# Context-Aware Adapter Tuning for Few-Shot Relation Learning in Knowledge Graphs

**Ran Liu**[1], **Zhongzhou Liu**[1], **Xiaoli Li**[2,3] **and Yuan Fang**[1]
[1]School of Computing and Information Systems, Singapore Management University, Singapore
[2]Institute for Infocomm Research, A*STAR, Singapore
[3]A*STAR Centre for Frontier AI Research, Singapore
{ran.liu.2020, zzliu.2020}@phdcs.smu.edu.sg, xlli@i2r.a-star.edu.sg, yfang@smu.edu.sg

## Abstract

Knowledge graphs (KGs) are instrumental in various real-world applications, yet they often suffer from incompleteness due to missing relations. To predict instances for novel relations with limited training examples, few-shot relation learning approaches have emerged, utilizing techniques such as meta-learning. However, the assumption is that novel relations in meta-testing and base relations in meta-training are independently and identically distributed, which may not hold in practice. To address the limitation, we propose RelAdapter, a context-aware adapter for few-shot relation learning in KGs designed to enhance the adaptation process in meta-learning. First, RelAdapter is equipped with a lightweight adapter module that facilitates relation-specific, tunable adaptation of meta-knowledge in a parameter-efficient manner. Second, RelAdapter is enriched with contextual information about the target relation, enabling enhanced adaptation to each distinct relation. Extensive experiments on three benchmark KGs validate the superiority of RelAdapter over state-of-the-art methods.

## 1 Introduction

Knowledge graphs (KGs) (Bollacker et al., 2008; Suchanek et al., 2007; Vrandečić and Krötzsch, 2014) have been widely adopted to describe real-world facts using triplets in the form of (head entity, relation, tail entity). However, curating and maintaining all the possible ground-truth triplets is impossible, and various approaches for knowledge graph completion (Bordes et al., 2013; Yang et al., 2014; Trouillon et al., 2016; Sun et al., 2019) have been proposed to discover missing facts. Many of these methods adopt a supervised learning paradigm, which require abundant training data for each relation. In real-world settings, novel and emerging relations, along with many relations in the long tail, are associated with very few instances (Xiong et al., 2018), limiting their performance.

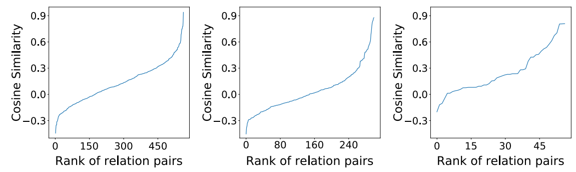
Subsequently, *few-shot relation learning* (FSRL) on KGs has emerged to handle novel relations with only a few known instances. An established line of work (Chen et al., 2019; Niu et al., 2021) employs *meta-learning*, most notably Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017). MAML aims to learn a prior from a series of meta-training tasks, which can be rapidly adapted to downstream meta-testing tasks. The meta-training tasks are specifically constructed in a few-shot setup to mimic downstream tasks. In the context of FSRL, each task contains only a few training instances of a single relation, and the objective is to predict more instances for a novel task (i.e., relation[1]) not seen in meta-training. For example, MetaR (Chen et al., 2019) aims to learn a relation-specific prior (also called meta-knowledge) during the meta-training stage, using a series of meta-training tasks constructed from a set of base relations with abundant instances. Subsequently, the meta-knowledge is leveraged for rapid adaptation, through a lightweight fine-tuning step, for few-shot predictions on novel relations in meta-testing.

**Limitation of prior work.** The major limitation of FSRL methods based on meta-learning lies in the assumption that the meta-training and meta-testing tasks are independently and identically distributed (i.i.d.). However, different relations may diverge significantly in their underlying distributions, thereby weakening the i.i.d. task assumption. To investigate this hypothesis, we randomly sample a large number of relation pairs from standard benchmark datasets, namely, WIKI, FB15K-237 and UMLS[2]. We perform a mean pooling across all entities within each relation task to derive an average embedding as the task representation. For every pair of relations, we plot their cosine similar-

[1]We use the terms "task" and "relation" interchangeably.
[2]Refer to Sect. 5 for dataset details.



*Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17525–17537
November 12-16, 2024 ©2024 Association for Computational Linguistics




(a) WIKI (b) FB15K-237 (c) UMLS

Figure 1: Pairwise cosine similarity of relations.

ity in Fig. 1. The results reveal a wide variance in the similarities between relations, suggesting that a uniform adaptation process may not suffice for all relations. In particular, for out-of-distribution relations, performance degradation is generally anticipated (Radstok et al., 2021; Li et al., 2022).

