Improving Temporal Knowledge Graph Forecasting via Multi-Rewards Mechanism and Confidence-Guided Tensor Decomposition Reinforcement Learning

Nam Le[1,2][a], Thanh Le[1,2][b] and Bac Le[1,2][c]
[1]*Faculty of Information Technology, University of Science, Ho Chi Minh City, Vietnam*
[2]*Vietnam National University, Ho Chi Minh City, Vietnam*
{lnnam, lnthanh, lhbac}@fit.hcmus.edu.vn

Keywords: Temporal Knowledge Graph Reasoning, Reinforcement Learning, Multi-Reward Function, Tensor Decomposition.

Abstract: Temporal knowledge graph reasoning, which has received widespread attention in the knowledge graph research community, is a task that predicts missing facts in data. When framed as a problem of forecasting future events, it becomes more challenging than the conventional completion task. Reinforcement learning is one of the potential techniques to address these challenges. Specifically, an agent navigates through a historical snapshot of a knowledge graph to find answers to the input query. However, these learning frameworks suffer from two main drawbacks: (1) a simplistic reward function and (2) candidate action selection being influenced by data sparsity issues. To address these problems, we propose a multi-reward function that integrates binary, adjusted path-based, adjusted ground truth-based, and high-frequency rule rewards to enhance the agent's performance. Furthermore, we incorporate recent advanced tensor decomposition methods such as TuckER, ComplEx, and LowFER to construct a reliability evaluation module for candidate actions, allowing the agent to make more reliable action choices. Our empirical results on benchmark datasets demonstrate significant improvements in performance while preserving computational efficiency and requiring fewer trainable parameters.

1 INTRODUCTION

Temporal knowledge graphs (TKGs) extend the representation of events from static KG triples in the form of (s,r,o)—where s and o are the subject and object entities, respectively, and r represents their relationship—to quadruples (s,r,o,t), with the inclusion of a timestamp to indicate the validity of the fact at a specific point or time interval. For instance: *(Japan, Make a visit, Thailand, 2014-09-22)*. As such, TKGs can evolve continuously over time. There is a substantial amount of research applying KGs and TKGs to fields like question-answering (Mavromatis et al., 2022) and recommendation systems (Chen et al., 2022).

KGs and TKGs are inherently incomplete. Therefore, the reasoning task of discovering missing or new facts from known ones plays a crucial role. This problem is typically studied under two different setups: 1) interpolation and 2) extrapolation. Most interpolation-based methods, such as TTransE (Leblay and Chekol, 2018a), TA-DistMult (García-Durán et al., 2018), and DE-SimplE (Goel et al., 2020), focus on completing data from known facts without temporal constraints, meaning they primarily predict missing facts associated with past timestamps. This work focuses on the extrapolation link prediction problem, designing a model to predict future links. For example, the question "Who will be the president of the USA in 2024?" can be converted into the problem of future link prediction as: *(?, president of, USA, 2024)*.

Reasoning on knowledge graph with the extrapolation setup presents more challenges than interpolation due to temporal constraints in the data. Furthermore, many unknown entities in the query make it difficult for learning models to adapt quickly. Recently, path-based reasoning methods for static knowledge graphs, such as DeepPath (Xiong et al., 2017), MINERVA (Das et al., 2017), and Multi-hop KG (Lin

[a] https://orcid.org/0000-0002-2273-5089
[b] https://orcid.org/0000-0002-2180-4222
[c] https://orcid.org/0000-0002-4306-6945

Le, N., Le, T. and Le, B.
Improving Temporal Knowledge Graph Forecasting via Multi-Rewards Mechanism and Confidence-Guided Tensor Decomposition Reinforcement Learning.
DOI: 10.5220/0013161400003890
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 1*, pages 68-79
ISBN: 978-989-758-737-5; ISSN: 2184-433X
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

et al., 2018), as well as for TKGs such as TimeTraveler (Sun et al., 2021), have shown significant improvements in both performance and interpretability for knowledge graph reasoning tasks. However, these methods still have several limitations: 1) The reward function is a critical component for the agent. Most current works focus on constructing a binary global reward function, which makes the agent's learning process inflexible. 2) The action space for the agent is too large, and there is limited research on how to select appropriate actions for the agent.

In this work, we propose a more flexible temporal path-based reasoning model to address the extrapolation reasoning task in TKGs. Our agent, named "CATTer" (Confidence-Augmented Time Traveler) based on TimeTraveler (Sun et al., 2021). We propose reward function criteria such as binary global, adjusted ground truth frequency, adjusted path length, and high-frequency rule rewards, with greater flexibility to stabilize the agent's learning process. Additionally, we integrate tensor decomposition models into the policy network to generate probabilities that represent the reliability of actions, helping the agent more easily select appropriate actions. Moreover, our policy network is implemented with Kolmogorov-Arnold Networks (KAN), achieving performance comparable to multi-layer perceptrons (MLP).

The main contributions of our work are as follows:

- Proposing a new multi-reward function, incorporating various reward criteria for the agent, such as binary global, adjusted ground truth frequency, path length, and high-frequency rules, with enhanced flexibility. This approach aims to improve the agent's learning and reasoning process.
- Incorporating Tensor decomposition architectures such as TuckER, ComplEx, and LowFER with MLP and KAN-Policy Network to generate reliability scores for actions. This enhances the agent's ability to select appropriate actions within the KG environment.
- Performing experiments and ablation study on standard datasets for the future link prediction task. Results based on MRR and Hit@K metrics demonstrate significant improvements compared to baseline models.

Our work is organized as follows: Section 2 introduces related works, focusing mainly on the existing path-based models for static and temporal KG reasoning. Section 3 details our proposed model. Section 4 discusses the experimental setup, main empirical results, and ablation studies. Finally, Section 5 summarizes our conclusions and future research discussion.

2 RELATED WORKS

RL has a wide range of applications in the field of KG reasoning, often referred to as path-based reasoning. This work applies this approach to reasoning tasks on static and temporal KGs.

2.1 RL for Static Knowledge Graph Reasoning

In contrast to traditional embedding-based methods that map entities and relations into low-dimensional continuous spaces, such as TransE (Bordes et al., 2013) and ComplEx (Trouillon et al., 2016), or deep learning techniques like Convolutional Neural Networks (e.g., ConvE (Dettmers et al., 2018)) or Graph Neural Networks (e.g., R-GCN (Schlichtkrull et al., 2018)), path reasoning methods such as DeepPath (Xiong et al., 2017), MINERVA (Das et al., 2017), and Multi-hop KG (Lin et al., 2018) treat the task of link prediction or knowledge graph reasoning as a Markov Decision Process (MDP). These approaches enhance link prediction performance by finding paths between the source and target entities. Moreover, such methods offer more significant potential for understanding the internal mechanics of learning models or agents.

2.2 RL for Temporal Knowledge Graph Reasoning

In TKG research, path-based reasoning or reinforcement learning-based methods are often applied to link prediction tasks in an extrapolation setting, also known as future link forecasting. In addition to Graph Neural Network-based methods like RE-NET (Jin et al., 2019) and CyGNet (Zhu et al., 2021a), or neural network-based methods enhanced by orthogonal differential equations such as TANGO (Han et al., 2021b), RL-based models like TAgent (Tao et al., 2021), TITer (Sun et al., 2021), TPath (Bai et al., 2021), DREAM (Zheng et al., 2023), and RLAT (Bai et al., 2023) also show strong potential for predicting future links with associated timestamps. TAgent (Tao et al., 2021) proposed an agent that utilizes binary terminal reward for learning. TPath (Bai et al., 2021) utilizes path length to construct a reward function. Recently, DREAM (Zheng et al., 2023) and RLAT (Bai et al., 2023) utilized an attention mechanism to design a black-box reward function for agent learning.

