SpeedE: Euclidean Geometric Knowledge Graph Embedding Strikes Back

Anonymous ACL submission

Abstract

Geometric knowledge graph embedding models (gKGEs) have shown great potential for knowledge graph completion (KGC), i.e., automatically predicting missing triples. However, contemporary gKGEs require *high embedding dimensionalities* or *complex embedding spaces* for good KGC performance, drastically limiting their space and time efficiency. Facing these challenges, we propose SpeedE, a lightweight Euclidean gKGE that (1) provides strong inference capabilities, (2) is competitive with state-of-the-art gKGEs, even significantly outperforming them on WN18RR, and (3) dramatically increases their efficiency, in particular, needing solely a fifth of the training time and a fourth of the parameters of the state-of-the-art ExpressivE model on WN18RR to reach the same KGC performance.

1 Introduction

Geometric knowledge graph embedding models (gKGEs) represent entities and relations of a *knowledge graph* (KG) as geometric shapes in the semantic vector space. gKGEs achieved promising performance on *knowledge graph completion* (KGC) and knowledge-driven applications (Wang et al., 2017; Broscheit et al., 2020); while allowing for an intuitive *geometric interpretation* of their captured patterns (Pavlović and Sallinger, 2023).

Efficiency Problem. Recently, increasingly *more complex* embedding spaces were explored to boost the KGC performance of gKGEs (Sun et al., 2019; Zhang et al., 2019; Cao et al., 2021). However, more complex embedding spaces typically require more costly operations, leading to an increased time complexity compared to Euclidean gKGEs (Wang et al., 2021). Even more, most gKGEs require *high-dimensional embeddings* to reach good KGC performance, leading to increased time and space requirements (Chami et al., 2020; Wang et al., 2021). Thus, the need for (1) complex embedding spaces and (2) high-dimensional embeddings increases the time complexity and storage space of gKGEs, hindering their application in resource-constrained environments, especially in mobile smart devices (Sun et al., 2019; Zhang et al., 2019; Wang et al., 2021).

Table 1: This table reports for WN18RR each gKGE's embedding dimensionality, final MRR, convergence time, and number of parameters.

Model	Dim.	MRR	Conv. Time	#Parameters
SpeedE	**50**	**.500**	**6min**	**2M**
ExpressivE	200	**.500**	31min	8M
HAKE	500	.497	50min	41M
ConE	500	.496	1.5h	20M
RotH	500	.496	2h	21M

Challenge and Methodology. Although there has been much work on scalable gKGEs, any such work has focused exclusively on either reducing the embedding dimensionality (Balazevic et al., 2019a; Chami et al., 2020; Bai et al., 2021) or using simpler embedding spaces (Kazemi and Poole, 2018; Zhang et al., 2020; Pavlović and Sallinger, 2023), thus addressing only one side of the efficiency problem. Facing these challenges, this work aims to design a *Euclidean* gKGE that performs well on KGC under *low-dimensional* conditions, reducing its storage space, inference, and training times. To reach this goal, we analyze ExpressivE (Pavlović and Sallinger, 2023), a Euclidean gKGE that has shown promising performance on KGC under high-dimensional conditions.

Contribution. Based on ExpressivE, we propose the lightweight SpeedE model that (1) halves ExpressivE's inference time and (2) enhances ExpressivE's distance function, significantly improving its KGC performance. We evaluate SpeedE on the two standard KGC benchmarks, WN18RR and FB15k-237, finding that it (3) is competitive with SotA gKGEs on FB15k-237 and even outperforms them significantly on WN18RR. Furthermore, we find that (4) SpeedE preserves ExpressivE's KGC performance on WN18RR with much fewer parameters, in particular, requiring solely a fourth of the number of parameters of ExpressivE and solely a fifth of its training time to reach the same KGC performance (Table 1, also c.f. Section 5.3). In total, we propose the SpeedE model, which reaches strong KGC performance using low-dimensional embeddings while maintaining the low space and time requirements of Euclidean gKGEs.

Organization. Our paper is organized as follows: Section 2 reviews related work to embed SpeedE in the context of contemporary literature. Section 3 discusses the KGC problem, evaluation methods, and the ExpressivE model. Section 4 disassembles ExpressivE's components to find a much simpler model that still supports the core inference patterns (c.f. Section 3.1) and continues by building on these results to introduce the lightweight SpeedE model. Section 5 empirically evaluates SpeedE's KGC performance and studies its space and time efficiency. Finally, Section 6 summarizes our results, and the appendix lists all proofs of theorems and additional experimental details.

2 Related Work

The main focus of our work lies on *gKGEs*, i.e., knowledge graph embedding models (KGEs) that allow for a geometric interpretation of their captured inference patterns. Thus, we have excluded neural KGEs as they are typically less interpretable (Dettmers et al., 2018; Socher et al., 2013; Nathani et al., 2019; Wang et al., 2021). In the following, we review relevant literature:

gKGEs can be grouped into families based on their scoring function, including: (1) *functional* and *spatial* models such as TransE (Bordes et al., 2013), RotatE (Sun et al., 2019), MuRP (Balazevic et al., 2019a), RotH (Chami et al., 2020), HAKE (Zhang et al., 2020), ConE (Bai et al., 2021), BoxE (Abboud et al., 2020), and ExpressivE (Pavlović and Sallinger, 2023); and (2) *factorization-based* models such as RESCAL (Nickel et al., 2011), DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), TuckER (Balazevic et al., 2019b), SimplE (Kazemi and Poole, 2018), QuatE (Zhang et al., 2019), and DualQuatE (Cao et al., 2021).

Embedding Space Problem. Although these families are vastly different, many contemporary gKGEs typically overcome the limitations of former ones by exploring increasingly *more complex* embedding spaces. For instance, while RESCAL and DistMult use the Euclidean space, ComplEx uses the complex space, QuatE the quaternion space, and DualQuatE even the dual-quaternion space. However, gKGEs based in increasingly more complex embedding spaces typically require increasingly more costly operations, raising their time complexity compared to Euclidean gKGEs (Wang et al., 2021).

High-Dimensionality Problem. Even more, most gKGEs require *high-dimensional* embeddings to reach good KGC performance (Chami et al., 2020; Wang et al., 2021). However, the need for high embedding dimensionalities of 200, 500, or even 1000 (Sun et al., 2019; Zhang et al., 2019) increases the time complexity and storage space of gKGEs, limiting their efficiency and application to resource-constrained environments, especially mobile smart devices (Wang et al., 2021).

Hyperbolic gKGEs such as RotH and AttH (Chami et al., 2020) achieved promising KGC performance using low-dimensional embeddings, addressing the high-dimensionality problem (Balazevic et al., 2019a; Chami et al., 2020). Moreover, hyperbolic gKGEs allow for high-fidelity and parsimonious representations of *hierarchical relations* (Balazevic et al., 2019a; Chami et al., 2020), i.e., relations that describe hierarchies between entities, such as *part_of*. Most hyperbolic gKGEs were limited to expressing a single global entity hierarchy per relation. ConE (Bai et al., 2021) solves this problem by embedding entities as hyperbolic cones and relations as transformations between these cones. However, any hyperbolic gKGE typically relies on far more costly operations — such as Möbius Addition and Möbius Matrix-Vector Multiplication — than their Euclidean counterparts. Thus, they fail to address the embedding space problem, which results in high time requirements for hyperbolic gKGEs (Wang et al., 2021).

Euclidean gKGEs have recently shown strong representation, inference, and KGC capabilities under high-dimensional conditions. On the one hand, HAKE (Zhang et al., 2020) achieved promising results for representing hierarchical relations on which hyperbolic gKGEs are typically most effective. On the other hand, BoxE (Abboud et al., 2020) managed to capture a large portion of the core inference patterns (c.f. Section 3). Moreover, ExpressivE (Pavlović and Sallinger, 2023) enhances BoxE by improving BoxE's inference capabilities while halving its space complexity. Although Euclidean gKGEs address the embedding space problem, their reported KGC results under low dimensionalities are dramatically lower than those of hyperbolic gKGEs (Chami et al., 2020). Thus, they currently fail to address the high-dimensionality problem.

Our work is inspired by (1) the gap of gKGEs addressing both sides of the efficiency problem, i.e., the use of (a) complex embedding spaces and (b) high-dimensional embeddings (Wang et al., 2021), and (2) the promising results of Euclidean gKGEs (Zhang et al., 2020; Abboud et al., 2020; Pavlović and Sallinger, 2023) under high-dimensional conditions. In contrast to prior work, this paper *jointly* focuses on both sides of the efficiency problem to design a highly resource-efficient gKGE.

3 Background

3.1 Knowledge Graph Completion

This section discusses the KGC problem and its empirical evaluation (Abboud et al., 2020). First, we introduce the *triple vocabulary*, consisting of a finite set of *relations* $\boldsymbol{R}$ and *entities* $\boldsymbol{E}$. We use this vocabulary to define triples, i.e., expressions of the form $r_j(e_h, e_t)$, where $r_j \in \boldsymbol{R}$, $e_h, e_t \in \boldsymbol{E}$, and where we call e_h the triple's *head* and e_t its *tail*. A finite set of triples over the triple vocabulary is called a knowledge graph G. KGC describes the problem of predicting missing triples of G.

Empirical Evaluation. To experimentally evaluate gKGEs, a set of true and corrupted triples is required. True triples $r_i(e_h, e_t) \in G$ are corrupted by substituting either e_h or e_t with any $e_c \in \boldsymbol{E}$ such that the corrupted triple does not occur in G. To estimate a given triple's truth, gKGEs define scores over triples and are optimized to score true triples higher than false ones. The KGC performance of a gKGE is measured with *the mean reciprocal rank* (MRR), the average of inverse ranks ($1/rank$), and Hits@k, the proportion of true triples within the predicted ones whose rank is at maximum k.

Inference Patterns. A gKGE's theoretical capabilities are commonly evaluated by studying the *inference patterns* it captures. An inference pattern is a logical rule $\phi \Rightarrow \psi$, where ϕ is called its body and ψ its head. Following (Sun et al., 2019; Abboud et al., 2020; Pavlović and Sallinger, 2023), a gKGE captures an inference pattern $\phi \Rightarrow \psi$ if there is an embedding instance such that the pattern is captured *exactly* and *exclusively* as formalized in the appendix. This means, at an intuitive level, that there needs to be an embedding instance such that (1) if the instance satisfies the pattern's body, then it also satisfies its head, and (2) the instance does not capture any unwanted inference pattern.

In the following, we briefly list a set of important inference patterns that are commonly studied in the gKGE literature (Sun et al., 2019; Abboud et al., 2020; Pavlović and Sallinger, 2023): (1) symmetry $r_1(X,Y) \Rightarrow r_1(Y,X)$, (2) anti-symmetry $r_1(X,Y) \Rightarrow \neg r_1(Y,X)$, (3) inversion $r_1(X,Y) \Leftrightarrow r_2(Y,X)$, (4) composition $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$, (5) hierarchy $r_1(X,Y) \Rightarrow r_2(X,Y)$, (6) intersection $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$, and (7) mutual exclusion $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$. We shall call these seven types of patterns *core inference patterns* henceforth.

3.2 The ExpressivE Model

This section reviews ExpressivE (Pavlović and Sallinger, 2023), a Euclidean gKGE with strong KGC performance under high dimensionalities.

