DataType-Aware Knowledge Graph Representation Learning in Hyperbolic Space

Yuxin Shen[1,2,†], Zhao Li[1,2,†], Xin Wang[1,2,*], Jianxin Li[3], Xiaowang Zhang[1,2]
[1] College of Intelligence and Computing, Tianjin University, Tianjin, China
[2] Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin, China
[3] School of Information Technology, Deakin University, Geelong, Victoria, Australia
{shenyuxin,lizh,wangx,xiaowangzhang}@tju.edu.cn,jianxin.li@deakin.edu.cn

ABSTRACT

Knowledge Graph (KG) representation learning aims to encode both entities and relations into a continuous low-dimensional vector space. Most existing methods only concentrate on learning representations from structural triples in Euclidean space, which cannot well exploit the rich semantic information with hierarchical structure in KGs. In this paper, we propose a novel *DataType-aware* hyperbolic knowledge representation learning model called DT-GCN, which has the advantage of fully embedding attribute values of data types information. We *refine* data types into five primitive modalities, including integer, double, Boolean, temporal, and textual. For each modality, an encoder is specifically designed to learn its embedding. In addition, we define a *unified space* based on Euclidean, spherical, and hyperbolic space, which is a *continuous curvature space* that combines advantages of three different spaces. Extensive experiments on both synthetic and real-world datasets show that our model is consistently better than the state-of-the-art models. The average performance is improved by 2.19% and 3.46% than the optimal baseline model on node classification and link prediction tasks, respectively. The results of ablation experiments demonstrate the advantages of embedding data types information and leveraging the unified space.

CCS CONCEPTS

• **Computing methodologies → Machine learning**; **Machine learning approaches**; **Learning latent representations**.

KEYWORDS

Knowledge Graph; Representation Learning; Data Type; Hyperbolic Space

ACM Reference Format:
Yuxin Shen, Zhao Li, Xin Wang, Jianxin Li, and Xiaowang Zhang. 2021. DataType-Aware Knowledge Graph Representation Learning in Hyperbolic Space. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM ’21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482421

† Both authors contributed equally to this research.
* Xin Wang is the corresponding author.

1 INTRODUCTION

As the latest achievement in the development of symbolism, knowledge graph (KG) is an important research direction of artificial intelligence. In recent years, a large number of KGs, such as DBpedia [20], Freebase [2], NELL [4], and WordNet [27], have been constructed in both academia and industry, and have become fundamental data sources for basic Natural Language Processing (NLP) tasks, such as semantic analysis, entity disambiguation, information extraction, and question answering [36]. The Resource Description Framework (RDF) standard recommended by the World Wide Web Consortium (W3C) [8] is a data format for describing network resources on the Web. In RDF, a KG is represented as a set of triples. In each triple (s, p, o), s is the subject, p is the predicate, and o is the object. Thus, a KG encodes structural information of entities and their relations in terms of triples to form a type of network with rich semantics, e.g, the KG shown in Figure 1.

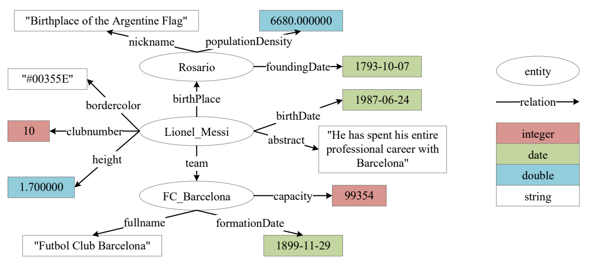


Figure 1: A KG with attribute values of various data types.

Knowledge graph representation learning aims to embed entities and relations as continuous low-dimensional vectors, which can be stored and computed efficiently. By using these vector representations, the semantic associations of entities and relations can be efficiently represented, and issues of computational efficiency and data sparsity can also be effectively addressed. Recently, a large number of models have been proposed to realize knowledge representation learning. TransE [3] is a representative KG embedding approach, and several enhanced models, including TransH [37], TransR [22], and TransD [15], have been proposed to improve the ability of KGs in prediction and reasoning. However, there still exists several challenges in KG embedding methods.

Challenge I. Fail to learn attribute values of data types. Most of existing KG embedding approaches only model structural information, while ignoring the rich semantic information, such as entity descriptions, types, and attributes. For example, Figure 1 shows a KG with attribute values of data types, which, in fact, are a kind of useful semantic information. Meanwhile, these semantic information can help handle the issue of data sparsity, and thus improve the performance of knowledge representation learning.

Challenge II. Fail to utilize the hierarchical information of KGs. Currently, the mainstream models on representation learning focus on embedding entities and relations into Euclidean space, which need more dimensions to represent the hierarchical information. However, hyperbolic space can better deal with this issue than Euclidean space, and plays an important role in modeling the hierarchical information in KGs.

To overcome the above challenges, we introduce *DataType-aware Graph Convolutional Network* (*DT-GCN*) in hyperbolic space to utilize data types as much as possible. Our DT-GCN model is elaborately designed to encode the data types information so that this information can be seamlessly fused into the embedding without affecting the original model. Moreover, we explore a *unified space* built on three kinds of spaces, including Euclidean, spherical, and hyperbolic space, which can flexibly adjust the proportion of the embedded spaces to achieve a better performance.

The contributions of this paper can be summarized as follows.

- We rationally *refine* the attribute values of *data types*, and design unique strategies and encoders for different data types to seamlessly embed their semantic information into the model, which effectively improves the performance of knowledge representation learning.
- On the basis of Euclidean, spherical, and hyperbolic space, we define a *unified continuous curvature space*, which fully combines advantages of other three spaces. In this unified space, the knowledge embedding model can flexibly adjust the proportion of different spaces used to adequately and efficiently capture the structural information in a knowledge graph to enhance the representation learning ability.
- Extensive experiments conducted on both synthetic and real-world datasets show that our DT-GCN can significantly outperform existing models on the tasks of link prediction and node classification.

The rest of this paper is organized as follows. We introduce preliminaries in Section 3. We describe our approach in detail in Section 4, and report experimental results in Section 5. Finally, we conclude this paper in Section 6.

2 RELATED WORK

In this section, we introduce existing methods that are related to our research, including translation-based, semantic matching, multi-source information, and hyperbolic models.

2.1 Translation-Based Models

Translation-based models apply the distance-based scoring function. The basic idea is that a relation between a head and a tail entity corresponds to their translation in vector space, and the distance between the two entities is used to measure the truth of the triple.

TransE [3] is the baseline work of the translation-based models, and a series of subsequent works are based on TransE. However, TransE cannot well model 1-to-N, N-to-1, and N-to-N relations. In order to solve this problem, TransH [37] and TransR [22] project the head and tail entities to the hyperplace and hyperspace corresponding to each relation, respectively, then employ the same method as TransE for training. TransD [15] and TranSparse [16] simplify TransR from two different perspectives. TransD decomposes the projection matrix of a relation into the product of two vectors, while TranSparse imposes sparsity on the projection matrix.

Another research thread for this problem is to relax requirements of the vector transformation in TransE. TransF [10] loosens the restriction of the translation and adopts the dot product operation instead of the addition operation to balance the constraints of head and tail entities. TransA [41] uses the Mahalanobis distance to realize adaptive parameter learning, and TransM [9] assigns a specific relation weight to each triple to distinguish different relation types. Moreover, RotatE [33] defines a relation as a rotation from a head entity to a tail entity in complex space. ManifoldE [42] extends the point model to the manifold model, and performs the translation operation on the hyperplane, which improves the accuracy of embeddings. The translation-based models can represent head and tail entities well, whereas, these methods lack representations of rich semantic information in relations and attributes.