3 PROPOSED METHODOLOGY

3.1 Problem Statement

Let $\mathcal{E}$, $\mathcal{R}$, $\mathcal{T}$, and $\mathcal{Q}$ denote the sets of entities, relations, timestamps, and quadruples, respectively. Each quadruplet in TKG can be defined as a tuple (e_s, r, e_o, t), where $r \in \mathcal{R}$ is a relation connecting a subject entity $e_s \in \mathcal{E}$ with an object entity $e_o \in \mathcal{E}$ at timestamp $t \in \mathcal{T}$. In this work, we consider TKG in as discrete form, i.e., a sequence of discrete snapshots over time $\mathcal{G}_{(1,T)} = \{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_T\}$, where $\mathcal{G}_t = \{\mathcal{E}_t, \mathcal{R}, \mathcal{Q}_t\}$ is a static multi-relational graph, and $\mathcal{E}_t$ and $\mathcal{Q}_t$ denote entities and facts that exist at time t.

In this work, we consider the problem of extrapolation-based TKG reasoning. In particular, given a TKG, the main goal is to predict the events that can occur in future time points to capture the evolution of TKG through the timeline, i.e., link prediction and future times. Formally, with given a query $(e_q, r_q, ?, t_q)$ or $(?, r_q, e_q, t_q)$, we have a set of known facts $\{(e_{s_i}, r_i, e_{o_i}, t_i) | t_i < t_q\}$, our goal is to predict the missing object or subject entity in the input query.

This work considers this problem as a Markov Decision Process (MDP) and uses Reinforcement Learning (RL) to solve it. Figure 1 illustrates an overview of our proposed model.

3.2 Reinforcement Learning Framework

Reinforcement learning frameworks for KG reasoning typically consist of four main components: states, actions, transitions, and reward functions. These components can be summarized as follows:

States. Let $\mathcal{S}$ be the state space. Each $s_\ell = (e_{(\ell)}, t_{(\ell)}, e_q, t_q, r_q) \in \mathcal{S}$ where represents a state in state space. The agent starts searching from (e_q, t_q) so the initial state is $s_0 = (e_q, t_q, e_q, t_q, r_q)$. Tuple $e_{(\ell)}, t_{(\ell)}$ and e_q, t_q, r_q are considered as local and global information, respectively.

Actions. Let $\mathcal{A}$ be the action space. At each step ℓ, let $\mathcal{A}_\ell$ be the set of actions for this step. Clearly, $\mathcal{A}_\ell \subset \mathcal{A}$. Formally, $\mathcal{A}_\ell = \{(r', e', t') | (e_\ell, r', e', t') \in \mathcal{Q}, t' \leq t_\ell, t' < t_q\}$ is sampled from the set of all feasible outgoing edges starting from $e_\ell^{t_\ell}$ for memory optimization.

Transitions. In the RL framework, the agent leverages a transition function to transfer from one state to another. Formally, the transition function $\xi : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ defined by:

$$(s_\ell, \mathcal{A}_\ell) \mapsto (e_{\ell+1}, t_{\ell+1}, e_q, t_q, r_q) = s_{\ell+1} \quad (1)$$

where $\mathcal{A}_\ell$ is the sampled from the feasible set of outgoing edges starting from $e_\ell^{t_\ell}$.

Rewards. Reward functions are important in reinforcement learning frameworks. One of the common types is the binary reward function. Specifically, if the agent captures the target entity e_{gt}, which mean it end up with terminal state $s_L = (e_L, t_L, e_q, t_q, r_q)$ where $e_L = e_{gt}$ and $(e_q, r_q, e_{gt}, t_q) \in \mathcal{Q}$, and 0 otherwise. Formally, the binary global reward function is defined by:

$$R_{bin}(s_L) = \mathbb{I}(e_\ell == e_{gt}), \quad (2)$$

where $\mathbb{I}(.)$ is a function that return 1 or 0.

3.3 Tensor Decomposition Confidence-Guided Policy Network

The policy network is one of the main components of the reinforcement learning framework. A general policy network for KG reasoning consists of three main components: dynamic embedding, path encoding, and action scoring. In this work, we design a confidence-augmented based MLP (Multi-Layer Perceptron) and KAN (Kolmogorov-Arnold Networks) policy network which allow us to calculate the probability distribution over all the candidate actions $\mathcal{A}_{(\ell)}$ at step ℓ, concerning the current state $s^{(\ell)}$, search history $h_{(\ell)} = (e_q, t_q, r_1, (e_1, t_1), \dots, r_\ell, (e_\ell, t_\ell))$ and confidence probability $c_{a|q}$ for each $a \in \mathcal{A}_{(\ell)}$

Dynamic Embedding. Following TITer (Sun et al., 2021), one dense vector embedding $\mathbf{r} \in \mathbb{R}^{d_r}$ is assigned for a relation $r \in \mathcal{R}$. To capture the character of changing over the timeline of entities, a dynamic embedding is used to represent variant features for each node $e_i^t = (e_i, t) \in \mathcal{G}_t$, and a static embedding $\mathbf{e} \in \mathbb{R}^{d_e}$ is used to represent latent invariant features of these nodes. For encoding timestamp, a relative time encoding function $\Phi : \mathbb{R} \rightarrow \mathbb{R}^{d_t}$ is defined by:

$$\Phi(t_q - t) = \sigma(\mathbf{w}\Delta t + \mathbf{b}) = \Phi(\Delta t) \quad (3)$$

where $\mathbf{w}, \mathbf{b} \in \mathbb{R}^{d_t}$ are learnable parameter vectors, and $\sigma(.)$ is an activation function (such as $\sin(.), \cos(.)$ or sigmoid$(.)$). d_r, d_e, and d_t are the embedding dimensions for relation, entity, and timestamp. Finally, the final representation of a node e_i^t is defined by:

$$\mathbf{e}_i^t = [\mathbf{e}_i; \Phi(\Delta t)] \quad (4)$$

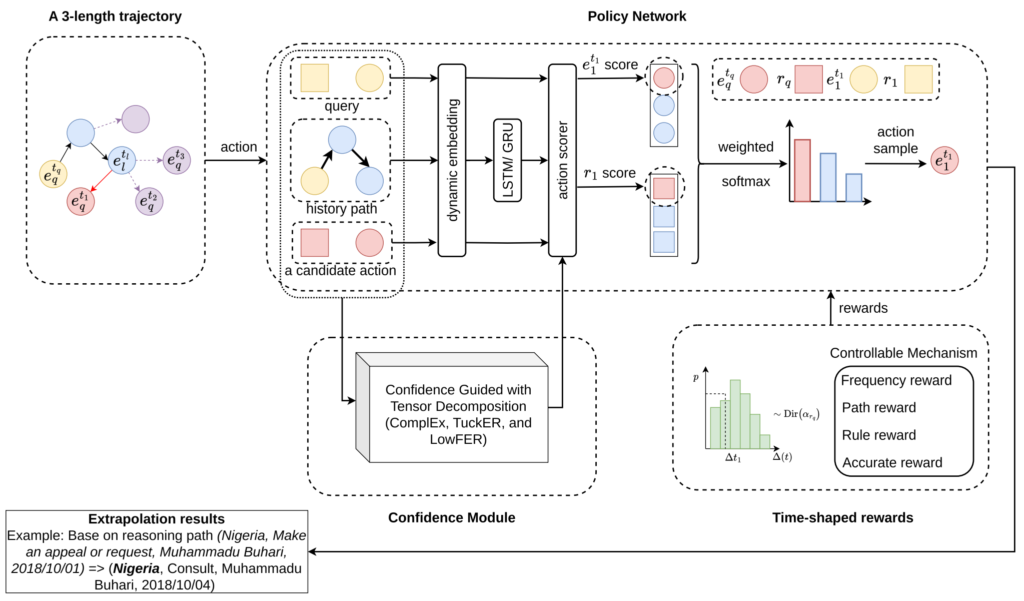

Figure 1: Overview of our proposed model CATTer. With the input $(e_q, r_q, ?\,(e_{gt}), t_q)$, model starts from node $e_q^{t_q}$ to search the answer. It samples an outgoing edge at each step and translates to a new node according to the results of the policy network. Suppose that $e_\ell^{t_\ell}$ is the current node. To compute the score for one candidate action (r_1, e_1, t_1), model sample actions based on the *joint probability* of *transition probability* calculated from all candidate scores and *confidence probability* obtained from a Tensor Decomposition such as ComplEx, TuckER, and LowFER. After that, the Dirichlet distribution-based time-shaped multi-reward function rewards the agent for its selected actions.

Historical Path Encoding. With the searching history $h_{(\ell)} = ((e_q, t_q), r_1, (e_1, t_1), \ldots, r_\ell, (e_\ell, t_\ell))$, we use two strategies to encode this sequence. First, the agent leverages an LSTM to encode this history sequence. This process is formulated as:

$$\begin{aligned} \mathbf{h}_\ell^{\text{lstm}} &= \text{LSTM}(\mathbf{h}_{l-1}, [\mathbf{r}_{l-1}; \mathbf{e}_{l-1}^{t_{l-1}}]), \\ \mathbf{h}_0^{\text{lstm}} &= \text{LSTM}(\mathbf{0}, [\mathbf{r}_0; \mathbf{e}_q^{t_q}]). \end{aligned} \tag{5}$$

In the second one, our agent leverages a GRU to encode this history sequence. This process is formulated as follows:

$$\begin{aligned} \mathbf{h}_\ell^{\text{gru}} &= \text{GRU}([\mathbf{r}_{l-1}; \mathbf{e}_{l-1}^{t_{l-1}}], \mathbf{h}_{l-1}), \\ \mathbf{h}_0^{\text{gru}} &= \text{GRU}([\mathbf{r}_0; \mathbf{e}_q^{t_q}, \mathbf{0}]). \end{aligned} \tag{6}$$

In the Eq. (5) and (6), $\mathbf{r}_0$ is dummy relation for initialization.

Confidence-Guided Multi-Layer Perceptron for Action Scoring. The action scoring module allows us to score each action and return the transition probability for the next state. We apply two strategies for designing this module: Multi-layer Perceptron (MLP) and Kolmogorov-Arnold Networks (KAN) with confidence rate augmentation via tensor decomposition sub-module. First, let $a_n = (e_n, t_n, r_n) \in \mathcal{A}_\ell$ denotes an action at step ℓ, the final candidate action score $\phi(a_n, s_\ell)$ can be formulated by:

$$\phi(a_n, s_\ell) = \beta_n \langle \widetilde{\mathbf{e}}, \mathbf{e}_n^{t_n} \rangle + (1 - \beta_n) \langle \widetilde{\mathbf{r}}, \mathbf{r}_n \rangle, \tag{7}$$

with

$$\begin{aligned} \widetilde{\mathbf{e}} &= \mathbf{W}_e \text{ReLU}(\mathbf{W}_1 [\mathbf{h}_\ell^{\text{lstm/gru}}; \mathbf{e}_q^{t_q}; \mathbf{r}_q]), \\ \widetilde{\mathbf{r}} &= \mathbf{W}_r \text{ReLU}(\mathbf{W}_1 [\mathbf{h}_\ell^{\text{lstm/gru}}; \mathbf{e}_q^{t_q}; \mathbf{r}_q]), \\ \beta_n &= \text{sigmoid}(\mathbf{W}_\beta [\mathbf{h}_\ell^{\text{lstm/gru}}; \mathbf{e}_q^{t_q}; \mathbf{r}_q; \mathbf{e}_n^{t_n}; \mathbf{r}_n]), \end{aligned}$$

where $\mathbf{W}_1$, $\mathbf{W}_e$, $\mathbf{W}_r$ and $\mathbf{W}_\beta$ are trainable parameters.

Then, we calculate the confidence rate $c_{a_n|q}$ of each $a_n \in \mathcal{A}_\ell$ via softmax function which receive the input vector from tensor decomposition such as TuckER (Balažević et al., 2019), ComplEx (Trouillon et al., 2016), and LowFER (Amin et al., 2020) as follow:

$$c_{a_n|q} = \frac{\exp(\psi_{a_n|q})}{\sum_{a'_\ell \in A_\ell} \exp(\psi_{a'_\ell|q})}, \tag{8}$$

where

$$\begin{aligned} \psi_{a_n|q} &= \mathcal{W} \times_1 \mathbf{e}_q^{t_q} \times_2 \mathbf{r}_q \times_3 \mathbf{e}_n^{t_n}, \text{ if use TuckER}, \\ \psi_{a_n|q} &= \text{Re}\left(\left\langle \mathbf{e}_q^{t_q}, \mathbf{r}_q, \overline{\mathbf{e}_n^{t_n}} \right\rangle\right) \text{ if use ComplEx}, \\ \psi_{a_n|q} &= (\mathbf{S}^k \text{diag}(\mathbf{U}^\top \mathbf{e}_q^{t_q}) \mathbf{V}^\top \mathbf{r}_q)^\top \mathbf{e}_n^{t_n}, \text{if use LowFER}, \end{aligned}$$

with $\mathcal{W} \in \mathbb{R}^{2d_e \times d_e \times d_e}$ is a learnable core tensor introduced in (Balažević et al., 2019), $\times_1, \times_2$, and $\times_3$ are tensor product in three different modes (see (Balažević et al., 2019; Tucker et al., 1964) for more details), Re(.) returns the real vector component for input embedding, $\mathbf{U} \in \mathbb{R}^{d_e \times kd_e}$, $\mathbf{V} \in \mathbb{R}^{d_r \times kd_e}$, $\operatorname{diag}(\mathbf{U}^\top \mathbf{e}_q^{t_q}) \in \mathbb{R}^{kd_e \times kd_e}$ and $\mathbf{S}^k \in \mathbb{R}^{d_e \times kd_e}$ is constant matrix which defined as

$$\mathbf{S}^k_{i,j} = \begin{cases} 1, & \forall j \in [(i-1)k+1, ik] \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

After scoring all candidate actions in $\mathcal{A}_\ell$ and calculating the confidence rate for all of them, the policy $\pi_\theta(a_\ell \mid s_\ell)$ at step ℓ is defined as:

$$\pi_\theta(a_\ell \mid s_\ell) = \frac{\exp(\phi(a_\ell, s_\ell) * c_{a_\ell|q})}{\sum_{a'_\ell \in \mathcal{A}_\ell} \exp(\phi(a'_\ell, s_\ell) * c_{a'_\ell|q})} \tag{10}$$

Confidence-Guided Kolmogorov-Arnold Networks for Action Scoring. Based on the Kolmogorov-Arnold representation (KAR) theorem, with a given smooth function $f : [0,1]^n \to \mathbb{R}$,

$$f(x) = f(x_1, ..., x_n) = \sum_{q=1}^{2n+1} \mathbf{W}_q \Big(\sum_{p=1}^{n} W_{q,p}(x_p) \Big) \tag{11}$$

where $W_{q,p} : [0,1] \to \mathbb{R}$ and $\mathbf{W}_q : \mathbb{R} \to \mathbb{R}$.