**Challenges and insights.** Specifically, we identify two open challenges, at the model and data levels, towards more universal downstream adaptation. At the *model level*, the first challenge (**C1**) is how to design a relation-specific adaptation module that can be tailored to each relation, while still leveraging the meta-knowledge learned from meta-training. The goal is to enable a more flexible adaptation process that allows for a relation-specific balance between the global prior and local task input. At the *data level*, the second challenge (**C2**) is how to augment few-shot relation instances during meta-testing in an unsupervised manner, thereby further enhancing adaptation to novel relations.

To address these challenges, we introduce a context-aware adapter framework, called RelAdapter, for few-shot relation learning. To overcome the model-level challenge (C1), we draw inspiration from parameter-efficient fine-tuning (Houlsby et al., 2019; Hu et al., 2021; Li and Liang, 2021; Brown et al., 2020). Specifically, we propose to integrate a lightweight *adapter* module into the meta-learning framework. The adapter module enables a relation-specific, tunable adaptation of the global prior to suit the local task in the meta-testing stage. For the data-level challenge (C2), we propose to inject additional *contextual information* about the target relation into meta-testing. The contextual information is extracted based on existing KG structures without requiring any extra annotation, serving as a form of data augmentation to enrich the few-shot relation instances. This strategy endows the adapter with more relation-specific contexts, and the context-aware adapter can enhance relation-specific adaptation.

**Contributions.** In summary, we make the following contributions. (1) We observe the i.i.d. limitation of prior work in FSRL, and first support it with an empirical analysis. (2) We propose RelAdapter, a context-aware adapter framework for FSRL. Specifically, we design a lightweight adapter module that leverages contextual information, enabling a relation-specific, tunable, and fine-grained adaptation for each novel relation in meta-testing. (3) We conduct extensive experiments on three benchmark datasets, and demonstrate the superiority of our proposed RelAdapter.

## 2 Related Work

**Supervised relation learning.** Knowledge graph embedding aims to transform entities and relations into a low-dimensional continuous vector space while preserving their semantic meaning. Conventional knowledge graph completion models can generally be classified into three main categories: (1) Translation-based methods, such as TransE (Bordes et al., 2013), TransH (Wang et al., 2014), and TransD (Ji et al., 2015), which are additive models that use distance-based constraint to optimize entity and relations embedding. (2) Semantic matching-based methods, such as DistMult (Yang et al., 2014) and ComplEx (Trouillon et al., 2016), which are multiplicative models that exploit the interaction between entity and relation vectors. (3) Graph-based models, include graph neural networks such as GCN (Kipf and Welling, 2017) and RGCN (Schlichtkrull et al., 2018), which considers higher-order structures in KGs. However, these supervised approaches rely on a large amount of training data and are not well-suited for few-shot relation learning.

**Few-shot relation learning.** To address one- or few-shot relation learning, many models have been proposed recently in two main categories: (1) Metric-based models that calculates a similarity score between support and query sets to learn the matching metrics. GMatching (Xiong et al., 2018) uses a one-hop neighbor encoder and a matching network, but assumes that all neighbors contribute equally. FSKGC (Zhang et al., 2020) extends the setting to more shots and seeks to merge information learnt from multiple reference triplets with a fixed attention mechanism. FAAN (Sheng et al., 2020) introduces an relation-specific adaptive neighbor encoder for one-hop neighbors. (2) Optimization-based models aim to learn an initial meta prior that can be generalized to a new relation given few-shot examples. MetaR (Chen et al., 2019) makes predictions by transferring shared relation-specific meta information from the support set to the queries through a mean pooling of the support set. GANA (Niu et al., 2021) improves on previous models by having a gated and attentive neighbor aggregator to capture the most valuable contextual semantics of a few-shot relation. HiRe (Wu et al., 2023) further considers triplet-level contextual information to generate relation meta. Meanwhile, REFORM (Wang et al., 2021) seeks to alleviate the impact of potential errors, which can be prevalent in KGs and further exacerbated under few-shot settings.

Despite these efforts, they fail to explicitly account for out-of-distribution relations. NP-FKGC (Luo et al., 2023) offers a solution to the out-of-distribution problem by leveraging neural processes to model complex distributions which can fit both training and test data. Unfortunately, it cannot be applied directly to existing meta-learning frameworks, and is therefore unable to benefit from meta-learned prior. Similarly, several other recent works (Liu et al., 2024; Meng et al., 2024; Huang et al., 2022) for few-shot relation learning do not utilize meta-learning frameworks, and thus do not leverage a set of meta-training tasks to learn a prior, as done in our work.

