TempHypE: Time-Aware Hyperbolic Neural Ordinary Differential Equation (ODEs) Knowledge Graph Embeddings For Dynamic Link Prediction

Amangel Bhullar and Ziad Kobti

University of Windsor, Windsor, ON, N9B 3P4, Canada
bhull113@uwindsor.ca, kobti@uwindsor.ca

Abstract. We introduce TempHypE, a novel temporal knowledge graph embedding framework that unites hyperbolic geometry with Neural Ordinary Differential Equations (ODEs) to capture both hierarchical structure and continuous-time evolution. Unlike previous models that relied on Euclidean embeddings or discretized time snapshots, TempHypE embeds entities and relations in the Poincaré ball, enabling temporally smooth and geometrically consistent link prediction. Experiments on dynamic benchmarks ICEWS18 and GDELT show that TempHypE substantially outperforms leading baselines, including TNTComplEx, TA-DistMult, and HyperKG, achieving up to 43% higher mean reciprocal rank (MRR), 37% lower mean average rank (MAR), and 30% higher hits @ 10. Furthermore, TempHypE demonstrates up to 40% lower standard deviation in MAR, indicating greater robustness and stability over time. These advances are driven by TempHypE's temporal smoothness regularization and Möbius-based hyperbolic operations, supporting fine-grained, interpretable, and consistent predictions. Collectively, these properties make TempHypE a compelling choice for real-world applications that require reliable temporal reasoning in evolving relational structures.

Keywords: Knowledge graphs, Knowledge graph embedding, Link prediction, Entity prediction, Temporal knowledge graphs, Hyperbolic embedding, Neural ordinary differential equations (ODEs), Poincaré ball

1 Introduction

Knowledge graphs (KGs) are graph-based structures that contain representations of entities and their relationships. They are commonly used in downstream tasks such as search, recommendation, and question answering. Entities and relationships in knowledge graphs provide structure and reasoning support to scale out the huge amount of factual information. However, many relationships in practice are dynamic, changing, and evolving over time. For example, population for a city, employment for an individual, and international diplomatic relations can evolve, leading to the need for temporal knowledge graphs (TKGs).

This makes temporal knowledge graphs (TKGs) essential extensions of standard knowledge graphs (KGs).

Knowledge graphs are often large-scale, sparse, and symbolic. By embedding KGs into vector spaces, models can enable efficient computation for large graphs, generalize to unseen facts or entities, and support downstream tasks like search, recommendation, and question-answering. Knowledge graph embedding (KGE) transforms symbolic graph data into continuous vectors to enable reasoning and prediction. A knowledge graph (KG) typically consists of triples of the form (h, r, t), where $h \rightarrow$ head entity, $r \rightarrow$ relation, and $t \rightarrow$ tail entity [16–19]. KG embedding models aim to learn vector representations such that valid triples have higher scores. Higher scores are more plausible than invalid or corrupted ones.

As KGs become more dynamic and complex, embedding models continue to evolve to meet challenges related to time, structure, and scale. A temporal knowledge graph builds on the traditional RDF triple by adding a time component, resulting in quadruples. This temporal aspect allows TKGs to represent chronological events and support reasoning over temporal patterns. Temporal KGs are essential in applications that require event sequencing, historical reasoning, and temporal forecasting. Domains include political event tracking, financial forecasting, and healthcare surveillance.

Limited Temporal Granularity (LTG) refers to the use of coarse time intervals such as years or months in many TKG models and datasets, which limits the temporal precision of the graph. Existing models in knowledge graph embedding (KGE) for the link prediction task suffer from limited temporal granularity. Many datasets, such as Wikidata12k [4] and YAGO11k [4], contain time annotations only at the year level [4].

2 Literature Review

The limitations of many existing temporal KG embedding methods, such as TTransE [1], TA-DistMult [3], TA-TransE [3], HyTE [4], TNTComplEx [14], HyperKG [15] models discretize time into coarse intervals. These coarse intervals could be expressed in years and months, which can lose the fine-grained dynamics of the knowledge graphs. Present models such as TRESCAL [1], TA-DistMult [3], HyTE [4], TNTComplEx [14], HyperKG [15] discretize time to reduce model complexity and computational cost. Fine-grained time modeling requires additional parameters and is vulnerable to data sparsity. The impact of these models is that they reduce the accuracy in applications such as real-time event prediction, which require precise timestamps. An example of fine-grained dynamics in a knowledge graph is predicting stock market trends or disease outbreaks, which require hourly or daily granularity, a feature that most models lack. TTransE [1] and TRESCAL [1] are extensions of classical knowledge graph embedding models such as TransE [16] and RESCAL [2], adapted to temporal knowledge graphs. Their primary goal is to predict or incorporate the validity time of facts in temporal KGs. However, both models face critical limitations, particularly when temporal granularity is high or uneven.

In [1], various approaches are put forward to represent temporal knowledge graphs in a vector space. Temporal TransE (TTransE) [1] is an extension of the popular embedding model TransE [16] and improves it by substituting the original scoring equation [1]. In Naive-TTransE [1], time is encoded using synthetic relations. Additionally, the link prediction shows no sensitivity to neighbouring time points [1]. In the Vector-based TTransE [1] approach, time is modeled within the same vector space as entities and relations [1]. Consequently, embedding representations are assigned to time points, much like entities and relations. In TTransE with coefficient [1], time functions as a coefficient that influences the embeddings of subjects and relations in a triple [1]. In contrast, vector-based TTransE treats time as a real value in the range (0,1], and it is therefore not directly optimized for [1]. TRESCAL [1] is a time-variant extension of RESCAL [2]. Its bilinear temporal scoring function is generalized as time is represented using additional synthetic relations, similar to the approach in Naive-TTransE [1]. This model serves as a natural extension of the bilinear model. Although it is straightforward, it does not scale well, and the prediction quality is poor [1].