Due to the limitation of expressiveness of the KAR Theorem, (Liu et al., 2024) design techniques to generalize this for arbitrary depths and widths. Formally, KAR can be written in matrix form as

$$f(x) = \mathbf{W}^{\text{kan}}_{\text{out}} \circ \mathbf{W}^{\text{kan}}_{\text{in}} \circ \mathbf{x}, \tag{12}$$

where

$$\mathbf{W}^{\text{kan}}_{\text{in}} = \begin{pmatrix} w_{1,1}(\cdot) & \cdots & w_{1,n}(\cdot) \\ \vdots & & \vdots \\ w_{2n+1,1}(\cdot) & \cdots & w_{2n+1,n}(\cdot) \end{pmatrix},$$

$$\mathbf{W}^{\text{kan}}_{\text{out}} = \begin{pmatrix} W_1(\cdot) & \cdots & W_{2n+1}(\cdot) \end{pmatrix}$$

Then, a Kolmogorov-Arnold layer is defined as:

$$\mathbf{W}^{\text{kan}} = \begin{pmatrix} w_{1,1}(\cdot) & \cdots & w_{1,n_{\text{in}}}(\cdot) \\ \vdots & & \vdots \\ w_{n_{\text{out}},1}(\cdot) & \cdots & w_{n_{\text{out}},n_{\text{in}}}(\cdot) \end{pmatrix}, \tag{13}$$

where $\mathbf{W}^{\text{kan}}_{\text{in}}$ corresponds to $n_{\text{in}} = n, n_{\text{out}} = 2n+1$, and $\mathbf{W}^{\text{kan}}_{\text{out}}$ corresponds to $n_{\text{in}} = 2n+1, n_{\text{out}} = 1$.

After defining the layer, we can construct a Kolmogorov-Arnold network for action scoring as:

$$\phi(a_n, s_\ell) = \beta_n \langle \widetilde{\mathbf{e}}, \mathbf{e}_n^{t_n} \rangle + (1-\beta_n) \langle \widetilde{\mathbf{r}}, \mathbf{r}_n \rangle, \tag{14}$$

with

$$\widetilde{\mathbf{e}} = \mathbf{W}^{\text{kan}}_e (\mathbf{W}^{\text{kan}}_1 [\mathbf{h}_\ell^{\text{lstm/gru}}; \mathbf{e}_q^{t_q}; \mathbf{r}_q]),$$

$$\widetilde{\mathbf{r}} = \mathbf{W}^{\text{kan}}_r (\mathbf{W}^{\text{kan}}_1 [\mathbf{h}_\ell^{\text{lstm/gru}}; \mathbf{e}_q^{t_q}; \mathbf{r}_q]),$$

$$\beta_n = \operatorname{sigmoid}(\mathbf{W}^{\text{kan}}_\beta [\mathbf{h}_l^{\text{lstm/gru}}; \mathbf{e}_q^{t_q}; \mathbf{r}_q; \mathbf{e}_n^{t_n}; \mathbf{r}_n]),$$

And then, by applying confidence techniques, we obtain the policy $\pi_\theta(a_\ell \mid s_\ell)$ at step ℓ via the softmax function.

3.4 Multi-Reward Mechanism with Rule Enhancing

To obtain more flexible reward functions, we adopt multi-type rewards, including binary global, adjusted ground truth frequency, adjusted path length, and high-frequency rule reward, into a weighted fusion scheme.

Binary Global Reward. Following the original RL framework, which is introduced in Section 3.2, we formulate a *binary global reward* that is defined by:

$$\mathrm{R}_{bin}(s_L) = \mathbb{I}(e_\ell == e_{gt}). \tag{15}$$

Adjusted Ground Truth Frequency Reward. Inspired by MPNet (Wang et al., 2024), we introduce a more flexible frequency reward, named *adjusted ground truth frequency reward.* With given (e_q, r_q, e_{gt}, t_q), $N_{\text{gt}} = \{n_1, n_2, \ldots, n_m\}$ denote the number of times that the e_{gt} occur in m snapshot $\{G_{t_q-1}, G_{t_q-2}, \ldots, G_{t_q-m}\}$, i.e., $n_i, (i = 1, \ldots, m)$ is the number of times that e_{gt} occurs in subgraph G_{t_q-i}. We define the ground truth frequency reward as follows:

$$\mathrm{R}_{\text{gt}}(s_L) = \begin{cases} f_i, & \text{if } t_{q-m} \le t_i \le t_q, \\ 0, & \end{cases} \tag{16}$$

where

$$f_i = \frac{n_i}{\max(N_{\text{gt}}) - \min(N_{\text{gt}})}.$$

Adjusted Path Length Reward. Following MPNet (Wang et al., 2024), we introduce a more flexible path length reward, named *adjusted path length reward* which can be defined as:

$$\mathrm{R}_{\text{path}}(s_L) = \frac{w_{\text{path}}}{p_\ell - 1} \tag{17}$$

where $p_\ell \le p_{\max}$ denotes the length of the path taken by the agent to capture the target entity from the source node at step ℓ, $p_{\max}$ is the maximum path length which agent can reach a node, and $w_{\text{path}} \in (0,1)$ is the weight for current path length which is taken.

High-Frequency Rule Reward. Knowledge graphs usually contain a pair entity relation, frequently appearing in the timelines. Formally, given a common pair entity-relation set, which is denoted as $\mathrm{ER} = \{(e_i, r_i)\}_{i=1}^k$. Each pair in ER has a frequency of occurrence greater than or equal to a threshold ϑ depending on the dataset. Then, we define a *high-frequency rule reward* for our agent as follows:

$$\mathrm{R}_{\text{rule}}(s_L) = \begin{cases} w_{\text{rule}}, & \text{if } (e_\ell, r_\ell) \in \mathrm{ER}, \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

where w_{rule} is reward value for matching rule.

Multi-Reward Fusion. After defining multi-reward, we put them into the primary reward function with $\alpha_1 \in (0,1)$, $\alpha_2 \in (0,1)$, and $\alpha_3 \in (0,1)$ are weights for binary reward (Section 3.4), Adjusted ground truth frequency reward (Section 3.4), Adjusted path length reward (Section 3.4), and high-frequency rule reward (Section 3.4), respectively, as follow:

$$R = (1+\alpha_1 R_{\text{gt}})(1+\alpha_2 R_{\text{rule}})(R_{\text{bin}}+\alpha_3 R_{\text{path}}) \quad (19)$$

3.5 Agent Parameter Learning

Following TITer (Sun et al., 2021), the search path length is fixed to a length of L. Then, the policy network π_θ generates a L-length trajectory as $\{a_1, a_2, ..., a_L\}$. The training objective is maximizing the expected multi-reward of the agent overall training set Q_{train}:

$$J(\theta) = \mathbb{E}_{(e_s,r,e_o,t)\sim Q_{train}}[\mathbb{E}_{a_1,...,a_L\sim\pi_\theta} [\widetilde{R}(s_L|e_s,r,t)]]. \quad (20)$$

where $\widetilde{R}(s_L) = (1 + p_{\Delta t_L} R(s_L)$, $\Delta t_L = t_q - t_L$, $(p_1,\ldots,p_k) \sim \text{Dirichlet}(\alpha_{r_q})$ with $\alpha_{r_q} \in \mathbb{R}^K$ is the Dirichlet distribution (Johnson et al., 1972; Ng et al., 2011) parameters vector of relation r_q.

