY.

	WN18RR				**FB15k-237**				**YAGO3-10**			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE†	0.226	-	-	0.501	0.294	-	-	0.465	-	-	-	-
RotatE‡	0.476	0.428	0.492	0.571	0.338	0.241	0.375	0.533	0.495	0.402	0.550	0.670
MuRP§	0.481	0.440	0.495	0.566	0.335	0.243	0.367	0.518	0.354	0.249	0.400	0.567
MuRE§	0.465	0.436	0.487	0.554	0.336	0.245	0.370	0.521	0.532	0.444	0.584	0.694
MDE§	0.458	-	-	0.560	0.344	-	-	0.531	-	-	-	-
DistMult‡	0.430	0.390	0.440	0.490	0.241	0.155	0.263	0.419	0.340	0.240	0.380	0.540
ComplEx‡	0.440	0.410	0.460	0.510	0.247	0.158	0.275	0.428	0.360	0.260	0.400	0.550
QuatE§	0.481	0.436	0.500	0.564	0.311	0.221	0.342	0.495	-	-	-	-
BoxE§	0.451	-	-	0.541	0.337	-	-	0.538	0.560	-	-	0.691
NagE§	0.476	0.429	0.493	**0.575**	0.340	0.243	0.376	0.532	-	-	-	-
ConvE‡	0.430	0.400	0.440	0.520	0.325	0.237	0.356	0.501	0.440	0.350	0.490	0.620
ConvKB♯	0.249	0.056	-	0.525	0.230	0.140	-	0.415	0.420	0.322	-	0.605
HypER§	0.465	0.436	-	0.522	0.341	0.252	-	0.520	0.533	0.455	-	0.678
KMAE§	0.448	0.415	0.465	0.524	0.326	0.240	0.358	0.502	-	-	-	-
MFAE§	0.467	0.437	0.482	0.530	0.355	0.263	0.390	0.540	0.549	0.472	0.596	0.688
LKE§	0.460	0.426	0.473	0.529	0.354	0.262	0.388	0.540	0.557	0.484	0.600	0.691
SDFormer§	0.458	0.425	0.471	0.528	0.356	**0.264**	0.390	0.541	-	-	-	-
ComplEx$_{Mixup}$§	0.401	-	-	0.466	0.279	-	-	0.452	0.391	-	-	0.576
LSU♢	0.475	0.402	0.468	0.499	0.336	0.251	0.364	0.508	0.484	0.409	0.511	0.653
AdaE-rule	0.448	0.425	0.458	0.496	0.322	0.236	0.353	0.490	0.528	0.449	0.570	0.679
AdaE-sim	0.470	0.436	0.482	0.533	0.341	0.251	0.376	0.521	0.552	0.478	0.598	0.686
AdaE-iter	0.475	0.439	0.490	0.573	0.345	0.253	0.380	0.527	0.564	0.493	0.607	0.691
AdaE	**0.489**∗	**0.452**∗	**0.504**∗	0.567	**0.357**∗	0.263	**0.393**∗	**0.546**∗	**0.572**∗	**0.501**∗	**0.618**∗	**0.697**∗

C. Dimension Size Analysis (RQ2)

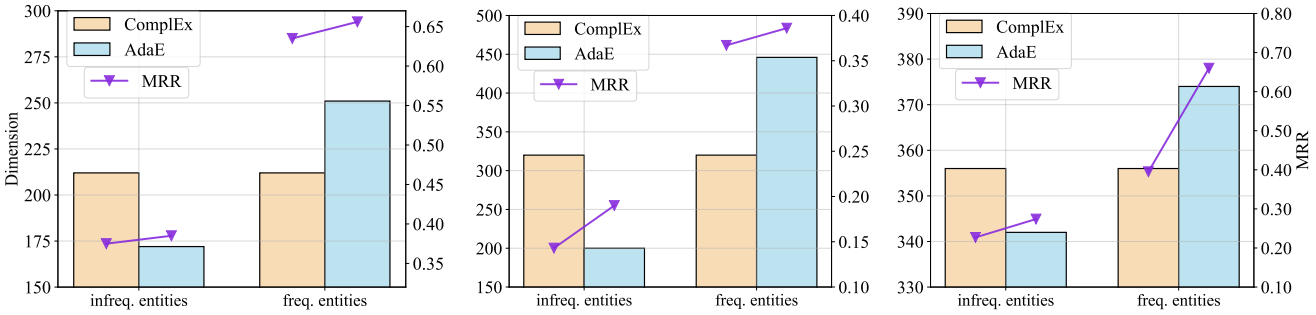
In this section, we present comprehensive analyses on the embedding sizes of entities with different frequencies.

