Knowledge Graph Representation Learning using Ordinary Differential Equations

Mojtaba Nayyeri
University Of Bonn
nayyeri@iai.uni-bonn.de

Chengjin Xu
University Of Bonn
xuc@iai.uni-bonn.de

Franca Hoffmann
University Of Bonn
franca.hoffmann@hcm.uni-bonn.de

Mirza Mohtashim Alam
Nature-Inspired Machine Intelligence-InfAI
mohtasim@infai.org

Jens Lehmann
Fraunhofer IAIS
jens.lehmann@iais.fraunhofer.de

Sahar Vahdati
Nature-Inspired Machine Intelligence-InfAI
vahdati@infai.org

Abstract

Knowledge Graph Embeddings (KGEs) have shown promising performance on link prediction tasks by mapping the entities and relations from a knowledge graph into a geometric space. The capability of KGEs in preserving graph characteristics including structural aspects and semantics, highly depends on the design of their score function, as well as the inherited abilities from the underlying geometry. Many KGEs use the Euclidean geometry which renders them incapable of preserving complex structures and consequently causes wrong inferences by the models. To address this problem, we propose a neuro differential KGE that embeds nodes of a KG on the trajectories of Ordinary Differential Equations (ODEs). To this end, we represent each relation (edge) in a KG as a vector field on several manifolds. We specifically parameterize ODEs by a neural network to represent complex manifolds and complex vector fields on the manifolds. Therefore, the underlying embedding space is capable to assume the shape of various geometric forms to encode heterogeneous subgraphs. Experiments on synthetic and benchmark datasets using state-of-the-art KGE models justify the ODE trajectories as a means to enable structure preservation and consequently avoiding wrong inferences.

1 Introduction

Knowledge Graphs (KGs) have a significant impact on machine learning approaches (Wang et al., 2017). A KG usually represents factual knowledge in triples of the form (entity, relation, entity) e.g., (Plato, influences, Kant). The nodes of a KG represent entities and the links denote the relations. Although quantitatively KGs are often large-scale with millions of triples, they are usually incomplete i.e. do not capture all knowledge within a domain of interest. To address this problem, various approaches have been used so far, among which link prediction using KG embeddings (KGE) attracted growing attention. KGE models map entities (e) and relations (r) of a KG from a symbolic domain to a geometric space (e.g. a vector space). Such embedding models employ a score function to perform the link prediction task, which uses the learned embedding vectors $(\boldsymbol{e_t}, \boldsymbol{r}, \boldsymbol{e_{t+1}})$ of a triple (e_t, r, e_{t+1}) to compute its plausibility. This allows to rank triples by their score, where a correct triple should obtain a higher score and a lower rank than an incorrect triple. KGE models have been predominantly studied with a focus on the triple level.

However, in a broader view, triples of a graph form specific subgraphs (i.e. structural patterns distributed in the graph) in local structures. The preservation of subgraphs in the learned representations is a major challenge. In this regard, the choice of geometry becomes crucial as the distribution of the corresponding embeddings for entities and relations depends on it. Furthermore, within each geometry, the mathematical operations used in the score function lead to differences in the encoding capability of KGEs. Several state-of-the-art KGE models (Zhang et al., 2019a; Sun et al., 2019; Trouillon et al., 2016; Bordes et al.,

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 9529–9548
November 7–11, 2021.

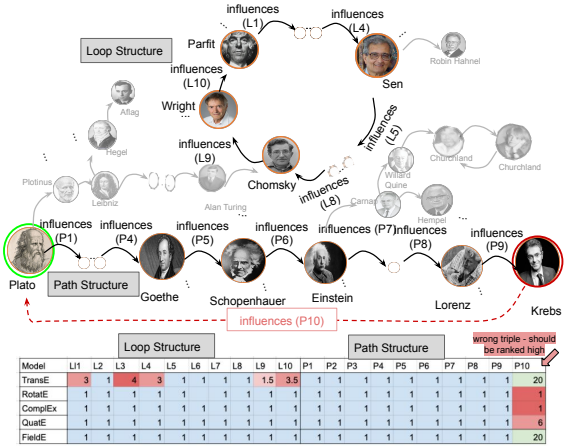


Figure 1: **Structure Preservation Challenges in KGs.** The figure illustrates a path and loop of the "influences" relation in a small excerpt of the YAGO KG for a chain of notable people who have influenced the other. The heat map shows that several other models incorrectly predict a loop for the chain of influencers at the bottom by assigning a high value for the triple P10 (marked red) as they model relations independent of entities.

2013a) are designed in Euclidean geometry which do not intrinsically support structural preservation. As an example, consider Figure 1, which contains both a loop/cycle and a path (without cycles) for the "influences" relation. Many KGE models incorrectly predict the path to be closed in such a scenario. The root cause of this problem lies in the entity-dependent nature of the "influences" relation. Most of the KGE models such as RotatE (as well as TransE, ComplEx, QuatE) consider relations independent of entities. This problem can usually only be mitigated at the cost of increasing the dimensionality of the model, which leads to higher computational costs and may negatively impact the usefulness of learned representations in downstream machine learning tasks.

For those KGEs with sophisticated geometry, only a limited number of structures such as hierarchical or tree-like have been studied by using hyperbolic geometry or Poincaré Ball models (Nickel and Kiela, 2017). In order to significantly improve the structure preservation capabilities of KGE models, we propose a novel KGE model named *FieldE* which employs differential equations (DEs) for embedding KGs into a vector space. The use of differential equations allows to overcome the entity-specific nature of previous KGE models. In this model, relations are viewed as trajectories connecting neighboring nodes in the graph, which implies a continuity of changes in the embedding space and consequently describes the underlying geometry. This is especially important, because the success of a KGE model depends on the way it correctly specifies the underlying geometry that describes the natural proximity of data samples in a geometric space (Mathieu and Nickel, 2020). Designing *FieldE* with a list of well-specified geometries (Euclidean, Poincare Ball, Hyperboloid, and Spherical) a) improves generalization and b) increases the interpretability. This is due to capturing the natural proximity of entities from KG space to the geometric space. We employ First-order Ordinary Differential equations, which are a special class of DEs that represent a vector field on a smooth Riemannian manifold. We selected first-order ODEs, because of their advantages over other classes of DEs: while being capable of capturing complex geometries, they are also efficiently implementable. To allow our approach to self-adapt to the complexity of the underlying knowledge graph, we developed a neural network based approach which learns a suitable geometry from the training graph itself. Therefore, *FieldE* combines substantial previous research and insights of DEs, embeddings and neural networks in order to provide a comprehensive model capable of representation learning on KGs with multiple heterogeneous subgraphs.

2 Related Work

We describe the KGE models exploiting geometric properties for structure preservation. We also review ODEs in ML in general, as our model is, to the best of our knowledge, the first approach employing ODEs in KGEs.

KGEs in Euclidean Geometry. Apart from some discussions about encoding relational patterns, capturing structures is not directly targeted by most previous work in KGE models. TransE (Bordes et al., 2013b) discussed simple 1-1 relational patterns and its follow up models (Ji et al., 2015; Lin et al., 2015; Wang et al., 2014) considered 1-many, many-1, and many-many patterns. RotatE (Sun et al., 2019) uses rotational transformations for encoding more complex patterns such as symmetry, transitivity, composition, reflexivity and inversion. Some other KGEs such as DisMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), only focus on effective choices of embedding by element-wise multiplication of transformed head and tail or angle transformation in QuatE (Zhang et al., 2019b), and RESCAL (Nickel et al., 2011),

KGEs in Non-Euclidean Geometry. Several works such as MuRP, ROTH, REFH, ATTH (Balazevic et al., 2019; Chami et al., 2020) use non-Euclidean geometry for hierarchical structures only (Suzuki et al., 2018; Ji et al., 2016). In (Dareddy et al., 2019), random walk and heterogeneous skip-gram models are used to generate the embeddings of different structures without leveraging DEs. Recently, (Lou et al., 2020) used Fréchet mean and geometries for capturing structures but not for link prediction task.

Use of Differential Equations in Machine Learning. An early work that uses differential equations in graphs is (Kozen and Stefansson, 1997), however it does not employ neural networks. In (Chen et al., 2018), a family of deep neural network models has been proposed which parameterizes the derivative of a hidden state instead of the usual specification of a discrete sequence of hidden layers. In this approach, ODEs are used in the design of continuous-depth networks with the purpose of providing an efficient computation of the network output, which improves memory efficiency, adaptive computation, parameter efficiency (Kobyzev et al., 2020), and continuous time-series models (Kidger et al., 2020). It is applied for supervised learning on an image dataset and time-series prediction. This work used ODEs in the proposed approach without considering knowledge graphs and embeddings for link prediction.

3 Preliminaries and Background

This section provides the preliminaries of the Riemannian Geometry (Franciscus, 2011; Hairer, 2011) and explains its key elements required to understand our model. The aim of our model is to embed nodes (entities) of a KG on trajectories of vector fields (relations) laid on the surface of a smooth Riemannian manifold. Therefore, we first provide the mathematical definitions for *manifold*, *tangent space* and *vector field* driving the dynamics of a DE on the manifold.

Manifold and Tangent Space We denote by $\mathcal{M}$ a smooth manifold of dimension d that is embedded in a higher dimensional Euclidean space $\mathbb{R}^n$ with $n \geq d$. Consider a particle moving on $\mathcal{M}$. The set of all possible directions that a particle passing a given point may go forms a space called the *tangent space* (the set of velocities). Formally, given a point p on a manifold $\mathcal{M}$, the tangent space $\mathcal{T}_p\mathcal{M} \subset \mathbb{R}^n$ is the set of all the vectors which are tangent to all the continuously differentiable curves passing through point p. A *Tangent Bundle* is the set of all tangent spaces on a manifold $\mathcal{M}$ and is defined as $\mathcal{TM} = \bigcup_{p \in \mathcal{M}} \mathcal{T}_p\mathcal{M}$.

Vector Field A function $f : \mathcal{M} \to \mathcal{TM}$ is called a *vector field* on $\mathcal{M}$ if $f(p) \in \mathcal{T}_p\mathcal{M} \subset \mathbb{R}^n$ for all $p \in \mathcal{M}$. Each such f defines trajectories $\gamma(t)$ on $\mathcal{M}$ via the ODE

$$\frac{d\gamma(t)}{dt} = f(\gamma(t)). \tag{1}$$

If f is continuously differentiable, then for each initial condition $\gamma(t_0) = p_0 \in \mathcal{M}$ there exists an open interval $[a, b]$ (of maximal size) with $t_0 \in [a, b]$ and a unique trajectory $\gamma : [a, b] \to \mathcal{M}$ solving (1) which is twice differentiable (Hairer, 2011).

Riemannian Manifold A smooth manifold $\mathcal{M}$ endowed with a Riemannian metric g is a *Riemannian manifold*, denoted by $(\mathcal{M}, g)$. For $p \in \mathcal{M}$, the function $g_p = g(p) = \langle ., . \rangle_p : \mathcal{T}_p\mathcal{M} \times \mathcal{T}_p\mathcal{M} \to \mathbb{R}$ defines an inner product on the associated tangent space. The metric tensor is used to measure angle, length of curves, surface area and volume locally. Global quantities can then be derived as integrals over the local contributions.