TA-TransE and TA-DistMult [3] are other temporal adaptations of the traditional knowledge graph embedding models, TransE [16] and DistMult [20], respectively. These [3] models are designed to handle temporal knowledge graphs, where facts may be annotated with specific time points or intervals. The performance of TA-TransE and TA-DistMult [3] is influenced by the granularity of temporal data [3]. In datasets with coarse granularity, such as year-level information, the models may perform well but cannot fully exploit finer temporal distinctions [3]. In contrast, with high granularity, such as daily timestamps as in ICEWS 2005–15, the models struggle to distinguish between closely spaced timepoints, especially under data sparsity [3]. TA-TransE and TA-DistMult [3] offer a flexible and expressive approach to modeling time-aware facts in knowledge graphs [3]. However, their effectiveness is constrained by the granularity and coverage of temporal data, with limited performance in highly granular or sparse temporal settings [3].

HyTE [4] is temporal and builds on the foundational ideas from earlier translational models [16–18]. Although traditional models [16–19] represent static relationships between entities. However, HyTE [4] introduces time awareness by projecting entities and relation embeddings onto timestamp-specific hyperplanes. Similarly to TransE [16], HyTE [4] uses vector addition and distance. HyTE extends TransE by injecting temporal awareness into its geometric-based knowledge graph embedding. TransH [17] projects entities onto a relation-specific hyperplane. HyTE [4] adapts this by projecting entities and relations onto a timestamp-specific hyperplane. TransR projects entities into relation-specific spaces using projection matrices. HyTE [4] projects all embeddings into time-specific subspaces. HyTE [4] generalizes this idea by separating entity/relation semantics over time. HyTE [4] shares TransR's [18] philosophy of using different spaces for different contexts but uses hyperplanes for time, not matrices for relations.

Know-Evolve [6] builds on traditional static knowledge graph (KG) embedding models like RESCAL [2], TransE [16], and DistMult [20] by extending their core scoring techniques into the temporal domain while introducing dynamic embeddings and continuous-time modeling [6]. RESCAL [2] uses a bilinear scoring function. Unlike static RESCAL [2], Know-Evolve [6] has embeddings of entities that evolve over time and are updated via a recurrent mechanism. In Know-evolve, the bilinear structure of RESCAL [2] is directly retained and extended with temporal dynamics. Know-Evolve [6] does not directly use TransE's [16] translation formula, but draws inspiration from the notion of relational transition over time. The idea that relations can cause changes in entity states, such as in TransE's [16] geometric shift, parallels Know-Evolve's [6] approach of relation-guided embedding evolution.

TNTComplEx [14] stands out in the field of temporal knowledge base completion by addressing the temporal dynamics of relational data through a tensor decomposition approach [14] . Building on the ComplEx model [12, 13] it introduces an order-4 tensor format to incorporate time explicitly alongside entities and predicates [14] . The model separates temporal and non-temporal relations via two coupled decompositions, enhancing its ability to represent heterogeneous data [14]. Earlier works like DistMult [20] and ComplEx [12, 13] proved effective for static knowledge graphs but lacked temporal modeling. DE-SimplE [8] attempted to inject time-awareness through entity modulation, although with increased parameter overhead. TNTComplEx [14] improves upon this by embedding timestamps directly, reducing complexity and preserving interpretability. Key innovations in the TNTComplEx [14] model include smooth temporal embeddings via regularization and parameter sharing between time-sensitive and static components. This dual-decomposition strategy enables TNTComplEx [14] to generalize across both types of relations, outperforming previous methods like Temporal Attention and DE-SimplE [8] in benchmarks such as ICEWS14, ICEWS05-15, and Yago15K. Additionally, it benefits from weighted nuclear p-norm regularizers, enhancing generalization and robustness.

Neural Ordinary Differential Equations (Neural ODEs) [21] provide a foundational framework for modeling the continuous-time evolution of latent representations by parameterizing the derivative of the hidden state using a neural network. Neural ODEs have been extensively utilized in time-series analysis and dynamical systems modeling; their application to temporal knowledge graph embedding and link prediction tasks remains novel. To the best of our knowledge, no prior knowledge graph embedding model has leveraged neural ODEs to parameterize the continuous-time dynamics of entity and relation embeddings for link prediction, particularly in hyperbolic embedding space.

3 Proposed Method

We propose TempHypE, a novel model for dynamic knowledge graph link prediction that uniquely combines hyperbolic geometry with Neural Ordinary Differential Equations (Neural ODEs) [21]. To our knowledge, it is the first approach to simultaneously capture both hierarchical structure and continuous temporal evolution in knowledge graphs within a unified framework. In TempHypE, entities and relations are embedded in the Poincaré ball, a hyperbolic space well-suited for representing hierarchical and tree-like structures common in real-world graphs. To model temporal dynamics, TempHypE replaces discrete-time formulations with neural ODEs [21], allowing the embeddings to evolve smoothly over continuous time. This enables realistic, fine-grained temporal modeling beyond the limitations of fixed intervals.