1) Evaluation on Dimension Sizes of Different Frequencies.: In this subsection, we analyze the embedding sizes of entities with different frequencies, and the results are shown in Figure 4. In Figure 4, the bars denote the average embedding sizes for frequent and infrequent entities, and the purple line represents the MRR score for frequent and infrequent triples (cf. Section III). For ComplEx, we set the embedding sizes as the median of dimensions search by AdaE. Under this setting, the AdaE has fewer training parameters than ComplEx, making the comparison even more challenging.

From this figure, we have the following findings: (1) From the perspective of embedding sizes, AdaE selects smaller sizes for infrequent entities and larger sizes for frequent ones. Taking the WN18RR dataset as an example, infrequent entities have an average embedding size of 172, whereas frequent entities have an average embedding size of 252. This result is in line with our expectations, which indicates that the proposed method is able to learn suitable embedding sizes. (2) From the perspective of model performance, AdaE can improve the performance of both frequent and infrequent triples. For instance, on FB15k-237 dataset, the MRR value for infrequent triples increases by 0.047. For frequent triples, it increases by 0.019. This result confirms our hypothesis that choosing appropriate embedding sizes leads to more comprehensive representations of entities with varying frequencies.

2) Dimension Size Distribution: The dimension distribution of entities with different frequencies is another crucial result that can validate the effectiveness of AdaE. To address this, we arrange entities in ascending order w.r.t. their frequencies and divide them into 10 groups. As the results are consistent across different datasets, we will use YAGO3-10 as an example. Figure 5 shows the change of dimension size as the entity frequency increases. From this figure, we find that the dimension size generally increases as entity frequency rises, demonstrating AdaE can effectively select suitable embedding sizes for entities with varying frequencies.

In our model, the selection of embedding sizes depends on both the frequency and semantic features. To explore the role of semantics in selecting embedding sizes, we remove the semantic features (i.e., entity embeddings) in the Picker module. In this setting, the dimension size of each entity is completely determined by entity frequency. The results are shown in Figure 6. Comparing Figure 6 with Figure 5, we find the two figures share similar trend, which shows that frequency information plays a decisive role in searching embedding sizes. While the role of semantic information is to help the model select a more precise dimension size via considering the characteristics of each entity. In our experiment, we find removing the semantic information resulted in a decreased MRR score (from 0.357 to 0.344), which shows the semantic feature is important in achieving good results. Indeed, even entities with exactly the same dimensions may have slightly different dimensions. For example, for two entities with the



(a) WN18RR (b) FB15k-237 (c) YAGO3-10

Fig. 4. Experimental results for dimension size analysis on three datasets.

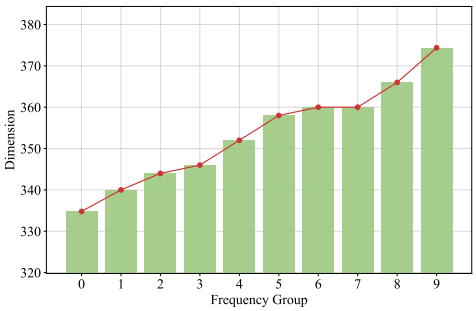


Fig. 5. The change of dimension size as the entity frequency increases.

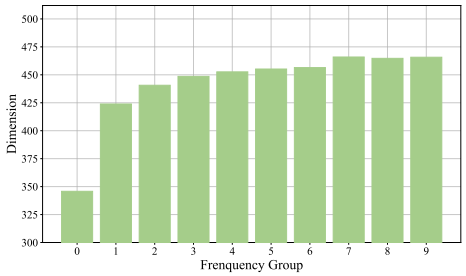


Fig. 6. The change of dimension size as the entity frequency increases without semantic feature.

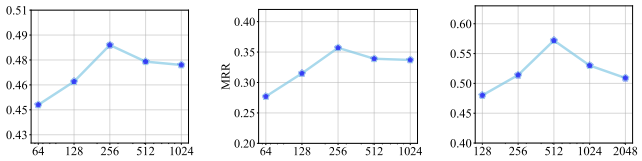
same frequency, the entity *Donald Trump* has richer semantic information than the entity *television*, since Donald Trump has multiple roles, e.g., president, businessman, husband, father, etc. As a result, the embedding size of *Donald Trump* is larger than that of *television*.

Overall, the entity frequency information plays a decisive factor in selecting optimal entity embedding sizes. And the semantic feature is also important and beneficial to the embedding size searching process because semantic feature can provide valuable information of the characteristics of each entity.

D. Hyperparameter Sensitivity Analysis (RQ3)

In this section, we analyze the influence of key hyperparameters in AdaE.

1) Search Space Size D.: During training, AdaE will explore embedding sizes from {1, 2, ..., D}. To investigate the influence of D, for FB15k-237 [14] and WN18RR [19], we set $D \in \{64, 128, 256, 512, 1024\}$. While for YAGO3-10 [19], owing to its substantial data volume, we set $D \in$



(a) WN18RR (b) FB15k-237 (c) YAGO3-10

Fig. 7. Performance under different search space size D.