Representation. ExpressivE embeds entities $e_h \in \boldsymbol{E}$ via vectors $\boldsymbol{e_h} \in \mathbb{R}^d$ and relations $r_j \in \boldsymbol{R}$ via hyper-parallelograms in $\mathbb{R}^{2d}$. The hyper-parallelogram of a relation r_j is parameterized via the following three vectors: (1) a *slope vector* $\boldsymbol{s_j} \in \mathbb{R}^{2d}$ representing the slopes of its boundaries, (2) a *center vector* $\boldsymbol{c_j} \in \mathbb{R}^{2d}$ representing its center, and (3) a *width vector* $\boldsymbol{w_j} \in (\mathbb{R}_{\geq 0})^{2d}$ representing its width. At an intuitive level, a triple $r_j(e_h, e_t)$ is captured to be *true* by an ExpressivE embedding if the concatenation of its head and tail embedding is within r_j's hyper-parallelogram. Formally, this means that a triple $r_j(e_h, e_t)$ is true if the following inequality is satisfied:

$$(\boldsymbol{e_{ht}} - \boldsymbol{c_j} - \boldsymbol{s_j} \odot \boldsymbol{e_{th}})^{|.|} \preceq \boldsymbol{w_j} \tag{1}$$

Where $\boldsymbol{e_{xy}} := (\boldsymbol{e_x} || \boldsymbol{e_y}) \in \mathbb{R}^{2d}$ with $||$ representing concatenation and $e_x, e_y \in \boldsymbol{E}$. Furthermore, the inequality uses the following operators: the element-wise less or equal operator $\preceq$, the element-wise absolute value $\boldsymbol{x}^{|.|}$ of a vector $\boldsymbol{x}$, and the element-wise (i.e., Hadamard) product $\odot$.

Scoring. ExpressivE employs the typical distance function $D : \boldsymbol{E} \times \boldsymbol{R} \times \boldsymbol{E} \rightarrow \mathbb{R}^{2d}$ of spatial gKGEs (Abboud et al., 2020; Pavlović and Sallinger, 2023), which is defined as follows:

$$D(h, r_j, t) = \begin{cases} \boldsymbol{\tau_{r_j(h,t)}} \oslash \boldsymbol{m_j}, & \text{if } \boldsymbol{\tau_{r_j(h,t)}} \preceq \boldsymbol{w_j} \\ \boldsymbol{\tau_{r_j(h,t)}} \odot \boldsymbol{m_j} - \boldsymbol{k_j}, & \text{otherwise} \end{cases} \quad (2)$$

Where $\oslash$ denotes the element-wise division operator, $\boldsymbol{\tau_{r_j(h,t)}} := (\boldsymbol{e_{ht}} - \boldsymbol{c_j} - \boldsymbol{s_j} \odot \boldsymbol{e_{th}})^{|.|}$ denotes the triple embedding, $\boldsymbol{m_j} := \boldsymbol{2} \odot \boldsymbol{w_j} + \boldsymbol{1}$ represents the distance function's slopes, and $\boldsymbol{k_j} := \boldsymbol{0.5} \odot (\boldsymbol{m_j} - \boldsymbol{1}) \odot (\boldsymbol{m_j} - \boldsymbol{1} \oslash \boldsymbol{m_j})$. Based on the distance function, ExpressivE defines the scoring function for quantifying the plausibility of a triple $r_j(h,t)$ as $s(h, r_j, t) = -||D(h, r_j, t)||_2$.

4 The Methodology

Our goal is to design a gKGE that addresses the efficiency problems raised by the use of (1) complex embedding spaces and (2) high-dimensional embeddings while (3) allowing for a geometric interpretation of its embeddings (Abboud et al., 2020; Pavlović and Sallinger, 2023). We reach this goal by designing a KGC model that (1) is based in the Euclidean space, (2) reaches high KGC performance under low-dimensional conditions while at the same time supports the *core inference patterns* (Section 3.1), and (3) is a gKGE.

Toward our goal, Section 4.1 analyzes the SotA ExpressivE model, finding that it uses redundant parameters that negatively affect its inference time. By redundant parameters, we mean parameters that can be removed while preserving the support of the core inference patterns (Section 3.1). Facing this problem, we propose the lightweight Min_SpeedE model that removes these redundancies, halving ExpressivE's inference time (Section 4.1).

However, Min_SpeedE loses the ability to adjust its distance function, which is important for representing hierarchical relations (as empirically verified in Section 5). Thus, Section 4.2 introduces SpeedE, a model that enhances Min_SpeedE by adding carefully designed parameters for flexibly adjusting the distance function while preserving Min_SpeedE's low inference times.

4.1 Min_SpeedE

To design Min_SpeedE, let us first analyze ExpressivE's parameters, particularly its width vector. Adjusting ExpressivE's width vector $\boldsymbol{w_j}$ has two competing effects: (1) it alters the distance function's slopes (by $\boldsymbol{m_j}$ in Inequality 2), and (2) it changes which entity pairs are inside the relation hyper-parallelogram (by $\boldsymbol{w_j}$ in Inequality 1). To increase ExpressivE's time efficiency substantially, we introduce Min_SpeedE, a constrained version of ExpressivE that replaces the relation-wise width vectors $\boldsymbol{w_j} \in (\mathbb{R}_{\geq 0})^{2d}$ by a constant value $w \in \mathbb{R}_{>0}$ - that is shared across all relations $r_j \in \boldsymbol{R}$. The following paragraphs theoretically analyze Min_SpeedE's inference capabilities and time efficiency.

Inference Capabilities. We find that Min_SpeedE surprisingly still captures the core inference patterns (given in Section 3.1) and prove this in Theorem 4.1. We give the full proof in the appendix and discuss one of the most interesting parts here, namely, hierarchy patterns.

Theorem 4.1. *Min_SpeedE captures the core inference patterns, i.e., symmetry, anti-symmetry, inversion, composition, hierarchy, intersection, and mutual exclusion.*

Hierarchy Patterns. According to Pavlović and Sallinger (2023), an ExpressivE model captures a hierarchy pattern $r_1(X, Y) \Rightarrow r_2(X, Y)$ iff r_1's hyper-parallelogram is a proper subset of r_2's. Thus, one would expect that ExpressivE's ability to capture hierarchy patterns is lost in Min_SpeedE, as the width parameter $w \in \mathbb{R}_{>0}$ (responsible for adjusting a hyper-parallelogram's size) is shared across all hyper-parallelograms. However, the actual size of a hyper-parallelogram does not solely depend on its width but also on its slope parameter $\boldsymbol{s_j} \in \mathbb{R}^{2d}$, allowing one hyper-parallelogram $\boldsymbol{H_1}$ to properly subsume another $\boldsymbol{H_2}$ even when they share the same width parameter w. We have visualized two hyper-parallelograms $\boldsymbol{H_2} \subset \boldsymbol{H_1}$ with the same width parameter w in Figure 1.

Intuition. Min_SpeedE can capture $\boldsymbol{H_2} \subset \boldsymbol{H_1}$ as w (depicted with orange dotted lines) represents the intersection of the bands (depicted with blue and green dotted lines), expanded from the hyper-parallelogram, and the axis of the band's corresponding dimension. Thus, a hyper-parallelogram's actual size can be adapted by solely changing its slopes, removing the need for a learnable width parameter per dimension and relation.

Inference Time. The most costly operations during inference are operations on vectors. Thus, we can estimate ExpressivE's and Min_SpeedE's inference time by counting the number of vector operations necessary for computing a triple's score: By reducing the width vector to a scalar, many operations reduce from a vector to a scalar operation. In particular, the calculation of m_j and k_j uses solely scalars in Min_SpeedE instead of vectors. Thus,

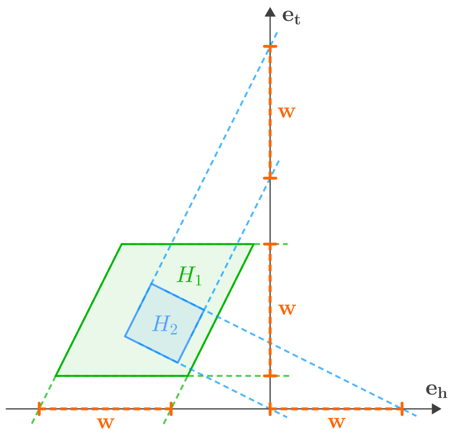

Figure 1: Representation of the two-dimensional relation hyper-parallelograms $\boldsymbol{H_1}$ and $\boldsymbol{H_2}$, such that $\boldsymbol{H_1}$ subsumes $\boldsymbol{H_2}$ and such that they share the same width parameter w in each dimension.

ExpressivE needs 15, whereas Min_SpeedE needs solely 8 vector operations to compute a triple's score. This corresponds to Min_SpeedE using approximately half the number of vector operations of ExpressivE for computing a triple's score, thus roughly halving ExpressivE's inference time, which aligns with Section 5.3's empirical results.

Key Insights. Fixing the width to a constant value w stops Min_SpeedE from adjusting the distance function's slopes. As we will empirically see in Section 5, the effect of this is a severely degraded KGC performance on hierarchical relations. Introducing independent parameters for adjusting the distance function's slopes solves this problem. However, these parameters must be designed carefully to (1) preserve ExpressivE's geometric interpretation and (2) retain the reduced inference time provided by Min_SpeedE. Each of these aspects will be covered in detail in the next section.

4.2 SpeedE

SpeedE further enhances Min_SpeedE by adding the following two carefully designed scalar parameters to each relation embedding: (1) the inside distance slope $s^i_j \in [0,1]$ and (2) the outside distance slope s^o_j with $s^i_j \leq s^o_j$. Let $m^i_j := 2s^i_j w + 1$, $m^o_j := 2s^o_j w + 1$, and $k_j := m^o_j(m^o_j - 1)/2 - (m^i_j - 1)/(2m^i_j)$, then SpeedE defines the following distance function:

$$D(h, r_j, t) = \begin{cases} \boldsymbol{\tau_{r_j(h,t)}} \oslash m^i_j, & \text{if } \boldsymbol{\tau_{r_j(h,t)}} \preceq w \\ \boldsymbol{\tau_{r_j(h,t)}} \odot m^o_j - k_j, & \text{otherwise} \end{cases} \quad (3)$$

Again, the distance function is separated into two piece-wise linear functions: (1) the inside distance $D_i(h, r_j, t) = \boldsymbol{\tau_{r_j(h,t)}} \oslash m^i_j$ for triples that are captured to be true (i.e., $\boldsymbol{\tau_{r_j(h,t)}} \preceq w$) and (2) the outside distance $D_o(h, r_j, t) = \boldsymbol{\tau_{r_j(h,t)}} \odot m^o_j - k_j$ for triples that are captured to be false (i.e., $\boldsymbol{\tau_{r_j(h,t)}} \succ w$). Based on this function, SpeedE defines the score as $s(h, r_j, t) = -||D(h, r_j, t)||_2$.

Geometric Interpretation. The intuition of s^i_j and s^o_j is that they control the slopes of the respective linear inside and outside distance functions. However, without any constraints on s^i_j and s^o_j, SpeedE would lose ExpressivE's intuitive geometric interpretation (Pavlović and Sallinger, 2023) as s^i_j and s^o_j could be chosen in such a way that distances of embeddings within the hyper-parallelogram are larger than those outside. By constraining these parameters to $s^i_j \in [0,1]$ and $s^i_j \leq s^o_j$, we preserve lower distances within hyper-parallelograms than outside and, thereby, the intuitive geometric interpretation of our embeddings.

Inference Time. The additional introduction of two scalar distance slope parameters $s^i_j, s^o_j \in \mathbb{R}$ per relation r_j does not change the number of vector operations necessary for computing a triple's score and, thus, does not significantly affect SpeedE's inference time. Thus, we expect that SpeedE retains the time efficiency of Min_SpeedE, as empirically validated in Section 5.3.