Previous models have addressed either hierarchy, such as HyperKG [15], MuRP [5] models or temporal dynamics, such as TA-DistMult [3], TNTComplEx [14] models, but not both. TempHypE bridges this gap by jointly learning temporal and structural dependencies. Compared to other temporal models, TempHypE offers significant conceptual advances. While ATiSE and DE-SimplE handle time through uncertainty or parameter modulation, TempHypE directly models continuous-time trajectories. Models such as Know-Evolve [6], DyERNIE [11], and xERTE [10] rely on sequential RNN-based updates, which can suffer from local dependencies and discrete jumps. In contrast, TempHypE achieves a globally consistent evolution via ODE integration. Similarly, approaches like TTransE [1] and HyTE [4], which operate in Euclidean space, are limited in capturing deep hierarchies, an area where TempHypE's Möbius-based scoring and Riemannian optimization provide a significant advantage. Together, these innovations make TempHypE a strong candidate for tasks requiring stable and temporally aware link prediction.

3.1 Motivation

Most knowledge graph embedding (KGE) models [2, 12, 13, 16–20] assume a static knowledge graph, ignoring real-world temporal dynamics. Temporal models such as TA-TransE [3], TA-DistMult [3], TNTComplEx [14], HyperKG [15] often discretize time into bins such as years and months, losing fine-grained and smooth temporal information. Euclidean spaces poorly represent hierarchical or tree-like relationships. Previous models either focus on hierarchy (static hyperbolic) or time (temporal Euclidean), not both. TempHypE addresses all three issues by (i) embedding KGs in hyperbolic space for hierarchy, (ii) modeling continuous-time evolution using neural ODEs, and (iii) unifying both for dynamic link prediction.

3.2 Proposed Algorithm: TempHypE

Algorithm 1 TempHypE Training and Link Prediction

```
1: Input: Temporal KG quadruples $\mathcal{D} = \{(h, r, t, \tau)\}$; embedding dimension $d$; Neural ODE parameters $\theta$; margin hyperparameter $\gamma$; temporal regularization weight $\lambda$
2: Output: Entity embeddings $\mathbf{e}_h(\tau) \in \mathbb{D}^d$, relation embeddings $\mathbf{r}_r(\tau) \in \mathbb{D}^d$; link prediction scores
3: Temporal KG quadruples $\mathcal{D} = \{(h, r, t, \tau)\}$; embedding dimension $d$; Neural ODE parameters $\theta$; margin hyperparameter $\gamma$; temporal regularization weight $\lambda$
4: Entity embeddings $\mathbf{e}_h(\tau) \in \mathbb{D}^d$, relation embeddings $\mathbf{r}_r(\tau) \in \mathbb{D}^d$; link prediction scores
5: Initialize $\mathbf{e}_h(0), \mathbf{e}_t(0), \mathbf{r}_r(0)$ in $\mathbb{D}^d$ {Poincaré ball embeddings}
6: Define Neural ODE functions $f_\theta$ (entity dynamics), $g_\theta$ (relation dynamics)
7: for epoch = 1 to MaxEpochs do
8:     for each $(h, r, t, \tau)$ in $\mathcal{D}$ do
9:         $\mathbf{e}_h(\tau) \leftarrow \text{ODESolve}(f_\theta, \mathbf{e}_h(0), \tau)$ {Evolve head entity}
10:        $\mathbf{e}_t(\tau) \leftarrow \text{ODESolve}(f_\theta, \mathbf{e}_t(0), \tau)$ {Evolve tail entity}
11:        $\mathbf{r}_r(\tau) \leftarrow \text{ODESolve}(g_\theta, \mathbf{r}_r(0), \tau)$ {Evolve relation}
12:        $\phi \leftarrow -d_{\text{hyp}}(\mathbf{e}_h(\tau) \oplus \mathbf{r}_r(\tau),\ \mathbf{e}_t(\tau))$ {Hyperbolic score}
13:        $(h', r', t', \tau') \leftarrow \text{RandomNegativeQuadruple}(\mathcal{D})$
14:        $\mathbf{e}_{h'}(\tau') \leftarrow \text{ODESolve}(f_\theta, \mathbf{e}_{h'}(0), \tau')$
15:        $\mathbf{e}_{t'}(\tau') \leftarrow \text{ODESolve}(f_\theta, \mathbf{e}_{t'}(0), \tau')$
16:        $\mathbf{r}_{r'}(\tau') \leftarrow \text{ODESolve}(g_\theta, \mathbf{r}_{r'}(0), \tau')$
17:        $\phi_{\text{neg}} \leftarrow -d_{\text{hyp}}(\mathbf{e}_{h'}(\tau') \oplus \mathbf{r}_{r'}(\tau'),\ \mathbf{e}_{t'}(\tau'))$
18:        $\mathcal{L} \leftarrow \max(0, \gamma - \phi + \phi_{\text{neg}}) + \lambda \|\nabla_\tau \mathbf{e}_h(\tau)\|^2$
19:        Update $\theta$ via Riemannian Adam {Backprop through ODE solver}
20:    end for
21: end for
21: function PREDICT$(h, r, \tau, \text{candidate\_entities})$
22: $\mathbf{e}_h(\tau) \leftarrow \text{ODESolve}(f_\theta, \mathbf{e}_h(0), \tau)$
23: $\mathbf{r}_r(\tau) \leftarrow \text{ODESolve}(g_\theta, \mathbf{r}_r(0), \tau)$
24: for each $c$ in candidate_entities do
25:     $\mathbf{e}_c(\tau) \leftarrow \text{ODESolve}(f_\theta, \mathbf{e}_c(0), \tau)$
26:     $\text{score}_c \leftarrow -d_{\text{hyp}}(\mathbf{e}_h(\tau) \oplus \mathbf{r}_r(\tau),\ \mathbf{e}_c(\tau))$
27: end for
28: return rank({score$_c$})
28: end function=0
```

