DANS-KGC: Diffusion Based Adaptive Negative Sampling for Knowledge Graph Completion

Haoning Li, Qinghua Huang*

School of Artificial Intelligence, OPtics and ElectroNics (iOPEN), Northwestern Polytechnical University
1134067491@qq.com, qhhuang@nwpu.edu.cn

Abstract

Negative sampling (NS) strategies play a crucial role in knowledge graph representation. In order to overcome the limitations of existing negative sampling strategies, such as vulnerability to false negatives, limited generalization, and lack of control over sample hardness, we propose DANS-KGC (Diffusion-based Adaptive Negative Sampling for Knowledge Graph Completion). DANS-KGC comprises three key components: the Difficulty Assessment Module (DAM), the Adaptive Negative Sampling Module (ANS), and the Dynamic Training Mechanism (DTM). DAM evaluates the learning difficulty of entities by integrating semantic and structural features. Based on this assessment, ANS employs a conditional diffusion model with difficulty-aware noise scheduling, leveraging semantic and neighborhood information during the denoising phase to generate negative samples of diverse hardness. DTM further enhances learning by dynamically adjusting the hardness distribution of negative samples throughout training, enabling a curriculum-style progression from easy to hard examples. Extensive experiments on six benchmark datasets demonstrate the effectiveness and generalization ability of DANS-KGC, with the method achieving state-of-the-art results on all three evaluation metrics for the UMLS and YAGO3-10 datasets.

Introduction

Knowledge graphs (KGs) encode real-world facts as triples $\langle h, r, t \rangle$ in a multi-relational graph, serving as a key foundation for recommender systems, search, and QA (Hogan et al. 2021). Yet data limitations and continual change leave many entities or relations missing. Knowledge-graph completion (KGC) therefore seeks to infer hidden semantic links and restore KG integrity (Liang et al. 2024).

After years of research, a rich spectrum of KGC techniques has been proposed and widely deployed across diverse domains, consistently delivering strong empirical performance. Current approaches can be roughly classified into: translation-based methods, rotation-based methods, semantic matching-based methods, and neural network-based methods (Liu et al. 2024). Although these methods differ in how they represent triples, most of them adopt a common training paradigm centered on positive&negative sample pairs. Positive samples correspond to factual triples in KG, whereas negative samples are constructed via negative sampling to act as semantic foils. During training, the model learns by pulling positive pairs closer together in the embedding space while pushing negative pairs farther apart, thereby capturing the decision boundary and relational semantics. Consequently, the negative sampling (NS) strategy is pivotal in KGC, the quality of generated negative pairs directly determines the fidelity of triple embeddings and, by extension, the performance of downstream tasks such as link prediction and multi-hop reasoning (Yu et al. 2024).

Negative sampling is one of the core components determining model performance in KGC. Its goal is to construct high-quality contrast triples (negative samples) for factual triples (positive samples), enabling the model to more clearly learn semantic boundaries and reasoning patterns (Madushanka and Ichise 2024). Traditional negative sampling strategies include random replacement (Bordes et al. 2013), heuristic rule–based approaches (Islam, Aridhi, and Smail-Tabbone 2022), and adversarial negative sampling (Qin et al. 2021). While these methods have progressively improved the quality of negative samples in knowledge graph completion, they still face limitations such as susceptibility to false negatives, poor generalizability, and difficulty in controlling the hardness distribution of negative samples (Yang et al. 2024). With the remarkable success of diffusion models in generative tasks, diffusion-based negative sampling strategies have emerged (Cao et al. 2024). These methods generate samples through a step-wise denoising Markov chain: early timesteps produce coarse, easy-to-distinguish negative samples, while later steps generate semantically closer and harder negatives (Nguyen and Fang 2024). This naturally forms a distribution of negative samples ranging from easy to hard. The approach not only enhances diversity but also offers controllability.

Although diffusion-based negative sampling can naturally produce negatives of different hardness across denoising stages, substantially increasing sample diversity and boosting knowledge-graph completion performance, it still has several shortcomings (Niu and Zhang 2025a). **Firstly, no adaptive mechanism for learning difficulty.** In complex knowledge graphs, entities differ widely in how hard they are to learn. Triples that involve harder entities typically need harder negatives than those involving easier

*Corresponding Author

entities. Existing diffusion frameworks struggle to sense and adapt to these differences, leaving hard entities undertrained while easy entities may be over-reinforced. **Secondly, possible generation of semantically impoverished negatives.** Step-wise denoising helps reduce false negatives, yet early stages can still yield many semantically irrelevant, low-information negative triples, wasting computational resources. **Thirdly, under-utilization of negative-sample hardness during training.** Current pipelines usually mix samples from different denoising stages using fixed ratios or simple thresholds, without dynamically adjusting the easy-to-hard balance as learning progresses. This prevents the model from fully leveraging hard negatives to continually improve its discriminative power.

To address the aforementioned limitations and inspired by the success of diffusion models in various generative tasks, this paper proposes a novel **D**iffusion-based **A**daptive **N**egative **S**ampling method for **K**nowledge **G**raph **C**ompletion (DANS-KGC). First, a **difficulty assessment module** (DAM) is introduced to perform fine-grained quantification of entity learning difficulty in complex knowledge graphs. Then, a conditional diffusion model is employed to achieve difficulty-aware **adaptive negative sampling** (ANS). In the forward noise-injection phase, a differentiated noise scheduling strategy is applied: entities with higher learning difficulty are injected with stronger noise, while those with lower difficulty receive weaker noise. In the denoising generation phase, semantic constraints from the positive samples and neighborhood structural information are integrated to ensure that the generated negatives are semantically related to but distinguishable from the positives. By sampling outputs at various timesteps during denoising, the model naturally acquires a multi-level distribution of negative samples with varying hardness. Finally, a **dynamic training mechanism** (DTM) is proposed to adjust the ratio of negative samples of different hardness levels in real time based on training progress, continuously enhancing triple representation quality and link prediction performance. Our contributions can be summarized as follows:

- To our knowledge, this study is the first to argue that negative sampling ought to reflect the varying learning difficulties of individual entities. Accordingly, we develop a **Difficulty Assessment Module (DAM)** that combines semantic cues with graph-structural signals to produce a quantitative score for each entity's learning difficulty.
- We devise an **Adaptive Negative Sampling framework driven by a diffusion process (ANS)**. In the forward diffusion stage, the noise schedule is modulated by each entity's difficulty score: easily learned entities receive lighter perturbations, whereas harder ones are subjected to stronger corruption. During the reverse denoising stage, semantic and neighborhood constraints steer the process so that the recovered triples remain plausible yet diverge from the original positives. By drawing samples at several points along the denoising trajectory, ANS produces a diverse set of negatives spanning a continuum of hardness levels.
- To maximize the utility of negatives with different hardness levels, we implement a **Dynamic Training Mechanism (DTM)** that progressively raises the share of hard negatives according to a preset curriculum. Training starts with mostly easy examples, giving the model a stable foundation, and then incrementally introduces more challenging negatives. This staged exposure sharpens the decision boundary step by step, boosting the model's discriminative power and delivering consistent gains in knowledge-graph completion performance.