**Adapters.** Parameter-efficient fine-tuning has gained traction in various domains (Houlsby et al., 2019; Li and Liang, 2021; Brown et al., 2020; Hu et al., 2021). The adapter (Houlsby et al., 2019) is a common parameter-efficient technique, which adds a sub-module to pre-trained language models to adapt them to downstream tasks. In the field of general graph learning, AdapterGNN (Li et al., 2024) employs adapters to bridge the gap between transformer-based models and graph neural networks. On the other hand, G-Adapter (Gui et al., 2024) leverages contextual graph structures as an inductive bias to aid the training of the adapter. Both approaches are designed for general graph learning, and cannot handle the problem of FSRL on KGs. Furthermore, our work focuses on integrating adapters into meta-learning frameworks to leverage meta-learned knowledge, which are missing in prior adapter designs for graphs.

**Contextual information.** Previous works have utilized contextual information in knowledge graphs. For example, Oh et al. (2018) tap on the contexts from multi-hop neighborhoods, and Tan et al. (2023) apply an attention mechanism to aggregate neighboring contexts. However, how contextual information can be utilized for few-shot relation learning, particularly within an adapter module, remains unclear. In contrast, our work introduces a context-aware adapter module to address the few-shot setting.

## 3 Preliminaries

In this section, we introduce the problem of few-shot relation learning (FSRL) and the meta-learning framework for this problem.

**Problem formulation.** A knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{R})$ comprises a set of triplets. Each triplet is represented by the form $(h, r, t)$ for some $h, t \in \mathcal{V}$ and $r \in \mathcal{R}$, where $\mathcal{V}$ is the set of entities and $\mathcal{R}$ is the set of relations.

Consider a novel relation $r \notin \mathcal{R}$ w.r.t. a knowledge graph $\mathcal{G}$. Further assume a support set $\mathcal{S}_r = \{(h_i, r, t_i) \mid i = 1, 2, \ldots, K\}$ for some $h_i, t_i \in \mathcal{V}$ for the novel relation $r$. The goal is to predict the missing tail entities in a query set, $\mathcal{Q}_r = \{(h_j, r, ?) \mid j = 1, 2, \ldots, \}$, w.r.t. the given $h_j \in \mathcal{V}$ and $r$.

In the paper, for each triplet in $(h_j, r, ?) \in \mathcal{Q}_r$, a type-constrained candidate set $\mathcal{C}_{h_j,r}$ is provided and the objective is to rank the true tail entities highest among the candidates. Together, the support and query sets form a task $\mathcal{T}_r = (\mathcal{S}_r, \mathcal{Q}_r)$ for $r$. The support set provides a few training instances, known as $K$-shot relation learning, where $|\mathcal{S}_r| = K$ is typically a small number.

**Meta-learning framework.** Given the success of meta-learning in few-shot problems, we adopt MetaR (Chen et al., 2019), a popular meta-learning-based approach for FSRL, as our learning framework. MetaR consists of two stages: *meta-training* and *meta-testing*, aiming to learn a prior $\Phi$ from the meta-training stage that can be adapted to the meta-testing stage. On one hand, meta-training involves a set of seen relations $\mathcal{R}^{\text{tr}}$, and operates on their task data $\mathcal{D}^{\text{tr}} = \{\mathcal{T}_r \mid r \in \mathcal{R}^{\text{tr}}\}$. On the other hand, meta-testing involves a set of novel relations $\mathcal{R}^{\text{te}}$ such that $\mathcal{R}^{\text{tr}} \cap \mathcal{R}^{\text{te}} = \emptyset$, and operates on their task data $\mathcal{D}^{\text{te}} = \{\mathcal{T}_r \mid r \in \mathcal{R}^{\text{te}}\}$. Note that the ground-truth tail entities are provided in the query sets of the meta-training tasks $\mathcal{D}^{\text{tr}}$, whereas the objective is to make predictions for the query sets of the meta-testing tasks $\mathcal{D}^{\text{te}}$.

**Meta-training.** During meta-training, the model learns a prior $\Phi$, which serves as a good initialization to extract a shared *relation meta*, $R_{\mathcal{T}_r} \in \mathbb{R}^d$, for each task $\mathcal{T}_r = (\mathcal{S}_r, \mathcal{Q}_r) \in \mathcal{D}^{\text{tr}}$. Specifically, the relation meta $R_{\mathcal{T}_r}$ is obtained through a mean pooling of all (head, tail) pairs in the support set of the task $\mathcal{T}_r$, as follows.

$$R_{\mathcal{T}_r} = \texttt{Mean}(\{\texttt{RML}(f(h), f(t); \Phi) \mid \mathcal{S}_r\}), \quad (1)$$

where $f(\cdot)$ is a pre-trained encoder that generates $d$-dimensional embeddings for the entities. Moreover, RML is the *relation-meta learner*, implemented as a two-layer multi-layer perceptron (MLP).