With this, we have finished our introduction and theoretical analysis of SpeedE. What remains to be shown is its empirical performance, which we shall evaluate next.

5 Experiments

This section empirically evaluates SpeedE: Section 5.1 describes the experimental setup. Section 5.2 studies SpeedE's KGC performance, finding that it achieves competitive performance on FB15k-237 to SotA gKGEs and even significantly outperforms them on WN18RR. Section 5.3 studies SpeedE's space and time efficiency, finding that on WN18RR, SpeedE needs a quarter of ExpressivE's parameters solely to reach the same KGC performance while training five times faster than it.

5.1 Experimental Setup

Datasets. We empirically evaluate SpeedE on the two standard KGC benchmarks, WN18RR (Bordes et al., 2013; Dettmers et al., 2018) and FB15k-237 (Bordes et al., 2013; Toutanova and Chen, 2015).

WN18RR is extracted from the WordNet database (Miller, 1995), representing lexical relations between English words, thus naturally containing many hierarchical relations (e.g., hypernym-of) (Chami et al., 2020). FB15k-237 is a subset of a collaborative database consisting of general knowledge (in English) called Freebase (Bollacker et al., 2007), which contains both hierarchical relations (e.g., part-of) and non-hierarchical ones (e.g., nationality) (Chami et al., 2020). Table 2 displays the following characteristics of both benchmarks: their number of entities $|\boldsymbol{E}|$ and relations $|\boldsymbol{R}|$, their curvature C_G (taken from Chami et al. (2020)), and the Krackhardt scores κ (taken from Bai et al. (2021)), consisting of the four metrics: (*connectedness*, *hierarchy*, *efficiency*, *LUBedness*). Both C_G and κ state how tree-like a benchmark is and, thus, how hierarchical its relations are. Following the procedure of Chami et al. (2020), we employ the standard augmentation protocol (Lacroix et al., 2018), adding inverse relations to the benchmarks.

Table 2: Benchmark dataset characteristics. Curvature C_G is from (Chami et al., 2020); the lower, the more hierarchical the data. Krackhardt scores κ are from (Bai et al., 2021); the higher, the more hierarchical the data.

Dataset	$\|\boldsymbol{E}\|$	$\|\boldsymbol{R}\|$	C_G	κ
FB15k-237	14,541	237	-0.65	(1.00, 0.18, 0.36, 0.06)
WN18RR	40,943	11	-2.54	(1.00, 0.61, 0.99, 0.50)

Setup. We compare our SpeedE model to (1) the Euclidean gKGEs ExpressivE (Pavlović and Sallinger, 2023), HAKE (Zhang et al., 2020), TuckER (Balazevic et al., 2019b), MuRE (Balazevic et al., 2019a), and RefE, RotE, and AttE (Chami et al., 2020), (2) the complex gKGEs ComplEx-N3 (Lacroix et al., 2018) and RotatE (Sun et al., 2019), and (3) the hyperbolic gKGEs ConE (Bai et al., 2021), MuRP (Balazevic et al., 2019a), and RefH, RotH, and AttH (Chami et al., 2020). Following Pavlović and Sallinger (2023), we train SpeedE and ExpressivE for up to 1000 epochs using gradient descent and the Adam optimizer (Kingma and Ba, 2015) and stop the training if the validation H@10 score does not increase by minimally 0.5% for WN18RR and 1% for FB15k-237 after 100 epochs. We average the experimental results over three runs on each benchmark to handle marginal performance fluctuations. Furthermore, as in (Chami et al., 2020), we evaluate SpeedE and ExpressivE in the low-dimensional setting using an embedding dimensionality of 32.

Reproducibility. We list further details on our experimental setup, hardware, hyperparameters, libraries (Ali et al., 2021), and definitions of metrics in the appendix. Furthermore, we include our code and a link to pre-trained gKGEs in the supplementary material to facilitate reproducibility.

5.2 Knowledge Graph Completion

This section evaluates SpeedE's and ExpressivE's KGC performance. Furthermore, we study how well they represent hierarchical relations, on which hyperbolic gKGEs are typically most effective (Balazevic et al., 2019a; Chami et al., 2020). Finally, we analyze the effect of embedding dimensionality on SpeedE's KGC performance.

Table 3: KGC performance under low dimensionalities ($d = 32$) of SpeedE, Min_SpeedE, ExpressivE, and SotA gKGEs on FB15k-237 and WN18RR split by embedding space. The results of: SpeedE, Min_SpeedE, and ExpressivE were obtained by us; ConE are from (Bai et al., 2021), HAKE are from (Zheng et al., 2022), and any other gKGE are from (Chami et al., 2020).

Space	Model	WN18RR			FB15k-237		
		MRR	H@1	H@10	MRR	H@1	H@10
Euclidean	SpeedE	**.493**	**.446**	**.584**	**.320**	**.227**	**.504**
	Min_SpeedE	.485	.442	.573	.319	.226	.502
	ExpressivE	.485	.442	.571	.298	.208	.476
	TuckER	.428	.401	.474	.306	.223	.475
	MuRE	.458	.421	.525	.313	.226	.489
	RefE	.455	.419	.521	.302	.216	.474
	RotE	.463	.426	.529	.307	.220	.482
	AttE	.456	.419	.526	.311	.223	.488
	HAKE	.416	.389	.467	.296	.323	.463
Non-Euclidean	RotatE	.387	.330	.491	.290	.208	.458
	ComplEx-N3	.420	.390	.460	.294	.211	.463
	MuRP	.465	.420	.544	.323	.235	.501
	RefH	.447	.408	.518	.312	.224	.489
	RotH	**.472**	.428	**.553**	.314	.223	.497
	AttH	.466	.419	.551	**.324**	**.236**	**.501**
	ConE	.471	**.436**	.537	-	-	-

Low-Dimensional KGC. Following the evaluation protocol of Chami et al. (2020), we evaluate each gKGE's performance under dimensionality $d = 32$. Table 3 presents the results of this evaluation. Our enhanced SpeedE model is competitive with SotA gKGEs on FB15k-237 and even outperforms ExpressivE and any other competing gKGE on WN18RR by a relative difference of 5% on H@10. Furthermore, SpeedE's performance gain over Min_SpeedE on the highly hierarchical dataset WN18RR (see Table 2) provides strong empirical evidence for the effectiveness of the distance slope parameters for representing hierarchical relations under low-dimensional conditions. SpeedE's performance on the more hierarchical WN18RR already questions the necessity of hyperbolic gKGEs for representing hierarchical relations, which will be further investigated in the following.

Hierarchical Relations (Chami et al., 2020; Zhang et al., 2020) describe hierarchies between entities, such as *part_of*. Hyperbolic gKGEs have shown great potential for representing hierarchical relations, outperforming Euclidean gKGEs under low-dimensional conditions and thereby justifying the increased model complexity added by the hyperbolic space (Chami et al., 2020). To study SpeedE's performance on hierarchical relations, we evaluate SpeedE on the triples of any hierarchical relation of WN18RR following the methodology of Bai et al. (2021). Table 4 presents the results of this study. It reveals that SpeedE significantly improves over ExpressivE on most relations and outperforms RotH on five out of the seven hierarchical ones. Most notably, SpeedE improves over RotH by a relative difference of 23% on H@10 on the hierarchical relation *_member_of_domain_usage*, providing empirical evidence for SpeedE's promising potential for representing hierarchical relations even under low-dimensional settings. The performance gain on hierarchical relations is likely due to the added distance slope parameters, which allow for independently adjusting the distance function's slopes.

Table 4: H@10 of ExpressivE, RotH, and SpeedE on hierarchical relations (Bai et al., 2021) of WN18RR.

Relation	ExpressivE	RotH	SpeedE
_member_meronym	0.362	**0.399**	0.379
_hypernym	0.276	0.276	**0.301**
_has_part	0.308	**0.346**	0.330
_instance_hypernym	0.509	0.520	**0.543**
_member_of_domain_region	0.365	0.365	**0.397**
_member_of_domain_usage	**0.545**	0.438	0.538
_synset_domain_topic_of	0.468	0.447	**0.502**

Dimensionality Study. To analyze the effect of the embedding dimensionality on the KGC performance, we evaluate state-of-the-art gKGEs on WN18RR under varied dimensionalities. Figure 2 visualizes the results of this study, displaying error bars for our SpeedE model with average MRR and standard deviation computed over three runs. The figure reveals that, surprisingly, ExpressivE significantly outperforms RotH, especially under low-dimensional conditions, and that the enhanced SpeedE model achieves an additional performance improvement over ExpressivE. This result provides further evidence for the great potential of Euclidean gKGEs under low-dimensional conditions.

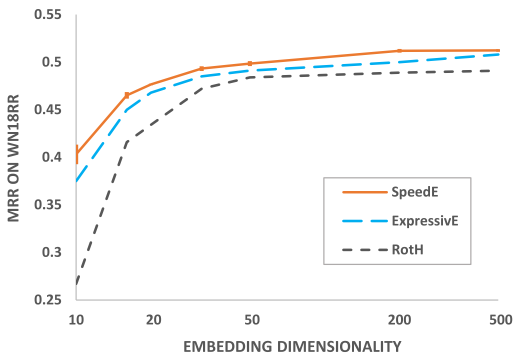

Figure 2: MRR of SotA gKGEs on WN18RR using $d \in \{10, 16, 20, 32, 50, 200, 500\}$.

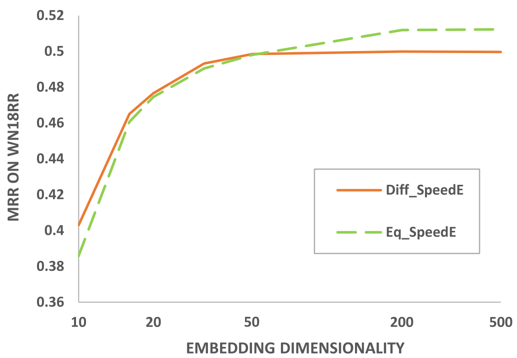

Figure 3: MRR of different ablations of SpeedE on WN18RR using $d \in \{10, 16, 20, 32, 50, 200, 500\}$

High-Dimensional KGC. The KGC performance of SotA gKGEs under high-dimensional conditions (i.e., $d \geq 200$) is listed in the appendix. It reveals that on FB15k-237, SpeedE achieves highly competitive KGC performance compared to gKGEs of its own family while dramatically outperforming any competing gKGE on WN18RR.

Ablation. Finally, to study the necessity of s^i_j and s^o_j in SpeedE, we introduce two versions of SpeedE: (1) Eq_SpeedE that forces $s^i_j = s^o_j$ and (2) Diff_SpeedE, where s^i_j and s^o_j can be different. We hypothesize that the flexibility of different s^i_j and s^o_j might be beneficial under lower dimensionalities, while under higher dimensionalities, reducing the number of parameters and thus setting $s^i_j = s^o_j$ might be beneficial. Figure 3 visualizes the result of this analysis, empirically supporting our hypothesis, as Diff_SpeedE outperforms Eq_SpeedE under low dimensionalities and vice-versa in high ones.

5.3 Space and Time Efficiency

This section empirically analyzes SpeedE's space and time efficiency compared to SotA gKGEs. Following the methodology of Wang et al. (2021), we first analyze the training time per epoch of

SpeedE, Min_SpeedE, and ExpressivE. Next, to allow for a fair comparison of the space and time efficiency of SpeedE and SotA gKGEs, we study each gKGE's model size and convergence time under hyper-parameter settings that achieve approximately equal KGC performance.