Geodesics and Exponential Map The manifold equivalent of the notion of straight lines in Euclidean space is given by geodesics and can be defined in terms of the metric tensor g. *Geodesics* are curves on the manifold such that given any two (sufficiently close) points on this curve, the geodesic minimizes the length of all curves joining these two points. For each pair (p, v) with $p \in \mathcal{M}$, $v \in \mathcal{T}_p\mathcal{M}$, there exists a unique geodesic $\gamma(t) \in \mathcal{M}$ such that $\gamma(0) = p$ and $v = \frac{d\gamma(t)}{dt}\Big|_{t=0}$. The *exponential map* exp_p is then defined as $exp_p : \mathcal{T}_p\mathcal{M} \to \mathcal{M}$ with $exp_p(v) = \gamma(1)$. Its inverse is given by $log_p : \mathcal{M} \to \mathcal{T}_p\mathcal{M}$.

4 Method

In this section, we propose *FieldE*, a new KGE model based on Ordinary Differential equations (ODEs). The approach relies on the following two key components: 1) choice of a smooth Riemannian manifold $\mathcal{M}$ on which the embedded entities lie; 2) choice of a vector field f such that a given relation between entities in the KG is encoded as trajectories on $\mathcal{M}$ solving the ODE in Equation (2) and connecting embedded entities that are related in

Table 1: Summary of operations in Euclidean, Poincaré Ball, Hyperbolid, and Spherical models (Lou et al., 2020; Weber and Nickel, 2018; Balazevic et al., 2019)

	Euclidean $\mathbb{R}^d$	Poincaré Ball $\mathbb{B}^d_K$	Hyperboloid $\mathbb{H}^d_K$	Spherical $\mathbb{S}^d$
Manifold $\mathcal{M}$	$\mathbb{R}^d$	$\{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x}\| < -\frac{1}{K}\}$	$\{\boldsymbol{x} \in \mathbb{R}^{d+1} : \langle \boldsymbol{x}, \boldsymbol{x} \rangle = \frac{1}{K}\}$	$\{\boldsymbol{x} \in \mathbb{R}^{d+1} : \langle \boldsymbol{x}, \boldsymbol{x} \rangle = 1\}$
Exponential map $exp_{\boldsymbol{x}}(\boldsymbol{v})$	$\boldsymbol{x} + \boldsymbol{v}$	$\boldsymbol{x} \oplus_K (tanh(\sqrt{\lvert K \rvert}\frac{\lambda_{\boldsymbol{x}K}\|\boldsymbol{v}\|}{2})\frac{\boldsymbol{v}}{\sqrt{\lvert K \rvert}\|\boldsymbol{v}\|})$	$cosh(\sqrt{\lvert K \rvert}\|\boldsymbol{v}\|)\boldsymbol{x} + \boldsymbol{v}\frac{sinh(\sqrt{\lvert K \rvert}\|\boldsymbol{v}\|)}{\sqrt{\lvert K \rvert}\|\boldsymbol{v}\|}$	$cos(\|\boldsymbol{v}\|)\boldsymbol{x} + sin(\|\boldsymbol{v}\|)\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}$
Distance $dist(\boldsymbol{x}, \boldsymbol{y})$	$\langle\sqrt{\boldsymbol{x} - \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{y}}\rangle$	$\frac{1}{\lvert K \rvert}cosh^{-1}(1 - \frac{2K\|\boldsymbol{x}-\boldsymbol{y}\|^2}{(1+K\|\boldsymbol{x}\|^2)(1+K\|\boldsymbol{y}\|^2)})$	$\frac{1}{\lvert K \rvert}cosh^{-1}(K\langle \boldsymbol{x}, \boldsymbol{y} \rangle)$	$cos^{-1}(\langle \boldsymbol{x}, \boldsymbol{y} \rangle)$
Curvature	$= 0$	< 0	< 0	> 0

the KG. These components can either be given explicitly, or be learned directly from the data. Table 1 includes a description of the manifolds we used in the application of *FieldE*. Below, we present the formulation of *FieldE* in five steps: *relation formulation*, *entity representation*, *triple learning*, *plausibility measurement*, *vector field parameterization*, and *manifold specification* which are discussed in the remainder of this section.

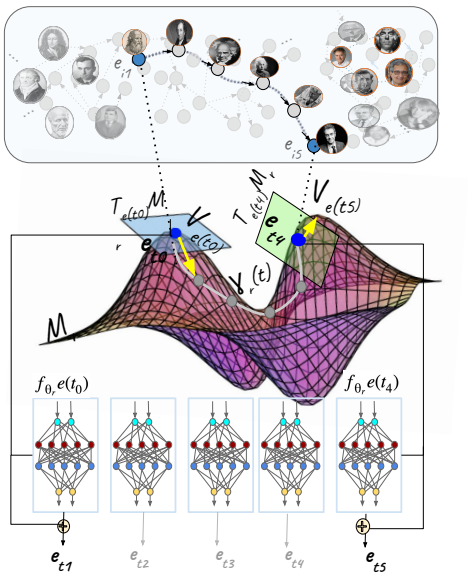


Figure 2: **The Architecture of the FieldE Model**. The input to FieldE is a KG (the upper part). A path structure is highlighted in the graph representation. The vector representation (lower part) illustrates a trajectory of an ODE on a manifold. Nodes of the path are sequentially embedded on the trajectory guided by an NN.

Relation Formulation *FieldE* represents each relation r in a KG as a vector field (f_{θ_r}) on a Riemannian manifold. Here, we presume a given functional form for the vector field f_{θ_r} (independent of time), which is determined by the choice of parameters θ_r. Let $\boldsymbol{e}(t)$ be a parametric trajectory that evolves in time, $t \in \mathbb{R}$, solving the following ODE corresponding to relation r of the KG:

$$\frac{d\boldsymbol{e}(t)}{dt} = f_{\theta_r}(\boldsymbol{e}(t)), \qquad \boldsymbol{e}(t) \in \mathcal{M}. \tag{2}$$

Given the above formulation, each relation of a KG corresponds to a relation-specific vector field. This is consistent with the nature of KGs where different relations form different structures and patterns.

Entity Representation We represent each entity e_i in the KG by a vector in $\mathbb{R}^n$ denoted by $\boldsymbol{e}(t)$, matching each subscript i to a time t where $\boldsymbol{e}(t)$ lies on a trajectory on the manifold $\mathcal{M}$ solving the ODE (2). In particular, consider $k+1$ entities $e_{i_n}, e_{i_{n+1}}, \ldots, e_{i_{n+k}}$ in the KG, each connected to the next by a relation r. The corresponding embeddings are then discrete points $\boldsymbol{e}(t_n), \boldsymbol{e}(t_{n+1}), \ldots, \boldsymbol{e}(t_{n+k}) \in \mathcal{M}$ lying on a trajectory $\boldsymbol{e}(t) \in \mathcal{M}$ solving the ODE in Equation (2).

Triple Learning Let e_{i_n} and $e_{i_{n+1}}$ be two subsequent nodes (e.g. the entities shown in the upper part of the Figure 2) of a graph connected by a relation r. This means the triple $(e_{i_n}, r, e_{i_{n+1}})$ is present in the KG. Let $\boldsymbol{e}(t_n), \boldsymbol{e}(t_{n+1}) \in \mathcal{M}$ be the embeddings corresponding to the entities $e_{i_n}, e_{i_{n+1}}$ respectively (lower part of Figure 2).

We then represent a triple $(e_{i_n}, r, e_{i_{n+1}})$ as a transition from head entity embedding $\boldsymbol{e}(t_n)$ to tail embedding $\boldsymbol{e}(t_{n+1})$ on a relation-specific vector field over the manifold. Therefore, in order to encode a triple $(e_{i_n}, r, e_{i_{n+1}})$ on the manifold, we first compute the tangent vector at $\boldsymbol{e}_{t_n} = \boldsymbol{e}(t_n)$, i.e. $\boldsymbol{v}^r_{\boldsymbol{e}_{t_n}} = \frac{d\boldsymbol{e}(t_n)}{dt} = f_{\theta_r}(\boldsymbol{e}_{t_n})$, which is the direction of movement at point $\boldsymbol{e}_{t_n}$ towards the point $\boldsymbol{e}_{t_{n+1}} = \boldsymbol{e}(t_{n+1})$. We then use the exponential map to map the tangent vector at the head embedding to the tail embedding (determining the direction of movement and moving towards the tail embedding to meet the tail on the manifold) as follows:

$$\boldsymbol{e}_{t_{n+1}} = exp_{\boldsymbol{e}_{t_n}}(\boldsymbol{v}^r_{\boldsymbol{e}_{t_n}}). \tag{3}$$

The triple $(e'_{i_n}, r, e'_{i_{n+1}})$ is *negative* if it does not appear in the KG. To encode negative triples, the following inequality should be satisfied:

$$\boldsymbol{e}'_{t_{n+1}} \neq exp_{\boldsymbol{e}'_{t_n}}(\boldsymbol{v}^r_{\boldsymbol{e}'_{t_n}}). \tag{4}$$

Equation 4 indicates that the triple is measured negative if moving in the direction of the tangent vector at the head point along the manifold does not coincide with the tail.

Plausibility Measurement Given a triple $(e_{i_n}, r, e_{i_{n+1}})$ in the KG, the *plausibility* of the triple is measured by computing the distance of the corresponding vectors $\boldsymbol{e}_{t_{n+1}}$ and $exp_{\boldsymbol{e}_{t_n}}(\boldsymbol{v}^r_{\boldsymbol{e}_{t_n}})$ where $\boldsymbol{v}^r_{e_{t_n}} = f_{\theta_r}(\boldsymbol{e}_{t_n})$. We denote the plausibility measure for a given relation r by the score function S_r, and consider two different choices: 1) the distance-based version named *DFieldE*:

$$S_r(\boldsymbol{e}_{t_n}, \boldsymbol{e}_{t_{n+1}}) = -dist(\boldsymbol{e}_{t_{n+1}}, exp_{\boldsymbol{e}_{t_n}}(\boldsymbol{v}^r_{e_{t_n}})), \tag{5}$$

where $dist$ is a suitable distance function on the manifold (see Table 1) , and 2) the semantic-matching version named *SFieldE*:

$$S_r(\boldsymbol{e}_{t_n}, \boldsymbol{e}_{t_{n+1}}) = \langle \boldsymbol{e}_{t_{n+1}}, exp_{\boldsymbol{e}_{t_n}}(\boldsymbol{v}^r_{e_{t_n}}) \rangle, \tag{6}$$

with $\langle \cdot, \cdot \rangle$ denoting the Euclidean inner product.

Vector Field Parameterization The selection of the function f_{θ_r} is key to our KGE approach. In this paper, we propose two approaches for determining the vector field: a) we parameterize the vector field function f_{θ_r} by a neural network (NN) and propose a neuro-differential KGE model, b) we consider a vector field given by a linear function, resulting in a linear version of our KGE model. Next, we explain these two choices in detail.