2.2 Semantic Matching Models

Semantic matching models make use of the similarity-based scoring function, and the basic idea is to match the latent semantics of entities and relations to achieve the authenticity of a triple.

RESCAL [30] represents an entity as a vector, and a relation as a matrix which models the pairwise interactions between potential factors. However, RESCAL is prone to over-fitting, and as the dimension of the relation matrix increases, the computation overhead becomes prohibitively high. To address this issue, DistMult [45] relaxes the constraint on the relation matrix and defines it as a diagonal matrix. However, DistMult oversimplifies RESCAL and can only deal with symmetric relations, but cannot handle other types of relations in KGs. HolE [29] combines the expressive power of RESCAL with the simplicity of DistMult, using circular operations to combine entity vectors, and matching them with relations, which makes the model more efficient and can also process symmetric and asymmetric relations. ComplEx [34] extends DistMult by introducing complex numbers, so that it can model not only symmetric and asymmetric relations, but also more complex symmetric relations. SimplE [17] is an enhancement of the canonical polyadic decomposition, which allows learning two embeddings of each entity independently. ANALOGY [23] extends RESCAL to model the analogical attributes of entities and relations. The semantic matching models further consider the relation information, however, it still fails to integrate the attribute information.

2.3 Multi-Source Information Models

In addition to using the triple structure for representation learning, multi-source information can also be integrated into the representation learning model, which mainly includes textual descriptions, entity types, and numeric literals.

2.3.1 Textual Descriptions. DKRL [43] uses CBOW to add word vectors in the text as text representations, and applies CNN to consider the word order information. The model in [47] represents text descriptions as a sequence of words, modeled by LSTM, and combines structure-based and description-based representations by gating mechanism. However, these two models still have difficulty in modeling strong associations between texts and triples. To deal with this problem, SSP [40] models the strong correlations between triples and text descriptions by performing the embedding process in a semantic subspace. Based on co-training, KDCoE [7] adopts the Attentive Gated Recurrent Unit encoder (AGRU) and multi-lingual word embeddings to characterize entity description and enhance the semi-supervised learning of multilingual KG embeddings.

2.3.2 Entity Types. TKRL [44] represents each relation type and its subtypes as a projection matrix, and constructs the projection matrix using the recursive and weighted hierarchical coding. TransT [26] can fuse the structural information and type information, construct multiple relation types from entity types, design the semantic similarity based on type for multiple embedded representations and prior knowledge, and propose a multiple embedding method, which represents each entity as multiple vectors with specific semantics. E2T and TRT [46] are proposed to handle the issue of incomplete entity types. E2T projects entities from entity space to relation space through the projection matrix, and TRT constructs a new entity type ternary group by replacing the head and tail entities in a triple with their types. By this way, the model can make full use of the global triple information in KGs. APR [25] is based on the hierarchical structure of entity types, and path patterns are generated from the data to infer the missing information in KGs by using LSTM to encode relation and entity type information.

2.3.3 Numeric Literals. RolEANE [21] employs a neighbor optimization strategy to modify the Skip-Gram model, which can seamlessly integrate the network topological structure and attribute information to improve the representation learning performance. KBLRN [12] utilizes the probabilistic Product of Experts method to combine relational, potential, and numerical features. By using the rule mining method AMIE+, each relation feature is formulated into a logical formula. The potential features are generated by using embedding methods such as DistMult. Assuming that some relation types leverage digital features, the difference between the head and tail entities is regarded as relation features. LiteralE [19], based on GRU, incorporates the numerical information into DistMult, ComplEx, and ConvE by replacing the entity representation with the representation of the numerical information. TransEA [39] includes two parts: structure embeddings and attribute embeddings, where attribute embeddings utilize attribute triples which contain the numerical information as inputs, and apply a linear regression model to learn entity and attribute embeddings. Although the multi-source information models integrate several aspects of attributes, the data types information is still ignored.

2.4 Hyperbolic Models

Recently, the non-Euclidean embedding has been exploited in several works as an effective method for learning representations of hierarchical data. Nickel [28] first applies Poincaré ball in hyperbolic geometry for the link prediction to model hierarchical structures. MuRP [1] is a translation-based model for multi-relational graph data, which can learn specific relation parameters by transforming entity embeddings through Möbius matrix vector multiplication and addition. However, the model still uses translation-based methods, and cannot learn complex relational patterns.

To address this issue, ATTH [5] is proposed to learn the curvature of a specific relation and leverage different curvatures to represent different degrees of bending in hyperbolic space. ATTH parameterizes the hyperbolic contour and applies its geometric features to capture logical patterns, such as symmetry and anti-symmetry. Also, ATTH utilizes the hyperbolic attention mechanism to combine geometric operators with captured logical patterns. Furthermore, based on the translation-based models, HyperKG is proposed to better reflect topological characteristics of knowledge bases in hyperbolic space. More recently, HGCN [6] and HGNN [24] generalize GNN to be manifold-agnostic and show that hyperbolic graph neural networks can provide substantial improvements for full-graph classification. HyperKA [32] is a knowledge association method, which combines the idea of TransE and GNN with hyperbolic space. In the input layer, hyperbolic TransE and GNN are used to perform relation transformations and neighbor aggregations, so as to obtain entity representations. Thus, the model does not need to aggregate separately according to different relations. Although the hyperbolic models exhibit high performance in representing hierarchical data, we are not aware any existing work that considers data types information in non-Euclidean space.

3 PRELIMINARIES

In this section, we present the preliminaries, including Riemannian manifold, hyperbolic geometry, and data type refinement.

3.1 Riemannian Manifold

An n-dimension Riemannian manifold $\mathcal{M}$ is a real and smooth manifold, which can locally be approximated by $\mathbb{R}^n$: it is a higher-dimensional generalization of the 2D surface. For each point $\mathbf{x} \in \mathcal{M}$, a Riemannian metric on $\mathcal{M}$ is a positive-definite inner product $g_{\mathbf{x}} = \langle \cdot, \cdot \rangle_{\mathbf{x}} : \mathcal{T}_{\mathbf{x}}\mathcal{M} \times \mathcal{T}_{\mathbf{x}}\mathcal{M} \to \mathbb{R}$, where $\mathcal{T}_{\mathbf{x}}\mathcal{M}$ is the tangent space of $\mathcal{M}$ at $\mathbf{x}$. A Riemannian manifold $(\mathcal{M}, g)$ is a manifold $\mathcal{M}$ equipped with a Riemannian metric g.

From the tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$, there exists an exponential mapping function $\exp_{\mathbf{x}}(\mathbf{v}) : \mathcal{T}_{\mathbf{x}}\mathcal{M} \to \mathcal{M}$ that maps a tangent vector $\mathbf{v}$ to the manifold $\mathcal{M}$, and conversely, the logarithmic mapping function $\log_{\mathbf{x}}(\mathbf{v}) : \mathcal{M} \to \mathcal{T}_{\mathbf{x}}\mathcal{M}$ maps $\mathbf{v}$ from $\mathcal{M}$ to $\mathcal{T}_{\mathbf{x}}\mathcal{M}$.

3.2 Hyperbolic Geometry

In this subsection, we introduce some basic operations of hyperbolic geometry, including vector translation, matrix-vector multiplication, transformation, and Hadamard product.