Related Works

In this section, we briefly review the related work, mainly covering knowledge graph completion methods, negative sampling techniques in knowledge graphs, and diffusion models.

Knowledge Graph Completion Methods

Translation-based models remain mainstream. TransE (Bordes et al. 2013) embeds entities and relations in a shared vector space, viewing each relation as a translation from head to tail. Variants such as TransH (Wang et al. 2014), TransR (Lin et al. 2015), and TransA (Xiao et al. 2015) extend this idea with relation-specific hyperplanes, mapping matrices, or adaptive metrics to better fit 1-to-N, N-to-1, and N-to-N patterns, yet they still struggle with symmetry, antisymmetry, and compositional relations. Rotation-based models interpret relations as rotations in complex or hypercomplex space. Starting with RotatE (Sun et al. 2019), subsequent works—QuatE (Zhang et al. 2019), Rotat3D (Gao et al. 2020), and DualE (Cao et al. 2021)—capture symmetric, antisymmetric, inverse, and compositional patterns, alleviating the limitations of translation models. Semantic-matching methods evaluate triples via latent semantic similarity. RESCAL (Bordes et al. 2014) uses a bilinear form with full-rank relation matrices; DistMult (Yang et al. 2015), HolE (Nickel, Rosasco, and Poggio 2016), and ComplEx (Trouillon et al. 2016) refine this by imposing structural constraints (diagonal matrices, circular convolution, or complex embeddings) to boost completion accuracy.

Most of the above reduce a knowledge graph to isolated triples, overlooking richer graph topology. GNN-based approaches fill this gap by propagating information along edges: convolutional models like R-GCN (Schlichtkrull et al. 2018) and CompGCN (Vashishth et al. 2019) aggregate neighbor features with relation-specific kernels, while attention models such as KBGAT (Nathani et al. 2019) and KRACL (Tan et al. 2023) weight neighbors adaptively. Both lines exploit multi-hop context, delivering state-of-the-art results in knowledge graph completion.

Negative Sampling Methods

Negative sampling (NS) is essential in training graph representation learning models, generating negative triples absent from the knowledge graph (KG) to contrast positive triples and improve model discriminative power. Traditional NS strategies, such as random replacement, are simple and efficient but often produce false negatives or trivial samples, weakening training signals. Heuristic-based methods

address these limitations through example popularity, prediction scores, or high-variance sample selection. Bernoulli sampling enhances negative sample quality via Bernoulli distributions (Wang et al. 2014). Graph-based heuristics, such as SANS (Ahrabian et al. 2020), select negatives from k-hop neighborhoods; MixGCF (Huang et al. 2021) synthesizes hard negatives by mixing hops and positives; and MCNS (Yang et al. 2020) employs Metropolis-Hastings sampling guided by sub-linear positivity theory. However, these methods typically rely on domain-specific knowledge, limiting their generalizability. Adversarial NS techniques like KBGAN (Cai and Wang 2018) and IGAN (Wang et al. 2021) utilize Generative Adversarial Networks (GANs) to dynamically create harder negatives, significantly boosting model discriminative capability. Despite improvements, adversarial and caching methods often lack explicit control over negative sample hardness. Recent diffusion-based NS approaches leverage denoising diffusion probabilistic models (DDPMs), generating negatives via step-wise denoising processes, naturally varying sample hardness and increasing diversity (Nguyen and Fang 2024). However, existing diffusion methods primarily operate globally, lacking fine-grained, entity-specific adaptive sampling.

Diffusion Models

Denoising diffusion probabilistic models (DDPMs) generate data by adding Gaussian noise via a forward Markov chain and then learning a reverse denoising path, achieving highly diverse, high-fidelity outputs (Ho, Jain, and Abbeel 2020). Adaptations to knowledge graphs include FDM, which treats link prediction as conditional generation (Long et al. 2024a), DHNS, which applies diffusion-based hierarchical negative sampling to multimodal KGs (Niu and Zhang 2025b), and other recent explorations (Long et al. 2024b). However, existing work cannot tailor negative samples to entity-specific learning difficulty. We address this by introducing a difficulty-aware noise schedule that adaptively generates negatives for each entity.

Methodology

Problem Definition and Overview

A knowledge graph can be defined as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, $\mathcal{E}$ and $\mathcal{R}$ denote the sets of entities and relations, and $\mathcal{T}$ is the set of triples in the form $\langle h, r, t\rangle$, where $h, t \in \mathcal{E}$ and $r \in \mathcal{R}$. Given an incomplete triple, such as $\langle h, r, ?\rangle$ or $\langle ?, r, t\rangle$, where h and t represent known head entities and tail entities, r is relation. The objective of KGC is predicting the missing head or tail entities. During the training process, the primary objective is to distinguish between positive triples $\langle h, r, t\rangle$ and their corresponding negative triples $\langle h', r, t\rangle$ or $\langle h, r, t'\rangle$. The aim is to enhance the discriminative ability of the model through contrastive learning between positive and negative triples. Where h' and t' are corrupted entities obtained through negative sampling.

Figure. 2 presents the overall architecture of DANS-KGC, consisting of three cooperating components. The **Difficulty Assessment Module (DAM)** assigns each entity a difficulty score that drives the subsequent noise scheduling and sampling strategies. The **Adaptive Negative Sampling Module (ANS)** leverages a diffusion process in which difficulty-aware noise is injected during the forward pass, and semantic as well as neighborhood constraints are enforced during reverse denoising, producing negatives that are both informative and graph-consistent. Finally, the **Dynamic Training Mechanism (DTM)** adjusts the mix of negatives on the fly, steadily shifting from easy to hard according to the curriculum. This staged exposure enables the model to refine its triple representations continuously and achieve superior completion accuracy.