Time per Epoch. Following the methodology of Wang et al. (2021), Table 5 displays the training time per epoch of SpeedE, Min_SpeedE, and ExpressivE for WN18RR and FB15k-237 with embedding dimensionality $d = 32$, negative sampling size $n = 500$, and batch size $b = 500$. The times per epoch were recorded on a GeForce RTX 2080 Ti GPU of our internal cluster. The empirical results of Table 5 align with the theoretical results of Sections 4.1 and 4.2, stating that SpeedE and Min_SpeedE approximately halve ExpressivE's inference time and, thus, also its time per epoch.

Table 5: Time per epoch of SpeedE, Min_SpeedE, and ExpressivE.

Model	Time per Epoch	
	WN18RR	FB15k-237
SpeedE	7s	22s
Min_SpeedE	6s	19s
ExpressivE	15s	46s

Next, to provide a fair comparison of each gKGE's space and time efficiency, we measure the convergence time of gKGEs with approximately equal KGC performance. Specifically, we observed that SpeedE with dimensionality $d = 50$ achieves comparable or slightly better KGC performance on WN18RR to ExpressivE with $d = 200$ and the best-published results of RotH, HAKE, and ConE with $d = 500$. In particular, the results are summarized in Table 1 (provided in Section 1).

Hypotheses. Since (1) the dimensionality of SpeedE embeddings is much smaller in comparison to RotH's, HAKE's, ConE's, and ExpressivE's dimensionality, while (2) SpeedE achieves comparable or even slightly better KGC performance, we expect a considerable improvement in SpeedE's space and time efficiency at comparable KGC performance. Next, based on Table 1's results, we analyze how strongly SpeedE reduces the model size and convergence time of competing gKGEs.

Model Size Analysis. Since $|\boldsymbol{R}| << |\boldsymbol{E}|$ in most graphs, (WN18RR: $|\boldsymbol{R}|/|\boldsymbol{E}| = 0.00012$) and since SpeedE, ExpressivE, ConE, and RotH embed each entity with a single real-valued vector, SpeedE ($d = 50$) needs solely a quarter of ExpressivE's ($d = 200$) and a tenth of ConE's and RotH's ($d = 500$) number of parameters, while preserving their KGC performance on WN18RR (Table 1). As HAKE requires two real-valued vectors per entity, SpeedE ($d = 50$) solely needs a twentieth of HAKE's ($d = 500$) parameters to achieve a slightly better KGC performance. Table 1 lists the number of parameters of trained SpeedE and SotA gKGE models, empirically confirming that SpeedE significantly reduces the size of competing gKGEs.

Convergence Time Analysis. To quantify the convergence time, we measure for each gKGE the time to reach a validation MRR score of 0.490, i.e., approximately 1% less than the worst reported MRR score of Table 1. As outlined in the table, SpeedE converges already after $6min$. Thus, while keeping strong KGC performance on WN18RR, SpeedE speeds up ExpressivE's convergence time by a factor of 5, HAKE's by a factor of 9, ConE's by a factor of 15, and RotH's by a factor of 20.

Discussion. These results show that SpeedE is not solely competitive with SotA gKGEs on FB15k-237 and significantly outperforms them on WN18RR, but even preserves their KGC performance on WN18RR with much fewer parameters and a dramatically shorter convergence time, in particular speeding up the convergence time of the SotA ExpressivE model by a factor of 5, while using solely a fourth of its number of parameters.

6 Conclusion

Although there has been much work on resource-efficient gKGEs, any such work has focused exclusively on reducing the embedding dimensionality (Balazevic et al., 2019a; Chami et al., 2020; Bai et al., 2021) or using simpler embedding spaces (Kazemi and Poole, 2018; Zhang et al., 2020; Pavlović and Sallinger, 2023), thus addressing only one side of the efficiency problem. In this work, we address the embedding space and dimensionality side jointly by introducing SpeedE, a lightweight gKGE that (1) provides strong inference capabilities, (2) is competitive with SotA gKGEs, even significantly outperforming them on WN18RR, and (3) dramatically increases the efficiency of current gKGEs, in particular, needing solely a fifth of the training time and a fourth of the number of parameters of the SotA ExpressivE model on WN18RR to reach the same KGC performance.

7 Limitations and Ethical Impact

As mentioned in the call for papers, we use this additional page to discuss our work's limitations and ethical impact.

7.1 Limitations

Since gKGEs naturally provide a geometric interpretation of their learned patterns, how to automatically and efficiently mine these learned patterns from the embeddings — to make the implicitly learned knowledge explicit and further raise the model's transparency — remains an open challenge and forms an exciting direction for future work.

7.2 Ethical Impact

We designed SpeedE with the goal of finding a highly resource-efficient model for KGC that, at the same time, provides a geometric interpretation of its captured patterns. Therefore, our work aligns with two pressing challenges of the machine learning community in general and the KGC community in particular, namely, (1) raising the resource efficiency of KGC models while (2) offering some degree of explainability via the geometric interpretation of captured patterns. Specifically, SpeedE reduces the training time — and thus the total compute — of the SotA ExpressivE model on WN18RR to one-fourth while sustaining ExpressivE's KGC performance and geometric interpretation. Therefore, we do not foresee any negative impact but even expect a potential positive *environmental* (see 1) and *social impact* (see 2) of our work by introducing a highly resource-efficient model that allows for some degree of explainability.

References

Ralph Abboud, İsmail İlkan Ceylan, Thomas Lukasiewicz, and Tommaso Salvatori. 2020. Boxe: A box embedding model for knowledge base completion. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Sahand Sharifzadeh, Volker Tresp, and Jens Lehmann. 2021. PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings. *Journal of Machine Learning Research*, 22(82):1–6.

Yushi Bai, Zhitao Ying, Hongyu Ren, and Jure Leskovec. 2021. Modeling heterogeneous hierarchies with relation-specific hyperbolic cones. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 12316–12327.

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019a. Multi-relational poincaré graph embeddings. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019b. TuckER: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5185–5194, Hong Kong, China. Association for Computational Linguistics.

Kurt D. Bollacker, Robert P. Cook, and Patrick Tufts. 2007. Freebase: A shared database of structured general human knowledge. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1962–1963. AAAI Press.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2787–2795.

Samuel Broscheit, Kiril Gashteovski, Yanjie Wang, and Rainer Gemulla. 2020. Can we predict new facts with open knowledge graph embeddings? A benchmark for open link prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2296–2308. Association for Computational Linguistics.

Zongsheng Cao, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. 2021. Dual quaternion knowledge graph embeddings. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6894–6902.

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*, pages 6901–6914, Online. Association for Computational Linguistics.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1811–1818. AAAI Press.

Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 4289–4300.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. 2019. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical tensor decomposition for knowledge base completion. In *International Conference on Machine Learning*.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4710–4723. Association for Computational Linguistics.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 809–816. Omnipress.

Aleksandar Pavlović and Emanuel Sallinger. 2023. Expressive: A spatio-functional embedding for knowledge graph completion. In *11th International Conference on Learning Representations, ICLR 2023, Kigal, Rwanda, May 1-5, 2023*.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 926–934.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2071–2080. JMLR.org.

Kai Wang, Yu Liu, Dan Lin, and Michael Sheng. 2021. Hyperbolic geometry is not necessary: Lightweight Euclidean-based models for low-dimensional knowledge graph embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 464–474, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans. Knowl. Data Eng.*, 29(12):2724–2743.

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 *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embeddings. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 2731–2741.

Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. 2020. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 3065–3072. AAAI Press.

Wenjie Zheng, Wenxue Wang, Fulan Qian, Shu Zhao, and Yanping Zhang. 2022. Hyperbolic hierarchical knowledge graph embeddings for link prediction in low dimensions. *CoRR*, abs/2204.13704.

A Organization

This appendix includes complete proofs, experimental setup details, and additional results. In particular, Section B reports the KGC performance of SpeedE and SotA gKGEs under high-dimensional conditions. Section C briefly summarizes the notation that is used throughout this paper. Section D formally defines vital concepts for SpeedE that we will use in our proofs. Based on the introduced concepts, Section E proves Theorem 4.1. Finally, Section F lists details on reproducing our results and on our implementation, training setup, evaluation protocol, and estimated CO2 emissions.

B High-Dimensional Knowledge Graph Completion

This section reports the KGC performance of SotA gKGEs under high-dimensional conditions (i.e., $d \geq 200$). Table 6 displays these results, where the result for SpeedE was obtained by us, the result for ExpressivE is from (Pavlović and Sallinger, 2023), the results for ConE and HAKE are from (Bai et al., 2021), the results for DistMult, ConvE, and ComplEx are from (Dettmers et al., 2018), and the results for any other gKGE are from (Chami et al., 2020). Table 6 reveals that on FB15k-237, SpeedE achieves highly competitive KGC performance compared to gKGEs of its own family while dramatically outperforming any competing gKGE on WN18RR.

Table 6: KGC performance under high dimensionalities of SpeedE and SotA gKGEs on FB15k-237 and WN18RR split by model family.

Family	Model	WN18RR			FB15k-237		
		MRR	H@1	H@10	MRR	H@1	H@10
Functional / Spatial	SpeedE	**.512**	.460	**.615**	.348	.253	.536
	ExpressivE	.508	**.464**	.597	**.350**	**.256**	.535
	HAKE	.497	.452	.582	.346	.250	**.542**
	ConE	.496	.453	.579	.345	.247	.540
	MuRE	.465	.436	.554	.336	.245	.521
	RefE	.473	.430	.561	.351	.256	.541
	RotE	.494	.446	.585	.346	.251	.538
	AttE	.490	.443	.581	.351	.255	.543
	MuRP	.481	.440	.566	.335	.243	.518
	RefH	.461	.404	.568	.346	.252	.536
	RotH	.496	.449	.586	.344	.246	.535
	AttH	.486	.443	.573	.348	.252	.540
Bilinear	RotatE	.476	.428	.571	.338	.241	.533
	ComplEx	.475	-	.547	.348	-	.536
	Quaternion	**.488**	.438	**.582**	.348	.248	.550
	TuckER	.470	**.443**	.526	**.358**	**.266**	**.544**

C Notation

In this section, we give a brief overview of the most important notations we use. Note that, for ease of readability and comparability, we use exactly the same language as ExpressivE (Pavlović and Sallinger, 2023).

v ... non-bold symbols represent scalars
$\boldsymbol{v}$ ... bold symbols represent vectors, sets or tuples
$\mathbf{0}$ ... represents a vector of zeros (the same semantics apply to $\mathbf{0.5}$, $\mathbf{1}$, and $\mathbf{2}$)
$\oslash$ ... represents the element-wise division operator
$\odot$ ... represents the element-wise (Hadamard) product operator
$\succeq$ ... represents the element-wise greater or equal operator

$\succ$ ... represents the element-wise greater operator
$\preceq$ ... represents the element-wise less or equal operator
$\prec$ ... represents the element-wise less operator
$\boldsymbol{x}^{|\cdot|}$ ... represents the element-wise absolute value
$||$ ... represents the concatenation operator

D Definition of Capturing

In this section, we introduce the formal semantics of SpeedE models. Note that, for ease of readability and comparability, we use exactly the same language as ExpressivE (Pavlović and Sallinger, 2023). In places where SpeedE significantly differs from ExpressivE, we will explicitly note this and compare the two. Specifically, this section introduces the notions of capturing a pattern in a SpeedE model that we informally discussed in Section 3.1. Furthermore, it introduces some additional notations, which will help us simplify the upcoming proofs and present them intuitively.