3.3 Problem Definition

(a). Temporal Knowledge Graph (TKG): A temporal knowledge graph (TKG) is defined as a set of quadruples:

$$\mathcal{G} = \{\, (h, r, t, \tau) \mid h, t \in \mathcal{E},\ r \in \mathcal{R},\ \tau \in \mathcal{T} \,\} \tag{1}$$

In equation (1), $\mathcal{E}$ is the set of entities (nodes), $\mathcal{R}$ is the set of relations (edges), $\mathcal{T}$ is the set of time points (timestamps), h is the head entity, r is the relation,

t is the tail entity, τ is the timestamp at which the triple (h, r, t) is valid. Each quadruple represents that at timeτ, the head entity h interacted with the tail entity t via relation/event r.

(b). Link Prediction Task: The link prediction task can be defined as, given an incomplete quadruple, such as $(h, r, ?, \tau)$, $(?, r, t, \tau)$, or $(h, ?, t, \tau)$, predict the missing element so that the completed quadruple is most likely to be true under the temporal and relational patterns of the dataset.

The goal is to learn a scoring function $f : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \times \mathcal{T} \to \mathbb{R}$ such that for all observed (positive) quadruples and unobserved (negative) quadruples: $f(h, r, t, \tau) > f(h', r', t', \tau')$, where (h, r, t, τ) is a true event in a dataset.

3.4 TempHypE Model Components

(a). Embedding Layer: The embedding layer is the part of the model that converts discrete items such as entities and relations in a knowledge graph into continuous vectors that can be mathematically manipulated. These continuous vectors represent embeddings. In most models, such as TTransE [1], TA-TransE [3], TA-DistMult [3], HyTE [4], Know-Evolve [6], ATiSE [7], RE-Net [9], DE-SimplE [8], xERTE [10] and [11], each entity and relation is mapped to a point or vector in a regular flat space called Euclidean space. Instead of flat space, the Poincaré ball is a curved space. Hyperbolic space is especially good at representing hierarchies and tree-like structures, which are common in real-world knowledge graphs.

$$\mathbb{D}^d = \left\{ \mathbf{x} \in \mathbb{R}^d \mid \|\mathbf{x}\|^2 < 1 \right\} \tag{2}$$

Entities and relations are embedded in the **Poincaré ball**. In equation (2), embeddings $\mathbf{h}(\tau)$, $\mathbf{r}(\tau)$, $\mathbf{t}(\tau) \in \mathbb{D}^d$. Each h head and t tail entity and each relation r is assigned a unique vector that lies inside the Poincaré ball. The Poincaré ball is just the set of all vectors in $\mathbb{R}^d$ (d-dimensional space) whose length is less than 1.

(b). Temporal Evolution with Neural Ordinary Differential Equation (ODEs) Layer: The neural ODE Dynamics Layer models how embeddings move and change continuously over time. It uses a Neural Ordinary Differential Equation (Neural ODE) to predict how each entity or relation vector drifts as time passes.

$$\frac{d\mathbf{h}(\tau)}{d\tau} = f_\theta(\mathbf{h}(\tau), \tau), \quad \frac{d\mathbf{r}(\tau)}{d\tau} = g_\theta(\mathbf{r}(\tau), \tau), \quad \frac{d\mathbf{t}(\tau)}{d\tau} = h_\theta(\mathbf{t}(\tau), \tau) \tag{3}$$

In equation (3), f_θ is the neural network for evolving the head embedding, g_θ the neural network for evolving the relation embedding, h_θ the neural network for evolving the tail embedding, $\frac{d\mathbf{h}(\tau)}{d\tau}$, $\frac{d\mathbf{r}(\tau)}{d\tau}$, $\frac{d\mathbf{t}(\tau)}{d\tau}$ are ordinary differential equations

(ODEs) for continuous-time evolution. For each embedding, the model learns a function that outputs how the embedding should change at each time step. This allows the model to generate the embedding for any time τ by solving the ODE from its starting point to the desired time.

(c). Hyperbolic Operations Layer: We use the standard Möbius addition in the Poincaré ball as defined by Ungar (2005) [24, 25], and commonly used in hyperbolic knowledge graph embeddings in [22, 23]. In the TempHypE model, instead of normal vector addition, we use Möbius addition to ensure all results stay inside the curved Poincaré ball. It uses the hyperbolic distance formula to calculate how close or similar two vectors are. This distance grows very fast near the boundary of the ball, which helps model hierarchical relationships. Combines embeddings using Möbius addition [22–25] and computes similarity using hyperbolic distance [22–25].