Diffculty Assessment Module (DAM)

As mentioned earlier, due to the complexity of KGs, different entities have varying levels of learning difficulty. We assess each entity's learning difficulty by integrating semantic and structural features of the graph. First, we obtain multi-dimensional features for each entity through pre-training and statistical computations. The semantic features are vector embeddings (e.g., obtained via KG embedding models like TransE), while structural features include metrics such as Betweenness Centrality (BC), Closeness Centrality (CC), Clustering Coefficient (CCoef), Triple Count (TC), Average Neighbor Degree (ADN), and PageRank (PR). Definitions and formulas for these features are provided in Appendix A.

Let $\mathbf{e}_{\text{sem}}$ denote the semantic feature vector of an entity e, and let $\mathbf{e}_{\text{str}} = (\mathbf{e}^1_{\text{str}}, \mathbf{e}^2_{\text{str}}, \ldots, \mathbf{e}^m_{\text{str}})$ be its m-dimensional structural feature vector. We normalize each feature and then concatenate them to form an entity difficulty representation: $\mathbf{e}^d = [\mathbf{e}_{\text{sem}};\ \mathbf{e}_{\text{str}}]$. Next, a multi-layer perceptron (MLP) maps e^d to a difficulty score:

$$\zeta(e) = sigmoid(\mathbf{W}_2 \cdot ReLU(\mathbf{W}_1 \cdot \mathbf{e}_d + b_1) + b_2) \quad (1)$$

where $\zeta(e) \in (0, 1)$ represents the diffculty score of eneity e, the closer it is to 0, the lower the learning difficulty of the entity; conversely, the farther it is from 0, the greater the difficulty. $\mathbf{W}_1$ and $\mathbf{W}_2$ represents learnable matrices, respectively.

Diffusion-based Adaptive Negative Sampling (ANS)

Difficulty-Aware Forward Noise Scheduling (DFS). After deriving each entity's difficulty score with DAM, we feed this information into the forward noise-injection stage of the diffusion model: entities judged harder receive stronger perturbations, whereas easier ones are only lightly corrupted. Concretely, for an entity x we set an entity-specific upper bound on the noise intensity that is proportional to its difficulty $\zeta(x)$:

$$\beta_{max}(x) = \beta_{low} + (\beta_{global} - \beta_{low})(\zeta(x))^{\mu} \quad (2)$$

where $\beta_{max}(x)$ denotes the noise upper bound for sample x, β_{global} is a global maximum noise level (set empirically based on prior experience), and β_{low} is a minimum noise level applied to all samples to ensure even low-difficulty entities receive some noise (maintaining a baseline level of

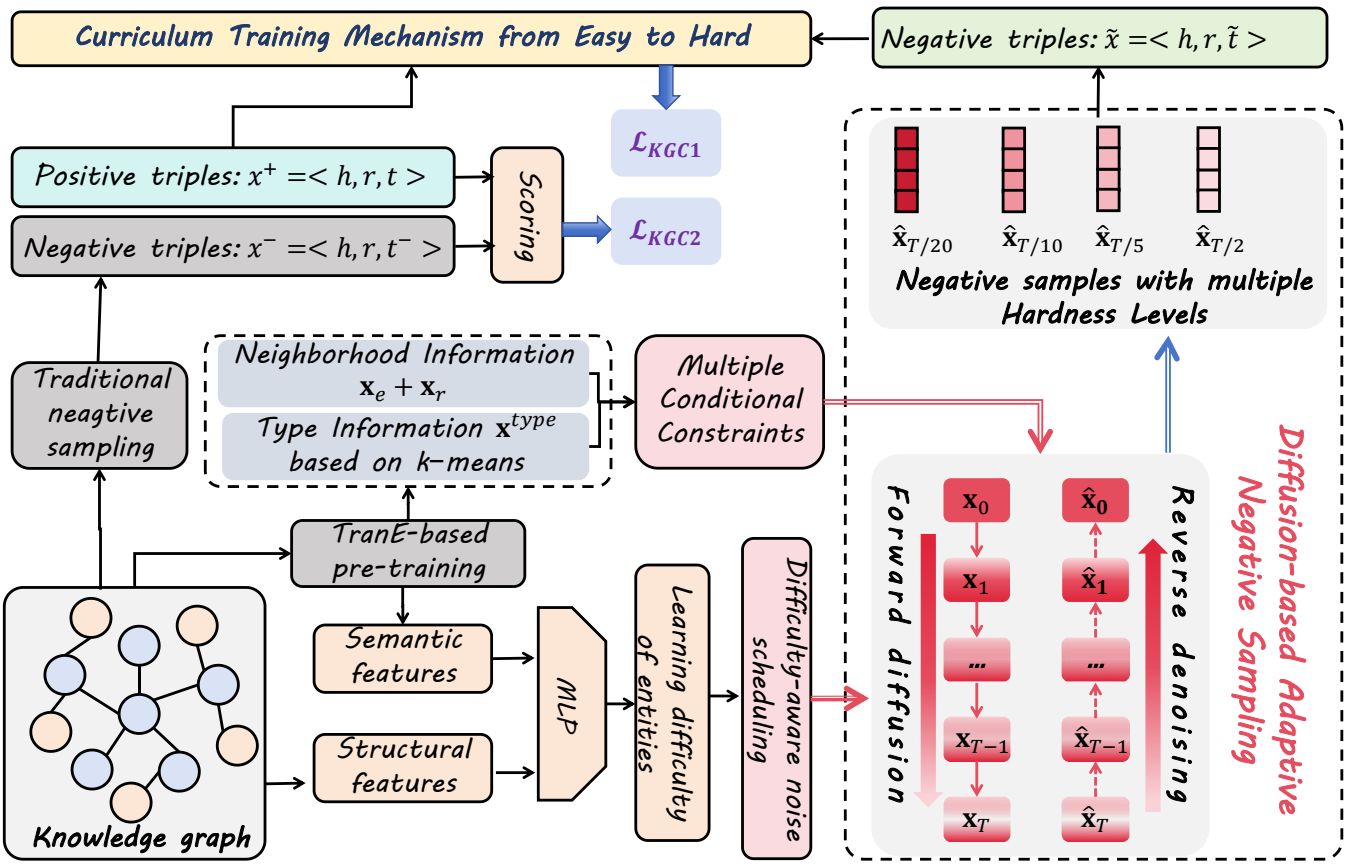

Figure 1: The overall framework of DANS-KGC.