$\{128, 256, 512, 1024, 2048\}$. Figure 7 presents the change of MRR value with the increasing search space size D. As shown in Figure 7, AdaE achieves the best performance when $D = 256$ on FB15k-237 and WN18RR. Due to the larger number of entities and triples in YAGO3-10, the optimal value for D increases to 512 on YAGO3-10. A too small D implies a more constrained maximum dimension for searching, resulting in diminished model performance. Conversely, with a excessively large D, the expanded search space introduces more intricate combinations of dimensions, which degrade model performance.

2) Picker Update Frequency p.: In Algorithm 1, we update the embedding module on the inner set and the picker module on the outer set. In the experiment, we find that the update frequency of the two modules has impact on the performance. Thus, we set the update frequency as p, which represents the picker module updates one time after the KGE module updates p times. We conduct experiments on WN18RR(similar conclusions can be found in other datasets) in Figure 9. We observe that the optimal performance is achieved when $p = 2$. When $p > 2$ or $p < 2$, the picker module obtains worse performance. Furthermore, we also expanded the range of p from $\{1, 2, 3\}$ to $\{1, 2, 3, 4, 5\}$, and observed the same phenomenon. The results are shown in Figure 8. With p increasing from 1 to 2, the model performance goes up. And then the performance falls down when p changes from 2 to 5. We conjecture the reason lies as follows. When p is very small, the Picker module updates too frequently. Specifically, the Picker module utilizes the embeddings from the KGE module, which only update once. The embeddings are not fully trained, leading to the Picker module selecting inappropriate embedding sizes, resulting in suboptimal performance. When p is very large, the KGE module updates many times after one update of the Picker module. In this case, the KGE module cannot get timely feedback from the Picker module, leading

TABLE V
EXPERIMENTAL RESULTS OF DIMENSION SEARCH FRAMEWORK UPON DIFFERENT REPRESENTATIVE KGE MODELS.

	WN18RR				FB15k-237				YAGO3-10			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE	0.226	-	-	0.501	0.294	-	-	0.465	-	-	-	-
TransE + DimS	**0.234**	**0.075**	**0.341**	**0.518**	**0.323**	**0.231**	**0.357**	**0.506**	**0.405**	**0.317**	**0.450**	**0.573**
DistMult	0.430	0.390	0.440	0.490	0.241	0.155	0.263	0.419	0.340	0.240	0.380	0.540
DistMult + DimS	**0.434**	**0.404**	**0.445**	**0.493**	**0.341**	**0.251**	**0.376**	**0.524**	**0.529**	**0.452**	**0.572**	**0.673**
ConvE	0.430	0.400	0.440	**0.520**	0.325	0.237	0.356	0.501	0.440	0.350	0.490	**0.620**
ConvE + DimS	**0.443**	**0.414**	**0.453**	0.503	**0.337**	**0.247**	**0.371**	**0.521**	**0.469**	**0.394**	**0.508**	0.608
ComplEx	0.440	0.410	0.460	0.510	0.247	0.158	0.275	0.428	0.360	0.260	0.400	0.550
AdaE	**0.489**	**0.452**	**0.504**	**0.567**	**0.357**	**0.263**	**0.393**	**0.546**	**0.572**	**0.501**	**0.618**	**0.697**

TABLE VI
EXPERIMENTAL RESULTS OF DIMENSION SEARCH FRAMEWORK UPON STATE-OF-THE-ART KGE MODEL COMPLEX-DURA.

	WN18RR				FB15k-237				YAGO3-10			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
SR-GNN(RNN)	0.490	0.455	0.504	0.558	0.360	0.267	0.398	0.543	-	-	-	-
RGAI	0.491	0.455	0.508	0.585	0.371	0.278	0.406	0.562	0.564	0.476	0.582	0.712
ComplEx-DURA	0.491	0.449	0.504	0.571	0.371	0.276	0.408	0.560	0.584	0.511	0.628	0.713
ComplEx-DURA+DimS	**0.500**	**0.456**	**0.516**	**0.586**	**0.377**	**0.282**	**0.419**	**0.565**	**0.589**	**0.517**	**0.629**	**0.719**

to inferior results. Thus setting an appropriate p is critical to obtain satisfactory results.

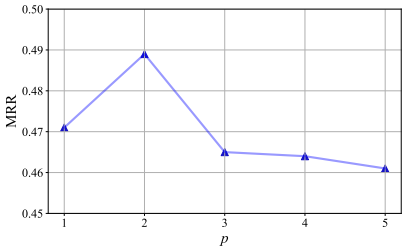


Fig. 8. Performance under different update frequencies p on WN18RR.