Then, a policy gradient method is applied to optimize the policy network. In this work, we apply REINFORCE algorithm (Williams, 1992) that will iterate through all quadruple in Q_{train} and update θ with the following stochastic gradient method such as SGD (Ruder, 2016), Adam (Ruder, 2016; Kingma, 2014) or AdaGrad (Duchi et al., 2011):

$$\nabla_\theta J(\theta) \approx \nabla_\theta \sum_{m\in[1,L]} \widetilde{R}(s_L|e_s,r,t) \log \pi_\theta(a_\ell|s_\ell) \quad (21)$$

4 EXPERIMENTS AND RESULTS

4.1 Experiment Setting

Standard Benchmark Datasets. During the experiment process, we use four common TKG datasets for evaluation, including ICEWS14, ICEWS18 (Boschee et al., 2015), WIKI (Leblay and Chekol, 2018b), and YAGO (Mahdisoltani et al., 2015).

(i) ICEWS14 and ICEWS18 are extracted from Integrated Crisis Early Warning System (ICEWS) (Boschee et al., 2015). These two datasets contain real-world facts from 2014 and 2018 with time granularity at the day level.

(ii) WIKI (Leblay and Chekol, 2018a) and YAGO (Mahdisoltani et al., 2013) are two KGs that contain real-world facts with time information. Following previous work (Sun et al., 2021), these two datasets are used with time granularity at year level. for performing the evaluation.

We meticulously adopt a train-test split strategy for the training and testing stage, as detailed in (Sun et al., 2021; Jin et al., 2020). This strategy involves splitting the dataset into three subsets, including train, validation, and test set in a specific order of timestamp, ensuring a comprehensive evaluation. Table 1 summarizes the statistical information about these four datasets.

Evaluation Protocol and Metrics. We compare our proposed model to the problem of predicting missing events at future timestamps. In a TKG, the number of relations is significantly smaller than the number of entities, making entity prediction more challenging than relation prediction. Consequently, TKG tasks often focus on predicting missing entities. Given a KG dataset, we address two types of entity prediction: $(e_s, r, ?, t)$ and $(?, r, e_o, t)$, where ? represents the missing entity. To enhance the evaluation consistency, we apply a time-aware filtering (Han et al., 2020) which is the same as TITer (Sun et al., 2021), which filters only those quadruples that match the query time t.

After ranking all the candidates according to their scores calculated by beam search according to the joint probability of transition probability and confidence probability, if the ground truth entity does not appear, the rank is set as the number of entities in the dataset. Then, we employ two metrics widely used in TKG research: Mean Reciprocal Rank (MRR) and Hits@k, where the higher MRR and Hits@k indicate better performance.

Baselines. We compare our model with the existing state-of-the-art KG reasoning model:

(i) Interpolation-based models: TTransE (Leblay and Chekol, 2018a), TA-DistMult (García-Durán et al., 2018), DE-SimplE (Goel et al., 2020), and TNTComplEx (Lacroix et al., 2020).

(ii) Extrapolation-based models: RE-NET (Jin et al., 2020), CyGNet (Zhu et al., 2021b), TANGO (Ding et al., 2021), xERTE (Han et al., 2021a), and TITer (Sun et al., 2021).

Implementation Details. Our proposed model is implemented based on TITer. The official source code

Table 1: Statistics information on benchmark datasets.

Dataset	#train	#valid	#test	#ent	#rel	Time granularity
ICEWS14 (Boschee et al., 2015)	63685	13823	13222	7128	230	24 hours
ICEWS18 (Boschee et al., 2015)	373018	45995	49545	23033	256	24 hours
WIKI (Leblay and Chekol, 2018a)	539286	67538	63110	12554	24	1 year
YAGO (Mahdisoltani et al., 2013)	161540	19523	20026	10623	10	1 year

of this model can be found at https://github.com/JHL-HUST/TITer. By default, we set the entity embedding, relation embedding, and relative time encoding dimension to 80, 100, and 20, respectively. For training and testing, we use the same setting as TITer. For the confidence module, we use $k = 30$ if the module is in LowFER mode and a dropout rate of 0.2 if the module is in TuckER mode. For reward fusion, we search suitable for α_1, α_2 and α_3 in range $[0, 1]$ and use rule weight $w_{\text{rule}} \in [0.01, 0.5]$. Our source code is available at https://github.com/lnhutnam/CATTer.

4.2 Results and Discussion

Performance and Efficiency Comparison. The experimental results evaluating the link prediction performance of the proposed model compared to baseline models are presented in Table 2. Overall, our proposed model shows significant improvements in performance compared to baseline models such as TNTComplEx and xERTE. For comparison with the TITer model, we re-ran the experiments using the same hardware setup as our proposed model. The reported results in the tables include those from the original papers and the re-experimented results. In our comparison, we focus on the re-experimented results.

For the ICEWS14 and ICEWS18 datasets, CATTer shows significant performance improvements compared to models like TTransE, TA-DistMult, DE-SimplE, and TNTComplEx, thanks to its high adaptability, which allows it to handle unknown entities in the data. Compared to other extrapolation models like RE-NET, xERTE, and TANGO, CATTer also demonstrates improvements in MRR and Hit@k. In comparison to TITer, the performance evaluation with MLP for the policy network shows improved efficiency over other methods.

For the YAGO and WIKI datasets, CATTer also demonstrates significant improvements compared to models like RE-NET and CyGNet. When compared to the baseline TITer model, the policy network with MLP continues to show notable performance across all metrics. There are two primary reasons for this phenomenon observed in the experiments on these datasets: 1) The spline approximation of KAN is insufficient to handle the complex characteristics of the environment, and 2) KAN is not fully stable during the learning process (based on the convergence analysis in Figure 2 and Figure 3).

Table 3 compares the number of trainable parameters between the proposed and baseline models. The computational cost is also assessed using the MACs (Multiply-Accumulate Operations) metric, representing the number of MACs. Based on the evaluation results, our proposed model maintains computational efficiency, requiring fewer trainable parameters and reduced operations while preserving performance.

Convergence Study. To evaluate the convergence speed of the proposed model, we assess the loss function values and the accumulated reward values of the agent over each training epoch. The provided results are illustrated in Figure 2 and Figure 3.

As shown in Figure 2, on the ICEWS14 and YAGO datasets, the fluctuation in the loss function remains relatively small for both the MLP and KAN networks, with the error levels being comparable across both methods. In contrast, the fluctuation amplitude in the loss function for the ICEWS18 and WIKI datasets is more significant, indicating instability during training on these datasets for both the KAN and MLP networks.

Regarding Figure 3, it is evident that for the YAGO and WIKI datasets, the model's performance on both KAN and MLP networks converges rapidly within approximately 100 epochs. On the other hand, for the ICEWS14 and ICEWS18 datasets, the model performance significantly improves after around 100 epochs and continues to increase slowly. Although the evaluation was conducted for 400 epochs, we believe the model's performance on these datasets has not yet fully converged and could improve further with additional training.

The Effect of LSTM and GRU for Historical Path Encoding. To assess the impact of deep learning techniques, specifically sequence-based architectures such as LSTM and GRU, on history path encoding, we evaluated their performance using the MRR metric across multiple datasets. The provided experimental results are illustrated in Figure 4.

Table 2: Performance comparison results on future link forecasting on ICEWS14 and ICEWS18. These results of MRR and Hits@1/3/10 are multiplied by 100. * denotes the re-experiment result on the same hardware with the proposed model. APG and RPG are calculated by $APG = R_{ours} - R_{baseline}$ and $RPG = (R_{ours} - R_{baseline})/R_{baseline}$ where R_{ours} and $R_{baseline}$ are the results of our models and baseline TITer*, respectively.