Vector translation. The vector translation in the Poincaré ball is defined by Möbius addition:

$$\mathbf{x} \oplus^c \mathbf{y} = \frac{(1 + 2c \langle \mathbf{x}, \mathbf{y} \rangle + c\|\mathbf{y}\|^2)\mathbf{x} + (1 - c\|\mathbf{x}\|^2)\mathbf{y}}{1 + 2c \langle \mathbf{x}, \mathbf{y} \rangle + c^2\|\mathbf{x}\|^2\|\mathbf{y}\|^2} \tag{1}$$

Matrix-vector multiplication. The Möbius matrix-vector multiplication in the Poincaré ball is defined as:

$$\mathbf{M} \otimes^c \mathbf{x} = \exp_0^c(\mathbf{M} \log_0^c(\mathbf{x})) \tag{2}$$

Transformation. The exponential map and logarithmic map on vector $\mathbf{x}$ are defined on the origin:

$$\exp_{\mathbf{0}}^c(\mathbf{x}) = \tanh(\sqrt{c}\|\mathbf{x}\|)\frac{\mathbf{x}}{\sqrt{c}\|\mathbf{x}\|} \tag{3}$$

$$\log_{\mathbf{0}}^c(\mathbf{x}) = \operatorname{arctanh}(\sqrt{c}\|\mathbf{x}\|)\frac{\mathbf{x}}{\sqrt{c}\|\mathbf{x}\|} \tag{4}$$

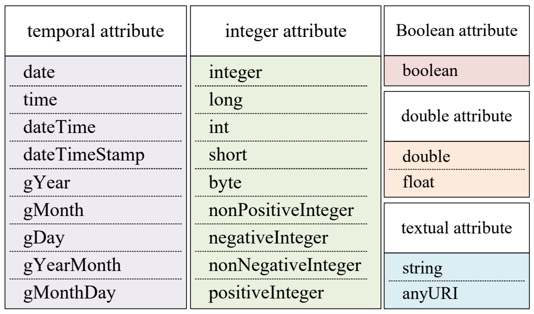
Hadamard product. Hadamard product is also known as pointwise product, which is defined by the square diagonal matrix:

$$\mathbf{x} \odot^c \mathbf{y} = \operatorname{diag}(\mathbf{x}) \otimes^c \mathbf{y} \tag{5}$$

3.3 Data Type Refinement

XML schema[1] is also known as XML Schema Definition (XSD), is the successor of DTD. Based on XML, XML schema can be used to determine the structure, order, data value, and the range of elements and attributes in XML documents.

By encoding different data types w.r.t. their specific characteristics, knowledge representation learning model can preserve more semantic information of the original KG. For the sake of simplicity, we *merge* data types, and *refine* these data types as integer, double, Boolean, temporal, and textual attributes (as shown in Figure 2). The notations used in our paper are summarized in Table 1.



temporal attribute	integer attribute	Boolean attribute
date	integer	boolean
time	long	double attribute
dateTime	int	double
dateTimeStamp	short	float
gYear	byte	textual attribute
gMonth	nonPositiveInteger	string
gDay	negativeInteger	anyURI
gYearMonth	nonNegativeInteger	
gMonthDay	positiveInteger	

Figure 2: Data Type Refinement.

4 OUR APPROACH

The key idea of DT-GCN is to more accurately represent the semantic information of data types in hyperbolic space. In this section, we first introduce the framework of DT-GCN. Then, we describe the general and dedicated encoders designed for each data type in detail. Finally, the definition of the unified space is given.

4.1 Overall Architecture

In this work, we consider five different types of attributes, including integer, double, Boolean, temporal, and textual attributes, and the specific refinement are shown in Figure 2.

Judging from the characteristics of numerical types, the similarity of integer, double, Boolean, and temporal attributes is that they are all composed of numbers. More specifically, double numbers add a fractional part on the basis of integers. A Boolean number is a special integer whose value is 0 or 1. The year, month, date, and time in temporal attributes are also numbers showing a certain regularity. For example, 7 is an integer, 7.0 is a double number, 1 is a Boolean number, and 07-01 is the temporal value. Therefore, we regard the integer attribute as the basis and use different encoders to distinguish double, Boolean, and temporal attributes, respectively. Besides the four numerical types, our model also considers more general textual attributes such that all literal values can be handled. Nevertheless, the existing models, e.g., LiteralE [19], TransEA [39], and KBLRN [12], treat values of data types `date`, `time`, `integer`, `double`, and `boolean` as numerical values, and use the same method to represent attributes of different data types, which cannot distinguish data types well due to the coarse-grained representations. Figure 3 shows an illustration of our model DT-GCN, which has two significant advantages at the theoretical level.

Data type refinement. Compared with the previous works, attribute values of data types are refined in our model in more detail, and expressed more precisely through the general and dedicated encoders. (1) In general encoders, we design a neural network (embedding layer) to embed the complex and potential non-linear information and construct the relation-aware representations of entities by considering the relation information (relation-aware layer). Note that the integer attribute is the basis of other attributes. Therefore, the output of the embedding layer can be used as the embedding results for the integer attributes. (2) In dedicated encoders, we use the attention mechanism (attention layer) to calculate the importance of double attributes to improve the accuracy of their representations. (3) Moreover, we utilize the Gated Recurrent Unit (GRU layer) to decide whether Boolean attributes should be contained in the results. (4) Furthermore, we define a bidirectional GRU (BiGRU layer) to effectively process textual attributes. (5) Finally, we apply a multi-layer feedforward neural network (FNN layer) with unfixed number of layers to capture temporal attributes.

The unified space. We explore the unified space built on Euclidean, spherical, and hyperbolic space and perform our model on

Table 1: Notations and Explanations.

Notation	Explanation
$\oplus^c$	Möbius addition
$\otimes^c$	Möbius matrix-vector multiplication
$\odot^c$	Hadmard product
$\mathbf{h}$	Initial node feature
$\mathbf{r}_{ij}$	Relation embedding
$\mathbf{t}_{ij}$	Attribute embedding
$\mathbf{c}_{ij}$	Relation-aware embedding
e_{ij}	Importance of the relation-aware attribute
α_{ij}	Attention value
$\mathbf{W}, \mathbf{U}$	Transformation matrices
$\mathbf{z}_t, \mathbf{r}_t$	Update gate and reset gate
$\mathbf{H}, \widetilde{\mathbf{H}}$	The old and new hidden states
$\overrightarrow{\mathbf{H}}, \overleftarrow{\mathbf{H}}$	Output of the forward and backward GRU
$\mathbf{h}^z, \mathbf{h}^f, \mathbf{h}^b, \mathbf{h}^w, \mathbf{h}^s$	The embedding results for five attributes