3) Learning Rate ξ: In this paper, we set the same learning rate for the Embedding module and Picker module. To evaluate the impact of different learning rates on model performance, we conducted experiments on the FB15k-237 dataset. We set candidate learning rates to $\{0.5, 0.25, 0.1, 0.05, 0.01\}$. As shown in Figure 10, the horizontal axis represents the learning rate on a logarithmic scale, while the vertical axis represents the MRR score.

The model's performance shows a trend of first rising and then declining as the learning rate increases. The peak performance is achieved at the learning rate of 0.1, with an MRR of 0.357. A learning rate that is too small leads to slow training and potential stagnation, while a learning rate that is too large causes instability and divergence. Both settings may lead to inferior performance. And finding the right balance is key to effective model training.

4) Decay Factor τ: We expect the Picker module output a one-hot vector, indicating the selected embedding size. In order to make the entire model differentiable and enable the entire model to be trained in an end-to-end manner, we apply the Gumbel-softmax trick to the output of the Picker module.

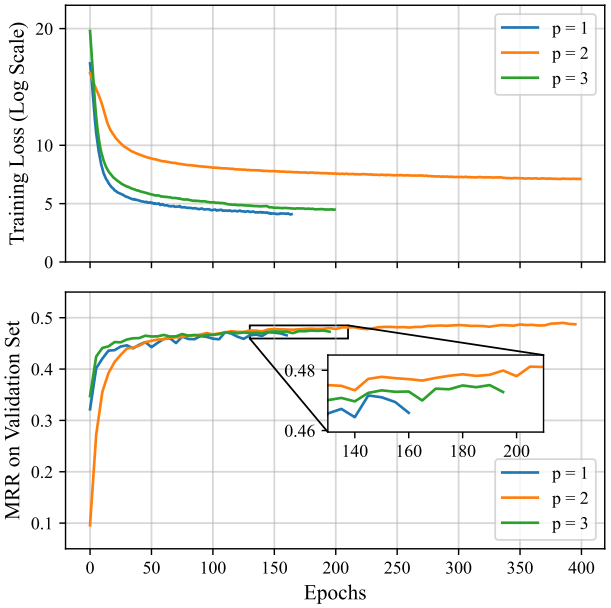


Fig. 9. Performance under different update frequencies p. Above: When epoch increases, the training loss of p=1 and p=3 are lower than that of p=2. Below: When epoch increases, the performance of p=1 and p=3 on validation set are lower than that of p=2.

Specifically, when τ approaches 0, the output tends to be close to one-hot vector.

We use an annealing algorithm to control the temperature τ, which means that the temperature is high at the beginning and slowly decays to a fixed value during training. In this experiment, we use the following formula: τ = max{1-epoch*β, 0.01}, where τ is the temperature and β is the decay factor.

We conduct experiments on the FB15k-237 dataset to evaluate the impact of different decay factors on model performance and set its candidate set to {0.5, 0.1, 0.05, 0.01, 0.005}. As

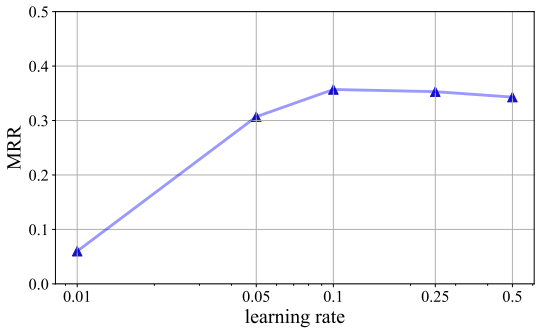


Fig. 10. Performance under different learning rates on FB15k-237.

shown in Figure 11, the horizontal axis represents the decay factor on a logarithmic scale, while the vertical axis represents the MRR score.

The model's performance also exhibits a trend of first rising and then declining as the decay factor increases. The best performance is achieved when the decay factor is set to 0.05. When the decay factor is smaller than 0.05, the temperature increases more gradually, enabling broader exploration of dimensions during the early stage of training process. However, it causes the temperature remaining relatively high in the later stages, preventing the model from focusing on specific dimensions, which slightly reduces performance. In contrast, when the decay factor is larger than 0.05, the temperature drops rapidly, making the Gumbel-Softmax output nearly one-hot. As a result, the model focuses exclusively on updating the weights of neurons in a specific dimension, prematurely abandoning the exploration of other dimensions, leading to suboptimal performance.

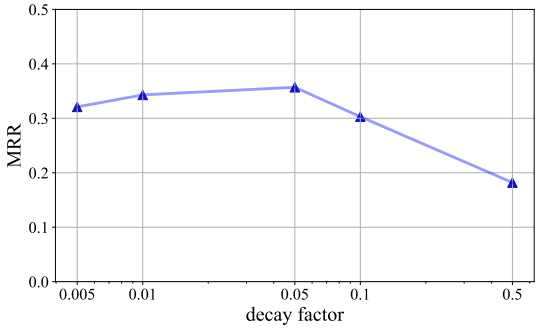


Fig. 11. Performance under different decay factors on FB15k-237.