negative distinguishability). μ is a hyperparameter controlling the influence of the difficulty score on noise intensity. After assigning difficulty-conditioned noise bounds, we define the noise schedule over time. Let T be the total number of diffusion steps. The noise variance at time step t for entity x is scheduled as:

$$\beta_t(x) = \beta_{init} + t/T(\beta_{max}(x) - \beta_{init}) \tag{3}$$

where β_{init} is the noise variance at the initial step $t = 0$. In this linear schedule, early diffusion steps inject relatively low noise (especially for easy entities with small $\beta_{\max}$), while later steps approach the entity's designated $\beta_{\max}(x)$. Following the DDPM framework, during forward diffusion the input entity embedding x_0 is progressively corrupted with noise, eventually yielding $x_T \sim \mathcal{N}(0, I)$ (pure Gaussian noise) after T steps. Formally, the forward diffusion process is a Markov chain:

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \tag{4}$$

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t \mathbf{I}) \tag{5}$$

where we abbreviate $\beta_t = \beta_t(x)$ for readability. Using the reparameterization trick, one can sample $\mathbf{x}_t$ at any arbitrary step t in closed form:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t \tag{6}$$

where $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, and $\epsilon_t \sim \mathcal{N}(0, I)$. In this way, we inject noise adaptively according to each entity's difficulty, thereby controlling the hardness of the negative sample that will be generated in the reverse process.

Condition-Constrained Reverse Denoising (CCD). After the forward diffusion process with adaptive noise scheduling, the entity embedding is progressively corrupted, ultimately resulting in a noisy entity representation $\mathbf{x}_t$. The reverse diffusion process iteratively denoises the noisy entity embedding $\mathbf{x}_t$, ultimately obtaining a synthesized entity embedding. Compared to existing reverse denoising processes used for negative sample generation, we impose semantic constraints to ensure the generated triples are semantically valid, thereby enhancing the quality of learning. Given the noisy embedding $\mathbf{x}_t$ at time step t, the reverse process is as follows:

$$\begin{aligned} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = &\mathcal{N}(\mathbf{x}_{t-1}; \varphi_\theta(\mathbf{x}_t, \mathbf{t}, \mathbf{x}^{\text{type}}, \\ &\mathbf{x}_e + \mathbf{x}_r), \beta_t \mathbf{I}) \end{aligned} \tag{7}$$

where θ denotes the parameterized configuration of the diffusion model, and $\mathbf{t}$ represents the embedding of time step t; $\mathbf{x}_e + \mathbf{x}_r$ is the combined representation of the observed entity e and its corresponding relation r, which guides the generation of the missing entity to form a negative sample triple. $\mathbf{x}^{\text{type}}$ represents the semantic type embedding of

the observed entity, which is used to impose semantic constraints on the generated negative samples. This ensures that the generated samples maintain a certain degree of semantic relevance to the observed sample, preventing the generation of implausible or meaningless triples. We obtain x_{type} by performing K-means clustering on the pre-trained entity embeddings and using the representation of each cluster center as the semantic type embedding for all entities within that cluster. Specifically, $\varphi_\theta(\mathbf{x}_t, \mathbf{t}, \mathbf{x}_e, \mathbf{x}_r)$ denotes the mean of the Gaussian distribution, which is defined as:

$$\begin{aligned}&\varphi_\theta(\mathbf{x}_t, \mathbf{t}, \mathbf{x}^{\text{type}}, \mathbf{x}_e + \mathbf{x}_r) = \\ &\frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t - \frac{1-\sqrt{\alpha_t}}{\sqrt{\alpha_t}\sqrt{1-\beta_t}}\epsilon_\theta(\mathbf{x}_t, \mathbf{t}, \mathbf{x}^{\text{type}}, \mathbf{x}_e + \mathbf{x}_r)\end{aligned} \tag{8}$$

After predicting the corresponding noise, the negative sample at the current time step can be obtained by removing the noise from the entity embedding at that time. The formula for generating the corresponding negative sample is as follows:

$$\hat{\mathbf{x}}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\hat{\mathbf{x}}_t - \frac{1-\sqrt{\alpha_t}}{\sqrt{\alpha_t}\sqrt{1-\alpha_t}}\epsilon_\theta(\mathbf{x}_t, \mathbf{t}, \mathbf{x}^{\text{type}}, \mathbf{x}_e + \mathbf{x}_r) + \sqrt{\beta_t}\epsilon_t \tag{9}$$

where $\hat{\mathbf{x}}_T = \mathbf{x}_T$ denotes the noisy entity embedding at the final time step T, and $\mathbf{x}_{t-1}$ represents the intermediate entity embedding at time step $t-1$ within the range $[0, T-1]$. To enhance the diversity of generated samples, $\sqrt{\beta_t}\epsilon_t$ is used to introduce random noise at each step, resulting in intermediate denoised outputs at each iteration. We use a multilayer perceptron combined with a layer normalization layer to predict the noise using learnable parameters θ as follows:

$$\epsilon_\theta(\mathbf{x}_t, \mathbf{t}, \mathbf{x}^{\text{type}}, \mathbf{x}_e + \mathbf{x}_r) = LayerNorm(MLP(\mathbf{x}_t, \mathbf{t}, \mathbf{x}^{\text{type}})) \tag{10}$$

We create negative samples by corrupting the tail entities of positive sample triples. Specifically, we feed the head entity, entity type, neighborhood, and temporal embeddings as conditional inputs to the reverse denoising function to generate semantically rich negative triples. To optimize this process, we minimize the denoising diffusion loss function to generate high-quality embeddings for the corrupted tail entities.

$$\mathcal{L}_{Diff} = ||\epsilon_\theta(\mathbf{x}_t, \mathbf{t}, \mathbf{x}^{\text{type}}, \mathbf{x}_e + \mathbf{x}_r) - \epsilon_t||^2 \tag{11}$$

which is the mean squared error between the predicted noise and the true noise at each diffusion step (with ϵ_t from the forward process). By training with $\mathcal{L}_{\text{Diff}}$, the model learns to generate high-quality negative entity embeddings that correspond to realistic corruptions of the original triple.