[1]https://www.w3.org/TR/xmlschema11-2

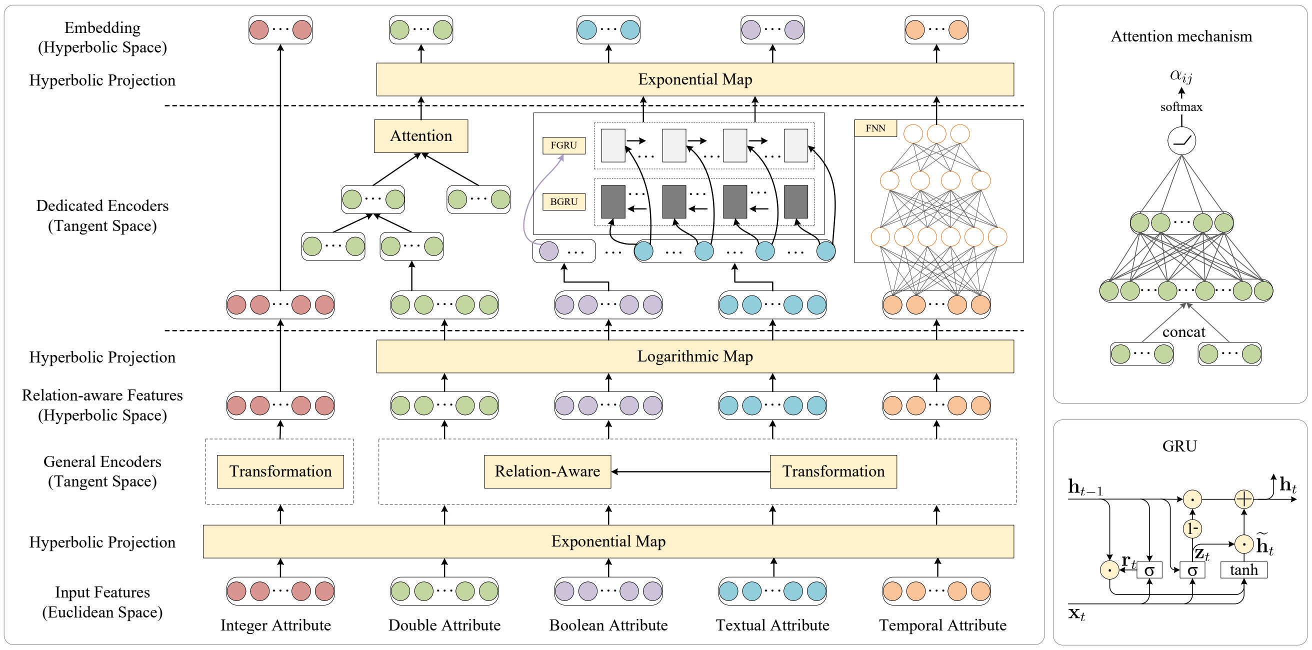


Figure 3: The framework of the DT-GCN model.

it to verify the significance of continuous curvatures. The embedding process for each data type is performed in the unified space we defined, which can efficiently capture the rich hierarchical information in KGs and improve the representation learning ability of our model. In addition, the proportion of different spaces can be flexibly adjusted in the unified space to seamlessly integrate the embeddings obtained in the three spaces, and taking full advantage of the continuous curvature space.

4.2 General Encoders

All the entities, relations, and attributes will be input into the general encoders, and the basic representations of five attributes are obtained through the embedding layer, among which the representation of the integer attribute is the final embddings. The representations of the remaining four attributes (double, Boolean, textual, and temporal) are obtained through the relation-aware layer, which is used as the input of the dedicated encoders, then the final embeddings are obtained.

4.2.1 Embedding Layer. Formally, we represent a knowledge graph as $\mathcal{G} = (N, R, T)$, where N, R, and T indicate the sets of nodes, edges, and triples, respectively. Specifically, the node set of a KG can be further formalized as $N = E \cup A$, where E represents the set of entities, and A represents the set of attributes. Therefore, the triple set T in a KG can be written as $T = \{(h, r, t) \mid h \in E, r \in R, t \in N\}$. Our knowledge graph representation learning model DT-GCN try to learn the effective representations of node features.

Given a KG $\mathcal{G}$ and input the node features $\mathbf{h}$ in Euclidean space, we first apply the exponential project function $\exp_{\mathbf{0}}^c(\cdot)$ to map $\mathbf{h}$ into hyperbolic space. To capture more complex and potential non-linear information, we introduce an embedding layer to non-linearly transform the node initializations into higher-level features. For the node h, we put it to the embedding layer as follows:

$$\mathbf{h} = \tanh((\mathbf{W} \otimes^c \mathbf{h}) \oplus^c \mathbf{b}) \tag{6}$$

where $\mathbf{W}$ and $\mathbf{b}$ denote the weight matrix and bias vector, while $\otimes^c$ and $\oplus^c$ denote Möbius matrix-vector multiplication and addition.

We take the output of the embedding layer as the final embeddings $\mathbf{h}^z$ for integer attributes of the data types: `integer`, `int`, `nonNegativeInteger`, `nonPositiveInteger`, `negativeInteger`, `positiveInteger`, `short`, `long`, and `byte`.

4.2.2 Relation-Aware Layer. According to TransE [3], given a triple (h, r, t), it assumes $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$. We apply it in our triples, thus, we represent embeddings for entities as $\{-\mathbf{r}_1 \oplus^c \mathbf{t}_1, \cdots, -\mathbf{r}_n \oplus^c \mathbf{t}_n\}$, and attribute values as $\{\mathbf{h}_1 \oplus^c \mathbf{r}_1, \cdots, \mathbf{h}_n \oplus^c \mathbf{r}_n\}$ in hyperbolic space.

After the preprocessing, we construct two cases of representations of head entities, $\mathbf{h}$ and $-\mathbf{r} \oplus^c \mathbf{t}$, corresponding to two cases of representations of attributes $\mathbf{h} \oplus^c \mathbf{r}$ and $\mathbf{t}$. In this way, we can not only learn entity and attribute representations, but consider relation representations. Since entities have attributes, rather than attributes have entities in KG triples, we only consider the first case, i.e., using relations and attributes to construct the entity representations.

In order to reflect the superiority of hyperbolic space to capture semantic information, after the transformation of node features and construction of relation-aware representation in hyperbolic space, we apply the logarithmic transformation function $\log_{\mathbf{0}}^c(\cdot)$ to process nodes into the tangent space.

4.3 Dedicated Encoders

Dedicated encoders include attention, GRU, BiGRU, and FNN layers. And float, Boolean, textual, and temporal attributes are processed in each layer, respectively. Note that, each layer only deals with its corresponding attributes and ignore other attributes.

4.3.1 *Attention Layer.*

The attention layer learns the importance of neighbors and aggregates neighboring messages according to their importance of the center node. Double numbers have more accurate representations than integers, e.g., Los Angeles has a population of 4.087 million, but if we use an integer to represent it, i.e., 4 million, the representation could be inaccurate. Thus, we use the attention mechanism to determine the contribution of double attributes of data types `double` and `float`.

As aforementioned in Section 4.2.2, for a specific triple (h_i, r_{ij}, t_{ij}), we apply TransE [3] to construct the representation of h_i by the relation r_{ij} and the attribute t_{ij}. We learn relation-aware embeddings $\mathbf{c}_{ij}$ by performing a linear transformation over the concatenation of the entity h_i, and its relation-aware attributes $-r_{tj} \oplus^c t_{ij}$ can be formulated as follows:

$$\mathbf{c}_{ij} = \mathbf{W}_1 \otimes^c [\mathbf{h}_i \| (-\mathbf{r}_{ij} \oplus^c \mathbf{t}_{ij})] \tag{7}$$

where $\mathbf{W}_1$ denotes the linear transformation matrix, t_{ij} represents the attribute of entity h_i with relation r_{ij}, $\|$ denotes the concatenation operation, while $\mathbf{h}_i$, $\mathbf{r}_{ij}$, and $\mathbf{t}_{ij}$ are embeddings of h_i, r_{ij}, and t_{ij}, respectively. Then, we learn the importance of relation-aware attribute $-r_{tj} \oplus^c t_{ij}$. We perform a linear transformation on $\mathbf{c}_{ij}$ by weight matrix $\mathbf{W}_2$ and a LeakyReLU activate function. The attention coefficient is calculated as follows:

$$e_{ij} = \exp_{\mathbf{0}}^{c}(\mathrm{LeakyReLU}(\log_{\mathbf{0}}^{c}(\mathbf{W}_2 \otimes^c \mathbf{c}_{ij}))) \tag{8}$$