Neuro-FieldE We parameterize the vector field by a multi-layer feedforward NN to approximate the underlying vector field,

$$f_{\theta_r}(\boldsymbol{e}) = \sum w^o_i z \left(\sum w^L_{ij} z \left(\sum w^{L-1}_{jk} \dots \sum w^2_{pq} z(w^1_{qz}\boldsymbol{e} + b^1_z) \right) \right), \tag{7}$$

where $\boldsymbol{e} \in \mathcal{M}$, L is the number of hidden layers, w^o denotes the output weight of the network and w^l_{mn} is the weight connecting the mth node of the layer $l-1$ to the nth node of the l-th layer (see Figure 2). All weights are collected in the vector of parameters θ_r which is learned during training. Parametrizing the vector field with an NN gives the model enough flexibility to learn various shapes of the manifold dynamics encoded in the vector field f_{θ_r} (representing complex geometry) from data. This is due to the fact that NNs are universal approximators (Hornik et al., 1989; Hornik, 1991; Nayyeri et al., 2017), i.e. NNs are capable of approximating any continuous function.

Linear-FieldE Linear ODEs are a class of differential equations which have been widely used for several applications (Massera and Schäffer, 1966). Here we model the vector field as a linear function

$$f_{\theta_r}(\boldsymbol{e}) = \mathcal{A}_r \boldsymbol{e}, \tag{8}$$

for $\boldsymbol{e} \in \mathcal{M}$, where $\mathcal{A}_r$ is an $n \times n$ matrix representing a projection on the tangent space. Depending on the eigenvalues of $\mathcal{A}_r$, the vector field can have various shapes.

Manifold Specification There has been a surge of efforts to appropriately select the underlying manifold for KGE models (Nickel and Kiela, 2017; Balazevic et al., 2019; Chami et al., 2020). However, the selection of a suitable manifold for representation learning still remains challenging because real world KGs contain heterogeneous multi-relational neighboring substructures. Thanks to the way FieldE is formulated, it lends itself well to applying techniques from manifold learning in order to explicitly identify the implicit geometry of the KG - a promising direction of future research. Here, we examine FieldE with the following choices of manifolds: the Euclidean space, the unit sphere, Hyperboloid and Poincare ball (see Table 1).

5 Model Analysis

In this part, we analyse the characteristics of the core formulation of *FieldE* compared to other models. We first show that while other models such as RotatE, and ComplEx face issues when learning relatively simple single-relational structures, *FieldE* is able to overcome those issues. Moreover, we show that *FieldE* subsumes popular KGEs and consequently inherits their capabilities in learning various well-studied patterns e.g. symmetry/inversion.

Flexible Relation Embedding Most of the state-of-the-art KGEs such as TransE, RotatE, QuatE, ComplEx etc., consider each relation of the KG

Table 2: Constraints in the embedding space.

MODEL	CONSTRAINTS
TRANSE	$\boldsymbol{r} + \boldsymbol{r} + \boldsymbol{r} = \boldsymbol{0}$
ROTATE	$\boldsymbol{r} \circ \boldsymbol{r} \circ \boldsymbol{r} = \boldsymbol{1}$
FIELDE	$\boldsymbol{f_{\theta_r}(e_1) + f_{\theta_r}(e_2) + f_{\theta_r}(e_3) = 0}$

as a constant vector to perform an algebraic operation such as translation or rotation. Therefore, the relation is entity-independent with regards to the applied algebraic operation. Table 2 shows the constraints in the vector space enforced by TransE, RotatE and FieldE for encoding a loop with three nodes i.e. $(e_1, r, e_2), (e_2, r, e_3), (e_3, r, e_1)$. The constraints obtained by TransE and RotatE are independent of the involved entities. In TransE, the relation embedding becomes null (in Euclidean geometry), which implies that the embeddings of all involved entities are equal. Using RotatE, the embeddings of entities in a loop are the same. However, the computed embedding for a relation with a loop will be entity-independent. Therefore, in one dimension of RotatE, all substructures formed by different groups of entities should be the same in terms of density and structure (e.g. all entities should form loops with density of 3 here in our example). This problem can be mitigated by increasing the dimension, however, it is not fully solved when restricting to low dimensional embeddings.

FieldE addresses the mentioned problem. The relation-specific constraint (see Table 2) is entity-dependent and the direction of translation is determined not only based on the relation, but also based on the entities connected by the relation and the way in which they are mapped to the manifold $\mathcal{M}$. In contrast to RotateE, which can only represent loops with a fixed number of entities, FieldE can capture different substructures locally (such as loops of different sizes, or loops and paths). Note that the relation-specific constraint for Neuro-DFieldE (Equation 2) can always be satisfied because NNs with bounded continuous activation functions are universal approximators and universal classifiers (Hornik et al., 1989; Hornik, 1991; Nayyeri et al., 2017) (complete proof in appendix).

In summary, the state-of-the-art KGE models like TransE, RotatE, ComplEx and QuatE are not capable of preserving more complex structures in the embedding space, because they always model the initial direction of the relation-specific movements independent of the involved entities.

Subsumption of Existing Models We show (see appendix) that *FieldE* subsumes popular models:

Definition 5.1 (from (Kazemi and Poole, 2018))**.** A model M_1 subsumes a model M_2 when any scoring over triples of a KG measured by model M_2 can also be obtained by model M_1.

Proposition 1. DFieldE *subsumes TransE and RotatE.* SFieldE *subsumes ComplEx and QuatE.*

Because *FieldE* subsumes existing models, it consequently inherits their advantages in learning various patterns including symmetry, and anti-symmetry, transitivity, inversion and composition. Moreover, because ComplEx is fully expressive (as defined in (Kazemi and Poole, 2018)) and it is subsumed by *SFieldE*, we conclude that Neuro-*SFieldE* is also fully expressive.

6 EXPERIMENTS AND RESULTS

In this section, we compare FieldE[1] against TransE, RotatE, ComplEx, QuatE, Dismult, MuRP, ATTH, ROTH, and REFH as those performed best on the presented benchmarks. The experiments are done over four benchmark datasets namely FB15K-237, WN18RR, YAGO3-10, and YouTube. We presented two versions of *FieldE* namely DFieldE and SFieldE. DFieldE uses the distance function ((5)) for score computation whereas SFieldE uses the inner product (see Equation (6)). The comparisons are performed in low and high dimensions. We also considered the evaluations with and without data augmentation (adding reverse triples).

Evaluation Metrics We use the standard metrics for link prediction: Mean Reciprocal Rank (MRR), and Hits@n ($n = 1, 3, 10$). MRR is measured by $\sum_{j=1}^{n_t} \frac{1}{r_j}$, where r_j is the rank of the j-th test triple and n_t is the number of triples in the test set. H@n is the number of test triples ranked less than n.

Hyperparameter Search We employed Adam/Adagrad as the optimizers and tune the hyperparameters based on a validation set. The learning rate (r) and batch size (b) are adjusted on $r = \{0.0002, 0.002, 0.02, 0, 1\}$, $b = \{100, 512, 1024\}$ respectively. The embedding dimension d is fixed to 100 for YAGO3-10, 1000 for FB15k-237, and 300 for YouTube. For experiments on high dimensions (Table 3), we used adversarial negative sampling on all the models, with 100 negative samples in FB15K-237, 500 for

[1] https://github.com/mojtabanayyeri/FieldE/tree/FieldE

YAGO3-10, 300 for YouTube. In addition, the experiments are done on low dimension of 32 for all of these datasets as well as WN18RR where we used bias in the score of DFieldE with the setting introduced in (Chami et al., 2020). To have a fair comparison, all the results in Table 4 have been regenerated using same setting. In these two tables, a NN (Equation 7) approximates the vector field. In Table 5, we use a linear function (equation 8).

Performance Evaluation. As shown in Table 3, in all datasets, *FieldE* outperforms all other models across all metrics. In all other cases, a consistent performance advantage can be observed, i.e. it appears that *FieldE* performs well across several datasets whereas other models show larger variations in performance. This evaluation is done on a setting without special boosting techniques except the above described hyperparameter search and using the adversarial loss (Sun et al., 2019) for all evaluated models (including *DFieldE* with Neural Network as a vector field, explained more in the appendix). Table 4 shows the evaluation of all models and all datasets using Poincaré manifold (similar results have been achieved by Hyperbloid). In this setting, *DFieldE* outperforms other models using similar manifolds (e.g., MuRP and ATTH). The experiments validated the suitability of Spherical for FB15k-237 and Euclidean manifold for YAGO3-10, and YouTube datasets (reported in Table 3), and Poincaré and Hyperboloid For WN18RR.

Visualization of Vector Fields. In Figure 3, we illustrate the learned vector fields by DFieldE for two structures on different manifolds: circular on Sphere and hierarchical on Hyperboloid. The arrows depict vector fields where each arrow at a point represents the direction of movement to the next point. For the relation $celebrities...friend$, the vector field on the sphere is circular, enabling to capture the loop structure. On the other hand, the arrows of the vector fields for the relation $partof$ on the Hyperboloid start from the narrow part of the manifold and move towards its wider sides – this is suitable for capturing tree-like or hierarchical structures. The opacity relates to the size of the arrows and means the distance between points on the narrow side is less than the distance on the wider side. This is consistent for hierarchical structures where the distance between the nodes grows by moving from a root entity towards the leafs.

In Figure 4, we provide two sample visualizations for the learned vector fields of the *influences* and *isConnectedTo* relations. The left figure shows circular vector fields illustrating a preservation of loop structure constructed by the *influences* relation (see Figure 1). The right figure depicts loops and paths created by the *isConnectedTo* relation (the connections between airports as entities of the KG) are preserved by our model.

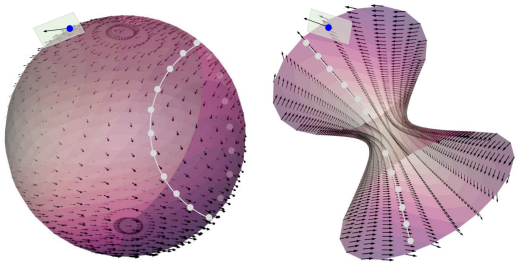


Figure 3: **Learned vector fields.** Showing vector field of $celebrities...friend$ on sphere and $partof$ on Hyperboloid in FB15k-237 and WN18RR.

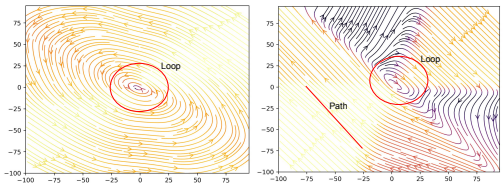


Figure 4: **Vector fields on Euclidean manifold.**. On the right side, the relation "isConnectedTo" and on the left side "influences" are shown. The X and Y axis correspond to the 2D dimensions of vector fields.

Evaluation with Data Augmentation. Table 5 shows the performance of our model in a boosted setting (Lacroix et al., 2018) using a full multiclass log-softmax loss function with applied N3 regularization and reciprocal (data augmentation) approaches. This setting is only suitable for semantic-matching models such as ComplEx and QuatE, due to the fast implementation of the matrix-vector product. Such feature enables taking the advantage of using full negative samples in the learning process. Our model significantly outperforms all the other models in all of the metrics on the YouTube dataset. For example, in H@3, the difference in performance is near 10%. On the YAGO3-10 dataset with these boosting techniques, *SFieldE* outperforms DisMult, QuatE and ComplEx with slightly better results. We believe that the performance differences correlate with the complexity of data structures: The YouTube and YAGO3-10 datasets have a similar size (1 million triples), but YouTube has

Table 3: Link prediction results on FB15k-237, YAGO3-10, and YouTube. Best score are colored.