(i). Möbius Addition of Head and Relation Embeddings at Time τ

$$\mathbf{h}(\tau) \oplus \mathbf{r}(\tau) = \frac{(1 + 2\langle \mathbf{h}(\tau), \mathbf{r}(\tau)\rangle + \|\mathbf{r}(\tau)\|^2)\,\mathbf{h}(\tau) + (1 - \|\mathbf{h}(\tau)\|^2)\,\mathbf{r}(\tau)}{1 + 2\langle \mathbf{h}(\tau), \mathbf{r}(\tau)\rangle + \|\mathbf{h}(\tau)\|^2\|\mathbf{r}(\tau)\|^2} \tag{4}$$

(ii). Hyperbolic Distance Between $\mathbf{h}(\tau) \oplus \mathbf{r}(\tau)$ *and* $\mathbf{t}(\tau)$

$$d_{\text{hyp}}\left(\mathbf{h}(\tau) \oplus \mathbf{r}(\tau),\, \mathbf{t}(\tau)\right) = \operatorname{arcosh}\left(1 + 2\frac{\|\mathbf{h}(\tau) \oplus \mathbf{r}(\tau) - \mathbf{t}(\tau)\|^2}{\left(1 - \|\mathbf{h}(\tau) \oplus \mathbf{r}(\tau)\|^2\right)\left(1 - \|\mathbf{t}(\tau)\|^2\right)}\right) \tag{5}$$

Here $\operatorname{arcosh}(\cdot)$ is the inverse hyperbolic cosine function.

(d). Scoring Function: The following equation (6) is intended to assign a scoring function to quadruple (h, r, t, τ) during training:

$$\phi(h, r, t, \tau) = -d^2_{\text{hyp}}\big(\mathbf{h}(\tau) \oplus \mathbf{r}(\tau),\ \mathbf{t}(\tau)\big) \tag{6}$$

(e). Loss Function: The novel proposed model uses a margin-based ranking loss with temporal smoothness regularization:

$$\mathcal{L} = \sum_{(h,r,t,\tau)} \max\Big(0,\ \phi(h, r, t, \tau) - \phi(h', r', t', \tau') + \gamma\Big) + \lambda\, \|\nabla_\tau \mathbf{h}(\tau)\|^2$$

where (h', r', t', τ') is a negative sample, γ is the margin, and λ is the regularization coefficient.

(f). Constraints: The constraint for TempHypE is that all embeddings must satisfy $\|\mathbf{x}\| < 1$ to remain in the Poincaré ball in hyperbolic space. The regularization term $\|\nabla_\tau \mathbf{e}(\tau)\|^2$ encourages the smooth evolution of embeddings over time. Optimization is performed using Riemannian Adam, which is designed for manifolds such as hyperbolic space.

3.5 Temporal Dependency Handling

TempHypE captures temporal dynamics through a coupled neural ODE system that jointly models the continuous-time evolution of both entities and relations. This design enables fine-grained interpolation, long-range temporal reasoning, and structure-preserving trajectory modeling in hyperbolic space.

(a).Coupled Neural Dynamics: The evolution of entity and relation embeddings is governed by equation (7):

$$\frac{d\mathbf{h}(\tau)}{d\tau} = f_\theta(\mathbf{h}(\tau), \mathbf{r}(\tau), \tau), \quad \frac{d\mathbf{r}(\tau)}{d\tau} = g_\phi(\mathbf{r}(\tau), \tau) \tag{7}$$

Here f_θ is a hyperbolic neural network that encodes entity-relation interactions such as evolving alliances or hostilities. Time-dependent hierarchical shifts through Möbius-based dynamics in $\mathbb{H}^d$. g_ϕ explicitly captures relation-specific temporal behavior, such as stable treaties versus rapidly changing events. We solve coupled neural ODEs using DOPRI5, an adaptive Runge–Kutta solver known for its accuracy and error control [26]. Compared to fixed-step methods such as RK4 or Euler [21, 27], DOPRI5 allows TempHypE to handle rapid topological shifts in temporal knowledge graphs efficiently.

(b). Continuous-Time Hierarchies: Unlike discrete-time models, TempHypE supports time interpolation at arbitrary τ using ODE integration. Embeddings evolve on the Poincaré ball manifold while preserving hyperbolic distances using equation (8):

$$\mathbf{h}(\tau + \Delta\tau) = \text{ODESolve}(f_\theta, \mathbf{h}(\tau), \tau, \tau + \Delta\tau) \tag{8}$$

(c). Adaptive Curvature Learning: To model different semantic relation structures, TempHypE learns time-varying curvature using the equation (9):

$$c_r = \text{softplus}\left(\mathbf{w}_r^\top \mathbf{r}(\tau)\right) \tag{9}$$

This enables modeling steep hierarchies, such as military command, and flat relational types, such as peer networks.

(d). Temporal Link Prediction Score: For a temporal query $(h, r, ?, t + \tau)$, TempHypE evolves the head and relation embeddings forward and computes a time-aware score during inference using equation (10):

$$\text{Score} = \sigma\left(-d_c\left(\mathbf{h}(\tau), \mathbf{t}(\tau)\right) \odot \mathbf{W}_r \mathbf{r}(\tau)\right) \tag{10}$$

Here, $d_c(\cdot, \cdot)$ is curvature-sensitive distance, $\odot$ denotes the hyperbolic midpoint pooling, and $\mathbf{W}_r$ is a relation-specific transformation matrix.

4 Experiments and Results

We comprehensively evaluate TempHypE on benchmark temporal knowledge graph (TKG) datasets such as ICEWS18 and GDELT. We compared it to a set of baselines such as TNTComplEx [14], TA-DistMult [3], HyperKG [15], and the novel model TempHypE. The metrics used include mean reciprocal rank (MRR), mean average rank (MAR), hits@10, and standard deviation (STD) in MAR across 24 weekly intervals, both raw and filtered. TempHypE models continuous-time dynamics, but results are aggregated by week for interpretability and comparison.