To make the coefficients between different nodes easy to compare, we normalize them by using the softmax function, and calculate the attention value as follows:

$$\alpha_{ij} = \mathrm{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})} \tag{9}$$

where n denotes the number of neighbor attributes of the entity h_i. The entity embedding can be calculated by the sum of each relation-aware embedding weighted by the corresponding attention value as follows:

$$\mathbf{h}_i^f = \sigma^c\Big(\sum_{j=1}^{n} \alpha_{ij} \otimes^c \mathbf{c}_{ij}\Big) \tag{10}$$

In addition, we consider the multi-head attention mechanism which can aggregate more information about neighbors. The entity embedding with the M-head attention is calculated as follows:

$$\mathbf{h}_i^f = \Big\|_{m=1}^{M} \sigma^c\Big(\sum_{j=1}^{n} \alpha_{ij}^m \otimes^c \mathbf{c}_{ij}^m\Big) \tag{11}$$

4.3.2 *GRU Layer.*

In this section, the gating mechanism, having the ability to remove or add information to states, is well-designed to embed Boolean attributes of the data type `boolean`. The characteristics of gating mechanism satisfy the need to determine whether `true` or `false` information should be propagated to the next state, which can achieve the effect of flexibly using attribute information.

We first introduce the Euclidean GRU, and then extend it to hyperbolic space to define the hyperbolic GRU. GRU contains two gates, i.e., the update gate and the reset gate, which are used to control the flow of information inside each state. The update gate determines the information that flows into the future, while the reset gate controls the amount of the past information needs to be forgotten.

We denote the input vector at step t as $\mathbf{h}_t$, and the hidden state at step $t-1$ as $\mathbf{H}_{t-1}$. $\mathbf{h}_t$ and $\mathbf{H}_{t-1}$ are transformed by weight matrices $\mathbf{W}_z$ and $\mathbf{U}_z$, respectively, and then activated with the sigmoid function σ to calculate the update gate $\mathbf{z}_t$ as follows:

$$\mathbf{z}_t = \sigma(\mathbf{W}_z\mathbf{h}_t + \mathbf{U}_z\mathbf{H}_{t-1}) \tag{12}$$

The calculation method of the reset gate $\mathbf{r}_t$ is similar to the update gate, and the weight matrices are $\mathbf{W}_r$ and $\mathbf{U}_r$. Therefore, we can calculate the reset gate $\mathbf{r}_t$ as follows:

$$\mathbf{r}_t = \sigma(\mathbf{W}_r\mathbf{h}_t + \mathbf{U}_r\mathbf{H}_{t-1}) \tag{13}$$

Then, the new memory, i.e., the new hidden state $\widetilde{\mathbf{H}}_t$, applies the reset gate to store the related information from the past. Thus, $\widetilde{\mathbf{H}}_t$ at step t is calculated as follows:

$$\widetilde{\mathbf{H}}_t = \tanh(\mathbf{W}_h\mathbf{h}_t + \mathbf{U}_h(\mathbf{r}_t \odot \mathbf{H}_{t-1})) \tag{14}$$

where tanh and $\odot$ are the activate function and the Hadamard product, respectively. The multiplication determines the previous information will be retained or forgotten. For example, a gate value 0 of a certain element means the information of this element has been completely forgotten.

The final memory $\mathbf{h}_t^b$ at current step t utilizes the update gate to determine the current memory $\widetilde{\mathbf{H}}_t$, and the information that needs to be collected at step $t-1$. To be more specific, $\mathbf{h}_t^b$ contains the information of the current state, and passes it to the next state, which can be calculated as follows:

$$\mathbf{h}_t^b = \mathbf{z}_t \odot \widetilde{\mathbf{H}}_t + (1-\mathbf{z}_t) \odot \mathbf{H}_{t-1} \tag{15}$$

According to the basic operations of hyperbolic geometry defined in Section 3.2, we define the hyperbolic GRU as follows:

$$\begin{aligned}
\mathbf{z}_t &= \sigma^c((\mathbf{W}_z \otimes^c \mathbf{h}_t) \oplus^c (\mathbf{U}_z \otimes^c \mathbf{H}_{t-1}))\\
\mathbf{r}_t &= \sigma^c((\mathbf{W}_r \otimes^c \mathbf{h}_t) \oplus^c (\mathbf{U}_r \otimes^c \mathbf{H}_{t-1}))\\
\widetilde{\mathbf{H}}_t &= \tanh^c((\mathbf{W}_h \otimes^c \mathbf{h}_t) \oplus^c (\mathbf{U}_h(\mathrm{diag}(\mathbf{r}_t) \otimes^c \mathbf{H}_{t-1}))\\
\mathbf{h}_t^b &= (\mathrm{diag}(\mathbf{z}_t) \otimes^c (-\mathbf{H}_{t-1} \oplus^c \widetilde{\mathbf{H}}_t)) \oplus^c \mathbf{H}_{t-1}
\end{aligned} \tag{16}$$

where σ^c, $\tanh^c$, and $\mathrm{diag}(\mathbf{r_t})$ denote the Möbius sigmoid function, activate function, and the square diagonal matrix of $\mathbf{r}_t$, respectively.

4.3.3 *BiGRU Layer.*

The BiGRU layer is designed for textual attributes of data types `string` and `anyURI`. The previous works [43, 47] concentrate on CBOW, CNN, and LSTM, which only encode strings in a single direction, and employ complicated tensor operations. BiGRU embeds strings in both forward and backward directions, which can capture more information contained in textual attributes with fewer tensor operations and less training time.

The BiGRU consists of two ordinary GRU, which process strings from two directions, including chronological and anti-chronological, named forward GRU and backward GRU. At step t, the hidden layer output $\mathbf{h}_t$ of the BiGRU can be calculated as follows:

$$\begin{aligned}
\overrightarrow{\mathbf{H}}_t &= \sigma^c((\mathbf{W}_b \otimes^c \mathbf{h}_t) \oplus^c (\mathbf{U}_b \otimes^c \overrightarrow{\mathbf{H}}_{t-1}))\\
\overleftarrow{\mathbf{H}}_t &= \sigma^c((\mathbf{W}_b \otimes^c \mathbf{h}_t) \oplus^c (\mathbf{U}_b \otimes^c \overleftarrow{\mathbf{H}}_{t+1}))
\end{aligned} \tag{17}$$

where $\overrightarrow{\mathbf{H}}_t$ and $\overrightarrow{\mathbf{H}}_{t-1}$ denote the output of the forward GRU at time t and $t-1$, $\overleftarrow{\mathbf{H}}_t$ and $\overleftarrow{\mathbf{H}}_{t+1}$ denote the output of the backward GRU

Table 2: Statistics of Datasets. The data types which are not included in the datasets are represented by −.