E. Model Flexibility (RQ4)

To evaluate the flexibility of the dimension search framework, we apply this framework to various KGE models. Specifically, we choose TransE from the distance-based methods, Distmult from tensor decomposition methods, and ConvE from the neural network methods, respectively. We divide the table into four groups, with each group comprising the basic model and the extended model with the **Dim**ension **S**earch framework (DimS). The results are shown in Table V. We find that the proposed dimension search framework is able to improve the performance of various KGE methods, demonstrating the effectiveness and flexibility of our framework. This result further emphasizes the necessity to explore and address the issue of data imbalance.

Moreover, our framework is extensible with numerous existing models as a plug-and-play component. To obtain better results, we just need to apply the dimension search framework to more recent and more advanced models. To verify this, we extend model ComplEx-DURA [68]. The results are shown in Table VI. We can see ComplEx-DURA+DimS not only outperforms the base model ComplEx-DURA, but also achieves better results than SOTA models RGAI [69] and SR-GNN (RNN) [70], which again verifies the effectiveness of our framework.

F. Time Efficiency and Memory Cost Analysis (RQ5)

Furthermore, we analyze the time and memory cost of AdaE. In particular, we conduct a comparative analysis between AdaE and its base model ComplEx. Our primary objective was to assess AdaE's time efficiency and memory cost. We conduct experiments on YAGO3-10, a large-scale knowledge graph dataset which contains 123,182 entities and more than 1 million triples. The results are shown in Table VII.

In terms of time efficiency, we use time per epoch as an evaluation metric. During the training phase, AdaE exhibits an approximately 15% increase in time compared to ComplEx, which primarily arises from the dimension selection process of the Picker module in AdaE. In the meanwhile, AdaE achieves a higher MRR score of 0.212 compared to the base model ComplEx (cf. Table V). Simultaneously, the inference time has decreased by 59%, which is attributed to the decrease in average dimensionality following dimensionality reduction. Building upon the aforementioned highlights, we posit that a 15% increase in time is deemed acceptable.

In terms of memory cost, we utilize the average dimension (Avg. Dim.) as an evaluation metric. This is because both AdaE and ComplEx are lightweight models, with primary memory consumption determined by the embedding lookup tables. During the training process, the dimension size for ComplEx is 512, and AdaE also selects appropriate dimension sizes from 1 to 512. Thus the memory cost of ComplEx and AdaE remains equivalent. While during inference, AdaE's average dimension size experiences a 32% reduction. In practical deployment for inference, AdaE exhibits both reduced memory requirements and faster processing speed compared to ComplEx.

TABLE VII
THE TIME EFFICIENCY AND MEMORY COST OF COMPLEX AND ADAE ON YAGO3-10

	Time per epoch		**Avg. Dim.**	
	Train	Inference	Train	Inference
ComplEx	135.7s	5.1s	512	512
AdaE	156.4s(+15%)	2.1s(-59%)	512	348(-32%)

G. Performance Analysis on Extreme Cases (RQ6)

To testify the effectiveness of AdaE on extreme cases, we evaluate the performance of AdaE on entities with a frequency of 1, 2, and 3, i.e., entities with extremely low frequencies. We use the average MRR score of triples containing such entities as the metric. And the results are shown in Figure 12. It can be observed that AdaE outperforms the base model ComplEx (with the same parameter capacity as AdaE) regarding entities with extremely low frequencies. For case study, the entity "Lancaster" has a frequency of 1, and after applying the dimension search framework, the MRR of its related triples improved from 0.106 to 0.169. The above observations indicate the robustness of AdaE and the proposed dimension search framework.

We also analyze the limitations of the proposed method. We find that AdaE is more suitable for datasets with imbalanced entity frequencies, while the performance improvement is relatively small on datasets with balanced entity frequencies. Evidence can be seen from Table V. In Table V, the data imbalance issue of YAGO3-10 is more serious than that of WN18RR. We find compared to the base model ComplEx, the improvement of AdaE on YAGO3-10 is more significant than that on WN18RR (MRR improvement 0.212 vs 0.049).

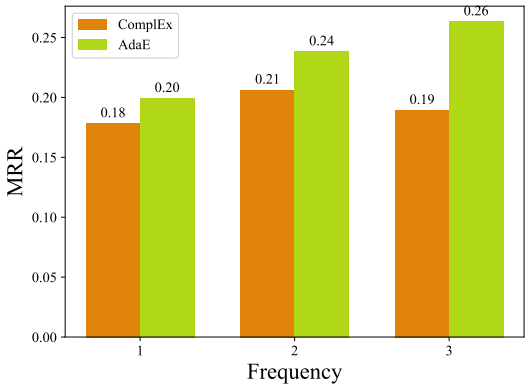


Fig. 12. Performance of entities with a frequency of 1, 2, and 3.