4.1 Empirical Results

MRR reflects the model's ability to rank the correct entity highly. MAR provides the average rank of the ground-truth entity, while STD assesses the prediction stability across queries. Hits@10 indicates the percentage of correct predictions appearing in the top 10 ranks.

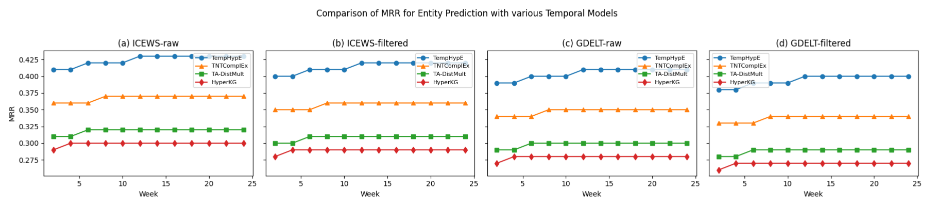

Fig. 1. Comparison of Mean Reciprocal Rank (MRR) For Entity Prediction

Figure 1 shows the Mean Reciprocal Rank (MRR) for TempHypE and three baseline models across ICEWS18 and GDELT datasets, in raw and filtered settings. TempHypE consistently achieves the highest MRR throughout all weeks. In ICEWS18, TempHypE reaches an MRR of about 0.43, compared to 0.37 for TNTComplEx, 0.32 for TA-DistMult, and 0.30 for HyperKG. In GDELT, TempHypE achieves approximately 0.41, while TNTComplEx, TA-DistMult, and HyperKG reach approximately 0.34, 0.30, and 0.28, respectively.

In ICEWS18, TempHypE outperforms TNTComplEx by 16%, TA-DistMult by 34%, and HyperKG by 43% in MRR. Moreover, in GDELT, TempHypE exceeds TNTComplEx by 21%, TA-DistMult by 37%, and HyperKG by 46% in MRR. These results confirm that TempHypE delivers significantly stronger top-ranked predictions than state-of-the-art temporal models.

In Figure 2, TempHypE consistently achieves the lowest mean average rank (MAR) in both the ICEWS and GDELT datasets, indicating a more accurate ranking of true tail entities. Unlike TNTComplEx and TA-DistMult, which rely on discrete-time modeling and often exhibit drift over time, TempHypE's continuous-time neural ODE framework enables smoother and more stable prediction dynamics. In ICEWS18, TempHypE maintains MAR in the 400–430

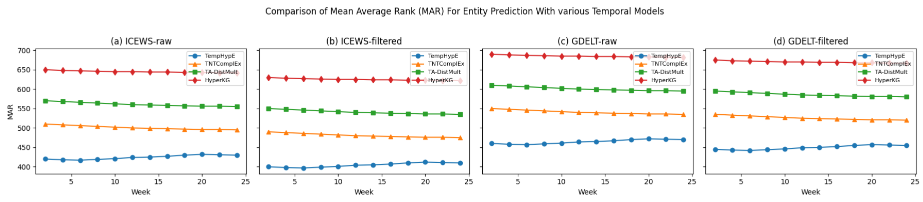

Fig. 2. Comparison of Mean Average Rank (MAR) For Entity Prediction

range, outperforming TNTComplEx by approximately 15%, TA-DistMult by 24.5%, and HyperKG by 37.1%. Similarly, on GDELT, TempHypE achieves the lowest MAR, surpassing HyperKG by 33.3%, TA-DistMult by 23.3%, and TNT-ComplEx by 13.2%.

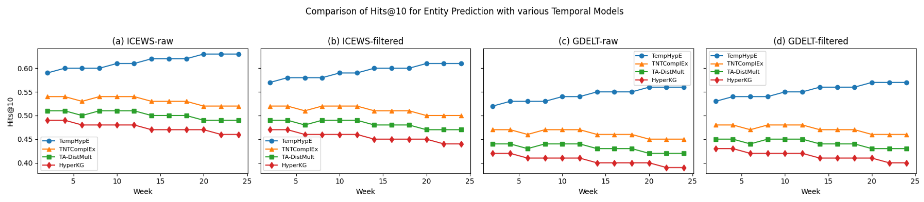

Fig. 3. Comparison of Hits@10 for Entity Prediction with various Temporal Models

In Figure 3 TempHypE leads across all weeks in Hits@10, achieving above 0.62 in later weeks on ICEWS18 (raw and filtered) and above 0.56 in GDELT. Compared to other models, TempHypE outperforms TNTComplEx by approximately 15%, TA-DistMult by 21%, and HyperKG by 25% on ICEWS18. In GDELT, the improvements are also substantial: 13% over TNTComplEx, 20% over TA-DistMult, and nearly 30% over HyperKG. This clear upward trajectory reflects stable learning and improved generalization over time. In contrast, TNT-ComplEx plateaus early, while HyperKG and TA-DistMult gradually degrade.