As we sample the reverse process at different intermediate time steps, the resulting negative samples exhibit different levels of difficulty: negatives obtained after fewer denoising steps (small t) are closer to the original positive and thus harder to distinguish, whereas those from later steps (large t) are more distorted by noise and thus easier. Therefore, by selecting particular diffusion steps, we can generate negative samples of varying difficulty for each positive triple. In this paper, we sample negatives at four diffusion steps (from near the middle to near the end of the process): $t \in \{\, T/20,\ T/10,\ T/5,\ T/2 \,\}$. This yields a set of negative triples with graded difficulty:

$$G^- = \{\ \langle h, r, \hat{x}_t \rangle \mid t \in \{T/20,\ T/10,\ T/5,\ T/2\}\ \}, \tag{12}$$

where $\hat{x}_t$ denotes the generated tail entity embedding at reverse step t (i.e., using $\hat{x}_t$ instead of continuing to $\hat{x}_0$).

In summary, the ANS module adaptively generates negative samples of varying hardness for entities of different difficulties via difficulty-based noise scheduling, conditional constrained denoising, and hierarchical sampling from multiple diffusion timesteps. Compared to simple random negative generation, this is a more refined strategy that provides a richer set of negatives. By integrating these diverse negatives into training, we can supply stronger and more informative training signals to the KGC model.

Dynamic Training Mechanism (DTM)

To effectively leverage the negative samples of varying difficulty produced by ANS, we propose a curriculum-style dynamic training mechanism (DTM). DTM gradually shifts training focus from easy negatives to hard negatives as learning progresses, stabilizing early training and progressively challenging the model with harder examples.

Difficulty-based partitioning. To fully exploit the difficulty-aware negatives produced at the four pre-defined diffusion timesteps $\mathcal{T} = \{\frac{T}{20}, \frac{T}{10}, \frac{T}{5}, \frac{T}{2}\}$,we redesign the curriculum schedule so that every difficulty band *exactly* corresponds to one sampling timestep. Let $t(\tilde{x})$ denote the diffusion timestep used to generate negative triple $\tilde{x} \in G^-$.We group G^- into four disjoint subsets:

$$\mathcal{N}^{(k)} = \left\{\tilde{x} \in G^- \;\middle|\; t(\tilde{x}) = \mathcal{T}_k\right\}, \quad k = 1, 2, 3, 4, \tag{13}$$

where $\mathcal{T}_1 < \mathcal{T}_2 < \mathcal{T}_3 < \mathcal{T}_4$ and the difficulty monotonically increases with k. Smaller timesteps preserve more noise, making the generated negatives harder; hence $\mathcal{T}_1 = \frac{T}{20}$ is the hardest band.

Curriculum sampling schedule. We denote the training epoch as e, and the total number of epochs as E_{max}, with the normalized training progress defined as $\tau_e = e/E_{max} \in [0, 1]$. Given the stage boundaries $(b_0, b_1, b_2, b_3, b_4) = (0, 0.25, 0.5, 0.75, 1)$, the sampling weight for difficulty band k at epoch e is:

$$w_e^{(k)} = \frac{\exp\left(\lambda\left[\tau_e - b_{k-1}\right]_+^\zeta\right)}{\sum_{j=1}^{4}\exp\left(\lambda\left[\tau_e - b_{j-1}\right]_+^\zeta\right)}, \quad k = 1, \ldots, 4, \tag{14}$$

where $[x]_+ = \max(0, x)$, $\zeta \in [0, 1]$ controls smoothness, and $\lambda > 0$ sharpens or softens the distribution.

Mini-batch Training. For each positive triple $x^+ = \langle h, r, t \rangle$, we sample a difficulty band $\kappa \sim \text{Cat}(w_e^{(1:4)})$ based on the current weights, and then select an entity from $\mathcal{N}^{(\kappa)}$ to corrupt the head or tail to obtain a negative triple $\tilde{x}$. Then we define the stage-aware weighted margin loss as follows:

$$\begin{aligned}\mathcal{L}_{KGC1} = &- log\sigma(\gamma_\kappa(\tau_e) - S(x^+)) \\ &- \frac{1}{|\mathcal{B}_e|}\sum_{\tilde{x}\in\mathcal{B}_e} w_e^{(\kappa(\tilde{x}))} log\sigma(S(\tilde{x}) - \gamma_\kappa(\tau_e))\end{aligned} \tag{15}$$

where

$$\gamma_\kappa(\tau_e) = \gamma_{base} \cdot \big[1 + \beta \cdot \mathrm{hard}_\kappa \cdot \tau_e\big], \quad \mathrm{hard}_\kappa = \frac{5-\kappa}{4}, \tag{16}$$

and γ_{base} is the base margin, $\beta \in [0, 1]$ controls the dynamic margin increment, and $S(\cdot)$ is the triple scoring function. As training proceeds, higher-difficulty weights and margins progressively dominate the loss calculation, focusing gradient updates on harder negatives. In addition, we also retained the negative triples obtained by negative sampling based on random replacement, which are also used to train knowledge graph representations, with the loss function being:

$$\begin{aligned} \mathcal{L}_{KGC2} = & - log\sigma(\gamma_{base} - S(x^+)) \\ & - \frac{1}{|\mathcal{B}^-|} \sum_{x^- \in \mathcal{B}^-} log\sigma(S(x^-) - \gamma_{base}) \end{aligned} \tag{17}$$

where γ_{base} indicates the fixed margin, x^- represents the negative sample from random replacement techniques. The total loss for KGC is as follows:

$$\mathcal{L}_{KGC} = \eta \mathcal{L}_{KGC1} + \mathcal{L}_{KGC2} \tag{18}$$

Furthermore, the loss function of the overall algorithm is as follows:

$$\mathcal{L} = \mathcal{L}_{KGC} + \mathcal{L}_{Diff} \tag{19}$$

To facilitate the understanding of the entire algorithm, Algorithm 1 presents the overall process of DANS-KGC.

Algorithm 1: DANS-KGC

Input: Training triples T_{train}; difficulty features for entities (from DAM); total diffusion steps T.
Compute initial difficulty scores $\zeta(e)$ for all entities e using DAM.
for epoch $ep = 1$ **to** $E_{\max}$ **do**
  Update curriculum weights $w_{ep}^{(1:4)}$ based on $\tau_{ep} = ep/E_{\max}$.
  for each positive triple $x^+ = \langle h, r, t\rangle$ in a batch B^+ **do**
    Sample band $\kappa \sim \text{Categorical}(w_{ep}^{(1:4)})$.
    Generate $G^- = \{\langle h, r, \hat{x}_t\rangle | t = T/20, T/10, T/5, T/2\}$ using ANS (forward diffusion with Eq. 2–3, reverse diffusion with Eq.7–9).
    Select a negative $\tilde{x}$ from $N^{(\kappa)} \subset G^-$ for x^+.
    Sample a random negative $x^- = \langle h, r, t'\rangle$ (or $\langle h', r, t\rangle$) by corrupting x^+.
  end for
  Compute $\mathcal{L}_{KGC1}$ on batch B^+ and corresponding negatives $\{\tilde{x}\}$ using Eq. 15.
  Compute $\mathcal{L}_{KGC1}$ on batch B^+ and random negatives $\{x^-\}$.
  Compute diffusion loss $\mathcal{L}_{\text{Diff}}$ on generated negatives.
  Update model parameters by minimizing $\mathcal{L} = \eta\mathcal{L}_{KGC1} + \mathcal{L}_{KGC2} + \mathcal{L}_{\text{Diff}}$.
end for

Experiments

In this section, to illustrate the superiority of DNS-KGC, we conducted comprehensive experiments on the link prediction task.