Model	FB15k-237				YAGO3-10				YouTube			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE	0.33	0.23	0.37	0.53	0.49	0.39	0.56	0.67	0.18	0.00	0.28	0.47
RotatE	0.34	0.24	0.37	0.53	0.49	0.40	0.55	0.67	0.25	0.14	0.30	0.46
ComplEx	0.32	0.23	0.35	0.51	0.36	0.26	0.40	0.55	0.32	0.21	0.36	0.54
QuatE	0.31	0.23	0.34	0.49	-	-	-	-	0.32	0.21	0.36	0.53
DistMult	0.24	0.15	0.26	0.42	0.34	0.24	0.28	0.54	0.04	0.01	0.03	0.10
DFieldE	0.36	0.27	0.39	0.55	0.51	0.41	0.58	0.68	0.33	0.24	0.39	0.55

Table 4: Link prediction results on FB15k-237, WN18RR, and YouTube (low dimension 32).

Model	FB15k-237				WN18RR				YouTube			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
MuRP	0.32	0.24	0.35	0.50	0.47	0.42	0.48	0.54	0.22	0.13	0.23	0.40
REFH	0.31	0.22	0.34	0.49	0.45	0.41	0.46	0.52	0.20	0.11	0.21	0.37
ROTH	0.31	0.22	0.35	0.50	0.48	0.44	0.49	0.55	0.20	0.11	0.21	0.37
ATTH	0.32	0.24	0.35	0.50	0.47	0.43	0.48	0.54	0.22	0.13	0.23	0.40
DFieldE	0.33	0.25	0.36	0.51	0.48	0.44	0.50	0.57	0.24	0.15	0.26	0.43

Table 5: Evaluation of models in boosting techniques.

Model	YouTube			
	MRR	H@1	H@3	H@10
ComplEx-N3	0.35	0.24	0.39	0.57
QuatE-N3	0.33	0.22	0.37	0.55
DistMult-N3	0.34	0.23	0.38	0.55
SFieldE-N3	0.41	0.28	0.47	0.64

fewer entities and relations. Therefore, YAGO3-10 is sparser than YouTube and generally contains less complex graph structures. The performance advantage of *FieldE* appears to increase when the underlying graph has higher density and contains more complex structures. We could also observe that increasing the number of hidden nodes for YouTube, leads to gradually improving results. This means the required complexity of the underlying vector field could necessitate a higher complexity of the underlying NN. This is reinforced when looking at the simpler YAGO3-10 results, where the best results are achieved with only 5 hidden nodes and any further increase leads to notable overfitting. In conclusion, we believe that complex graphs necessitate complex geometries. While this is not entirely surprising, this hypothesis could be directly investigated empirically as *FieldE* can vary the complexity using the underlying NNs.

Run-time Evaluation In terms of runtime per batch, TransE, ComplEx and RotatE have completed the tasks within 0.10 s, 0.11 s, and 0.13 s respectively. FieldE performs learning within 0.14 s which is mostly due to the time required for learning the underlying vector fields. FieldE is capable of learning complex geometries whereas TransE, ComplEx and RotatE represent simple geometry.

7 Conclusion

We presented *FieldE* – the first representation learning model for knowledge graphs based on Ordinary Differential Equations. In contrast to previous models, *FieldE* models relations as vector fields on a Riemannian Manifold and thereby overcomes the drawbacks of previous works, in which relations are entity-independent. Furthermore, we developed a neural network based approach allowing to learn a suitable geometry from the training graph. We have both empirically and analytically shown that *FieldE* can preserve subgraph structures in the embedding space better than state-of-the-art models. Formally, *FieldE* is a generalisation of several state-of-the-art KGE models and we could formally show that it subsumes TransE, RotatE, ComplEx and QuatE. Evaluation on standard benchmark datasets shows competitive or superior performance of FieldE across all datasets and metrics.

Acknowledgements

We acknowledge the support of the following projects: SPEAKER (BMWi FKZ 01MK20011A), JOSEPH (Fraunhofer Zukunftsstiftung), the EU projects Cleopatra (GA 812997), PLATOON(GA 872592), TAILOR(EU GA 952215), CALLISTO(101004152), the BMBF projects MLwin(01IS18050) and the BMBF excellence clusters ML2R (BmBF FKZ 01 15 18038 A/B/C) and ScaDS.AI (IS18026A-F).

References

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. Multi-relational poincaré graph embeddings. In *Advances in Neural Information Processing Systems*, pages 4465–4475.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013a. Translating embeddings for modeling multi-relational data. pages 2787–2795.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013b. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation learning for attributed multiplex heterogeneous network. In *ACM SIGKDD*, pages 1358–1368.

Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-dimensional hyperbolic knowledge graph embeddings. *arXiv preprint arXiv:2005.00545*.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583.

Manoj Reddy Dareddy, Mahashweta Das, and Hao Yang. 2019. motif2vec: Motif aware node representation learning for heterogeneous networks. In *International Conference on Big Data (Big Data)*, pages 1052–1059. IEEE.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

M.J. Franciscus. 2011. *Riemannian Geometry*. International Book Market Service Limited.

Ernst Hairer. 2011. Solving differential equations on manifolds. *Lecture Notes, Université de Geneve*.

Kurt Hornik. 1991. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257.

Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. 1989. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. 1:687–696.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *Thirtieth AAAI conference on artificial intelligence*.

Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *Advances in neural information processing systems*, pages 4284–4295.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. 2020. Neural controlled differential equations for irregular time series. *arXiv preprint arXiv:2005.08926*.

Ivan Kobyzev, Simon Prince, and Marcus Brubaker. 2020. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Dexter Kozen and Kjartan Stefansson. 1997. Computing the newtonian graph. *Journal of Symbolic Computation*, 24(2):125–136.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical tensor decomposition for knowledge base completion. In *ICML*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

Aaron Lou, Isay Katsman, Qingxuan Jiang, Serge Belongie, Ser-Nam Lim, and Christopher De Sa. 2020. Differentiating through the fréchet mean. In *International Conference on Machine Learning*, pages 6393–6403. PMLR.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian M Suchanek. 2013. Yago3: A knowledge base from multilingual wikipedias.

José Luis Massera and Juan Jorge Schäffer. 1966. *Linear differential equations and function spaces*, volume 21. Academic Press New York.

Emile Mathieu and Maximilian Nickel. 2020. Riemannian continuous normalizing flows. *arXiv preprint arXiv:2006.10605*.

Mojtaba Nayyeri, Hadi Sadoghi Yazdi, Alaleh Maskooki, and Modjtaba Rouhani. 2017. Universal approximation by using the correntropy objective function. *IEEE transactions on neural networks and learning systems*, 29(9):4515–4521.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. 11:809–816.

Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*, pages 6338–6347.

Natalia Ostapuk, Jie Yang, and Philippe Cudré-Mauroux. 2019. Activelink: deep active learning for link prediction in knowledge graphs. In *The World Wide Web Conference*, pages 1398–1408.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

Atsushi Suzuki, Yosuke Enokida, and Kenji Yamanishi. 2018. Riemannian transe: Multi-relational graph embedding in non-euclidean space.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *CVPMC Workshoo*, pages 57–66.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080.

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

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):2724–2743.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*.

Melanie Weber and Maximilian Nickel. 2018. Curvature and representation learning: Identifying embedding spaces for relational data. *NeurIPS Relational Representation Learning*.

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 *Conference on Learning Representations (ICLR)*.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019a. Quaternion knowledge graph embedding. *arXiv preprint arXiv:1904.10281*.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019b. Quaternion knowledge graph embeddings. In *Advances in Neural Information Processing Systems*, pages 2731–2741.

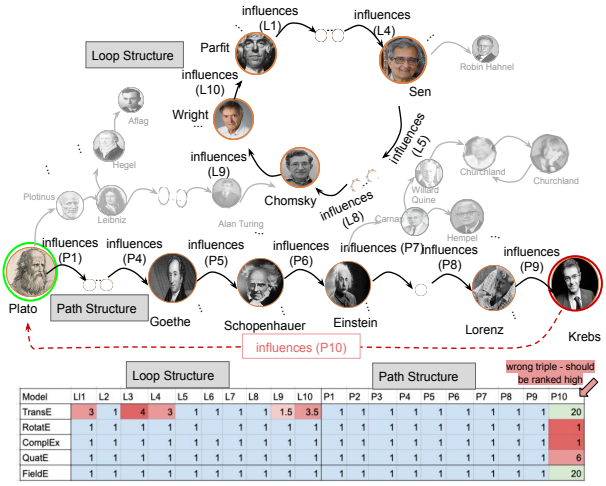
A Problem Statement

In this part, we use an example to explore the problem of KGE models in preservation of heterogeneous structures of the underlying KGs. In order to do so, let us focus on a scenario where heterogeneous structures such as loops and paths are created by an individual relation. The scenario is explored with RotatE which is a recent state-of-the-art among KGE models (Sun et al., 2019). Despite the high performance of RotatE in comparison to other models, this exploration justifies that it returns wrong inferences for link prediction when loops and paths appear in a subgraph with the same relation. To show this, let us represent the loop and path structures as s_L (L = loop) and s_P (P = path) – each with a set of 10 connected nodes $(e_1 \dots e_{10})$ with one relation (r). Therefore, $({e_1}^{s_L} \dots {e_{10}}^{s_L})$ represents the nodes of structure (s_L) where entities form a loop, and $({e_1}^{s_P} \dots {e_{10}}^{s_P})$ corresponds to the nodes of another structure with the same relation but forming a path. This is illustrated by an example in Figure 5 where the influences relation creates such loops and paths. For each triple in this graph e.g. $({e_i}^{s_k}, r, {e_j}^{s_k})$, $k \in \{L, P\}$, the vector representation using RotatE is $\boldsymbol{e_i}^{s_k} \circ \boldsymbol{r} = \boldsymbol{e_j}^{s_k}$ ($\circ$ is the multiplication between complex numbers). The complete representations of the loop and path are then the following:

$$s_{Loop}: \begin{cases} ({e_1}^{s_L}, r, {e_2}^{s_L}), \rightarrow \boldsymbol{e_1}^{s_L} \circ \boldsymbol{r} = \boldsymbol{e_2}^{s_L}, \\ \vdots \\ ({e_9}^{s_L}, r, {e_{10}}^{s_L}), \rightarrow \boldsymbol{e_9}^{s_L} \circ \boldsymbol{r} = \boldsymbol{e_{10}}^{s_L}, \\ ({e_{10}}^{s_L}, r, {e_1}^{s_L}). \rightarrow \boldsymbol{e_{10}}^{s_L} \circ \boldsymbol{r} = \boldsymbol{e_1}^{s_L}. \end{cases}$$

$$s_{Path}: \begin{cases} ({e_1}^{s_P}, r, {e_2}^{s_P}), \rightarrow \boldsymbol{e_1}^{s_P} \circ \boldsymbol{r} = \boldsymbol{e_2}^{s_P}, \\ \vdots \\ ({e_9}^{s_P}, r, {e_{10}}^{s_P}). \rightarrow \boldsymbol{e_9}^{s_P} \circ \boldsymbol{r} = \boldsymbol{e_{10}}^{s_P}. \end{cases}$$

In order to compute the embedding for the r relation, let us first take the loop structure. Starting from $\boldsymbol{e_2}$ in the right side of the first triple, the equivalent vector $(\boldsymbol{e_1}^{s_L} \circ \boldsymbol{r})$ is replaced in its subsequent notation (e.g., by replacing the left side of $\boldsymbol{e_2}$ in the second triple equation of s_L, we get $\boldsymbol{e_1}^{s_L} \circ \boldsymbol{r} \circ \boldsymbol{r} = \boldsymbol{e_3}^{s_L}$).