Knowledge Graph. A tuple $(\boldsymbol{G}, \boldsymbol{E}, \boldsymbol{R})$ is called a knowledge graph, where $\boldsymbol{R}$ is a finite set of relations, $\boldsymbol{E}$ is a finite set of entities, and $\boldsymbol{G} \subseteq \boldsymbol{E} \times \boldsymbol{R} \times \boldsymbol{E}$ is a finite set of triples. W.l.o.g., we assume that any relation is non-empty since removing any virtual entity pair embedding from a hyper-parallelogram would be trivial, just adding unnecessary complexity to the proofs.

SpeedE Model. We define a SpeedE model as a tuple $\boldsymbol{M^+} = (\boldsymbol{\epsilon}, \boldsymbol{\sigma}, w, \boldsymbol{\rho})$, where $\boldsymbol{\epsilon} \subset 2^{\mathbb{R}^d}$ is the set of entity embeddings, $\boldsymbol{\sigma} \subset 2^{\mathbb{R}^d}$ is the set of center embeddings, $w \in \mathbb{R}_{>0}$ represents the width constant, and $\boldsymbol{\rho} \subset 2^{\mathbb{R}^d}$ is the set of slope vectors. Note that this definition is slightly different from an ExpressivE model $\boldsymbol{M} = (\boldsymbol{\epsilon}, \boldsymbol{\sigma}, \boldsymbol{\delta}, \boldsymbol{\rho})$, where instead of the width constant w, we have $\boldsymbol{\delta} \subset 2^{\mathbb{R}^d}$ that represents the set of width embeddings.

Linking Embeddings to KGs. A SpeedE model $\boldsymbol{M^+} = (\boldsymbol{\epsilon}, \boldsymbol{\sigma}, w, \boldsymbol{\rho})$ and a KG $(\boldsymbol{G}, \boldsymbol{E}, \boldsymbol{R})$ are linked via the following assignment functions: The entity assignment function $\boldsymbol{f_e} : \boldsymbol{E} \rightarrow \boldsymbol{\epsilon}$ assigns to each entity $e_h \in \boldsymbol{E}$ an entity embedding $\boldsymbol{e_h} \in \boldsymbol{\epsilon}$. Based on $\boldsymbol{f_e}$, the virtual assignment function $\boldsymbol{f_v} : \boldsymbol{E} \times \boldsymbol{E} \rightarrow \mathbb{R}^{2d}$ defines for any pair of entities $(e_h, e_t) \in \boldsymbol{E}$ a virtual entity pair embedding $\boldsymbol{f_v}(e_h, e_t) = (\boldsymbol{f_e}(e_h)||\boldsymbol{f_e}(e_t))$, where $||$ represents the concatenation operator. Furthermore, we define SpeedE's relation assignment function $\boldsymbol{f_h^+}(r_j) : \boldsymbol{R} \rightarrow \mathbb{R}^{2d} \times \mathbb{R} \times \mathbb{R}^{2d}$ as $\boldsymbol{f_h^+}(r_j) = (\boldsymbol{c_j^{ht}}, w, \boldsymbol{s_j^{th}})$, where $\boldsymbol{c_j^{ht}} = (\boldsymbol{c_j^h}||\boldsymbol{c_j^t})$ with $\boldsymbol{c_j^h}, \boldsymbol{c_j^t} \in \boldsymbol{\sigma}$ and where $\boldsymbol{s_j^{th}} = (\boldsymbol{s_j^t}||\boldsymbol{s_j^h})$ with $\boldsymbol{s_j^t}, \boldsymbol{s_j^h} \in \boldsymbol{\rho}$. Note that this is different from ExpressivE's relation assignment function $\boldsymbol{f_h}(r_j) : \boldsymbol{R} \rightarrow \mathbb{R}^{2d} \times \mathbb{R}^{2d} \times \mathbb{R}^{2d}$, where $\boldsymbol{f_h}(r_j) = (\boldsymbol{c_j^{ht}}, \boldsymbol{w_j^{ht}}, \boldsymbol{s_j^{th}})$ with $\boldsymbol{w_j^{ht}} = (\boldsymbol{w_j^h}||\boldsymbol{w_j^t})$ being two concatenated width embeddings.

Virtual Triple Space. To be able to assign a geometric interpretation to $\boldsymbol{f_h^+}(r_j)$, we briefly recap the definition of the virtual triple space $\mathbb{R}^{2d}$ introduced by Pavlović and Sallinger (2023). Specifically, the virtual triple space is constructed by concatenating the head and tail entity embeddings. In detail, this means that any pair of entities $(e_h, e_t) \in \boldsymbol{E} \times \boldsymbol{E}$ defines a point in the virtual triple space by concatenating their entity embeddings $\boldsymbol{e_h}, \boldsymbol{e_t} \in \mathbb{R}^d$, i.e., $(\boldsymbol{e_h}||\boldsymbol{e_t}) \in \mathbb{R}^{2d}$. We will henceforth call the first d dimensions of the virtual triple space the *head dimensions* and the second d dimensions the *tail dimensions*. A set of important sub-spaces of the virtual triple space are the 2-dimensional spaces created from the k-th dimension of the head and tail dimensions — i.e., the k-th and $(d+k)$-th virtual triple space dimensions. We call them *correlation subspaces* as they visualize the captured relation-specific dependencies of head and tail entity embeddings. Moreover, we call the correlation subspace spanned by the k-th and $(d+k)$-th virtual triple space dimension the k-th correlation subspace. Now, the geometric interpretation of $\boldsymbol{f_h^+}(r_j)$ within the virtual triple space is a hyper-parallelogram whose edges are solely crooked in each correlation subspace, representing the relationship between head and tail entity embeddings.

Model Configuration. We call a SpeedE model $\boldsymbol{M^+}$ together with a concrete relation assignment function $\boldsymbol{f_h^+}$ a relation configuration $\boldsymbol{m_h^+} = (\boldsymbol{M^+}, \boldsymbol{f_h^+})$. If $\boldsymbol{m_h^+}$ additionally has a virtual assignment function $\boldsymbol{f_v}$, we call it a complete model configuration $\boldsymbol{m^+} = (\boldsymbol{M^+}, \boldsymbol{f_h^+}, \boldsymbol{f_v})$. Note that an ExpressivE relation configuration $\boldsymbol{m_h} = (\boldsymbol{M}, \boldsymbol{f_h})$ and a complete ExpressivE model configuration $\boldsymbol{m} = (\boldsymbol{M}, \boldsymbol{f_h}, \boldsymbol{f_v})$ are defined differently by replacing $\boldsymbol{M^+}$ and $\boldsymbol{f_h^+}$ with their ExpressivE equivalents, i.e., $\boldsymbol{M}$ and $\boldsymbol{f_h}$.

Definition of Truth. A triple $r_j(e_h, e_t)$ is captured to be true in some $\boldsymbol{m}^+$, with $r_j \in \boldsymbol{R}$ and $e_h, e_t \in \boldsymbol{E}$ iff Inequality 4 holds for the assigned embeddings of h, t, and r. This means more precisely that Inequality 4 needs to hold for $\boldsymbol{f_v}(e_h, e_t) = (\boldsymbol{f_e}(e_h)||\boldsymbol{f_e}(e_t)) = (\boldsymbol{e_h}, \boldsymbol{e_t})$ and $\boldsymbol{f_h^+}(r_j) = (\boldsymbol{c_j^{ht}}, w, \boldsymbol{s_j^{th}})$. Note that, for ExpressivE, the definition of a triple's truth is slightly different, as w in Inequality 4 would be exchanged by the respective width embedding $\boldsymbol{w_j^{ht}}$.

$$(\boldsymbol{e_{ht}} - \boldsymbol{c_j^{ht}} - \boldsymbol{s_j^{th}} \odot \boldsymbol{e_{th}})^{|.|} \preceq w, \tag{4}$$

Intuition. At an intuitive level, a triple $r_j(e_h, e_t)$ is captured to be true by some complete SpeedE model configuration $\boldsymbol{m}^+$ iff the virtual pair embedding $\boldsymbol{f_v}(e_h, e_t)$ of entities e_h and e_t lies within the hyper-parallelogram of relation r_j defined by $\boldsymbol{f_h^+}(r_j)$.

Simplifying Notations. Therefore, to simplify the upcoming proofs, we denote with $\boldsymbol{f_v}(e_h, e_t) \in \boldsymbol{f_h^+}(r_j)$ that the virtual pair embedding $\boldsymbol{f_v}(e_h, e_t)$ of an entity pair $(e_h, e_t) \in \boldsymbol{E} \times \boldsymbol{E}$ lies within the hyper-parallelogram $\boldsymbol{f_h^+}(r_j)$ of some relation $r_j \in \boldsymbol{R}$ in the virtual triple space. Accordingly, for sets of virtual pair embeddings $\boldsymbol{P} := \{\boldsymbol{f_v}(e_{h_1}, e_{t_1}), \dots, \boldsymbol{f_v}(e_{h_n}, e_{t_n})\}$, we denote with $\boldsymbol{P} \subseteq \boldsymbol{f_h^+}(r_j)$ that all virtual pair embeddings of $\boldsymbol{P}$ lie within the hyper-parallelogram of the relation r_j. Furthermore, we denote with $\boldsymbol{f_v}(e_h, e_t) \notin \boldsymbol{f_h^+}(r_j)$ that a virtual pair embedding $\boldsymbol{f_v}(e_h, e_t)$ does not lie within the hyper-parallelogram of a relation r_j and with $\boldsymbol{P} \not\subseteq \boldsymbol{f_h^+}(r_j)$ we denote that an entire set of virtual pair embeddings $\boldsymbol{P}$ does not lie within the hyper-parallelogram of a relation r_j.

Capturing Inference Patterns. Based on the previous definitions, we define capturing patterns formally: A relation configuration $\boldsymbol{m_h^+}$ captures a pattern ψ *exactly* if for any ground pattern $\phi_{B_1} \wedge \dots \wedge \phi_{B_m} \Rightarrow \phi_H$ within the deductive closure of ψ and for any instantiation of $\boldsymbol{f_e}$ and $\boldsymbol{f_v}$ the following conditions are satisfied:

- if ϕ_H is a triple and if $\boldsymbol{m_h^+}$ captures the body triples to be true — i.e., $\boldsymbol{f_v}(args(\phi_{B_1})) \in \boldsymbol{f_h^+}(rel(\phi_{B_1})), \dots, \boldsymbol{f_v}(args(\phi_{B_m})) \in \boldsymbol{f_h^+}(rel(\phi_{B_m}))$ — then $\boldsymbol{m_h^+}$ also captures the head triple to be true — i.e., $\boldsymbol{f_v}(args(\phi_H)) \in \boldsymbol{f_h^+}(rel(\phi_H))$.
- if $\phi_H = \bot$, then $\boldsymbol{m_h^+}$ captures at least one of the body triples to be false — i.e., there is some $j \in \{1, \dots, m\}$ such that $\boldsymbol{f_v}(args(\phi_{B_j})) \notin \boldsymbol{f_h^+}(rel(\phi_{B_j}))$.

where *args*() is the function that returns the arguments of a triple, and *rel*() is the function that returns the relation of the triple. Furthermore, a relation configuration $\boldsymbol{m_h^+}$ captures a pattern ψ *exactly* and *exclusively* if (1) $\boldsymbol{m_h^+}$ exactly captures ψ and (2) $\boldsymbol{m_h^+}$ does not capture any *positive* pattern ϕ (i.e., $\phi \in \{symmetry,\ inversion,\ hierarchy,\ intersection,\ composition\}$) such that $\psi \not\models \phi$ except where the body of ϕ is not satisfied over $\boldsymbol{m_h^+}$.