Method	**ICEWS14**				**ICEWS18**			
	MRR ↑	Hit@1 ↑	Hit@3 ↑	Hit@10	MRR ↑	Hit@1 ↑	Hit@3 ↑	Hit@10 ↑
TTransE	13.43	3.11	17.32	34.55	8.31	1.92	8.56	21.89
TA-DistMult	26.47	17.09	30.22	45.41	16.75	8.61	18.41	33.59
DE-SimplE	32.67	24.43	35.69	49.11	19.30	11.53	21.86	34.80
TNTComplEx	32.12	23.35	36.03	49.13	27.54	19.52	30.80	42.86
CyGNet	32.73	23.69	36.31	50.67	24.93	15.90	28.28	42.61
RE-NET	38.28	28.68	41.34	54.52	28.81	19.05	32.44	**47.51**
xERTE	40.79	32.70	45.67	57.30	29.31	21.03	**33.51**	46.48
TANGO-Tucker	–	–	–	–	28.68	19.35	32.17	47.04
TANGO-DistMult	–	–	–	–	26.75	17.92	30.08	44.09
TITer	**41.73**	**32.74**	**46.46**	**58.44**	**29.98**	**22.05**	33.46	44.83
TITer*	40.33	31.00	45.30	57.71	29.42	21.63	32.83	43.96
CATTer-MLP	41.21	32.47	45.75	57.37	29.54	21.60	32.99	44.51
CATTer-KAN	40.13	31.04	44.80	57.19	29.11	21.37	32.46	43.60
APG (%) ↑ (MLP)	**0.62**	**0.54**	**0.61**	**0.64**	**-0.22**	**-0.39**	**-0.02**	**0.03**
RPG (%) ↑ (MLP)	**2.18**	**4.74**	**0.99**	**-0.59**	**0.41**	**-0.14**	**0.49**	**1.25**
APG (%) ↑ (KAN)	**0.65**	**0.48**	**0.77**	**0.92**	**-0.61**	**-0.68**	**-0.54**	**-0.53**
RPG (%) ↑ (KAN)	**-0.49**	**0.13**	**-1.10**	**-0.90**	**-1.05**	**-1.20**	**-1.13**	**-0.82**

Method	**WIKI**				**YAGO**			
	MRR ↑	Hit@1 ↑	Hit@3 ↑	Hit@10	MRR ↑	Hit@1 ↑	Hit@3 ↑	Hit@10 ↑
TTransE	29.27	21.67	34.43	42.39	31.19	18.12	40.91	51.21
TA-DistMult	44.53	39.92	48.73	51.71	54.92	48.15	59.61	66.71
DE-SimplE	45.43	42.6	47.71	49.55	54.91	51.64	57.30	60.17
TNTComplEx	45.03	40.04	49.31	52.03	57.98	52.92	61.33	66.69
CyGNet	33.89	29.06	36.10	41.86	52.07	45.36	56.12	63.77
RE-NET	49.66	46.88	51.19	53.48	58.02	53.06	61.08	66.29
xERTE	71.14	68.05	76.11	79.01	84.19	80.09	88.02	89.78
TANGO-Tucker	50.43	48.52	51.47	53.58	57.83	53.05	60.78	65.85
TANGO-DistMult	51.15	49.66	52.16	53.35	62.70	59.18	60.31	67.90
TITer	**75.50**	**72.96**	**77.49**	**79.02**	87.47	84.89	**89.96**	90.27
TITer*	73.56	71.48	74.86	76.40	87.80	85.52	89.92	90.31
CATTer-MLP	74.18	72.02	75.47	77.04	**87.58**	**85.13**	89.90	**90.34**
CATTer-KAN	74.21	71.96	75.63	77.32	87.19	84.84	89.38	89.78
APG (%) ↑ (MLP)	**0.88**	**1.47**	**0.45**	**-0.34**	**0.12**	**-0.03**	**0.16**	**0.55**
RPG (%) ↑ (MLP)	**0.84**	**0.76**	**0.81**	**0.83**	**-0.25**	**-0.46**	**-0.02**	**0.03**
APG (%) ↑ (KAN)	**-0.2**	**0.04**	**-0.5**	**-0.52**	**-0.31**	**-0.26**	**-0.37**	**-0.36**
RPG (%) ↑ (KAN)	**0.88**	**0.67**	**1.03**	**1.20**	**-0.69**	**-0.80**	**-0.60**	**-0.59**

Based on the results, we observe that the performance gap between LSTM and GRU is not significant. LSTM shows a slight advantage on the ICEWS14 and ICEWS18 datasets, with the difference being more pronounced on the WIKI dataset. This observation argues that LSTM may be more effective in tasks with more complex data. However, on the YAGO dataset, GRU marginally outperforms LSTM. This indicates that GRU may be more suitable for tasks with smaller and simpler data. Overall, the performance difference between the two models remains relatively small, suggesting that both LSTM and GRU are viable options for history path encoding.

The Effect of MLP and KAN for Policy Networks. The MLP and KAN models were employed in designing policy networks, a critical component of the RL framework for link prediction on knowledge graphs. We conducted experiments to evaluate the impact of these two models on the MRR metric across different evaluation datasets. The experimental re-

Table 3: Number of trainable parameters and calculation of our proposed models and baselines. MACs stand for Multi-Adds operations, and M stand for million.

Method	# Params	# MACs
RE-NET	5.459M	4.370M
CyGNet	8.568M	8.554M
xERTE	2.927M	225.895M
TITer	1.455M	0.225M
CATTer	1.425M	0.220M

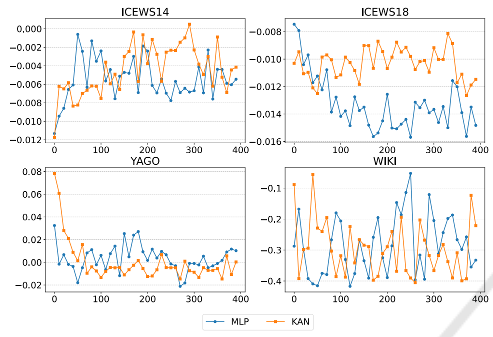

Figure 2: The change of the loss function over each epoch with MLP and KAN Policy Network.

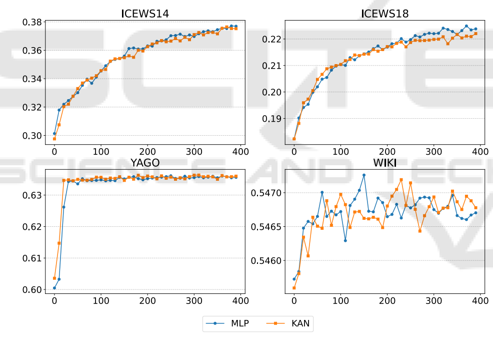

Figure 3: The change of the multi-reward function over each epoch with MLP and KAN Policy Network.

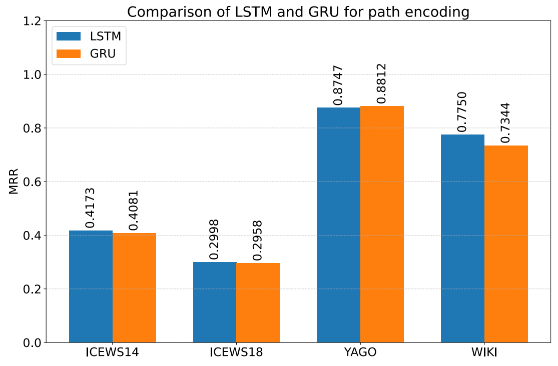

Figure 4: The effect of LSTM and GRU for path encoding on ICEWS14, ICEWS18, YAGO and WIKI dataset in term of MRR.

sults are illustrated in Figure 5. The results indicate that the MLP model performs slightly better on the ICEWS14, ICEWS18, and YAGO datasets. However, the KAN network achieves better training results for the WIKI dataset than the MLP network.