VI. Conclusion

In this paper, we propose an adaptive KGE model AdaE, which automatically assigns appropriate embedding dimensions to different entities. To be specific, we first propose an end-to-end differentiable framework that can navigate dimension search in a continuous integer space. Then, we tailor bilevel optimization to efficiently learn the embedding sizes and representations of entities. Finally, extensive experiments on benchmark datasets validate the effectiveness. In the future, we would like to utilize the proposed method to tackle the data imbalance issue of relations.

Acknowledgements

The research work is supported by the National Natural Science Foundation of China under Grant No.62206266.

References

[1] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.
[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 1247–1250.
[3] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 697–706.
[4] Z. Su, Z. Dou, Y. Zhu, and J.-R. Wen, "Knowledge enhanced search result diversification," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 1687–1695.
[5] F. Zhang, Z. Zhang, X. Ao, D. Gao, F. Zhuang, Y. Wei, and Q. He, "Mind the gap: Cross-lingual information retrieval with hierarchical knowledge enhancement," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 4, 2022, pp. 4345–4353.
[6] H. Ren, H. Dai, B. Dai, X. Chen, M. Yasunaga, H. Sun, D. Schuurmans, J. Leskovec, and D. Zhou, "Lego: Latent execution-guided reasoning for multi-hop question answering on knowledge graphs," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8959–8970.
[7] Z. Jia, S. Pramanik, R. Saha Roy, and G. Weikum, "Complex temporal question answering on knowledge graphs," in *Proceedings of the 30th ACM international conference on information & knowledge management*, 2021, pp. 792–802.
[8] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, and Q. He, "A survey on knowledge graph-based recommender systems," *IEEE Transactions on Knowledge and Data Engineering*, 2020.
[9] J. Zhou, B. Wang, R. He, and Y. Hou, "Crfr: Improving conversational recommender systems via flexible fragments reasoning on knowledge graphs," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 4324–4334.
[10] S. M. Kazemi and D. Poole, "Simple embedding for link prediction in knowledge graphs," *Advances in neural information processing systems*, vol. 31, 2018.
[11] W. Xiong, T. Hoang, and W. Y. Wang, "Deeppath: A reinforcement learning method for knowledge graph reasoning," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 564–573.
[12] Z. Zhang, F. Zhuang, M. Qu, F. Lin, and Q. He, "Knowledge graph embedding with hierarchical relation structure," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3198–3207.
[13] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *The International Conference on Machine Learning*, 2016, pp. 2071–2080.
[14] K. Toutanova and D. Chen, "Observed versus latent features for knowledge base and text inference," in *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, 2015, pp. 57–66.
[15] Z. Zhang, Z. Guan, F. Zhang, F. Zhuang, Z. An, F. Wang, and Y. Xu, "Weighted knowledge graph embedding," in *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023, pp. 867–877.
[16] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.
[17] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, 2013.
[18] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *International Conference on Learning Representations*, 2015.
[19] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.
[20] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proceedings of the AAAI conference on artificial intelligence*, 2014, pp. 1112–1119.
[21] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion." in *Proceedings of the AAAI conference on artificial intelligence*, 2015, pp. 2181–2187.
[22] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th*

international joint conference on natural language processing (volume 1: Long papers), 2015, pp. 687–696.

[23] H. Xiao, M. Huang, Y. Hao, and X. Zhu, "Transa: An adaptive approach for knowledge graph embedding," *arXiv preprint arXiv:1509.05490*, 2015.

[24] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," in *International Conference on Learning Representations*, 2019.

[25] G. Ji, K. Liu, S. He, and J. Zhao, "Knowledge graph completion with adaptive sparse transfer matrix," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[26] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data," in *International Conference on Machine Learning*, 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:1157792

[27] S. M. Kazemi and D. L. Poole, "Simple embedding for link prediction in knowledge graphs," in *Neural Information Processing Systems*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:3674966

[28] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Journal of Mathematics and Physics*, vol. 6, pp. 164–189, 1927. [Online]. Available: https://api.semanticscholar.org/CorpusID:124183279

[29] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*, 2018, pp. 593–607.

[30] T. D. Nguyen, D. Q. Nguyen, D. Phung *et al.*, "A novel embedding model for knowledge base completion based on convolutional neural network," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2018, pp. 327–333.

[31] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "A comparative study of data sampling and cost sensitive learning," in *2008 IEEE international conference on data mining workshops*. IEEE, 2008, pp. 46–52.

[32] G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[33] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special issue on learning from imbalanced data sets," *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 1–6, 2004.