Discussion. The following provides some intuition of the above definition of capturing a pattern. Capturing a pattern *exactly* is defined straightforwardly by adhering to the semantics of logical implication $\phi := \phi_B \Rightarrow \phi_H$, i.e., a relation configuration $\boldsymbol{m_h^+}$ needs to be found such that for any complete model configuration $\boldsymbol{m}^+$ over $\boldsymbol{m_h^+}$ if the body ϕ_B of the pattern is satisfied, then its head ϕ_H can be inferred.

Capturing a pattern *exactly* and *exclusively* imposes additional constraints. Here, the aim is not solely to capture a pattern but additionally to showcase that a pattern can be captured independently from any other pattern. Therefore, some notion of minimality/exclusiveness of a pattern is needed. As in Abboud et al. (2020); Pavlović and Sallinger (2023), we define minimality by means of *solely* capturing those positive patterns ϕ that directly follow from the deductive closure of the pattern ψ, except for those ϕ that are captured trivially, i.e., except for those ϕ where their body is not satisfied over the constructed $\boldsymbol{m_h^+}$.

The authors of (Pavlović and Sallinger, 2023) have shown that any core inference patterns (given in Section 3.1) can be expressed by means of spatial relations of the corresponding relation hyper-parallelograms in the virtual triple space. Therefore, *exclusiveness* is formulated intuitively as the ability to limit the intersection of hyper-parallelograms to only those intersections that directly follow from the captured pattern ψ for any known relation $r_j \in \boldsymbol{R}$, which is in accordance with the notion of exclusiveness of the literature (Abboud et al., 2020; Pavlović and Sallinger, 2023).

Note that the definition of capturing patterns solely depends on relation configurations. This is vital for SpeedE to capture patterns in a *lifted* manner, i.e., SpeedE shall be able to capture patterns without grounding them first. Furthermore, being able to capture patterns in a lifted way is not only efficient but also natural, as the aim is to capture patterns between relations. Thus, it would be unnatural if constraints on entity embeddings were necessary to capture such relation-specific patterns.

As outlined in the previous paragraphs, the definition of capturing patterns is in accordance with the literature (Abboud et al., 2020; Pavlović and Sallinger, 2023), focuses on efficiently capturing patterns, and gives us a formal foundation for the upcoming proofs, which will show that SpeedE can capture the core inference patterns.

E Proof of Theorem 4.1

In Section 3.1, we have already briefly introduced inference patterns. To prove that SpeedE captures the core inference patterns exactly and exclusively (Theorem 4.1), let us now first recall the full, formal definition of these patterns.

Definition E.1. *(Abboud et al., 2020; Pavlović and Sallinger, 2023) Let the inference patterns be defined as follows:*

- *Patterns of the form* $r_1(X,Y) \Rightarrow r_1(Y,X)$ *with* $r_1 \in \boldsymbol{R}$ *are called* symmetry patterns.
- *Patterns of the form* $r_1(X,Y) \Rightarrow \neg r_1(Y,X)$ *with* $r_1 \in \boldsymbol{R}$ *are called* anti-symmetry patterns.
- *Patterns of the form* $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ *with* $r_1, r_2 \in \boldsymbol{R}$ *and* $r_1 \neq r_2$ *are called* inversion patterns.
- *Patterns of the form* $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$ *with* $r_1, r_2, r_3 \in \boldsymbol{R}$ *and* $r_1 \neq r_2 \neq r_3$ *are called (general)* composition patterns.
- *Patterns of the form* $r_1(X,Y) \Rightarrow r_2(X,Y)$ *with* $r_1, r_2 \in \boldsymbol{R}$ *and* $r_1 \neq r_2$ *are called* hierarchy patterns.
- *Patterns of the form* $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ *with* $r_1, r_2, r_3 \in \boldsymbol{R}$ *and* $r_1 \neq r_2 \neq r_3$ *are called* intersection patterns.
- *Patterns of the form* $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \bot$ *with* $r_1, r_2 \in \boldsymbol{R}$ *and* $r_1 \neq r_2$ *are called* mutual exclusion patterns.

Based on these definitions, we will prove that SpeedE captures the core inference patterns exactly and exclusively, thereby proving Theorem 4.1. To prove Theorem 4.1, we give the relevant propositions obtained from and proved by Pavlović and Sallinger (2023) and adapt them to SpeedE. For each of them, we give proofs, which in some situations follow from the ones in Pavlović and Sallinger (2023), and in other situations are entirely new constructions.

The key change of SpeedE that will be of our concern in the following proofs is fixing the width to a constant value, as this will require new proofs for some of the properties. Observe that SpeedE additionally changes the distance function of ExpressivE. However, this does not affect ExpressivE's inference capabilities, i.e., which inference patterns can be captured. Careful inspection of the proofs of inference capabilities given in (Pavlović and Sallinger, 2023) shows that the only property required of the distance function is that scores within the hyper-parallelogram are larger than those outside. As the newly defined distance function of SpeedE keeps this property, the change of distance function between the two models does not affect the proofs of the inference capabilities given in (Pavlović and Sallinger, 2023). Hence, the same proof argument can be applied.

The other observation that we will make in general before giving the specific proofs is that the "exactly" part, proved in (Pavlović and Sallinger (2023), Propositions F.1-F.7), of "exactly and exclusively" capturing patterns is not affected by the changes in the model. These proofs are all based on embedding pairs of entities as points in the virtual triple space and relations as hyper-parallelograms, which is still the case in SpeedE. Thus, we now proceed to proving that SpeedE captures the core inference patterns exactly and exclusively.

Proposition E.2 (Inversion (Exactly and Exclusively)). *Let $\boldsymbol{m_h^+} = (\boldsymbol{M^+}, \boldsymbol{f_h^+})$ be a relation configuration and $r_1, r_2 \in \boldsymbol{R}$ be relations where $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ holds for any entities $X, Y \in \boldsymbol{E}$. Then $\boldsymbol{m_h^+}$ can capture $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ exactly and exclusively.*

Proof. The proof of this property in Expressive (Pavlović and Sallinger (2023), Proposition G.3) is based on a key assumption, namely that there is an $\boldsymbol{m_h}$ such that $\boldsymbol{f_h(r_1)}$ is the mirror image of $\boldsymbol{f_h(r_2)}$ with $\boldsymbol{f_h(r_1)} \neq \boldsymbol{f_h(r_2)}$. This is straightforward in ExpressivE but more complex in SpeedE. We will show this next.

Let us first observe that in SpeedE, it is not trivially given that there is an $\boldsymbol{m_h^+} = (\boldsymbol{M^+}, \boldsymbol{f_h^+})$ such that $\boldsymbol{f_h^+(r_1)}$ is the mirror image of $\boldsymbol{f_h^+(r_2)}$ with $\boldsymbol{f_h^+(r_1)} \neq \boldsymbol{f_h^+(r_2)}$, as $\boldsymbol{f_h(r_j)}$'s width embedding $\boldsymbol{w_j^{ht}}$ has been replaced by a shared width constant w in $\boldsymbol{f_h^+(r_j)}$ with $j \in \{1,2\}$. Thus, what needs to be shown is that there is a relation configuration $\boldsymbol{m_h^+}$ such that $\boldsymbol{f_h^+(r_1)}$ is the mirror image of $\boldsymbol{f_h^+(r_2)}$ with $\boldsymbol{f_h^+(r_1)} \neq \boldsymbol{f_h^+(r_2)}$, as then the original proof of ExpressivE can be directly applied to prove Proposition E.2's claim, i.e., that $\boldsymbol{m_h^+}$ can capture $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ exactly and exclusively. Now, it is interesting to see that fixing the width parameter in SpeedE as opposed to ExpressivE not only changes the model but actually allows a quite elegant construction witnessing this property.

Let us now give this construction, thereby showing the claim. Specifically, let $\boldsymbol{f_h^+(r_1)} = (\boldsymbol{c_1^{ht}}, w, \boldsymbol{s_1^{th}})$ with $\boldsymbol{c_1^{ht}} = (\boldsymbol{c_1^h} || \boldsymbol{c_1^t}) \in \mathbb{R}^{2d}$, $w \in \mathbb{R}_{>0}$, and $\boldsymbol{s_1^{th}} = (\boldsymbol{s_1^t} || \boldsymbol{s_1^h}) \in \mathbb{R}^{2d}$. Furthermore, let $\boldsymbol{f_h^+(r_2)} = (\boldsymbol{c_2^{ht}}, w, \boldsymbol{s_2^{th}})$ with $\boldsymbol{c_2^{ht}} = (\boldsymbol{c_1^t} || \boldsymbol{c_1^h}) \in \mathbb{R}^{2d}$, $w \in \mathbb{R}_{>0}$, and $\boldsymbol{s_2^{th}} = (\boldsymbol{s_1^h} || \boldsymbol{s_1^t}) \in \mathbb{R}^{2d}$. We will, in the following, show that the constructed $\boldsymbol{f_h(r_2)}$ is the mirror image of $\boldsymbol{f_h(r_1)}$ to prove our claim. Let $X, Y \in \boldsymbol{E}$ be arbitrary entities and let $\boldsymbol{f_v}$ be an arbitrary virtual assignment function defined over (X,Y) and (Y,X) with $\boldsymbol{f_v}(X,Y) = \boldsymbol{e_{xy}}$ and $\boldsymbol{f_v}(Y,X) = \boldsymbol{e_{yx}}$. Then by Inequality 4, a triple $r_1(X,Y)$ is captured to be true by $\boldsymbol{m^+} = (\boldsymbol{M^+}, \boldsymbol{f_h^+}, \boldsymbol{f_v})$ if Inequality 5 is satisfied.

$$(\boldsymbol{e_{xy}} - \boldsymbol{c_1^{ht}} - \boldsymbol{s_1^{th}} \odot \boldsymbol{e_{yx}})^{|.|} \preceq w \tag{5}$$

$$(\boldsymbol{e_{yx}} - \boldsymbol{c_1^{th}} - \boldsymbol{s_1^{ht}} \odot \boldsymbol{e_{xy}})^{|.|} \preceq w \tag{6}$$

$$(\boldsymbol{e_{yx}} - \boldsymbol{c_2^{ht}} - \boldsymbol{s_2^{th}} \odot \boldsymbol{e_{xy}})^{|.|} \preceq w \tag{7}$$

Since Inequality 5 is element-wise, one can equivalently reformulate it by arbitrarily exchanging its dimensions. Using this insight, we can replace the head and tail dimensions for each embedding, thereby obtaining Inequality 6. Finally, by our construction of $\boldsymbol{f_h^+}(r_2)$, we have that $\boldsymbol{c_2^{ht}} = \boldsymbol{c_1^{th}}$ and $\boldsymbol{s_2^{th}} = \boldsymbol{s_1^{ht}}$. We substitute these equations into Inequality 6, thereby obtaining Inequality 7. Now, Inequality 7 states by the definition of a triple's truth (i.e., Inequality 4) that $r_2(Y,X)$ is captured by m_h^+. Since Inequalities 5-7 are all equivalent, we have shown that $\boldsymbol{f_h^+}(r_1)$ is the mirror image of $\boldsymbol{f_h^+}(r_2)$. Since, it is now easy to see that an $\boldsymbol{m_h^+}$ exists such that $\boldsymbol{f_h^+(r_1)}$ is the mirror image of $\boldsymbol{f_h^+(r_2)}$ with $\boldsymbol{f_h^+(r_1)} \neq \boldsymbol{f_h^+(r_2)}$, the proof of (Pavlović and Sallinger (2023), Proposition G.4) can be directly applied to SpeedE. Thus, we have proven Proposition E.2, i.e., that $\boldsymbol{m_h^+}$ can capture $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ exactly and exclusively. □

Table 7: Relation embeddings of a relation configuration $\boldsymbol{m}_h^+$ that captures hierarchy (i.e., $r_1(X,Y) \Rightarrow r_2(X,Y)$) exactly and exclusively using width $w = 1$.