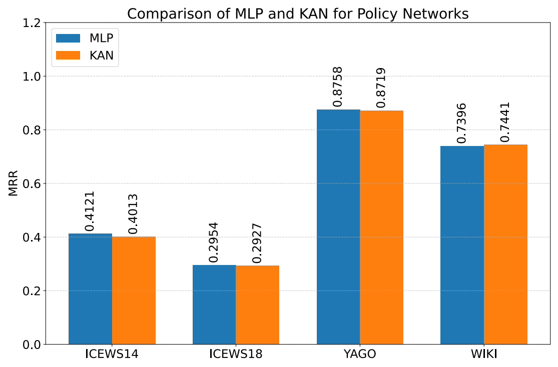

Figure 5: The effect of MLP and KAN for Policy Networks on ICEWS14, ICEWS18, YAGO and WIKI dataset in term of MRR.

The Effect of Tensor Decomposition Methods for Action Confidence Generation. The action confidence generation module is designed based on tensor decomposition models. Recent studies have demonstrated the significantly improved performance of these models in the link prediction task. We experimented with three tensor decomposition models to evaluate their ability to enhance action selectivity: TuckER, ComplEx, and LowFER. The experimental results are illustrated in Figure 6 with MLP-Policy Networks and Figure 7 with KAN-Policy Networks. Overall, using LowFER to generate confidence probabilities for action selection had a positive impact compared to ComplEx and TuckER. LowFER generalizes TuckER and is better able to fuse information between entities and relations than ComplEx. As a result, the probabilities generated by this module led to significantly improved performance.

4.3 Ablation Study

In this section, we perform some ablation experiments to evaluate the impact of modules such as action confidence generation, multi-rewards, and multi-reward reshaping on the agent's learning performance.

The Effect of Using Action Confidence Generation. To evaluate the role of the action confidence generation module, we conducted an experiment comparing two scenarios: with and without using this module. The experimental results are shown in Figure 8. As demonstrated in Figure 8, the action confidence gen-

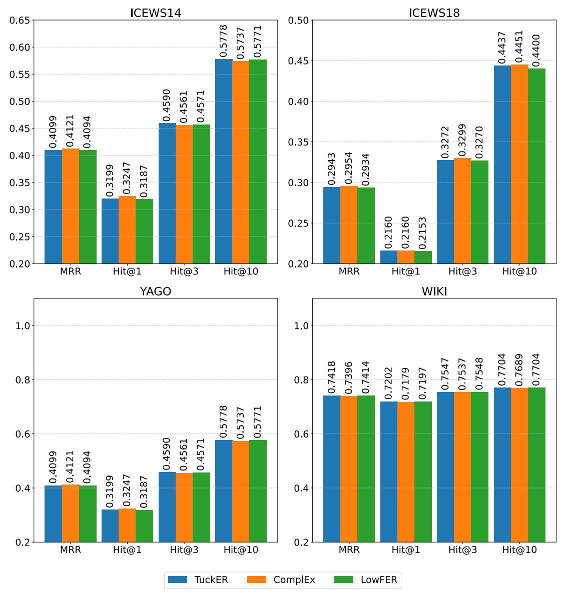

Figure 6: The effect of different tensor decomposition methods with MLP-Policy Network for action confidence generation on ICEWS14, ICEWS18, YAGO, and WIKI dataset.

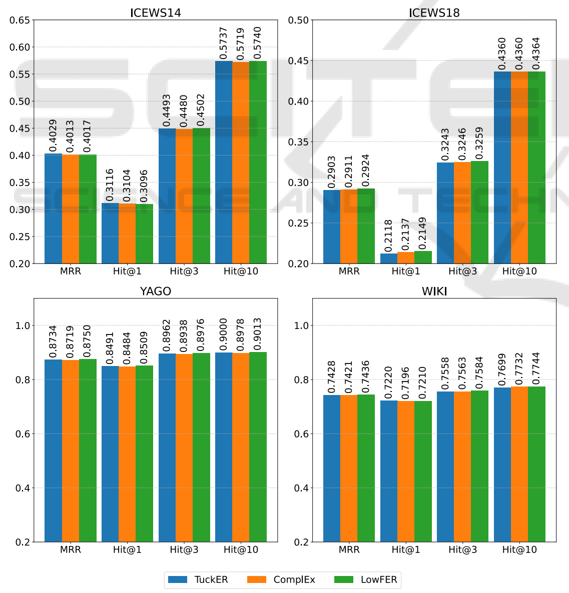

Figure 7: The effect of different tensor decomposition methods with KAN-Policy Network for action confidence generation on ICEWS14, ICEWS18, YAGO, and WIKI dataset.

eration module was effective across all four evaluation datasets. By incorporating the confidence rate into the action selection and previous steps for analysis, the agent can identify better and more reliable actions to interact with the environment. Thus, it ultimately enhances the reasoning ability of an agent.

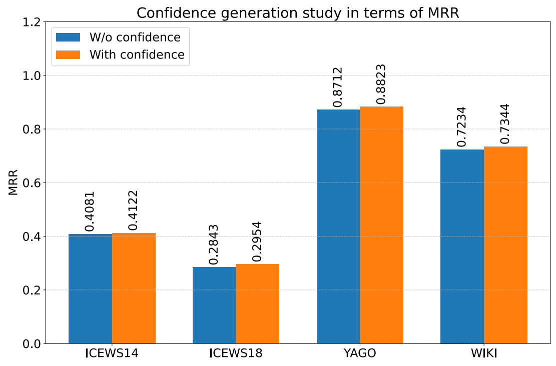

Figure 8: The effect of using action confidence for agent learning on ICEWS14, ICEWS18, YAGO and WIKI dataset.

The Effect of Multi-Rewards. The multi-reward function is a crucial component of the RL framework. In this ablation study, we examine the impact of the multi-reward function on the agent's learning process by considering two scenarios: 1) using only a binary reward function, and 2) employing the multi-reward function with the criteria introduced in Section 3. The experimental results are illustrated in Figure 9. We observe that when using the standard binary reward function, the performance difference compared to the multi-reward function is insignificant for the ICEWS14 and ICEWS18 datasets. However, providing additional rewards for the YAGO and WIKI datasets allows the agent to gain a more comprehensive understanding of the learning environment and further optimize its strategy.

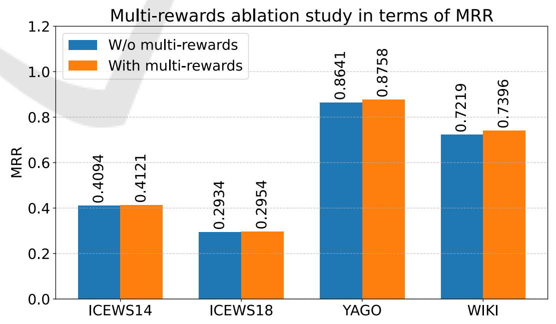

Figure 9: The effect of multi-reward mechanism for agent learning on ICEWS14, ICEWS18, YAGO and WIKI dataset.