[34] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, 2011.

[35] N. V. Chawla, D. A. Cieslak, L. O. Hall, and A. Joshi, "Automatically countering imbalance and its empirical relationship to cost," *Data Mining and Knowledge Discovery*, vol. 17, pp. 225–252, 2008.

[36] Z. Zhang, F. Zhuang, M. Qu, Z.-Y. Niu, H. Xiong, and Q. He, "Knowledge graph embedding with shared latent semantic units," *Neural Networks*, vol. 139, pp. 140–148, 2021.

[37] A. Brock, T. Lim, and N. Weston, "Smash: one-shot model architecture search through hypernetworks," *arXiv preprint arXiv:1708.05344*, 2017.

[38] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, "Neural architecture optimization," *Advances in neural information processing systems*, vol. 31, 2018.

[39] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *International conference on machine learning*. PMLR, 2018, pp. 4095–4104.

[40] A. Wan, X. Dai, P. Zhang, Z. He, Y. Tian, S. Xie, B. Wu, M. Yu, T. Xu, K. Chen *et al.*, "Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 12 965–12 974.

[41] A. Chavan, Z. Shen, Z. Liu, Z. Liu, K.-T. Cheng, and E. P. Xing, "Vision transformer slimming: Multi-dimension searching in continuous optimization space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4931–4941.

[42] Y. Liu, S. Ju, J. Wang, C. Su *et al.*, "A new feature selection method for text classification based on independent feature space search," *Mathematical Problems in Engineering*, vol. 2020, 2020.

[43] X. Zhaok, H. Liu, W. Fan, H. Liu, J. Tang, C. Wang, M. Chen, X. Zheng, X. Liu, and X. Yang, "Autoemb: Automated embedding dimensionality search in streaming recommendations," in *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2021, pp. 896–905.

[44] X. Zhao, H. Liu, H. Liu, J. Tang, W. Guo, J. Shi, S. Wang, H. Gao, and B. Long, "Autodim: Field-aware embedding dimension searchin recommender systems," in *Proceedings of the Web Conference 2021*, 2021, pp. 3015–3022.

[45] B. Yan, P. Wang, K. Zhang, W. Lin, K.-C. Lee, J. Xu, and B. Zheng, "Learning effective and efficient embedding via an adaptively-masked twins-based layer," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 3568–3572.

[46] Y. Qu, T. Chen, X. Zhao, L. Cui, K. Zheng, and H. Yin, "Continuous input embedding size search for recommender systems," in *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023, pp. 708–717.

[47] M. R. Joglekar, C. Li, M. Chen, T. Xu, X. Wang, J. K. Adams, P. Khaitan, J. Liu, and Q. V. Le, "Neural input search for large scale recommendation models," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2387–2397.

[48] S. Rendle, "Learning recommender systems with adaptive regularization," in *Proceedings of the fifth ACM international conference on Web search and data mining*, 2012, pp. 133–142.

[49] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[50] F. Mahdisoltani, J. Biega, and F. Suchanek, "Yago3: A knowledge base from multilingual wikipedias," in *7th biennial conference on innovative data systems research*. CIDR Conference, 2014.

[51] I. Balazevic, C. Allen, and T. Hospedales, "Multi-relational poincaré graph embeddings," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[52] A. Sadeghi, D. Graux, H. Shariat Yazdi, and J. Lehmann, "Mde: Multiple distance embeddings for link prediction in knowledge graphs," in *ECAI 2020*. IOS Press, 2020, pp. 1427–1434.

[53] S. Zhang, Y. Tay, L. Yao, and Q. Liu, "Quaternion knowledge graph embeddings," *Advances in neural information processing systems*, vol. 32, 2019.

[54] R. Abboud, I. Ceylan, T. Lukasiewicz, and T. Salvatori, "Boxe: A box embedding model for knowledge base completion," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9649–9661, 2020.

[55] T. Yang, L. Sha, and P. Hong, "Nage: Non-abelian group embedding for knowledge graphs," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 1735–1742.

[56] I. Balažević, C. Allen, and T. M. Hospedales, "Hypernetwork knowledge graph embeddings," in *Artificial Neural Networks and Machine Learning–ICANN 2019: Workshop and Special Sessions: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings 28*. Springer, 2019, pp. 553–565.

[57] D. Jiang, R. Wang, J. Yang, and L. Xue, "Kernel multi-attention neural network for knowledge graph embedding," *Knowledge-Based Systems*, vol. 227, p. 107188, 2021.

[58] D. Jiang, R. Wang, L. Xue, and J. Yang, "Multiview feature augmented neural network for knowledge graph embedding," *Knowledge-Based Systems*, vol. 255, p. 109721, 2022.