	$\boldsymbol{c}^h$	$\boldsymbol{s}^t$	$\boldsymbol{c}^t$	$\boldsymbol{s}^h$
r_1	-2.5	0.5	1.5	0
r_2	1	-2	4.5	2

Proposition E.3 (Hierarchy (Exactly and Exclusively)). *Let $\boldsymbol{m}_h^+ = (\boldsymbol{M}^+, \boldsymbol{f}_h^+)$ be a relation configuration and $r_1, r_2 \in \boldsymbol{R}$ be relations where $r_1(X,Y) \Rightarrow r_2(X,Y)$ holds for any entities $X, Y \in \boldsymbol{E}$. Then $\boldsymbol{m}_h^+$ can capture $r_1(X,Y) \Rightarrow r_2(X,Y)$ exactly and exclusively.*

Proof. The proof of this property in Expressive (Pavlović and Sallinger (2023), Proposition G.4) is based on a key assumption, namely that there is an $\boldsymbol{m_h}$ such that $\boldsymbol{f_h}(\boldsymbol{r_1}) \subset \boldsymbol{f_h}(\boldsymbol{r_2})$ with $\boldsymbol{f_h}(\boldsymbol{r_1}) \neq \boldsymbol{f_h}(\boldsymbol{r_2})$. This is straightforward in ExpressivE but much more complex in SpeedE. We will show this next.

Let us first observe that in SpeedE, it is not trivially given that there is an $\boldsymbol{m}_h^+ = (\boldsymbol{M}^+, \boldsymbol{f}_h^+)$ such that $\boldsymbol{f}_h^+(\boldsymbol{r_1}) \subset \boldsymbol{f}_h^+(\boldsymbol{r_2})$ with $\boldsymbol{f}_h^+(\boldsymbol{r_1}) \neq \boldsymbol{f}_h^+(\boldsymbol{r_2})$, as $\boldsymbol{f_h}(\boldsymbol{r_j})$'s width embedding $\boldsymbol{w}_j^{ht}$ has been replaced by a shared width constant w in $\boldsymbol{f}_h^+(\boldsymbol{r_j})$ with $j \in \{1, 2\}$. Thus, what needs to be shown is that there is a relation configuration $\boldsymbol{m}_h^+$ such that $\boldsymbol{f}_h^+(\boldsymbol{r_1}) \subset \boldsymbol{f}_h^+(\boldsymbol{r_2})$ with $\boldsymbol{f}_h^+(\boldsymbol{r_1}) \neq \boldsymbol{f}_h^+(\boldsymbol{r_2})$, as then the original proof of ExpressivE can be directly applied to prove Proposition E.3's claim, i.e., that $\boldsymbol{m}_h^+$ can capture $r_1(X,Y) \Rightarrow r_2(X,Y)$ exactly and exclusively. In the following, we construct such a relation configuration $\boldsymbol{m}_h^+ = (\boldsymbol{M}^+, \boldsymbol{f}_h^+)$, where $\boldsymbol{f}_h^+(\boldsymbol{r_1}) \subset \boldsymbol{f}_h^+(\boldsymbol{r_2})$ with $\boldsymbol{f}_h^+(\boldsymbol{r_1}) \neq \boldsymbol{f}_h^+(\boldsymbol{r_2})$ to prove the claim of Proposition E.3:

Figure 1 (given on Page 5 of the main body) visualizes the relation configuration $\boldsymbol{m}_h^+ = (\boldsymbol{M}^+, \boldsymbol{f}_h^+)$ provided in Table 7. As can be easily seen in Figure 1, $\boldsymbol{m}_h^+$ captures $\boldsymbol{f}_h^+(\boldsymbol{r_1}) \subset \boldsymbol{f}_h^+(\boldsymbol{r_2})$ with $\boldsymbol{f}_h^+(\boldsymbol{r_1}) \neq \boldsymbol{f}_h^+(\boldsymbol{r_2})$. Thus, we have proven Proposition E.3, as (1) we have shown the existence of an $\boldsymbol{m}_h^+$ that captures $\boldsymbol{f}_h^+(\boldsymbol{r_1}) \subset \boldsymbol{f}_h^+(\boldsymbol{r_2})$ with $\boldsymbol{f}_h^+(\boldsymbol{r_1}) \neq \boldsymbol{f}_h^+(\boldsymbol{r_2})$ and (2) the proof of (Pavlović and Sallinger (2023), Proposition G.4) can be directly applied to SpeedE since an $\boldsymbol{m}_h^+$ exists such that $\boldsymbol{f}_h^+(\boldsymbol{r_1}) \subset \boldsymbol{f}_h^+(\boldsymbol{r_2})$ with $\boldsymbol{f}_h^+(\boldsymbol{r_1}) \neq \boldsymbol{f}_h^+(\boldsymbol{r_2})$. □

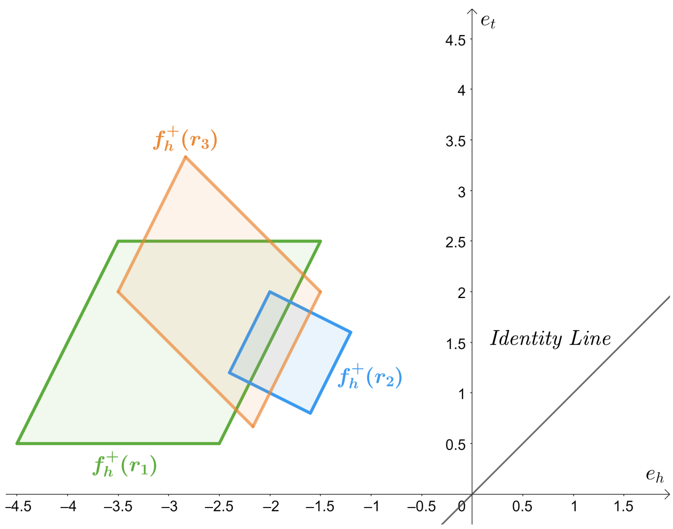

Figure 4: Relation embeddings of a relation configuration $\boldsymbol{m_h}$ that captures intersection (i.e., $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$) exactly and exclusively using width $w = 1$.

Table 8: Relation embeddings of a relation configuration $\boldsymbol{m}_h^+$ that captures intersection (i.e., $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$) exactly and exclusively using width $w = 1$.

	$\boldsymbol{c^h}$	$\boldsymbol{s^t}$	$\boldsymbol{c^t}$	$\boldsymbol{s^h}$
r_1	-3.75	0.5	1	0
r_2	1	-2	5	2
r_3	-3.5	0.5	0.5	-1

Proposition E.4 (Intersection (Exactly and Exclusively)). *Let $\boldsymbol{m}_h^+ = (\boldsymbol{M}^+, \boldsymbol{f}_h^+)$ be a relation configuration and $r_1, r_2, r_3 \in \boldsymbol{R}$ be relations where $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ holds for any entities $X, Y \in \boldsymbol{E}$. Then $\boldsymbol{m}_h^+$ can capture $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ exactly and exclusively.*

Proof Sketch. This is similar in construction to the previous proof. Hence, we only give a proof sketch for ease of readability. To prove Proposition E.4, observe that in (Pavlović and Sallinger (2023), Proposition G.5) an ExpressivE relation configuration m_h with several different width embeddings is constructed. However, the key observation we will make is that choosing the width embeddings differently is not necessary. In fact, an interested reader inspecting the original proof can obtain a proof applicable to SpeedE by following the proof of (Pavlović and Sallinger (2023), Proposition G.5) analogously for the SpeedE relation configuration m_h^+ described in Table 8 and visualized by Figure 4. Thus, the proof for Proposition E.4 is straightforward given m_h^+ defined in Table 8 and (Pavlović and Sallinger (2023), Proposition G.5). □

Table 9: Relation embeddings of a relation configuration $\boldsymbol{m}_h^+$ that captures composition (i.e., $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$) exactly and exclusively using width $w = 1$.

	$\boldsymbol{c^h}$	$\boldsymbol{s^t}$	$\boldsymbol{c^t}$	$\boldsymbol{s^h}$
r_1	-7	3	5	1
r_2	-7.5	1	2	3
r_3	-19.5	2	13	2

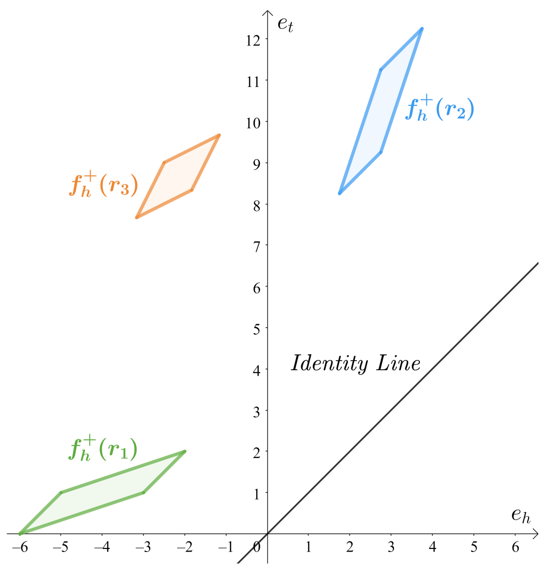

Figure 5: Relation embeddings of a relation configuration $\boldsymbol{m}_h$ that captures composition (i.e., $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$) exactly and exclusively using width $w = 1$.