Model	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
TransE	3	1	4	3	1	1	1	1	1.5	3.5	1	1	1	1	1	1	1	1	1	20
RotatE	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1
ComplEx	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
QuatE	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	6
FieldE	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	20

Figure 5: **Structure Preservation Challenges in KGs.** The figure illustrates a path and loop of the "influences" relation in a small excerpt of the YAGO KG for a chain of notable people who have influenced each other. The heat map below shows that several state-of-the-art models incorrectly predict a loop for the chain of influencers at the bottom by assigning a high value for the triple P10 (marked red) as they model relations independant of entities. The only exception (apart from the newly proposed model) is TransE, which is however generally a lot more limited in terms of structure preservation due to using Euclidean Geometry.

$$\begin{array}{l|l} \boldsymbol{e_1}^{s_L} \circ \boldsymbol{r} = \boldsymbol{e_2}^{s_L} & \\ \boldsymbol{e_2}^{s_L} \circ \boldsymbol{r} = \boldsymbol{e_3}^{s_L} & \boldsymbol{e_1}^{s_L} \circ \boldsymbol{r} \circ \boldsymbol{r} = \boldsymbol{e_3}^{s_L}, \\ \boldsymbol{e_3}^{s_L} \circ \boldsymbol{r} = \boldsymbol{e_4}^{s_L} & \boldsymbol{e_1}^{s_L} \circ \boldsymbol{r} \circ r \circ \boldsymbol{r} = \boldsymbol{e_4}^{s_L}, \\ \vdots & \vdots \\ \boldsymbol{e_9}^{s_L} \circ \boldsymbol{r} = \boldsymbol{e_{10}}^{s_L} & \boldsymbol{e_1}^{s_L} \circ \boldsymbol{r} \cdots \circ \boldsymbol{r} = \boldsymbol{e_{10}}^{s_L}, \\ \boldsymbol{e_{10}}^{s_L} \circ \boldsymbol{r} = \boldsymbol{e_1}^{s_L}. & \boldsymbol{e_1}^{s_L} \circ \boldsymbol{r} \cdots \circ \boldsymbol{r} = \boldsymbol{e_1}^{s_L}. \end{array}$$

By doing this to the end, we conclude that $\boldsymbol{e_1}^{s_L} \circ \boldsymbol{r} \cdots \circ \boldsymbol{r} = \boldsymbol{e_1}^{s_L}$ which means $\boldsymbol{r} \circ \cdots \circ \boldsymbol{r} = 1$ where $\boldsymbol{r}$ is a complex number ($\boldsymbol{r} = e^{i\theta_r}$), therefore $\theta_r = \frac{2\pi}{10}$. This value is static for r in the whole graph, therefore, this can be used to check whether the second structure (path) is preserved.

Here, we replace the vectors the same way as above and additionally include the value of r derived in the first calculation.

$$\begin{array}{l|l} \boldsymbol{e_1}^{s_P} \circ \boldsymbol{r} = \boldsymbol{e_2}^{s_P} & \\ \boldsymbol{e_2}^{s_P} \circ \boldsymbol{r} = \boldsymbol{e_3}^{s_P} & \boldsymbol{e_1}^{s_P} \circ \boldsymbol{r} \circ \boldsymbol{r} = \boldsymbol{e_3}^{s_P}, \\ \boldsymbol{e_3}^{s_P} \circ \boldsymbol{r} = \boldsymbol{e_4}^{s_P} & \boldsymbol{e_1}^{s_P} \circ \boldsymbol{r} \circ \boldsymbol{r} \circ \boldsymbol{r} = \boldsymbol{e_4}^{s_P}, \\ \vdots & \vdots \\ \boldsymbol{e_9}^{s_P} \circ \boldsymbol{r} = \boldsymbol{e_{10}}^{s_P} & \boldsymbol{e_1}^{s_P} \circ \boldsymbol{r} \cdots \circ \boldsymbol{r} = \boldsymbol{e_{10}}^{s_P}, \end{array}$$

After some derivations, we have $\boldsymbol{e_1}^{s_P} \circ e^{\frac{20\pi i}{10}} = \boldsymbol{e_{10}}^{s_P} \circ \boldsymbol{r}$. With a simplification step, this results in $\boldsymbol{e_1}^{s_P} = \boldsymbol{e_{10}}^{s_P} \circ \boldsymbol{r}$, from which the model infers that the triple $({e_{10}}^{s_P}, r, {e_1}^{s_P})$ is positive. However, this is a path structure and the wrong inference yields it to be a loop structure. This shows how such heterogeneous structures are challenging for the RotatE model. This problem is not limited to RotatE. Other rotation based KGE models such as QuatE and ComplEx also have this problem. Generally, every KGE model which uses constant relation-based transformation such as TransE also suffer from such limitations. The heat map in Figure 5 also indicates the wrong inferences by the state-of-the-art models namely TransE, RotatE, ComplEx, and QuatE. These wrong inferences lead to difficulties in preservation of loop and path structures. However, FieldE is capable of correct inferences for different sub-structures, thus, heterogeneous structure preservation is also satisfied. Later, we will discuss KGEs by comparing the TransE model with FieldE from the flexibility point of view for relation transformation (constant vs varied vector field).

B Flexible Relation Embedding

Here, we analyse TransE for modeling the mentioned subgraphs. We specifically focus on the loop structure in this part for TransE. The model considers a relation as a constant vector to perform translations as

$$\boldsymbol{e_t} + \boldsymbol{r} = \boldsymbol{e_{t+1}}. \tag{9}$$

Therefore, a relation-specific transformation (here translation) is performed in the same direction with the same length, regardless of different entities. This causes an issue on the learning outcome of complex structures and patterns. To show this, without loss of generality, let us consider a loop in a graph with a relation r which connects three entities

$$\begin{aligned} {\boldsymbol{e_1}}^{s_L} + \boldsymbol{r} &= \boldsymbol{e}_2^{s_L}, \\ \boldsymbol{e}_2^{s_L} + \boldsymbol{r} &= \boldsymbol{e}_3^{s_L}, \\ \boldsymbol{e}_3^{s_L} + \boldsymbol{r} &= \boldsymbol{e}_1^{s_L}. \end{aligned} \tag{10}$$

After substituting the first line in Equation 10 in the second one and comparing the result with the third equation, we conclude that $\boldsymbol{r} = \boldsymbol{0}$. This is indeed problematic because embedding of all the entities will be the same i.e. different entities are not distinguishable in the geometric space. Now we prove that our model can encode loops, overcoming these issues. In order to learn the loop mentioned above, FieldE should fulfill the following equations

Table 6: Constraints in the embedding space for encoding a loop with three nodes.

MODEL	CONSTRAINTS
TRANSE	$\boldsymbol{r} + \boldsymbol{r} + \boldsymbol{r} = \boldsymbol{0}$
ROTATE	$\boldsymbol{r} \circ \boldsymbol{r} \circ \boldsymbol{r} = \boldsymbol{1}$
FIELDE	$\boldsymbol{f_{\theta_r}}(\boldsymbol{e_1}^{s_L}) + \boldsymbol{f_{\theta_r}}(\boldsymbol{e_2}^{s_L}) + \boldsymbol{f_{\theta_r}}(\boldsymbol{e_3}^{s_L}) = \boldsymbol{0}$

$$\begin{aligned} \boldsymbol{e}_1^{s_L} + f_{\theta_r}(e_1^{s_L}) &= \boldsymbol{e}_2^{s_L}, \\ \boldsymbol{e}_2^{s_L} + f_{\theta_r}(e_2^{s_L}) &= \boldsymbol{e}_3^{s_L}, \\ \boldsymbol{e}_3^{s_L} + f_{\theta_r}(e_3^{s_L}) &= \boldsymbol{e}_1^{s_L}. \end{aligned} \tag{11}$$

In FieldE, after substituting the first line of this equation in the second one, and again substituting the result in the third equation, we obtain the following

$$f_{\theta_r}(\boldsymbol{e}_1^{s_L}) + f_{\theta_r}(\boldsymbol{e}_2^{s_L}) + f_{\theta_r}(\boldsymbol{e}_3^{s_L}) = \boldsymbol{0}. \tag{12}$$

The above equation can be satisfied by FieldE because neural networks with bounded continuous activation functions (here tangent hyperbolic function) are universal approximator and universal classifiers (Hornik et al., 1989; Hornik, 1991; Nayyeri et al., 2017). Therefore, a well-specified neural network for the vector field f_{θ_r} can ensure that equation (12) is satisfied.

We additionally show that our model can also embed a path structure with other three entities $e_1^{s_P}, e_2^{s_P}, e_3^{s_P}$ while preserving a loop structure with $e_1^{s_L}, e_2^{s_L}, e_3^{s_L}$.

$$\begin{aligned} {\boldsymbol{e_1}}^{s_P} + f_{\theta_r}(\boldsymbol{e}_1^{s_P}) &= \boldsymbol{e}_2^{s_P}, \\ \boldsymbol{e}_2^{s_P} + f_{\theta_r}(\boldsymbol{e}_2^{s_P}) &= \boldsymbol{e}_3^{s_P}, \\ \boldsymbol{e}_3^{s_P} + f_{\theta_r}(\boldsymbol{e}_3^{s_P}) &\neq \boldsymbol{e}_1^{s_P}. \end{aligned} \tag{13}$$

After substituting the first line in this equation in the second equation, and again substituting the results in the third equation, we have

$$f_{\theta_r}(\boldsymbol{e}_1^{s_P}) + f_{\theta_r}(\boldsymbol{e}_2^{s_P}) + f_{\theta_r}(\boldsymbol{e}_3^{s_P}) \neq \boldsymbol{0}. \tag{14}$$

Because the embeddings of $\boldsymbol{e}_1^{s_L}, \boldsymbol{e}_2^{s_L}, \boldsymbol{e}_3^{s_L}$, and $\boldsymbol{e}_1^{s_P}, \boldsymbol{e}_2^{s_P}, \boldsymbol{e}_3^{s_P}$ are distinct points, there is a neural

network that approximates the vector field f_{θ_r} in such a way that both Equations (12) and (14) are satisfied due to the universal approximation ability of the underlying network. Therefore, FieldE can learn two different sub-graph structures with the same relation.

C Subsumption

Here we show that variants of FieldE subsume other KGE models.