Dataset	#entities	#relations	#train	#valid	#test	#integer	#double	#Boolean	#temporal	#textual
YAGO-10	123,182	44	1,345,752	6,214	16,970	–	–	–	111,406	107,326
FB15k-237	14,541	237	272,115	17,535	20,466	63,550	5,604	–	1,373	15,633
WN18RR	40,943	11	86,835	3,034	3,134	–	–	–	–	–
DBpedia-literals	9,142	171	65,960	10,870	10,989	21,111	10,695	37	35,647	–

at time t and $t-1$, while $\mathbf{W}_b$ and $\mathbf{U}_b$ are two weight matrices, respectively. Then, we merge them together to get the final output as follows:

$$\mathbf{h}_t^w = \mathbf{W}_f \otimes^c [\overrightarrow{\mathbf{H}}_t \| \overleftarrow{\mathbf{H}}_t] \tag{18}$$

where $\mathbf{h}_t^w$ represents the entity embedding of the BiGRU layer, and $\mathbf{W}_f$ is the weight matrix.

4.3.4 FNN Layer. The feedforward neural network is designed for the temporal attributes of data types `time`, `date`, `dateTime`, `gYearMonth`, `gYear`, `gMonthDay`, `gDay`, `gMonth`, and `dateTimeStamp`.

We define the temporal hierarchy as century, decade, year, quarter, month, week, day, hour, minute, and second with the maximum sizes 100, 10, 10, 4, 3, 5, 7, 24, 60, and 60, respectively. Note that the size of week is 5 since four weeks cannot completely represent all the days in a month. In addition, we limit the range of the year to $[-9999, 9999]$. For example, for the date April 18, 2010, we split it into: the first year of the second decade in the 21st century, the first month of the second quarter, and the fourth day of the third week, which can be written as $(21, 2, 1, 2, 1, 3, 4)$.

We use a multi-layer feedforward neural network with unfixed number of layers to represent the temporal hierarchy. Since we have defined 10 temporal levels, we set the maximum number of layers to 10 and define a weight matrix $\mathbf{W}_i$ for each layer. The level of the temporal attribute determines the feedforward neural network that the attribute can be forwarded. In this way, we can learn the representation of temporal attributes $\mathbf{h}^s$ flexibly.

4.4 The Unified Space

There are three different types of curvatures, including a zero curvature, a positive curvature, and a negative curvature, corresponding to Euclidean, spherical, and hyperbolic space, respectively. In this work, we define the curvature as a non-negative number ($c > 0$), thus, the curvature of three spaces are 0, c, and $-c$, respectively. For feasible gradient optimizations, we choose the *Poincaré ball model* for negative curvatures. The characteristics and definitions of three spaces are shown in Table 3. $\langle \cdot, \cdot \rangle_2$ denotes the standard Euclidean inner product.

We define the *unified space* based on the product of manifolds, which closely combines three spaces of Euclidean, spherical, and hyperbolic to construct a *continuous curvature space*. The product of manifolds is defined as the Cartesian product:

$$\mathcal{M} = \alpha\mathcal{M}_1 \times \beta\mathcal{M}_2 \times \gamma\mathcal{M}_3 \tag{19}$$

where $\mathcal{M}_i$ is the sequence of smooth manifolds with curvature c, while α, β, and γ are the number of manifolds for Euclidean, spherical, and hyperbolic space, respectively. Note that, the embedding dimension should be divisible by the sum of α, β, and γ. Thus, the restrictions on the number of manifolds can be written as follows:

Table 3: Characteristics and definitions of Euclidean ($\mathbb{E}^n$), spherical ($\mathbb{S}_c^n$), and hyperbolic space ($\mathbb{P}_c^n$).

Space	Curvature c	Riemannian Manifold $\mathcal{M}_c^n$
Euclidean	0	$\mathbb{E}^n = \mathbb{R}^n$
Spherical	> 0	$\mathbb{S}_c^n = \{\mathbf{x} \in \mathbb{R}^n : \langle \mathbf{x}, \mathbf{x} \rangle_2 > -\frac{1}{c}\}$
Hyperbolic	< 0	$\mathbb{P}_c^n = \{\mathbf{x} \in \mathbb{R}^n : \langle \mathbf{x}, \mathbf{x} \rangle_2 < -\frac{1}{c}\}$

$$(\alpha + \beta + \gamma) \mid d \tag{20}$$

where $\mid$ and d are the exact division operator and the embedding dimension, respectively. We evaluate knowledge graph link prediction and node classification tasks in the unified space, and design a series of ablation experiments to explore the effect of continuous curvatures on the embedding performance.

5 EXPERIMENTS AND RESULTS

In this section, we evaluate the proposed DT-GCN model, and present its performance on both synthetic and real-world datasets. We first introduce the experimental setup. Next, we show the effectiveness of our model. We further conduct several ablation studies to demonstrate the effectiveness of the unified space, dimension, and various data types.

5.1 Experimental Setup

5.1.1 Datasets. We evaluate our model on node classification and link prediction tasks using both synthetic and real-world datasets, including YAGO-10 [31], FB15k-237 [12], WN18RR, and DBpedia-literals. The statistics of datasets are summarized in Table 2.

5.1.2 Baselines. To evaluate the effectiveness of our model, we compare DT-GCN with several state-of-the-art models for knowledge graph link prediction and node classification tasks, including 2 shallow models, 2 NN-based models, 4 Euclidean GNN-based models, and 3 hyperbolic GNN-based models:

- **EUC**: Euclidean-based shallow model with Euclidean distance function as the objective function.
- **HYP** [28]: Poincaré ball-based shallow model with hyperbolic distance function as the objective function.
- **MLP**: Traditional feature-based deep learning model without considering graph topology.
- **HNN** [11]: Deep learning model in hyperbolic space.
- **GCN** [18]: Graph-based convolutional neural network.
- **SGC** [38]: The simplified linear model of GCN.

Table 4: Knowledge graph link prediction (LP) and node classification (NC) results for the embeddings on both synthetic and real-world datasets. The best results are in bold and the second best results are underlined.

$\mathcal{M}$	Model	YAGO-10		FB15k-237		WN18RR		DBpedia-literals	
		LP	NC	LP	NC	LP	NC	LP	NC
$\mathbb{E}$	EUC	0.6928	0.5765	0.6332	0.5921	0.6364	0.4880	0.5418	0.3841
$\mathbb{P}$	HYP	0.6941	0.5816	0.6971	0.5730	0.6851	0.5476	0.5670	0.4129
$\mathbb{E}$	MLP	0.7260	0.6362	0.7144	0.6129	0.7040	0.5019	0.5953	0.5159
$\mathbb{P}$	HNN [11]	0.6830	0.6667	0.7258	0.6101	0.7094	0.5765	0.5947	0.5916
$\mathbb{E}$	GCN [18]	0.7890	0.6979	0.7233	0.6752	0.7126	0.7037	0.6360	0.6111
$\mathbb{E}$	SGC [38]	0.7820	0.7041	0.7593	0.6644	0.7222	0.6953	0.7129	0.6514
$\mathbb{E}$	GAT [35]	0.7884	0.7094	0.7408	0.7138	0.7240	0.6852	0.7312	0.6202
$\mathbb{E}$	GraphSAGE [14]	0.7745	0.7018	0.6953	0.6813	0.7407	0.6534	0.6444	0.6150
$\mathbb{P}$	HGNN [24]	0.8007	0.8214	0.8127	0.7653	0.8154	0.7826	0.7501	0.6312
$\mathbb{P}$	HGCN [6]	0.8704	0.8326	0.8816	0.7713	0.9080	0.7803	0.7449	0.6458
$\mathbb{P}$	HGAT [13]	0.8811	0.8162	0.9030	0.7742	0.8763	0.7832	0.7513	0.6229
$\mathbb{E}$	EGCN*	0.9286	0.8623	0.9381	0.8527	0.7126	0.7037	0.8130	0.7236
$\mathbb{S}$	SGCN*	0.8929	0.8574	0.9167	0.8462	0.9271	0.8066	0.7783	0.6922
$\mathbb{P}$	HGCN*	0.9381	0.8816	0.9402	0.8971	0.9080	0.7803	0.8351	0.7748
$\mathbb{ESP}$	DT-GCN	**0.9643**	**0.8933**	**0.9785**	**0.9080**	**0.9513**	**0.8363**	**0.8714**	**0.7945**

Table 5: Ablation results on the unified space.