Experiment Setting

Datasets: We evaluate on six public KGC datasets: **Family**, **UMLS**, **WN18RR**, **FB15K-237**, **NELL-995**, and **YAGO3-10**. Statistics of each dataset are provided in the Table 1.

Dataset	Entity	Relation	Train	Valid	Test
Family	3.0k	12	5.9k	2.0k	2.8k
UMLS	135	46	1.3k	569	633
WN18RR	40.9k	11	21.7k	3.0k	3.1k
FB15k-237	14.5k	237	68.0k	17.5k	20.4k
NELL-995	74.5k	200	37.4k	543	2.8k
YAGO3-10	123.1k	37	269.7k	5.0k	5.0k

Table 1: Statistics of datasets used in experiments. Train, Valid, and Test represent the number of training, validation, and test queries, respectively.

Baseline: To comprehensively evaluate the effectiveness of DNS-KGC, we compared it with the following three categories of methods. **Traditional embedding methods**: ConvE (Dettmers et al. 2018), QuatE (Zhang et al. 2019), RotatE (Sun et al. 2019), DRUM (Sadeghian et al. 2019), and RNNLogic (Qu et al. 2020); **Graph neural network-based methods**: CompGCN, NBFNet (Zhu et al. 2021), ConGLR (Lin et al. 2022), and RED-GNN (Zhang and Yao 2021); **Diffusion-based generative methods**: FDM (Long et al. 2024a) and DiffusionE (Cao et al. 2024).

Evaluation Protocols: During the testing process, for each triple $\langle h, r, t\rangle$, we construct two queries $\langle h, r, ?\rangle$ and $\langle ?, r, t\rangle$, whose answers are t and h respectively. We then calculate the scores of the candidate triples and rank the candidate entities in descending order based on the scores. The evaluation metrics used are Mean Reciprocal Rank (MRR) and Hit@N (N=1,10). Higher MRR and Hit@N values indicate better performance.

Implementation Details: Our model is implemented in PyTorch and trained on four NVIDIA A6000 GPUs. We use the Adam optimizer. The hyperparameters (learning rate, diffusion steps T, noise hyperparameters $\beta_{\text{global}}, \beta_{\text{low}}, \mu$, curriculum parameters λ, ζ, β, etc.) are tuned on validation sets; their chosen values are provided in the Table 2.

Comparison Experiment Results

As shown in Table 3 and Table 4, **DANS-KGC achieves superior performance across all six benchmark datasets**. On **UMLS** and **YAGO3-10**, our method achieves the highest scores on all three metrics, with MRRs of 0.972 and 0.572 respectively, demonstrating strong modeling and generalization capability. On **WN18RR**, DANS-KGC also obtains the best MRR (0.558), outperforming all baselines including recent diffusion-based methods. For the **Family** dataset, DANS-KGC reaches an MRR of 0.994 and Hits@1 of 0.991, confirming its excellent reasoning ability on this domain. On **FB15K-237** and **NELL-995**, our model ranks

Hyperparameter	Family	UMLS	WN18RR	FB15K-237	NELL-995	YAGO3-10
Learning rate	$8e^{-5}$	$5e^{-5}$	$3e^{-4}$	$3e^{-4}$	$5e^{-4}$	e^{-3}
Batchsize	256	256	512	512	1024	2048
Epoch	1500	1500	1000	2000	2000	2500
Embedding size	200	200	400	500	500	600
λ	10	10	10	5	5	5
ζ	1	1	1	0.75	1	0.75
η	0.30	0.40	0.20	0.25	0.25	0.25
γ	1	1	1	1	1	1
μ	1	1	1.5	2	2	2
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
β	0.4	0.4	0.4	0.4	0.4	0.3

Table 2: Hyperparameter settings for different datasets.

Models	Family			UMLS			WN18RR		
	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
ConvE	0.912	0.837	0.982	0.937	0.922	0.967	0.427	0.392	0.498
QuatE	0.941	0.896	0.991	0.944	0.905	0.993	0.480	0.440	0.551
RotatE	0.921	0.866	0.988	0.925	0.863	0.993	0.477	0.428	0.571
DRUM	0.934	0.881	0.996	0.813	0.674	0.976	0.486	0.425	0.586
RNNLogic	0.881	0.857	0.907	0.842	0.772	0.965	0.483	0.446	0.558
CompGCN	0.933	0.883	0.991	0.927	0.867	<u>0.994</u>	0.479	0.443	0.546
NBFNet	0.989	0.988	0.989	0.948	0.920	**0.995**	0.551	0.497	**0.666**
RED-GNN	<u>0.992</u>	<u>0.988</u>	**0.997**	0.964	0.946	0.990	0.533	0.485	0.624
FDM	-	-	-	0.922	0.893	0.970	0.506	0.456	0.592
DiffusionE	0.990	0.989	0.992	<u>0.970</u>	<u>0.957</u>	0.992	<u>0.557</u>	<u>0.504</u>	0.658
DANS-KGC	**0.994**	**0.991**	<u>0.995</u>	**0.972**	**0.962**	**0.995**	**0.558**	**0.509**	<u>0.661</u>

Table 3: Comparison on Family, UMLS, and WN18RR. Best results are in **bold**, second-best in <u>underline</u>.

among the top performers, with an MRR of 0.546 on NELL-995 and the highest Hits@10 (0.669) on NELL-995. These results highlight the effectiveness of our adaptive sampling and dynamic training mechanisms in improving link prediction across diverse scenarios.