Proposition E.5 (**Composition (Exactly and Exclusively)**)**.** *Let $r_1, r_2, r_3 \in \boldsymbol{R}$ be relations and let $\boldsymbol{m_h^+} = (\boldsymbol{M}^+, \boldsymbol{f_h^+})$ be a relation configuration, where $\boldsymbol{f_h^+}$ is defined over r_1, r_2, and r_3. Furthermore let r_3 be the composite relation of r_1 and r_2, i.e., $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$ holds for all entities $X, Y, Z \in \boldsymbol{E}$. Then $\boldsymbol{m_h^+}$ can capture $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$ exactly and exclusively.*

Proof Sketch. This is similar in construction to the proof of Proposition E.3. Hence, we only give a proof sketch for ease of readability. To prove Proposition E.5, observe that in (Pavlović and Sallinger (2023), Proposition G.6), an ExpressivE relation configuration m_h with several different width embeddings is constructed. However, choosing the width embeddings differently is not necessary. In fact, an interested reader inspecting the original proof can obtain a proof applicable to SpeedE by following the proof of (Pavlović and Sallinger (2023), Proposition G.6) analogously for the SpeedE relation configuration m_h^+ described in Table 9 and visualized by Figure 5. Thus, the proof for Proposition E.5 is straightforward given m_h^+ defined in Table 9 and (Pavlović and Sallinger (2023), Proposition G.6). □

Proposition E.6 (**Symmetry (Exactly and Exclusively)**)**.** *Let $\boldsymbol{m_h^+} = (\boldsymbol{M}^+, \boldsymbol{f_h^+})$ be a relation configuration and $r_1 \in \boldsymbol{R}$ be a symmetric relation, i.e., $r_1(X,Y) \Rightarrow r_1(Y,X)$ holds for any entities $X, Y \in \boldsymbol{E}$. Then $\boldsymbol{m_h^+}$ can capture $r_1(X,Y) \Rightarrow r_1(Y,X)$ exactly and exclusively.*

Proposition E.7 (**Anti-Symmetry (Exactly and Exclusively)**)**.** *Let $\boldsymbol{m_h^+} = (\boldsymbol{M}^+, \boldsymbol{f_h^+})$ be a relation configuration and $r_1 \in \boldsymbol{R}$ be an anti-symmetric relation, i.e., $r_1(X,Y) \Rightarrow \neg r_1(Y,X)$ holds for any entities $X, Y \in \boldsymbol{E}$. Then $\boldsymbol{m_h^+}$ can capture $r_1(X,Y) \Rightarrow \neg r_1(Y,X)$ exactly and exclusively.*

The proofs for Proposition E.6-E.7 are straightforward and work analogously to the proofs of (Pavlović and Sallinger (2023), Proposition G.1-G.2). This is the case, as (1) any of these patterns contain at most one relation, (2) thus we solely need to show that no unwanted patterns over at most one relation are captured, as any considered pattern over more than one relation (precisely inversion, hierarchy, intersection, and composition) requires by Definition E.1 at least two or three *distinct* relations and thus is not applicable, and (3) it is easy to see that, for instance, a relation hyper-parallelogram can be symmetric without being anti-symmetric, or vice versa (i.e., without capturing any unwanted pattern).

Proposition E.8 (**Mutual Exclusion (Exactly and Exclusively)**)**.** *Let $\boldsymbol{m_h^+} = (\boldsymbol{M}^+, \boldsymbol{f_h^+})$ be a relation configuration and $r_1, r_2 \in \boldsymbol{R}$ be mutually exclusive relations, i.e., $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ holds for any entities $X, Y \in \boldsymbol{E}$. Then $\boldsymbol{m_h^+}$ can capture $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ exactly and exclusively.*

The proof for Proposition E.8 is trivial, as it is straight-forward to see that (1) there is an $\boldsymbol{m_h^+} = (\boldsymbol{M}^+, \boldsymbol{f_h^+})$ such that $\boldsymbol{f_h^+}(r_1) \cap \boldsymbol{f_h^+}(r_2) = \emptyset$, thereby $\boldsymbol{m_h^+}$ captures $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ exactly, (2) neither $\boldsymbol{f_h^+}(r_1)$ nor $\boldsymbol{f_h^+}(r_2)$ need to be symmetric, thereby no unwanted symmetry pattern is captured, (3) $\boldsymbol{f_h^+}(r_1)$ does not need to be the mirror image of $\boldsymbol{f_h^+}(r_2)$, thus no unwanted inversion pattern is captured, and finally (4) since $\boldsymbol{f_h^+}(r_1)$ and $\boldsymbol{f_h^+}(r_2)$ are disjoint, neither $\boldsymbol{f_h^+}(r_1)$ can subsume $\boldsymbol{f_h^+}(r_2)$ nor vice versa, thus no unwanted hierarchy pattern is captured. Thus by Points 1-4, we have shown that $\boldsymbol{m_h^+}$ captures $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ exactly and that it does not capture any unwanted positive pattern that is applicable, i.e., requires at most two different relations (symmetry, inversion, and hierarchy). Thus, we have shown Proposition E.8, i.e., that $\boldsymbol{m_h^+}$ can capture $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ exactly and exclusively.

Finally, by Propositions E.2-E.8, we have shown Theorem 4.1, i.e., that SpeedE captures the core inference patterns exactly and exclusively.

F Experimental Details

The details of our experiment's setup, benchmarks, and evaluation protocol are covered in this section. Specifically, details on SpeedE's implementation and about reproducing our results are covered in Section F.1. Each benchmark's properties are discussed in Section F.2. Our experimental setup is described in Section F.3, including details about the chosen learning setup, hardware, and hyperparameters. The evaluation protocol and the used metrics are discussed in Section F.4. Finally, the size of CO2 emissions resulting from our experiments is estimated in Section F.5.

F.1 Implementation Details & Reproducibility

Following Pavlović and Sallinger (2023), we have implemented our gKGE using PyKEEN 1.7 (Ali et al., 2021), a Python library that runs under the MIT license and offers support for numerous benchmarks and gKGEs. In doing so, we facilitate the comfortable reuse of SpeedE for upcoming benchmarks and experiments. To ease reproducing our findings, we have included our code in the supplementary material, and, in addition, we have included a ReadMe.md file stating library dependencies, running instructions, and a link to pre-trained SpeedE models. Upon our paper's acceptance, we will make SpeedE's source code available in a public GitHub repository.

F.2 Benchmarks and Licenses

The details of the two standard benchmarks, WN18RR (Dettmers et al., 2018) and FB15k-237 (Toutanova and Chen, 2015), used in our experiments are discussed in this section. Specifically, Table 2 (given on Page 6 of the main body) has already stated important characteristics of the benchmarks, including their number of entities, relations, and metrics describing how hierarchical the relations within the benchmark are. WN18RR and FB15k-237 already provide a split into a training, validation, and testing set, which we directly adopted in any reported experiments. Table 10 lists characteristics of these splits, specifically the number of training, validation, and testing triples. Furthermore, the table lists the number of entities and relations of each benchmark. Finally, concerning licensing, we did not find a license for WN18RR nor its superset WN18 (Bordes et al., 2013). Also, we did not find a license for FB15k-237, but we found that its superset FB15k (Bordes et al., 2013) uses the CC BY 2.5 license.

Table 10: Benchmark split characteristics: Number of entities, relations, and training, validation, and testing triples.

Dataset	$\|\boldsymbol{E}\|$	$\|\boldsymbol{R}\|$	#training triples	#validation triples	#testing triples
FB15k-237	14,541	237	272,115	17,535	20,466
WN18RR	40,943	11	86,835	3,034	3,134

F.3 Training Setup

Training Details. We have trained each model on one of four GeForce RTX 2080 Ti GPUs of our internal cluster. In particular, during the training phase, we optimize the self-adversarial negative sampling loss (Sun et al., 2019) using the Adam optimizer (Kingma and Ba, 2015). We use gradient descent to optimize SpeedE's parameters, stopping the training after 1000 epochs early if the H@10 score did not rise by at least 0.5% for WN18RR and 1% for FB15k-237. Any experiment was run three times to average over light performance variations. We will discuss the optimization of hyperparameters in the following paragraph.

Hyperparameter Optimization. Following similar optimization principles as Balazevic et al. (2019a); Chami et al. (2020); Pavlović and Sallinger (2023), we manually tuned the following hyperparameters within the listed ranges: (1) the learning rate $\lambda \in \{b * 10^{-c} \mid b \in \{1, 2, 5\} \wedge c \in \{2, 3, 4, 5, 6\}\}$, (2) the negative sample size $n \in \{100, 150, 200, 250\}$, (3) the loss margin $\gamma \in \{2, 3, 4, 5, 6\}$, (4) the adversarial temperature $\alpha \in \{1, 2, 3, 4\}$, (5) the batch size $b \in \{100, 250, 500, 1000\}$, and (6) constraining the distance slope parameters to be equal — i.e., $s^i_j = s^o_j$ for each relation $r_j \in \boldsymbol{R}$ — or not $EqDS \in \{true, false\}$. In accordance with Pavlović and Sallinger (2023), we chose self-adversarial negative sampling (Sun et al., 2019) for generating negative triples. We list the best hyperparameters for SpeedE split by benchmark and embedding dimensionality in Table 11. Following Chami et al. (2020), we used one parameter set for any low-dimensional experiment (i.e., $d \leq 50$) and one parameter set for any high-dimensional experiment (i.e., $d > 50$). Furthermore, for ExpressivE, we used the hyperparameters of Pavlović and Sallinger (2023) under high-dimensional conditions, as they report the best-published results for ExpressivE. For low-dimensional conditions, ExpressivE's best hyperparameter setting was unknown. Thus, we optimized ExpressivE's hyperparameters manually, finding the hyperparameters of Table 12 to produce the best KGC results for ExpressivE under low dimensionalities. For RotH, we used the hyperparameters of Chami et al. (2020), as they report the best-published results for RotH. Finally, we used the same hyperparameters for each of SpeedE's model variants to directly compare SpeedE to them, i.e., Min_SpeedE, Diff_SpeedE, and Eq_SpeedE.

Table 11: Hyperparameters of SpeedE models that achieve the best performance on WN18RR and FB15k-237 split by low-dimensional (i.e., $d \leq 50$) and high-dimensional setting (i.e., $d > 50$).

Dataset	Embedding Dimensionality	Margin	Learning Rate	Adversarial Temperature	Negative Sample Size	Batch Size	EqDS
WN18RR	$d \leq 50$	3	$5 * 10^{-3}$	2	200	250	false
WN18RR	$d > 50$	3	$1 * 10^{-3}$	2	200	250	true
FB15k-237	$d \leq 50$	2	$5 * 10^{-4}$	4	250	100	false
FB15k-237	$d > 50$	4	$1 * 10^{-4}$	4	150	1000	false

Table 12: Hyperparameters of ExpressivE that achieve the best performance on WN18RR and FB15k-237 under low-dimensional conditions (i.e., $d \leq 50$).

Dataset	Embedding Dimensionality	Margin	Learning Rate	Adversarial Temperature	Negative Sample Size	Batch Size
WN18RR	$d \leq 50$	2	$5 * 10^{-3}$	3	200	250
FB15k-237	$d \leq 50$	2	$5 * 10^{-4}$	4	250	100

F.4 Evaluation Protocol

Following the standard KGC evaluation protocol as described by Sun et al. (2019); Balazevic et al. (2019b); Chami et al. (2020); Pavlović and Sallinger (2023), we have evaluated ExpressivE by measuring the ranking quality of each test set triple $r_i(e_h, e_t)$ over all possible heads e'_h and tails e'_t: $r_i(e'_h, e_t)$ for all $e'_h \in \boldsymbol{E}$ and $r_i(e_h, e'_t)$ for all $e'_t \in \boldsymbol{E}$. The typical metrics for evaluating the KGC performance are the mean reciprocal rank (MRR) and Hits@k (Bordes et al., 2013). In particular, we have presented the filtered metrics (Bordes et al., 2013), i.e., all triples occurring in the training, validation, and testing set are deleted from the ranking (apart from the test triple that must be ranked), as scoring these triples highly does not indicate a wrong inference. The most used metrics for assessing gKGEs are the filtered MRR, Hits@1, and Hits@10 (Sun et al., 2019; Trouillon et al., 2016; Balazevic et al., 2019b; Abboud et al., 2020). Finally, we will briefly review how these metrics are defined: The proportion of true triples among the predicted triples whose rank is at maximum k is represented by Hits@k, whereas the MRR reflects the average of inverse ranks ($1/rank$).

F.5 CO2 Emissions

The sum of all reported experiments took less than 150 GPU hours. This corresponds to an estimate of approximately $16.20kg$ CO_2-eq, based on the OECD's 2014 carbon efficiency average of $0.432kg/kWh$ and the usage of an RTX 2080 Ti on private infrastructure. We computed these estimates using the MachineLearning Impact calculator (Lacoste et al., 2019).