[59] Q. Zhang, S. Huang, Q. Xie, F. Zhao, and G. Wang, "Fair large kernel embedding with relation-specific features extraction for link prediction," *Information Sciences*, vol. 668, p. 120533, 2024.

[60] D. Li, T. Xia, J. Wang, F. Shi, Q. Zhang, B. Li, and Y. Xiong, "Sdformer: A shallow-to-deep feature interaction for knowledge graph embedding," *Knowledge-Based Systems*, vol. 284, p. 111253, 2024.

[61] T. Xie and Y. Ge, "Enhance knowledge graph embedding by mixup," *IEEE Transactions on Knowledge & Data Engineering*, vol. 36, no. 02, pp. 569–580, 2024.

[62] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of machine learning research*, vol. 12, no. 7, 2011.

[63] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[64] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *International Conference on Learning Representations*, 2016.

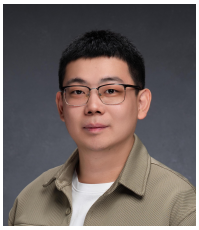
[65] Z. Zhang, J. Cai, Y. Zhang, and J. Wang, "Learning hierarchy-aware knowledge graph embeddings for link prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 03, 2020, pp. 3065–3072.

[66] S. Wang, X. Wei, C. N. Nogueira dos Santos, Z. Wang, R. Nallapati, A. Arnold, B. Xiang, P. S. Yu, and I. F. Cruz, "Mixed-curvature multi-

relational graph neural network for knowledge graph completion," in *Proceedings of the Web Conference 2021*, 2021, pp. 1761–1771.
[67] A. Rossi, D. Barbosa, D. Firmani, A. Matinata, and P. Merialdo, "Knowledge graph embedding for link prediction: A comparative analysis," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 15, no. 2, pp. 1–49, 2021.
[68] Z. Zhang, J. Cai, and J. Wang, "Duality-induced regularizer for tensor factorization based knowledge graph completion," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 604–21 615, 2020.
[69] B. Shang, Y. Zhao, and J. Liu, "Knowledge graph representation learning with relation-guided aggregation and interaction," *Information Processing & Management*, vol. 61, no. 4, p. 103752, 2024.
[70] X. Li, Y. Tian, and S. Ji, "Semantic-and relation-based graph neural network for knowledge graph completion," *Applied Intelligence*, pp. 1–23, 2024.

Zhanpeng Guan is currently pursuing his M.S. degree in the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. He recieved his B.E. degree from Xidian University of Technology in 2020. His main research interests include knowledge graph, graph neurnal network and data mining.

Fuwei Zhang received the B.S. degree from Xiamen University, Fujian, China, in 2020, and the M.S. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2023. He is currently a Ph.D. student at the Institute of Artificial Intelligence, Beihang University. His research interests include data mining and applied machine learning, with a special focus on the representation and application of knowledge graphs.



Zhao zhang received the BE degree in computer science and technology from the Beijing Institute of Technology (BIT), Beijing, China, in 2015, and the PhD degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2021. He is currently an associate professor at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. His research interests include data mining and applied machine learning, with a special focus on the representation and application of knowledge graphs.

Fuzhen Zhuang is a professor in Institute of Artificial Intelligence, Beihang University. His research interests include transfer learning, machine learning, data mining, multi-task learning and recommendation systems. He has published over 100 papers in the prestigious refereed journals and conference proceedings, such as Nature Communications, TKDE, Proc. of IEEE, TNNLS, TIST, KDD, WWW, SIGIR, NeurIPS, AAAI, and ICDE.

Fei Wang Fei Wang , born in 1988, PhD, associate professor. He received the B.S. degree in computer science from the Beijing Institute of Technology, Beijing, China, in 2011. He received the PhD degree in computer architecture from Institute of Computing Technology, Chinese Academy of Sciences in 2017. From 2017 to 2020, he was a research assistant with the Institute of Technology, Chinese Academy of Sciences. Since 2020, he has been working as associate professor in Institute of Computing Technology, Chinese Academy of Sciences. His main research interest includes spatiotemporal data mining, Information fusion, graph neural networks.

Zhulin An Zhulin An received the B.Eng. and M.Eng. degrees in computer science from Hefei University of Technology, Hefei, China, in 2003 and 2006, respectively and the Ph.D. degree from the Chinese Academy of Sciences, Beijing, China, in 2010. He is currently with the Institute of Computing Technology, Chinese Academy of Sciences, where he became a Senior Engineer in 2014. His current research interests include optimization of deep neural network and lifelong learning.

Yongjun Xu is a professor at Institute of Computing Technology, Chinese Academy of Sciences (ICT-CAS) in Beijing, China. He received his B.Eng. and Ph.D. degree in computer communication from Xi'an Institute of Posts Telecoms (China) in 2001 and Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China in 2006, respectively. His current research interests include artificial intelligence systems, and big data processing.