Abaltion Study

We perform ablation experiments on the **Family** dataset to quantify the contribution of each proposed component. Specifically, we consider variants: **DANS w/o DFS**, which uses a simple linear noise schedule (removing the difficulty-based forward scheduling); **DANS w/o CCD**, which removes the entity type and neighborhood semantic constraints during reverse diffusion (the diffusion model generates negatives without conditional guidance); and **DANS w/o DTM**, which replaces the curriculum-based dynamic training with a traditional static training regime (e.g., mixing easy and hard negatives in fixed proportion). Specific ablation study results and analysis are presented in the Table 5.

Difficulty-based forward noise scheduling. Removing the DFS module results in a significant performance drop, indicating that adapting the noise intensity according to entity difficulty is essential for generating appropriately hard negative samples. This mechanism ensures that entities with higher learning difficulty are exposed to stronger perturbations, enabling the model to better distinguish them from similar negatives.

Condition-constrained reverse denoising. The absence of the CCD module also leads to a noticeable decline in performance, which confirms the importance of incorporating semantic type and neighborhood constraints in the denoising process. These constraints guide the generation of semantically plausible but still challenging negative triples, enhancing both the quality and informativeness of the training signals.

Dynamic training mechanism. Replacing the DTM module with a static training process results in a marked reduction in Hits@1, underscoring the value of curriculum-style training. The dynamic adjustment of negative sample difficulty over training epochs helps the model to first stabilize with easier examples and then gradually focus on harder ones, thereby achieving more robust optimization and improved generalization.

Hyperparameter Senstivity Analysis

To comprehensively evaluate the performance of the DANS-KGC model under different configurations, we conducted sensitivity experiments on two key hyperparameters—μ and η. These parameters govern the model's core mechanisms and significantly impact its final performance.

First, μ controls the intensity of adaptive noise scheduling. It determines how the model generates challenging neg-

Models	FB15K-237			NELL-995			YAGO3-10		
	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
ConvE	0.325	0.237	0.501	0.511	0.446	0.619	0.520	0.450	0.660
QuatE	0.350	0.256	0.538	0.533	0.466	0.643	0.379	0.301	0.534
RotatE	0.337	0.241	0.533	0.508	0.448	0.608	0.495	0.402	0.670
DRUM	0.343	0.255	0.516	0.532	0.460	0.662	0.531	0.453	0.676
RNNLogic	0.344	0.252	0.530	0.416	0.363	0.478	0.554	0.509	0.622
CompGCN	0.355	0.264	0.535	0.463	0.383	0.596	0.421	0.392	0.577
NBFNet	0.415	0.321	0.599	0.525	0.451	0.639	0.550	0.479	0.686
RED-GNN	0.374	0.283	0.558	0.543	0.476	0.651	0.559	0.483	0.689
FDM	**0.485**	**0.386**	**0.681**	-	-	-	-	-	-
DiffusionE	0.376	0.294	0.539	**0.552**	0.490	0.654	0.566	0.494	0.692
DANS-KGC	0.404	0.324	0.596	0.546	**0.512**	**0.669**	**0.572**	**0.512**	**0.701**

Table 4: Comparison on FB15K-237, NELL-995, and YAGO3-10. Best results are in **bold**, second-best in underline.

Model	MRR	Hit@1	Hit@10
DANS-KGC	0.994	0.991	0.995
DANS w/o DFS	0.965	0.943	0.988
DANS w/o CCD	0.989	0.976	0.983
DANS w/o DTM	0.978	0.921	0.974

Table 5: Ablation study of DANS-KGC on Family dataset.

ative samples based on the learning difficulty of entities. By adjusting the value of μ, we aim to explore the model's reliance on this adaptive negative sampling strategy. Understanding the sensitivity of μ will help reveal the effectiveness of injecting stronger noise into harder entities and its contribution to the model's overall performance. Second, η balances the loss between DANS negative samples and traditional random negative samples. This parameter directly reflects the model's dependence on the proposed DANS sampling mechanism. By varying η, we can analyze how the DANS mechanism influences the training process and final performance under different weight settings. In particular, when η approaches zero, the model primarily relies on traditional random negative sampling, which provides strong evidence for the irreplaceability of the DANS mechanism.

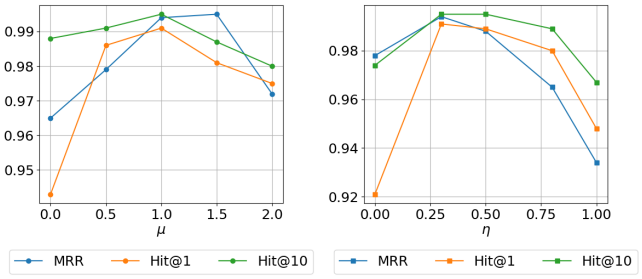

Figure 2: The sensitivity analysis results of parameters μ and η on the Family dataset.

The model shows notable sensitivity to the hyperparameters μ and η. Specifically, μ controls the strength of adaptive noise scheduling. The best performance (MRR, Hit@1, Hit@10) is achieved when $\mu = 1$, indicating its effectiveness in generating challenging negative samples. However, extreme values (e.g., $\mu = 0.0$ or 2.0) lead to performance degradation, suggesting the necessity of properly tuning noise intensity. The parameter η balances the loss between DANS-generated and randomly sampled negatives. The optimal performance occurs at $\eta = 0.25$, highlighting the effectiveness of the DANS mechanism. Performance drops significantly as $\eta \to 0.0$, reflecting over-reliance on random sampling, while excessive values of η also impair performance, implying the need for a balanced sampling strategy.

Conclusions

To address the issue that existing methods struggle to achieve adaptive negative sampling for different entities, this paper proposes the DANS-KGC method. This method introduces a noise scheduling mechanism based on difficulty and a denoising module with conditional constraints, utilizing diffusion models to generate diverse negative samples for entities of different difficulties. Moreover, a dynamic training mechanism is introduced during the training process to gradually make full use of various negative samples. Experimental results show that DANS-KGC achieves excellent performance on multiple datasets.

References

Ahrabian, K.; Feizi, A.; Salehi, Y.; Hamilton, W. L.; and Bose, A. J. 2020. Structure Aware Negative Sampling in Knowledge Graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6093–6101.

Bordes, A.; Glorot, X.; Weston, J.; and Bengio, Y. 2014. A semantic matching energy function for learning with multi-relational data: Application to word-sense disambiguation. *Machine learning*, 94(2): 233–259.

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.

Cai, L.; and Wang, W. Y. 2018. KBGAN: Adversarial Learning for Knowledge Graph Embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 1470–1480.