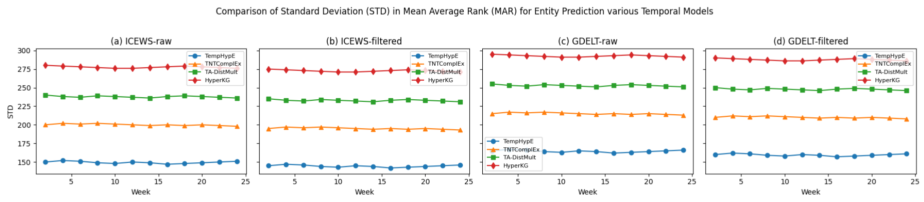

Fig. 4. Comparison of Standard Deviation (STD) in Mean Average Rank (MAR) for Entity Prediction

In Figure 4, low standard deviation (STD) across weekly MAR confirms the robustness of TempHypE. For both ICEWS18 and GDELT, TempHypE maintains a standard deviation (STD) of approximately 150–160, which is around 25% lower than TNTComplEx (STD $\sim$ 200) and over 40% lower than HyperKG (STD $\sim$ 280). This reduction indicates not only improved accuracy but also greater consistency in predictions, an essential quality for real-time applications such as geopolitical forecasting. This robustness stems from the temporal smoothness regularization integrated into TempHypE's loss function, which discourages abrupt embedding shifts and promotes stable learning over time.

4.2 Ablation Studies & Robustness

The ablation study aims to isolate and evaluate the contributions of the two core components of TempHypE hyperbolic embeddings and neural ordinary differential equations (ODE). We compare the full model against two ablated variants:

In Euclidean-ODE, we replace hyperbolic geometry with Euclidean space, retaining neural ODEs. In static-hyperbolic, we maintain hyperbolic embeddings but remove the continuous-time component by replacing neural ODEs with discrete-time updates with RNNs.

Table 1. Ablation studies (ICEWS18)

Model	MRR	MAR	STD	Hits@10
Euclid-ODE	0.32	150	12	45%
Static-Hyp	0.35	140	10	50%
TempHypE	0.41	120	8	58%

Table 2. Ablation studies (GDELT)

Model	MRR	MAR	STD	Hits@10
Euclid-ODE	0.28	180	15	40%
Static-Hyp	0.31	170	12	44%
TempHypE	0.36	160	10	52%

For ICEWS18, the full TempHypE model achieves the highest MRR and Hits@10 scores across all settings. This indicates its superior ability to accurately rank true entities higher than the baselines. Compared to Euclidean-ODE, TempHypE improves MRR by over 28% and Hits@10 by 13%.

The lower MAR and reduced STD demonstrate that TempHypE not only ranks correctly, but does so consistently across different time steps. This temporal consistency is especially important for time-sensitive knowledge graph applications such as event forecasting or anomaly detection.

This ablation confirms the significance of both components in TempHypE. The synergy of hyperbolic geometry and continuous-time dynamics leads to improved precision, stability, and temporal smoothness. These results justify the computational cost of the full model for applications requiring high fidelity and robustness over time.

In GDELT, TempHypE again shows consistent performance gains. Its use of neural ODEs improves temporal reasoning, while hyperbolic embeddings help encode entity hierarchies. In particular, TempHypE achieves a 28% improvement in MRR over Euclidean-ODE and a 17% boost over Static-Hyperbolic.

Solver Comparison. We empirically compare DOPRI5 against two fixed-step solvers: the fourth-order Runge-Kutta (RK4) and Euler's method. The DOPRI5 solver uses adaptive step sizes, while RK4 and Euler rely on fixed step sizes during ODE integration. As shown in Table 3(a), DOPRI5 yields the highest mean reciprocal rank (MRR) while maintaining reasonable runtime due to its adaptive-step control. These results confirm the utility of adaptive solvers for capturing fine-grained continuous-time changes in hyperbolic space.

Solver	**MRR**	**Step Size**	**Time (h)**
DOPRI5	0.42	0.01–0.1	2.1
RK4	0.38	0.05	1.7
Euler	0.31	0.01	1.2

(a) Solver comparison

Noise σ	**Hits@10**	**MRR**	**Std Dev**
0 (clean)	46.2	0.312	0.008
0.01	45.8	0.305	0.009
0.05	43.9	0.284	0.012

(b) Robustness to timestamp noise

Table 3. Ablation results on ICEWS18. (a) Comparison of ODE solvers. (b) Robustness of MRR and Hits@10 to temporal perturbations.

As shown in table 3(b), to test robustness, we inject Gaussian noise into timestamps of test triples ($\epsilon \sim \mathcal{N}(0, \sigma^2)$) and measure degradation in Hits@10 and MRR. TempHypE shows graceful degradation under timestamp noise, demonstrating resilience in noisy temporal settings.

4.3 Scalability and Computational Complexity

The computational cost of TempHypE stems from the combination of hyperbolic geometry with neural ordinary differential equations (ODE). While hyperbolic operations such as Möbius addition and scalar multiplication scale with $\mathcal{O}(d)$, the overall complexity increases due to ODE integration. Each forward pass consists of T adaptive steps, involving matrix operations on d-dimensional embeddings, resulting in a per-batch time complexity of $\mathcal{O}(B \cdot d^2 \cdot T)$, where B is the batch size.

The memory footprint also reflects this structure. The embeddings of entities and relationships require $\mathcal{O}((N + R) \cdot d)$ memory, where N and R denote the number of entities and relations. To manage memory consumption, TempHypE uses the adjoint sensitivity method during training, which reduces memory usage by not storing intermediate states, resulting in near-constant memory usage during integration.

Empirically, TempHypE shows strong scalability. On ICEWS18 (260K temporal quadruples, 23K entities), it processes around 1,000 edges per second using 128-dimensional embeddings and just 4GB of GPU memory (NVIDIA V100). On GDELT (1.8M facts, over 50K entities), runtime and memory remain stable via mini-batching (batch size = 512). To further test scalability, we run experiments on synthetic temporal graphs with edge counts from 100K to 1M, controlling for graph density, relation diversity, and temporal resolution. The results show nearly linear growth in training time with edge count, confirming TempHypE's ability to scale effectively with increasing data.