The Effect of Multi-Reward Reshaping. Based on TITer (Sun et al., 2021), we applied a strategy to reshape the initial distribution of the multi-reward using Dirichlet distributions. To assess the effectiveness of this strategy, we conducted experiments in two scenarios: 1) using multi-reward reshaping and 2) not using reshaping for the original multi-reward distribution. The experimental results are visualized in Figure 10. The results indicate that reshaping the distribution yields improvements across most experimental datasets. This demonstrates that distribution reshaping enables the agent to receive better rewards within the multi-reward module, allowing it to make more informed decisions in complex environments.

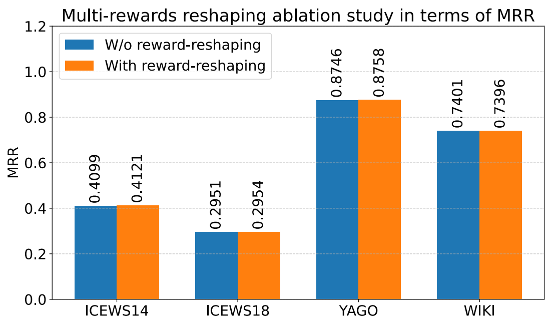

Figure 10: The effect of multi-reward reshaping for agent learning on ICEWS14, ICEWS18, YAGO and WIKI dataset.

5 CONCLUSION

In this work, we propose strategies for a new model to improve the temporal-path-based reinforcement learning model based on the TimeTraveler framework, namely CATTer. These strategies include employing GRU to encode historical paths; integrating confidence probability into MLP and KAN layers, thereby designing a more flexible Policy Network capable of selecting appropriate actions for the agent during learning; and utilizing a multi-reward function with various reward criteria to enhance the agent's adaptability in Temporal Knowledge Graphs (TKG) environment. The experimental results demonstrate that these enhancements positively impact the model's performance in future link prediction. Looking ahead, we plan to incorporate informative sub-graph patterns and temporal rules into the model to further enhance its link prediction capabilities.

ACKNOWLEDGEMENTS

This research is partially funded by the Vingroup Innovation Foundation (VINIF) under the grant number VINIF.2021.JM01.N2

REFERENCES

Amin, S., Varanasi, S., Dunfield, K. A., and Neumann, G. (2020). Lowfer: Low-rank bilinear pooling for link prediction. In *International Conference on Machine Learning*, pages 257–268. PMLR.

Bai, L., Chai, D., and Zhu, L. (2023). Rlat: Multi-hop temporal knowledge graph reasoning based on reinforcement learning and attention mechanism. *Knowledge-Based Systems*, 269:110514.

Bai, L., Yu, W., Chen, M., and Ma, X. (2021). Multi-hop reasoning over paths in temporal knowledge graphs using reinforcement learning. *Applied Soft Computing*, 103:107144.

Balažević, I., Allen, C., and Hospedales, T. M. (2019). Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Boschee, E., Lautenschlager, J., O'Brien, S., Shellman, S., Starz, J., and Ward, M. (2015). ICEWS Coded Event Data.

Chen, W., Wan, H., Guo, S., Huang, H., Zheng, S., Li, J., Lin, S., and Lin, Y. (2022). Building and exploiting spatial–temporal knowledge graph for next poi recommendation. *Knowledge-Based Systems*, 258:109951.

Das, R., Dhuliawala, S., Zaheer, M., Vilnis, L., Durugkar, I., Krishnamurthy, A., Smola, A., and McCallum, A. (2017). Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*.

Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S. (2018). Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*.

Ding, Z., Han, Z., Ma, Y., and Tresp, V. (2021). Temporal knowledge graph forecasting with neural ode. *arXiv preprint arXiv:2101.05151*.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

García-Durán, A., Dumancic, S., and Niepert, M. (2018). Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4816–4821.

Goel, R., Kazemi, S. M., Brubaker, M., and Poupart, P. (2020). Diachronic embedding for temporal knowledge graph completion. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 3988–3995.

Han, Z., Chen, P., Ma, Y., and Tresp, V. (2021a). Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *International Conference on Learning Representations*.

Han, Z., Ding, Z., Ma, Y., Gu, Y., and Tresp, V. (2021b). Temporal knowledge graph forecasting with neural ode. *arXiv preprint arXiv:2101.05151*.

Han, Z., Ma, Y., Wang, Y., Günnemann, S., and Tresp, V. (2020). Graph hawkes neural network for forecasting on temporal knowledge graphs. In *Conference on Automated Knowledge Base Construction*.

Jin, W., Qu, M., Jin, X., and Ren, X. (2019). Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. *arXiv preprint arXiv:1904.05530*.

Jin, W., Qu, M., Jin, X., and Ren, X. (2020). Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6669–6683.

Johnson, N. L., Kotz, S., and Balakrishnan, N. (1972). *Continuous multivariate distributions*, volume 7. Wiley New York.

Kingma, D. P. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lacroix, T., Obozinski, G., and Usunier, N. (2020). Tensor decompositions for temporal knowledge base completion. In *International Conference on Learning Representations*.

Leblay, J. and Chekol, M. W. (2018a). Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference 2018*, pages 1771–1776.

Leblay, J. and Chekol, M. W. (2018b). Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference*, pages 1771–1776.

Lin, X. V., Socher, R., and Xiong, C. (2018). Multi-hop knowledge graph reasoning with reward shaping. *arXiv preprint arXiv:1808.10568*.

Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., Hou, T. Y., and Tegmark, M. (2024). Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*.

Mahdisoltani, F., Biega, J., and Suchanek, F. M. (2013). Yago3: A knowledge base from multilingual wikipedias. In *CIDR*.

Mahdisoltani, F., Biega, J., and Suchanek, F. M. (2015). YAGO3: A knowledge base from multilingual wikipedias. In *Seventh Biennial Conference on Innovative Data Systems Research*.

Mavromatis, C., Subramanyam, P. L., Ioannidis, V. N., Adeshina, A., Howard, P. R., Grinberg, T., Hakim, N., and Karypis, G. (2022). Tempoqr: temporal question reasoning over knowledge graphs. In *Proceedings of the AAAI conference on artificial intelligence*, pages 5825–5833.

Ng, K. W., Tian, G.-L., and Tang, M.-L. (2011). Dirichlet and related distributions: Theory, methods and applications.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. (2018). Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*, pages 593–607. Springer.

Sun, H., Zhong, J., Ma, Y., Han, Z., and He, K. (2021). Timetraveler: Reinforcement learning for temporal knowledge graph forecasting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8306–8319.

Tao, Y., Li, Y., and Wu, Z. (2021). Temporal link prediction via reinforcement learning. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3470–3474. IEEE.

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. (2016). Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR.

Tucker, L. R. et al. (1964). The extension of factor analysis to three-dimensional matrices. *Contributions to mathematical psychology*, 110119:110–182.

Wang, J., Wu, R., Wu, Y., Zhang, F., Zhang, S., and Guo, K. (2024). Mpnet: temporal knowledge graph completion based on a multi-policy network. *Applied Intelligence*, 54(3):2491–2507.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Xiong, W., Hoang, T., and Wang, W. Y. (2017). Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*.

Zheng, S., Yin, H., Chen, T., Nguyen, Q. V. H., Chen, W., and Zhao, L. (2023). Dream: Adaptive reinforcement learning based on attention mechanism for temporal knowledge graph reasoning. *arXiv preprint arXiv:2304.03984*.

Zhu, C., Chen, M., Fan, C., Cheng, G., and Zhang, Y. (2021a). Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *Proceedings of the AAAI conference on artificial intelligence*, pages 4732–4740.

Zhu, C., Chen, M., Fan, C., Cheng, G., and Zhang, Y. (2021b). Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 4732–4740.