Cao, Z.; Li, J.; Wang, Z.; and Li, J. 2024. Diffusione: Reasoning on knowledge graphs via diffusion-based graph neural networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 222–230.

Cao, Z.; Xu, Q.; Yang, Z.; Cao, X.; and Huang, Q. 2021. Dual quaternion knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 6894–6902.

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Gao, C.; Sun, C.; Shan, L.; Lin, L.; and Wang, M. 2020. Rotate3d: Representing relations as rotations in three-dimensional space for knowledge graph embedding. In *Proceedings of the 29th ACM international conference on information & knowledge management*, 385–394.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33: 6840–6851.

Hogan, A.; Blomqvist, E.; Cochez, M.; d'Amato, C.; Melo, G. D.; Gutierrez, C.; Kirrane, S.; Gayo, J. E. L.; Navigli, R.; Neumaier, S.; et al. 2021. Knowledge graphs. *ACM Computing Surveys (Csur)*, 54(4): 1–37.

Huang, T.; Dong, Y.; Ding, M.; Yang, Z.; Feng, W.; Wang, X.; and Tang, J. 2021. Mixgcf: An improved training method for graph neural network-based recommender systems. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 665–674.

Islam, M. K.; Aridhi, S.; and Smail-Tabbone, M. 2022. Negative sampling and rule mining for explainable link prediction in knowledge graphs. *Knowledge-Based Systems*, 250: 109083.

Liang, K.; Meng, L.; Liu, M.; Liu, Y.; Tu, W.; Wang, S.; Zhou, S.; Liu, X.; Sun, F.; and He, K. 2024. A survey of knowledge graph reasoning on graph types: Static, dynamic, and multi-modal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12): 9456–9478.

Lin, Q.; Liu, J.; Xu, F.; Pan, Y.; Zhu, Y.; Zhang, L.; and Zhao, T. 2022. Incorporating context graph with logical reasoning for inductive relation prediction. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, 893–903.

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29.

Liu, J.; Ke, W.; Wang, P.; Shang, Z.; Gao, J.; Li, G.; Ji, K.; and Liu, Y. 2024. Towards continual knowledge graph embedding via incremental distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 8759–8768.

Long, X.; Zhuang, L.; Li, A.; Li, H.; and Wang, S. 2024a. Fact embedding through diffusion model for knowledge graph completion. In *Proceedings of the ACM Web Conference 2024*, 2020–2029.

Long, X.; Zhuang, L.; Li, A.; Wei, J.; Li, H.; and Wang, S. 2024b. KGDM: A diffusion model to capture multiple relation semantics for knowledge graph embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 8850–8858.

Madushanka, T.; and Ichise, R. 2024. Negative sampling in knowledge graph representation learning: A review. *arXiv preprint arXiv:2402.19195*.

Nathani, D.; Chauhan, J.; Sharma, C.; and Kaul, M. 2019. Learning Attention-based Embeddings for Relation Prediction in Knowledge Graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4710–4723.

Nguyen, T.-K.; and Fang, Y. 2024. Diffusion-based negative sampling on graphs for link prediction. In *Proceedings of the ACM Web Conference 2024*, 948–958.

Nickel, M.; Rosasco, L.; and Poggio, T. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Niu, G.; and Zhang, X. 2025a. Diffusion-based hierarchical negative sampling for multimodal knowledge graph completion. *arXiv preprint arXiv:2501.15393*.

Niu, G.; and Zhang, X. 2025b. Diffusion-based hierarchical negative sampling for multimodal knowledge graph completion. *arXiv preprint arXiv:2501.15393*.

Qin, X.; Sheikh, N.; Reinwald, B.; and Wu, L. 2021. Relation-aware graph attention model with adaptive self-adversarial training. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 9368–9376.

Qu, M.; Chen, J.; Xhonneux, L.-P.; Bengio, Y.; and Tang, J. 2020. Rnnlogic: Learning logic rules for reasoning on knowledge graphs. *arXiv preprint arXiv:2010.04029*.

Sadeghian, A.; Armandpour, M.; Ding, P.; and Wang, D. Z. 2019. Drum: End-to-end differentiable rule mining on knowledge graphs. *Advances in neural information processing systems*, 32.

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 *European semantic web conference*, 593–607. Springer.

Sun, Z.; Deng, Z.-H.; Nie, J.-Y.; and Tang, J. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

Tan, Z.; Chen, Z.; Feng, S.; Zhang, Q.; Zheng, Q.; Li, J.; and Luo, M. 2023. KRACL: Contrastive learning with graph context modeling for sparse knowledge graph completion. In *Proceedings of the ACM web conference 2023*, 2548–2559.

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

Vashishth, S.; Sanyal, S.; Nitin, V.; and Talukdar, P. 2019. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082*.

Wang, M.; Wang, S.; Yang, H.; Zhang, Z.; Chen, X.; and Qi, G. 2021. Is visual context really helpful for knowledge graph? A representation learning perspective. In *Proceedings of the 29th ACM international conference on multimedia*, 2735–2743.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28.

Xiao, H.; Huang, M.; Hao, Y.; and Zhu, X. 2015. TransA: An adaptive approach for knowledge graph embedding. *arXiv preprint arXiv:1509.05490*.

Yang, B.; Yih, S. W.-t.; He, X.; Gao, J.; and Deng, L. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *Proceedings of the International Conference on Learning Representations (ICLR) 2015*.

Yang, Z.; Ding, M.; Huang, T.; Cen, Y.; Song, J.; Xu, B.; Dong, Y.; and Tang, J. 2024. Does negative sampling matter? a review with insights into its theory and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8): 5692–5711.

Yang, Z.; Ding, M.; Zhou, C.; Yang, H.; Zhou, J.; and Tang, J. 2020. Understanding negative sampling in graph representation learning. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 1666–1676.

Yu, J.; Ge, Q.; Li, X.; and Zhou, A. 2024. Heterogeneous graph contrastive learning with meta-path contexts and adaptively weighted negative samples. *IEEE Transactions on Knowledge and Data Engineering*, 36(10): 5181–5193.

Zhang, S.; Tay, Y.; Yao, L.; and Liu, Q. 2019. Quaternion knowledge graph embeddings. *Advances in neural information processing systems*, 32.

Zhang, Y.; and Yao, Q. 2021. Knowledge graph reasoning with relational directed graph. *arXiv preprint*.

Zhu, Z.; Zhang, Z.; Xhonneux, L.-P.; and Tang, J. 2021. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in neural information processing systems*, 34: 29476–29490.