$\mathcal{M}$	Model	YAGO-10		FB15k-237		WN18RR		DBpedia-literals	
		LP	NC	LP	NC	LP	NC	LP	NC
$\mathbb{ES}$	ES-GCN*	0.9013	0.8627	0.9374	0.8416	0.9287	0.7937	0.8197	0.7358
$\mathbb{EP}$	EH-GCN*	0.9637	0.8871	0.9412	0.8873	0.9417	0.8004	0.8673	0.7933
$\mathbb{SP}$	SH-GCN*	0.9295	0.8830	0.9398	0.8928	0.9391	0.8172	0.8243	0.7901
$\mathbb{ESP}$	DT-GCN	**0.9643**	**0.8933**	**0.9785**	**0.9080**	**0.9513**	**0.8363**	**0.8714**	**0.7945**

- **GAT** [35]: GCN with the attention mechanism.
- **GraphSAGE** [14]: Inductive GCN model on large graph.
- **HGNN** [24]: Hyperbolic version of GNN.
- **HGCN** [6]: Hyperbolic version of GCN.
- **HGAT** [13]: Hyperbolic version of GAT.

5.1.3 Ablations. To analyze the advantage of the unified space, we consider several variants of the proposed model, including:

- **EGCN***: *DataType-aware* Euclidean embedding.
- **SGCN***: *DataType-aware* spherical embedding.
- **HGCN***: *DataType-aware* hyperbolic embedding.
- **ES-GCN***: *DataType-aware* embedding with Euclidean and spherical spaces.
- **EH-GCN***: *DataType-aware* embedding with Euclidean and hyperbolic spaces.
- **SH-GCN***: *DataType-aware* embedding with spherical and hyperbolic spaces.

Also, we consider several variants to analyze the advantage of data types, including HGCN with attributes of integer, double, Boolean, temporal, and textual types.

5.1.4 Training. Following the previous work [6], we train our model by minimizing the cross-entropy loss and optimize all models with Adam except Poincaré embeddings which are optimized with Riemannian SGD. We report the hyper-parameters, including learning rate, dimension, number of hidden layers, activation function, bias, dropout probability, weight decay, and number of attention heads as this example: 0.01, 16, 2, relu, 1, 0, 0, and 4.

5.1.5 Implementation. We regard ROC AUC as the evaluation metric for the link prediction task, and treat F1 score as the evaluation metric for the node classification task. We perform the optimization in the tangent space and adopt standard Euclidean optimizers. We have implemented the proposed model in PyTorch and conducted the experiments on an NVIDIA Tesla V100 GPU.

5.2 Results and Analysis

In this subsection, we compare the proposed model with the state-of-the-art models, and present some ablations with different spaces. The experimental results are shown in Table 4. $\mathbb{E}$, $\mathbb{S}$, and $\mathbb{P}$ correspond to Euclidean, spherical, and hyperbolic space, respectively. $\mathbb{ESP}$ is the unified space defined in our model. Note that the performance of GCN* and HGCN* on the WN18RR dataset is the same as that of GCN and HGCN, since the WN18RR dataset does not include attribute values. As we can see, the proposed DT-GCN outperforms all the baseline models on both synthetic and real-world datasets, which verifies the effectiveness of the proposed model on fine-grained data types. We also observe that the model based on

Table 6: Ablation results on data types.

$\mathcal{M}$	Model	YAGO-10		FB15k-237		DBpedia-literals	
		LP	NC	LP	NC	LP	NC
$\mathbb{P}$	HGCN	0.8704	0.8326	0.8816	0.7713	0.7449	0.6458
$\mathbb{P}$	HGCN+integer	-	-	0.9172	<u>0.8767</u>	0.7982	0.7102
$\mathbb{P}$	HGCN+double	-	-	0.8993	0.8423	0.7774	0.6835
$\mathbb{P}$	HGCN+Boolean	-	-	-	-	0.7503	0.6612
$\mathbb{P}$	HGCN+temporal	<u>0.9237</u>	<u>0.8756</u>	0.9236	0.8451	<u>0.8123</u>	<u>0.7319</u>
$\mathbb{P}$	HGCN+textual	0.8929	0.8673	<u>0.9271</u>	0.8637	-	-
$\mathbb{P}$	HGCN*	**0.9381**	**0.8816**	**0.9402**	**0.8971**	**0.8351**	**0.7748**

the unified space outperforms any of the three kinds of single spaces (i.e., Euclidean, spherical, and hyperbolic), which proves the ability of continuous curvature space. In addition, in the comparison of the three kinds of single-space embedding models, we can find that the performance of the hyperbolic embedding model is superior to the models based on other kinds of spaces. Moreover, the models in spherical space perform the worst in most cases, since most of the current embedding models are in two-dimensional Euclidean space rather than three-dimensional spherical space.

5.3 Ablations on the Unified Space

In this subsection, we study the effectiveness of the unified space by comparing variants of the proposed model with various spaces. The ablation results are shown in Table 5, where $\mathbb{ES}$ denotes the inner product of Euclidean and spherical spaces, $\mathbb{EH}$ denotes the inner product of Euclidean and hyperbolic spaces, and $\mathbb{SP}$ denotes the inner product of spherical and hyperbolic spaces. We can observe that the performance of the models in $\mathbb{ES}$ and $\mathbb{SP}$ spaces is inferior to that in $\mathbb{EH}$ space in most cases due to the unsatisfying representation ability in spherical space.

5.4 Ablations on Dimension

In this subsection, we study the influence of dimension. We evaluate DT-GCN, HGCN*, and HGCN in the unified space with dimension $d \in \{4, 8, 16, 32, 64\}$ on the DBpedia-literals dataset. Note that the sum of the dimensions of each space must be a divisor of the input dimension. For example, for dimension 16, the model can be $\mathbb{E}1\mathbb{S}1\mathbb{P}2$, due to 16 is divisible by 4. As shown in Figure 4, it can be seen that with the increase of dimensions, the performance of models gradually improves and the F1 score-dimension curves tend to be stable when the dimension reaches 16.

5.5 Ablations on Data Types

In this subsection, we study the contribution of different data types, including integer, double, Boolean, temporal, and textual. We apply HGCN as the baseline, and compare each variant with HGCN. The experimental results are shown in Table 6. The WN18RR dataset is excluded from this ablation experiment since it does not contain attributes. From the results, we can find that HGCN* variants with the fine-grained data types achieve a better performance than the original HGCN model. Furthermore, each variant of HGCN with

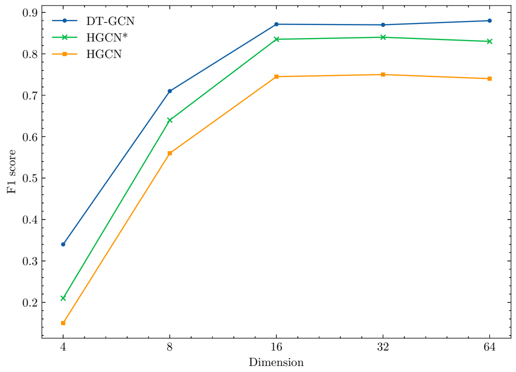


Figure 4: F1 score-dimension curves of DT-GCN, HGCN*, and HGCN with $d \in \{4, 8, 16, 32, 64\}$ on the DBpedia-literals dataset.

attribute values outperforms the basic HGCN, which demonstrates the significance of incorporating attribute values.