Proposition 2. DFieldE *subsumes TransE and RotatE.* SFieldE *subsumes ComplEx and QuatE.*

Proof. Here we prove that *DFieldE* subsumes TransE. Note that TransE and RotateE use distance for calculation of scores. Choosing as the manifold $\mathcal{M}$ the Euclidean space $\mathbb{R}^d$, we have for any $\boldsymbol{x} \in \mathbb{R}^d$ that $\mathcal{T}_{\boldsymbol{x}}\mathcal{M} = \mathcal{M}$, and the exponential map is given by $\exp_{\boldsymbol{x}}(\boldsymbol{v}) = \boldsymbol{x} + \boldsymbol{v}$. Then, the FieldE assumption is

$$\boldsymbol{e}_{t+1} = \boldsymbol{e}_t + f_{\theta_r}(\boldsymbol{e}_t).$$

If we set $f_{\theta_r} = \boldsymbol{r}$ (constant vector field), then we have $\boldsymbol{e_{t+1}} = \boldsymbol{e}_t + \boldsymbol{r}$ which is the assumption of the TransE model for triple learning. □

Proof. We now prove that *DFieldE* subsumes RotatE. The assumption in RotatE is

$$\boldsymbol{e}_{t+1} = \boldsymbol{e}_t \circ \boldsymbol{r}, \tag{15}$$

where entities and relations are complex vectors and the modulus of the complex coefficients of each relation is 1 i.e. $|\boldsymbol{r}| = 1$. In the vector form, Equation (15) can be written in real (rotation) matrix-vector multiplication as following

$$\boldsymbol{e}^v_{t+1} = \boldsymbol{R}_r \boldsymbol{e}^v_t,$$

where $\boldsymbol{R}_r$ is a rotation matrix and $\boldsymbol{e}^v_t$ denotes the vector representation of complex numbers (with two components of real and imaginary). Given the assumption of *DFieldE* in Euclidean space i.e. $\boldsymbol{e}^v_{t+1} = \boldsymbol{e}^v_t + f_{\theta_r}(\boldsymbol{e}^v_t)$, and setting $f_{\theta_r}(\boldsymbol{e}^v_t) = (\boldsymbol{R}_r - I)\boldsymbol{e}^v_t$ where I is the identity matrix, the assumption of RotatE is obtained. We conclude that the RotatE model is a special case of *DFieldE*. □

Proof. Here we present the proof of subsumption of the ComplEx model. With the manifold given by Euclidean space, the *SFieldE* uses the following score function

$$S_r(\boldsymbol{e}^v_t, \boldsymbol{e}^v_{t+1}) = \langle \boldsymbol{e}^v_{t+1}, \boldsymbol{e}^v_t + f_{\theta_r}(\boldsymbol{e}^v_t) \rangle. \tag{16}$$

Table 7: Symbols

Symbol	Description
r	relation
e	entity
$\boldsymbol{r}$	head and tail and relation embedding
$\boldsymbol{e}_t$	t-th entity embedding
$\mathcal{M}$	manifold
$\mathcal{T}_p\mathcal{M}$	Tangent Space
d	embedding dimension
v_{e_t}, f_θ	vector field
$S_r(\boldsymbol{e}_t, \boldsymbol{e}_{t+1})$	score of triple (e_t, r, e_{t+1})
η	regularizer
L	number of hidden layer in NN
w^o_i	output weight of NN
w^k_{ij}	kth hidden layer weight of NN
g	activation function
b	bias
$\mathcal{A}_r$	matrix vector field
$\boldsymbol{R}_r$	rotation matrix
$\boldsymbol{e}^v_t r$	complex vector/quaternion number

Now, let us focus on the score function of ComplEx which is

$$S_r(\boldsymbol{e}_t, \boldsymbol{e}_{t+1}) = Re(\langle \bar{\boldsymbol{e}}_{t+1}, \boldsymbol{r}, \boldsymbol{e}_t \rangle), \tag{17}$$

where $\bar{\boldsymbol{e}}$ denotes the complex conjugate of $\boldsymbol{e}$. We represent the above equation in vectored version of complex numbers as following

$$S_r(\boldsymbol{e}^v_t, \boldsymbol{e}^v_{t+1}) = \langle \boldsymbol{e}^v_{t+1}, \alpha_r \boldsymbol{R}_r \boldsymbol{e}^v_t \rangle. \tag{18}$$

We can see if $f_{\theta_r}(\boldsymbol{e}_t) = \alpha_r(\boldsymbol{R}_r - \frac{1}{\alpha_r} I)\boldsymbol{e}^v_t$ in Equation (16), we obtain the score of the ComplEx model in the vectorized form shown in Equation 18. Therefore, ComplEx is also a special case of *SFieldE*. □

Proof. Here, we show that *SFieldE* subsumes QuatE. QuatE uses the following formula for the score function

$$S_r(\boldsymbol{e}_t, \boldsymbol{e}_{t+1}) = \boldsymbol{e}_{t+1} \cdot \boldsymbol{r} \otimes \boldsymbol{e}_t, \tag{19}$$

where $\otimes$ and $\cdot$ denote the Hamilton product and element-wise dot product between two quaternion vectors, respectively. Similar to RotatE, Equation (19) can be written in matrix vector multiplication shown in the following equation

$$S_r(\boldsymbol{e}^v_t, \boldsymbol{e}^v_{t+1}) = \langle \boldsymbol{e}^v_{t+1}, \boldsymbol{R}_r \boldsymbol{e}^v_t \rangle, \tag{20}$$

where $\boldsymbol{R}_r$ is a $4d \times 4d$ matrix and $\boldsymbol{e}^v_t$ is a vectorized version of quaternion numbers. Indeed, the above

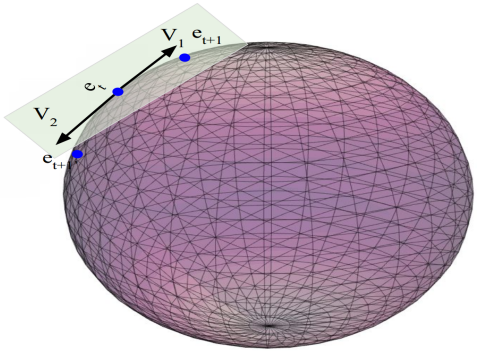


Figure 6: **Handling one-to-many relations.** Movement on the tangent space will be determined based on the head as source of movement and the tail as target.

equation can be constructed by the score function of *SFieldE* which is given by Equation (16). This will be the same as the score function of QuatE in vectorized form, if the vector Field is set to $f_{\theta_r}(\boldsymbol{e}_t^v) = (\boldsymbol{R}_r - I)\boldsymbol{e}_t^v$. Therefore, *SFieldE* subsumes the QuatE model, as well. □

D Modeling Studied Relational Patterns by FieldE

We have theoretically shown that FieldE subsumes other state-of-the-art KGE models. Therefore, it inherits their capability to encode well-studied relational patterns (such as symmetry, anti-symmetry, inversion etc.). Here, we show the approach for encoding one-to-many relations by the FieldE model. The meaning of one-to-many relation is when an entity, say e_t is connected to several entities, $e_{t+1}^1, e_{t+1}^2, \ldots, e_{t+1}^N$. The score of a triple is computed by $dist(\boldsymbol{e}_{t_{n+1}}, exp_{\boldsymbol{e}_{t_n}}(\boldsymbol{v}_{e_{t_n}}^r))$ which is upper-bounded by using the loss function (Sun et al., 2019), i.e. $dist(\boldsymbol{e}_{t_{n+1}}, exp_{\boldsymbol{e}_{t_n}}(\boldsymbol{v}_{e_{t_n}}^r)) \leq \eta_1$. Because smooth Riemanian manifolds are locally Euclidean, there will be an area (with the center of $\boldsymbol{e}_t$) on the manifold where the tails are embedded in that area and all the corresponding triples $(e_t, r, e_{t+1}^1), (e_t, r, e_{t+1}^2), \ldots, (e_t, r, e_{t+1}^N)$ are measured as positive. This is the way how FieldE handles one-to-many relations.

Another way to handle one-to-many relations is to obtain tangent vectors considering tail to encode a triple in the vector space i.e.

$$\boldsymbol{e}_{t+1}^i = exp_{\boldsymbol{e}_t}(f_{\theta_r^i}(\boldsymbol{e}(t))),\ i = 1, \ldots, N. \quad (21)$$

In this way, as shown in Figure 6, the direction of the movement on the tangent space will be determined based on the head as source of movement and the tail as target of movement which consequently enables the model to handle one-to-many relations.

E Training and the Algorithm of FieldE

In order to optimize the parameters of the FieldE model (θ_r and embedding vectors), we employ the loss function used in RotatE (Sun et al., 2019), which is defined as

$$E = -\sum_{(e_t, r, e_{t+1}) \in \mathcal{T}} \Big(\log \sigma(\eta - S_r(\boldsymbol{e}_t, \boldsymbol{e}_{t+1})) + \sum_{(e_t', r, e_{t+1}') \in \mathcal{T}'} p(\boldsymbol{e}_t', r, \boldsymbol{e}_{t+1}') \log \sigma(S_r(\boldsymbol{e}_t', \boldsymbol{e}_{t+1}') - \eta) \Big), \quad (22)$$

where $\sigma(.)$ is the Sigmoid function, $\mathcal{T}, \mathcal{T}'$ are two distinct sets of positive and negative samples respectively, and η is the hyper-parameter of the loss and is adjusted through the validation process. Further,

$$p(\boldsymbol{e}_t', \boldsymbol{r}, \boldsymbol{e}_{t+1}') = \frac{exp(\alpha S_r(\boldsymbol{e}_t', \boldsymbol{e}_{t+1}'))}{\sum exp(\alpha S_r(\boldsymbol{e}_t', \boldsymbol{e}_{t+1}'))}$$

denotes the probability of the triple $(\boldsymbol{e}_t', \boldsymbol{r}, \boldsymbol{e}_{t+1}')$ to be true negative, and the constant α is the temperature of sampling. Note that a negative sample (e_t', r, e_{t+1}') is created from a positive sample (e_t, r, e_{t+1}) by randomly corrupting either e_t or e_{t+1}.

Datasets We run our experiments on several public datasets with diversity in the covered content and graph structure, namely FB15k-237 (Toutanova and Chen, 2015), WN18RR (Dettmers et al., 2018), YAGO3-10 (Mahdisoltani et al., 2013), WikiMovie-300k (Ostapuk et al., 2019), and YouTube (Cen et al., 2019). Statistics of these datasets including the number of their entities and relations as well as the split of train, test, and validation sets are shown in Table 9.

- **FB15k-237** contains a subset of FreeBase dataset (Bollacker et al., 2008) in the form of a standard KG. It is created for experimental purposes, and covers general world knowledge for example science, politic, and sport (Dettmers et al., 2018). FB15k-237 is a refined subset of FB15k (Toutanova and Chen, 2015) in which most of the triples involved in inverse relational patterns are removed from the training set.

Table 8: Link prediction on d WikiMovie-300k, and YouTube. The highlight of performances for different models are marked.