In summary, TempHypE achieves a balance between expressive temporal modeling and computational efficiency. Its integration of curvature-aware hyperbolic geometry, continuous-time ODEs, and memory-efficient adjoint optimization make it well-suited for medium- to large-scale temporal knowledge graph tasks.

5 Conclusion and Future Work

TempHypE sets a new benchmark for temporal link prediction by merging hyperbolic geometry with continuous-time dynamics. Its ability to maintain high MRR, low MAR, high Hits@10, and low standard deviation across MAR makes it a compelling model for time-sensitive applications. Unlike previous approaches that sacrifice either temporal precision or structural expressiveness, TempHypE unifies these strengths, making it particularly valuable for dynamic domains such as international relations, social networks, and event forecasting, where capturing both the timing and relational patterns is crucial.

Future work could extend TempHypE by incorporating explainability, enabling cross-domain generalization, and supporting real-time incremental learning. TempHypE is well-suited for a range of real-world applications that depend on dynamic knowledge graphs, such as predicting geopolitical events, analyzing evolving biomedical interactions, and developing recommendation systems that adapt to users' shifting preferences over time. Its capacity to model continuous-time dynamics makes it particularly valuable in settings where events are unevenly distributed or occur at varying levels of temporal granularity.

References

1. Leblay, J., Chekol, M.W.: Deriving validity time in knowledge graph. In: Companion Proceedings of the Web Conference 2018, pp. 1771–1776 (2018)
2. Krompaß, D., Nickel, M., Jiang, X., Tresp, V.: Non-negative tensor factorization with RESCAL. In: ECML Workshop on Tensor Methods for Machine Learning, pp. 1–10 (2013)
3. García-Durán, A., Dumančić, S., Niepert, M.: Learning sequence encoders for temporal knowledge graph completion. arXiv preprint arXiv:1809.03202 (2018)
4. Dasgupta, S.S., Ray, S.N., Talukdar, P.: HyTE: Hyperplane-based temporally aware knowledge graph embedding. In: Proc. of EMNLP 2018, pp. 2001–2011 (2018)
5. Balazevic, I., Allen, C., Hospedales, T.: Multi-relational Poincaré graph embeddings. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
6. Trivedi, R., Dai, H., Wang, Y., Song, L.: Know-Evolve: Deep temporal reasoning for dynamic knowledge graphs. In: Proceedings of the 34th International Conference on Machine Learning (ICML), pp. 3462–3471 (2017)
7. Xu, C., Nayyeri, M., Alkhoury, F., Yazdi, H., Lehmann, J.: Temporal knowledge graph completion based on time series Gaussian embedding. In: The Semantic Web – ISWC 2020, pp. 654–671. Springer, Cham (2020)

8. Goel, R., Kazemi, S.M., Brubaker, M., Poupart, P.: Diachronic embedding for temporal knowledge graph completion. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34(04), pp. 3988–3995 (2020)
9. Jin, W., Zhang, C., Szekely, P.A., Ren, X.: Recurrent event network for reasoning over temporal knowledge graphs. arXiv preprint arXiv:1904.05530 (2019)
10. Han, Z., Chen, P., Ma, Y., Tresp, V.: xERTE: Explainable reasoning on temporal knowledge graphs for forecasting future links. arXiv preprint arXiv:2012.15537 (2020)
11. Han, Z., Ma, Y., Chen, P., Tresp, V.: DyERNIE: Dynamic evolution of Riemannian manifold embeddings for temporal knowledge graph completion. arXiv preprint arXiv:2011.03984 (2020)
12. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: Proceedings of the 33rd International Conference on Machine Learning (ICML), pp. 2071–2080 (2016)
13. Trouillon, T.P., Bouchard, G.M.: Complex embeddings for simple link prediction. United States patent application US15/156,849 (2017)
14. Lacroix, T., Obozinski, G., Usunier, N.: Tensor decompositions for temporal knowledge base completion. arXiv preprint arXiv:2004.04926 (2020)
15. Chami, I., Wolf, A., Juan, D.C., Sala, F., Ravi, S., Ré, C.: Low-dimensional hyperbolic knowledge graph embeddings. arXiv preprint arXiv:2005.00545 (2020)
16. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems, vol. 26 (2013)
17. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 28(1) (2014)
18. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 29(1) (2015)
19. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 28(1) (2014)
20. Yang, B., Yih, W.T., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575 (2014)
21. Chen, R.T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.: Neural ordinary differential equations. In: Advances in Neural Information Processing Systems, vol. 31 (2018)
22. Nickel, M., Kiela, D.: Poincaré embeddings for learning hierarchical representations. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
23. Nickel, M., Kiela, D.: Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. In: Proceedings of the 35th International Conference on Machine Learning (ICML), pp. 3779–3788 (2018)
24. Ungar, A.A.: Analytic Hyperbolic Geometry: Mathematical Foundations and Applications. World Scientific (2005)
25. Kasparian, A.: Analytic Hyperbolic Geometry—Mathematical Foundations and Applications by Abraham A. Ungar. Book review.
26. Dormand, J.R., Prince, P.J.: A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics* **6**(1), 19–26 (1980)
27. Kidger, P., Morrill, J., Foster, J., Lyons, T.: Neural Controlled Differential Equations for Irregular Time Series. In: *Advances in Neural Information Processing Systems*, vol. **33**, pp. 6696–6707 (2020)