6 CONCLUSION

In this paper, we propose a novel *DataType-aware* hyperbolic knowledge representation learning model (DT-GCN) to take full advantage of attribute values of various data types. Specifically, DT-GCN projects each entity in a unified space and further enhances the embedding learning by attribute values of fine-grained data types. The experimental results verify that our proposed DT-GCN significantly outperforms the state-of-the-art models on benchmark datasets on the tasks of link prediction and node classification.

In the future work, we will further expand this research by considering: (1) the data type hierarchy built in XML schema that includes forty-six different data types and contains rich structural information. (2) the more effective embedding space that can achieve better knowledge representation learning.

ACKNOWLEDGMENTS

This work is supported by the National Key R&D Program of China (2020AAA0108504), the National Natural Science Foundation of China (61972275), and the Australian Research Council Linkage Project (LP180100750).

REFERENCES

[1] Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. Multi-relational Poincaré Graph Embeddings. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 4465–4475.

[2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylo. 2008. Freebase: A Collaboratively Created Graph Database For Structuring Human Knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. 1247–1250.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Proceedings of the 27st International Conference on Neural Information Processing Systems*. 2787–2795.

[4] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam Hruschka, and Tom Mitchell. 2010. Toward an Architecture for Never-Ending Language Learning. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. 1306–1313.

[5] Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-Dimensional Hyperbolic Knowledge Graph Embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 6901–6914.

[6] Ines Chami, Rex Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic Graph Convolutional Neural Networks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 4869–4880.

[7] Muhao Chen, Yingtao Tian, Kai-Wei Chang, Steven Skiena, and Carlo Zaniolo. 2018. Co-training Embeddings of Knowledge Graphs and Entity Descriptions for Cross-lingual Entity Alignment. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 3998–4004.

[8] Richard Cyganiak, David Hyland-Wood, and Markus Lanthaler. 2014. RDF 1.1 Concepts and Abstract Syntax. *W3C Recommendation* 25, 02 (2014), 1–8.

[9] Miao Fan, Qiang Zhou, Emily Chang, and Thomas Fang Zheng. 2014. Transition-based Knowledge Graph Embedding with Relational Mapping Properties. In *Proceedings of the 28th Pacific Asia Conference on Language, Information and Computing*. 328–337.

[10] Jun Feng, Mantong Zhou, Yu Hao, Minlie Huang, and Xiaoyan Zuh. 2016. Knowledge Graph Embedding by Flexible Translation. In *Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning*. 557–560.

[11] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. 2018. Hyperbolic Neural Networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 5345–5355.

[12] Alberto García-Durán and Mathias Niepert. 2018. KBLRN: End-to-End Learning of Knowledge Base Representations with Latent, Relational, and Numerical Features. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence*. 372–381.

[13] Caglar Gulcehre, Misha Deni, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, and Nando de Freitas. 2019. Hyperbolic Attention Networks. In *Proceedings of the 7th International Conference on Learning Representations*.

[14] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[15] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Kowledge 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*. 687–696.

[16] Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge Graph Completion with Adaptive Sparse Transfer Matrix. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. 985–991.

[17] Seyed Mehran Kazemi and David Poole. 2018. SimplE Embedding for Link Prediction in Knowledge Graphs. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 4284–4295.

[18] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*.

[19] Agustinus Kristiadi, Mohammad Asif Khan, Denis Lukovnikov, Jens Lehmann, and Asja Fischer. 2019. Incorporating Literals into Knowledge Graph Embeddings. In *Proceedings of the 2019 International Semantic Web Conference*. 347–363.

[20] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostasa, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia - A Large-Scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web* 6, 2 (2015), 167–195.

[21] Zhao Li, Xin Wang, Jianxin Li, and Qingpeng Zhang. 2021. Deep Attributed Network Representation Learning of Complex Coupling and Interaction. *Knowledge-Based Systems* 212 (2021), 106618.

[22] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence*. 2181–2187.

[23] Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical Inference for Multi-Relational Embeddings. In *Proceedings of the 34th International Conference on Machine Learning*. 2168–2178.

[24] Qi Liu, Maximilian Nickel, and Douwe Kiela. 2019. Hyperbolic Graph Neural Networks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 8228–8239.

[25] Weiyu Liu, Angel Daruna, Zsolt Kira, and Sonia Chernova. 2020. Path Ranking with Attention to Type Hierarchies. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*. 2893–2900.

[26] Shiheng Ma, Jianhui Ding, Weijia Jia, Kun Wang, and Minyi Guo. 2017. TransT: Type-based Multiple Embedding Representations for Knowledge Graph Completion. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 717–733.

[27] George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[28] Maximillian Nickel and Douwe Kiela. 2017. Poincaré Embeddings for Learning Hierarchical Representations. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 6338–6347.

[29] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic Embeddings of Knowledge Graphs. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. 1955–1961.

[30] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-way Model for Collective Learning on Multi-relational Data. In *Proceedings of the 28th International Conference on Machine Learning*. 809–816.

[31] Pouya Pezeshkpour, Liyan Chen, and Sameer Singh. 2018. Embedding Multimodal Relational Data for Knowledge Base Completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 3208–3218.

[32] Zequn Sun, Muhao Chen, Wei Hu, Chengming Wang, Jian Dai, and Wei Zhang. 2020. Knowledge Association with Hyperbolic Knowledge Graph Embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. 5704–5716.

[33] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *Proceedings of the 7th International Conference on Learning Representations*.

[34] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *Proceedings of the 33th International Conference on Machine Learning*. 2071–2080.

[35] Petar Veličković, Guillem Cucurul, Arantxa Casanov, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attentinon Networks. In *Proceedings of the 6th International Conference on Learning Representations*.

[36] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[37] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*. 1112–1119.

[38] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr., Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In *Proceedings of 36th International Conference on Machine Learning*. 6861–6871.

[39] Yanrong Wu and Zhichun Wang. 2018. Knowledge Graph Embedding with Numeric Attributes of Entities. In *Proceedings of The 3rd Workshop on Representation Learning for NLP*. 132–136.

[40] Han Xiao, Minlie Huang, Lian Meng, and Xiaoyan Zhu. 2017. SSP: Semantic Space Projection for Knowledge Graph Embedding with Text Description. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. 3104–3110.

[41] Han Xiao, Minlie Huang, Hao Yu, and Xiaoyan Zhu. 2015. TransA: An Adaptive Approach for Knowledge Graph Embedding. arXiv:1509.05490

[42] Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. From One Point to A Manifold: Knowledge Graph Embedding For Precise Link Prediction. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. 1315–1321.

[43] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation Learning of Knowledge Graphs with Entity Descriptions. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. 2659–2665.

[44] Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Representation Learning of Knowledge Graphs with Hierarchical Types. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. 2965–2971.

[45] 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 *Proceedings of the 3rd International Conference on Learning Representations*.

[46] Yu Zhao, Anxiang Zhang, Ruobing Xie, Kang Liu, and Xiaojie Wang. 2020. Connecting Embeddings for Knowledge Graph Entity Typing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 6419–6428.

[47] Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. 2015. Aligning Knowledge and Text Embeddings by Entity Descriptions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 267–272.