Model	WikiMovie-300k				YouTube			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
TransE	0.32	0.25	0.36	0.46	0.18	0.00	0.28	0.47
RotatE	0.32	0.25	0.35	0.45	0.25	0.14	0.30	0.46
TuckEr	0.27	0.21	0.30	0.39	0.32	0.24	0.34	0.47
ComplEx	0.28	0.21	0.30	0.40	0.32	0.21	0.36	0.54
QuatE	0.13	0.07	0.14	0.24	0.32	0.21	0.36	0.53
DistMult	0.14	0.08	0.15	0.24	0.04	0.01	0.03	0.10
MuRP	0.32	0.26	0.36	0.45	-	-	-	
DFieldE	0.34	0.27	0.37	0.47	0.33	0.24	0.39	0.55

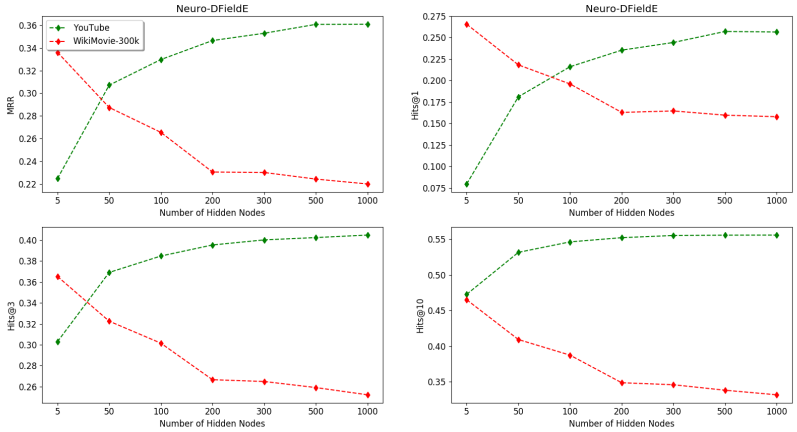


Figure 7: Performance variations with regard to the number of hidden nodes on the YouTube and WikiMovie-300k datasets.

- **WikiMovie-300k** contains extracted knowledge about films such as directors, actors and genre from Wikidata (Vrandečić and Krötzsch, 2014). This dataset contains 300K triples only with entities that appear in at least two triples.

- **YAGO3-10** is a subset of information collected from multilingual Wikipedias including: English, German, French, Dutch, Italian, Spanish, Romanian, Polish, Arabic, and Farsi. The knowledge is general such as people influencing each other, cities and airports of them connected to each other, or organizations etc.

- **YouTube** is a social network dataset which contains information about interactions between 15k users. This information is captured by five relations namely contact, shared friends, shared subscription, shared subscriber, and shared favorite videos between users.

- **WN18RR** is a subset of WordNet dataset where the inverse relations are deleted, and the main relation patterns are symmetry/antisymmetry and composition.

Specific Test Sets from FB15k-237 In order to further analyse the model, we generated four test sets considering the characteristics of the relations. In addition, we explored the relations to choose the most suitable metric for analysis as depending on the semantics of the relations, the common metrics used for comparison in KGE models such as Hits@1 is possible not to be the best criteria for performance judgement of a model.

- **Dataset1**: This test set contains the relation of $/people/person/profession$ which appears 1311 times in the main data, and is chosen for Right Rank Hits@k. This relation is a

one-to-many. Based on the statistics, the lowest number of profession for one person is one and the highest number of professions is recorded to be 13. Therefore, the results for evaluations on this dataset will be shown on Hits@1,3,10.

- **Dataset2**: This test set corresponds to the relations that create triples with only one possible option in the tail. Example of such relations is FB15k-237 is $/common/\ldots/category$ which appears 402 times in the train dataset. Therefore, for such relations, the evaluation using the Hits@k metric is not appropriate. We provided the results using the F-Measure metric.

- **Dataset3**: This test set contains those relations that can have limited options in the tail but not only one. An example of such relations is $/people/person/gender$ which has two possibilities in the tail per each head. It covers 436 triples in the dataset. Considering such relations in the evaluation of Hits@k for the models reduces the fairness of the comparisons as it is not the mistake of the models ranking triples for genders high. This test set has 8 more relations of similar kind and we provide the F-Measure for these as well.

- **Dataset4**: This test set includes two relations with multiple options and always more than 10 possibilities for the tails. Therefore, here we only consider the Hits@10 of right rank for the evaluations. By averaging the left and right ranks in computation of Hits@k for such relations, the performance of the models was not properly measured. An example of such relations is $/film/\ldots/film\_crew\_role$ and appears in 606 triples which is highly effecting the performance of the models if measured on Hits@1,3 and also taking the left rank in account.

Table 9: **Dataset Statistics.** Number of entities, relations, and the splits.

Dataset	#Ent.	#Rel.	#Train	#Valid.	#Test
FB15k-237	15k	237	272k	20k	18k
WikiMovie-300k	36k	588	240k	23k	23k
YAGO3-10	123k	37	1m	5k	5k
YouTube	2k	5	1m	65k	131k
WN18RR	40k	11	86k	3k	3k

Hyperparameter Search We implemented our model in Python using the PyTorch library [2]. We used Adam and Adagrad as the optimizers and tuned the hyperparameters based on the validation set. The learning rate (r) and batch size (b) are tuned on $r = \{0.0002, 0.002, 0.02, 0, 1\}$, $b = \{100, 512, 1024\}$ respectively. The embedding dimension d is fixed to 100 for YAGO3-10, 1000 for FB15k-237, 300 for YouTube and 100 for WikiMovie-300k. We set the number of negative sample to 100 for FB15K-237, 500 for YAGO3-10, 300 for YouTube, 100 for WikiMovie-300k and used adversarial negative sampling for our model as well as the other models we have re-implemented. We presented two versions of FieldE namely DFieldE and SFieldE. DFieldE uses the distance function to compute the score of a triple. On the other hand, SFieldE uses the inner product for score computation. Here we define *DFieldE* as

$$S_r(\boldsymbol{e}_{t_n}, \boldsymbol{e}_{t_{n+1}}) = -dist(\boldsymbol{e}_{t_{n+1}}, exp_{\boldsymbol{e}_{t_n}}(\boldsymbol{v}^r_{e_{t_n}})), \tag{23}$$

and *SFieldE* as

$$S_r(\boldsymbol{e}_{t_n}, \boldsymbol{e}_{t_{n+1}}) = \langle \boldsymbol{e}_{t_{n+1}}, exp_{\boldsymbol{e}_{t_n}}(\boldsymbol{v}^r_{e_{t_n}}) \rangle, \tag{24}$$

with $\langle \cdot, \cdot \rangle$ denoting the Euclidean inner product in the ambient space.

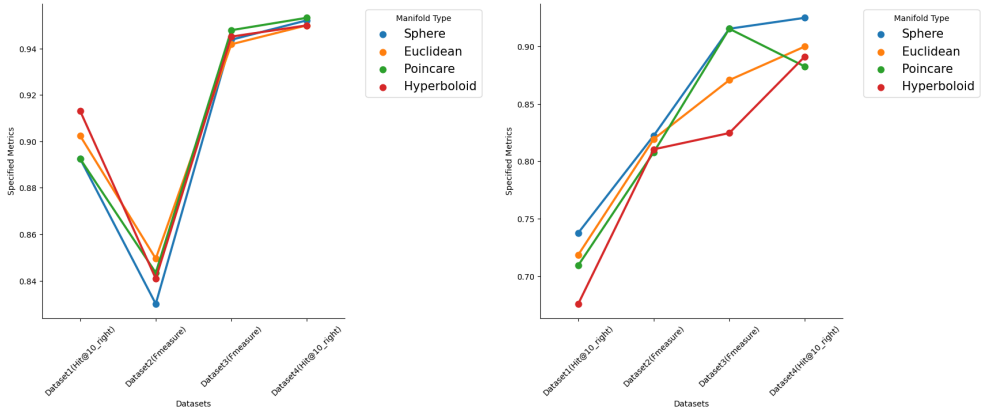
Each of the above versions of FieldE can either use a neural network to approximate the vector field, or use an explicit linear function as a vector field. For the Neural version of FieldE, we used a neural network with two hidden layers. The linear version of FieldE depends on the same parameters except for the hidden layers and having full negative samples with N3 regularization (Lacroix et al., 2018).

For the Neural version of FieldE, we used a neural network with two hidden layers (details in Table 10). with (500,100) hidden nodes for YAGO3-10, (100,100) for FB15K-237a as well as YouTube, and (5,5) for WikiMovie-300k. We fixed the parameter η to 0.5 in equations 5 and 6.

For *SFieldE*, all the following parameters apply except the hidden layers and having full negative samples. For all other models, the same parameters were used. We also performed evaluations in low dimensions where we set the same hypermarameters reported in (Chami et al., 2020) for all the models.

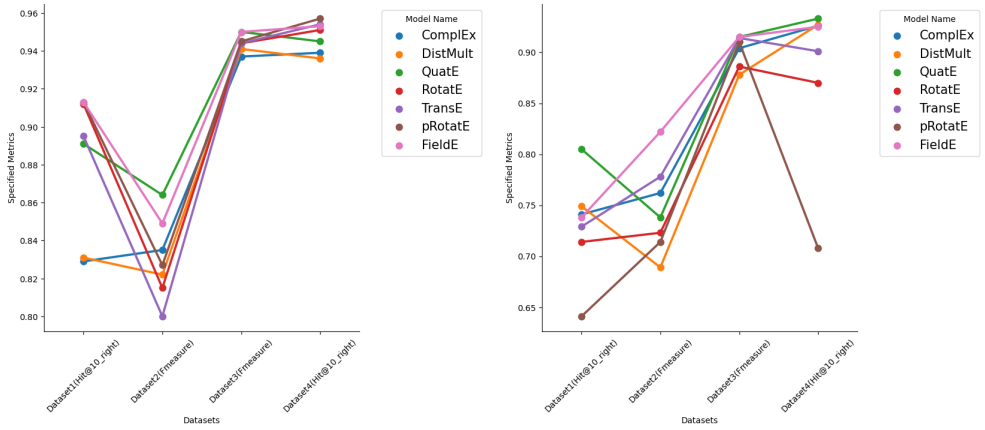
Dataset Complexity and Size of Neural Network. An interesting insight from the performance

[2] https://pytorch.org



(a) Comparison of manifolds in high dimension (500) (b) Comparison of manifolds in low dimension (32)

Figure 8: Performance of the FieldE model using different manifolds of Sphere, Euclidean, Hyperbloid, and Poincaré Ball.



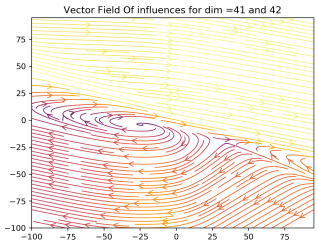
(a) Comparison of models in high dimension (500) (b) Comparison of models in low dimension (32)

Figure 9: Performance comparison of FieldE model and other State-of-the-art models.

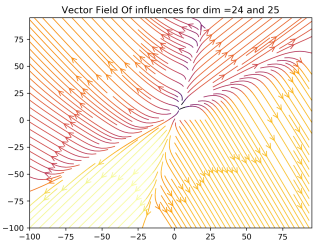
Table 10: Best hyper-parameter settings for *DFieldE* for the three different datasets.

Dataset	**dimension**	**learning rate**	**batch size**	**hidden nodes**	**active function**	**neg.sample**	**margin**
YAGO3-10	100	0.002	512	(500,100)	tanh	500	24
FB15k-237	1000	0.1	1024	(100,100)	tanh	100	9
YouTube	300	0.0002	512	(100,100)	tanh	300	6

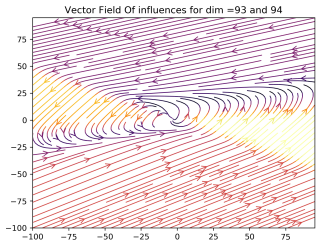
results and the characteristics of the datasets was the connection between the complexity of knowledge graphs and the complexity in design of the FieldE's Neural Network (i.e. the neural network which parameterizes the vector field). As shown in Table 9, the statistics of the datasets also reveals their sparsity and density. For example, YouTube with 5 relations, 2k entities and 1m triples is a more dense and complex knowledge graph than WikiMovie-300k with 588 relations, 36K entities and 240K triples. In other words, the WikiMovie-300k dataset is much sparser than the YouTube dataset. Considering these characteristics, we explored the performance of FieldE in Hits@1,3,10 and MRR by increasing the number of hidden layers in the Neural Network characterising the vector field f_{θ_r}. In Figure 7, we show the results of comparison for YouTube and WikiMovie-300K, where increasing the number of hidden nodes on the YouTube dataset, the results are gradually improved but not for WikiMovie-300K. This means the complexity of the underlying vector field re-



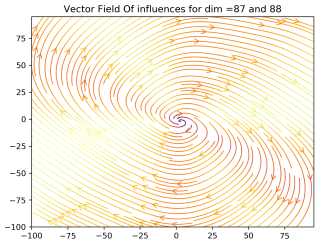
(a) some people influencing many (loop and path)



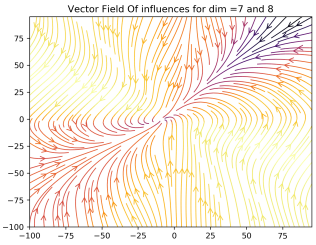
(b) few people influence many (source)



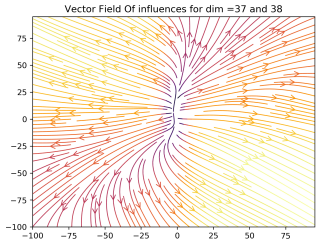
(c) dense influencing of many others (loop and path)



(d) many people influencing each other (loop)



(e) few people being influenced by many (sink)



(f) density in few people influencing others (source)

Figure 10: Vector field visualization of different relations.

quired the corresponding neural network to be complex as well. In the case of WikiMovie-300k dataset, the best results are always achieved by only using 5 nodes in each hidden layers, and further increasing the hidden nodes resulted in a decrease of performance in all of the metrics. This clearly leads to overfitting of the model for such sparse datasets. The results presented in Table 8 justify these observations. As can be seen, the performance of TransE and FieldE are quite close for WikiMovie-300K which is a result of the simplicity of the KG. However, on a more dense and complex dataset such as YouTube, the performance difference of TransE and FieldE are significant. For example, comparing to TransE, our model FieldE performs 15% better in MRR, 24% better in Hits@1, 11% better in Hits@3, and 8% better in Hits@10. Overall, FieldE provides a flexible model by parameterizing the vector field via a neural network with arbitrary architecture to work on KGs with different degrees of complexity. Note that FieldE can be as simple as TransE if the vector field is set to be constant.

Comparison of Manifolds Certain experiments have been done with the aim of differentiating the effect of different manifold choices for the learning performance of the FieldE model. To this end, we performed FieldE on four subdatasets (test sets) of FB15k-237, Namely Dataset1, Dataset2, Dataset3, Dataset4, each of which is analysed with a suitable metric (either Hits@10 right rank or F-Measure). These evaluations have been done using different embedding dimensions, a high dimension of 500 and a low dimension of 32 and the results are shown in Figure 8. As can be seen, the difference of various manifolds choices in high dimensions is small. Non-Euclidean manifolds are performing slightly better in three out of four datasets. The results in low dimensions show a significant effect of the choice of manifold. Generally, the sphere dominates performance in all of the datasets. Additionally, FieldE on a Poincaré Ball performs better than FieldE using Euclidean space on Dataset3 considering the F-Measure metric.

Visualization of Vector Fields. Tracing the vector fields created by different relations gives the intuition about the underlying structures that are preserved by the model. Due to the high dimensionality, the vector fields are usually beyond human perception capabilities, however, in order to provide a presentable illustration, we plot each vector field in pair of dimensions. Therefore, for FieldE with $d = 100$, we created 99 pairs which we selected six graphs constructed from $\{(7, 8), (24, 25), (37, 38), (41, 42), (87, 88), (93, 94)\}$ dimension.

In Figure 10, we illustrate corresponding vec-

tor fields for *influences* relation in the YAGO3-10 knowledge graph. The printed vector fields correspond exactly to the case shown in Figure 5 where some people are influencing others in a loop structure, and some people influence others in a path structure (without a return link). These results show full structure preservation from the graph representation to the vector representation. As discussed before, this capability also avoids incorrect inferences. Subgraph 10(a) shows some people influencing many others both in path and loop structures. Subgraph 10(b) shows trajectories of some people being a *source* influencer for many others. For those source points, the divergence is positive. The subfigure 10(c) is another loop and path occurrence with more density.

A source vector fields is illustrated in subfigure 10(d). The interpretation of this vector field is that there is a person influencing many others. The subfigure 10(e) shows a set of sink nodes (with negative divergence) where they have been influenced by many. And finally, the subfigure 10(f) shows some more dense source entities. Overall, all of these illustrations for the *influences* relation illustrate the capability of FieldE inherited from ODEs and facilitated by the concept of vector field and trajectories.

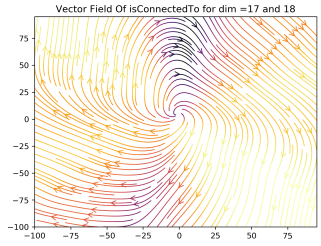
Four other relations (*inConnectedTo*, *hasGender*, *owns*, and *livesIn*) have been selected to provide broader visualizations of vector fields. In Figure 11, we represent the vector fields of these relations for subgraphs with different structures including paths and loops. Each row corresponds to the visualizations of one relation for which three different learned structures are selected to be shown. For example, subgraphs of 11(a), 11(b), and 11(c) correspond to the "isconnectedto" relation that shows which airports are connected to each other in different structures (loops and paths). Our visualizations capture different preserved structures including paths and loops which show some airports are connected in a loop form and some not. In the subgraphs of 11(d), 11(e), 11(f), different structures of "hasGender" relation are captured. Same for the "livesIn" relation, we show different visualizations of the vector fields in 11(j), 11(k), and 11(l). By these visualizations, we aim at giving clarity on the effect of ODEs in learning vector fields. All of these are trajectories lying on relation-specific vector fields learned by the neural network of our model. The arrows show the direction of structure evolution in the vector space for each subfigure.

Further Evaluations. A comparison of our model to a list of six state-of-the-art models has been provided, in addition to four test sets. The purpose of this evaluation is to deepen the analysis by taking more specific metrics into account and by designing test sets based on characteristics of the relations. The results are presented in Figure 9 where ComplEx, Dismult, Quate, RotatE, TransE, and pRotatE mdoels are compared to FieldE. We performed these evaluations in low (32), and high (500) dimensions. As can be seen, our model outperforms all the other models in low dimension in all the datasets except QuatE with which there is a very close competition in Dataset3. In high dimension, our model outperforms other models in Dataset1 and Dataset3. For the other two datasets, QuatE and pRotatE perform close to our model.

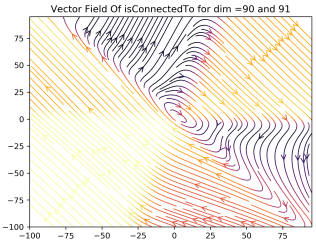
Additionally, we investigate the effect of coordinate transformation by using a neural network i.e.

$$\frac{d\boldsymbol{e}_L(t)}{dt} = f_{\theta_r}(\boldsymbol{e}_L(t)), \tag{25}$$

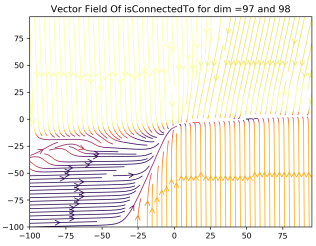
where $\boldsymbol{e}_L(t) = \phi(\boldsymbol{e}(t))$ with ϕ being represented by a neural network. This corresponds to posing the manifold dynamics in a (possibly lower-dimensional) space similar to the encoder step in auto-encoders. We observed that in Dataset4, this variant of our model outperforms the original Euclidean version without coordinate transformation in low dimension of 32 (98 vs 88%) using right Hits@10 (tail ranking). However, on the other test sets, the original version of FieldE obtained a slightly better/closer performance than FieldE with neural coordinate transformation. We conjecture that such results are due to the advantage of the vector field used in the original FieldE which can capture data complexity without using neural coordinate transformation.



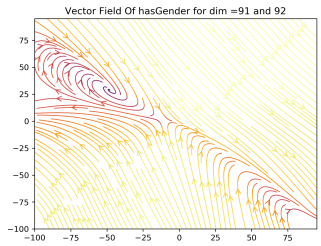
(a) one airport is connected to many (source)



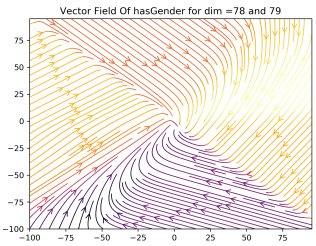
(b) airports are connected to others (path and loop)



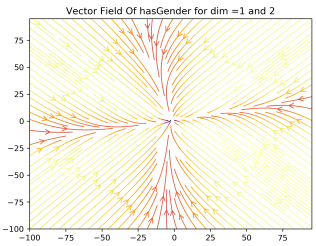
(c) some airports having more connection (mix)



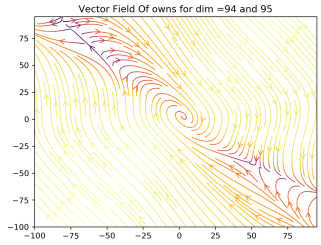
(d) separation of male and female (2 sinks)



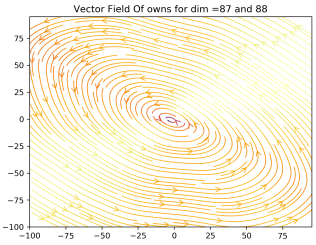
(e) people with one gender (sink)



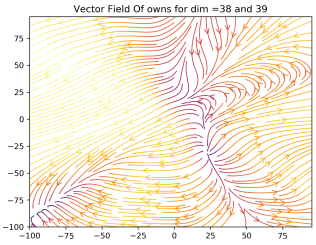
(f) people with one gender (sink)



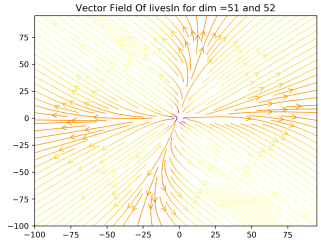
(g) shared stock (loop and path)



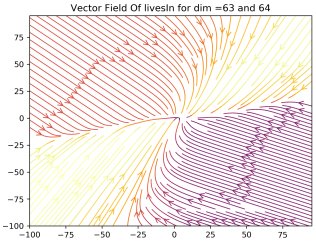
(h) shared stock (loop)



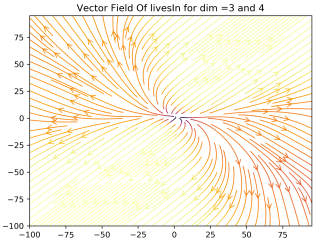
(i) complexity in shared stocks (source and sink)



(j) one person lives in several place (source)



(k) many people lives in one place (sink)



(l) one person lives in several place (source)

Figure 11: Illustration of vector fields for *isConnectedTo*, *hasGender*, *owns*, and *livesIn* relations.