It is worth noting that the prior $\Phi$ initializes the relation-meta learner RML, which further generates the relation meta $R_{\mathcal{T}_r}$. Subsequently, $R_{\mathcal{T}_r}$ is rapidly updated by a *gradient meta* $G_{\mathcal{T}_r}$ calculated from the loss on the support set, i.e., $R'_{\mathcal{T}_r} = R_{\mathcal{T}_r} - \beta G_{\mathcal{T}_r}$, where $\beta$ is the step size. The resulting $R'_{\mathcal{T}_r}$ serves as the relation meta adapted to the support set, which is used to calculate the loss on the query set. Finally, the query loss is backpropagated to update the prior $\Phi$ and the embedding matrix emb for the entities.

More specifically, the support and query losses are calculated as follows.

$$\begin{aligned} L_{\mathcal{S}_r} = \textstyle\sum_{(h,r,t)\in\mathcal{S}_r} [\gamma + s(\texttt{emb}(h), R_{\mathcal{T}_r}, \\ \texttt{emb}(t)) - s(\texttt{emb}(h), R_{\mathcal{T}_r}, \texttt{emb}(t'))]_+ \quad (2) \\ L_{\mathcal{Q}_r} = \textstyle\sum_{(h,r,t)\in\mathcal{Q}_r} [\gamma + s(\texttt{emb}(h), R'_{\mathcal{T}_r}, \\ \texttt{emb}(t)) - s(\texttt{emb}(h), R'_{\mathcal{T}_r}, \texttt{emb}(t'))]_+ \quad (3) \end{aligned}$$

Here, $s(\cdot,\cdot)$ is a scoring function such as TransE (Bordes et al., 2013), i.e., $s(\mathbf{h},\mathbf{r},\mathbf{t}) = \|\mathbf{h}+\mathbf{r}-\mathbf{t}_i\|$, where $\|\cdot\|$ denotes the $L_2$ norm. emb is an entity embedding matrix, which is initialized by a pre-trained model and can be further optimized during meta-training. $(h, r, t')$ refers to a negative triplet of the relation $r$, where $t'$ is randomly sampled from the type-constrained candidate set. $\gamma$ represents the margin which is a hyper-parameter. $[\cdot]_+$ denotes the positive part of the input value.

**Meta-testing.** The meta-testing stage follows the same pipeline as meta-training, except that we cannot compute and backpropagate the query loss to update model parameters ($\Phi$ and emb). For each novel task $\mathcal{T}_r \in \mathcal{D}^{\text{te}}$, like meta-training, we first generate the relation meta $R_{\mathcal{T}_r}$ using the prior $\Phi$, then update it as $R'_{\mathcal{T}_r}$ based on the support set. Then, for each partial triplet $(h_j, r, ?)$ in the query set, we rank the candidate entities $\mathcal{C}_{h_j,r}$ in ascending order of the scoring function $s(h_j, R'_{\mathcal{T}_r}, t_j)$ for every $t_j \in \mathcal{C}_{h_j,r}$.

# 4 Methodology

In this section, we introduce the proposed approach RelAdapter. As depicted in Fig. 2, RelAdapter has two important components, namely, (a) adapter network and (b) entity context. On one hand, the adapter network aims to facilitate relation-specific and tunable adaptation of meta-learned prior in a parameter-efficient manner. On the other hand, the entity context aims to enrich the adapter with contextual information pertinent to the target relation, enabling more precise adaptation to each distinct relation. The two components are integrated into a *context-aware adapter*, which enhances FSRL for the novel relations in meta-testing.

In the rest of the section, we first introduce the context-aware adapter, followed by the details of the meta-training and meta-testing stages.

## 4.1 Context-aware Adapter

We enhance the adaptation to novel relations at the model level through an adapter module, and at the data level through context-aware adaptation.

**Adapter.** The objective of the adapter is to achieve a relation-specific adaptation for the novel relations in meta-testing, to overcome the divergence from the seen relations in meta-training. Specifically, we adapt the relation meta $R_{\mathcal{T}_r}$ to the target relation $r$ through the adapter module, as follows.

$$\begin{aligned} R^A_{\mathcal{T}_r} &= \texttt{Adapter}(R_{\mathcal{T}_r}; \Theta_r) \\ &= \alpha \cdot \texttt{FFN}(R_{\mathcal{T}_r}; \Theta_r) + (1-\alpha) \cdot R_{\mathcal{T}_r} \quad (4) \end{aligned}$$

where the output from the adapter is $R^A_{\mathcal{T}_r}$, the *adapted* relation meta specific to the relation $r$. The adapter module consists of a lightweight feed-forward network (FFN) and a residual layer, as shown in Fig. 2(a), where $\alpha$ is a hyper-parameter to balance the FFN and the residual, and $\Theta_r$ is the $r$-specific parameter of the adapter. Note that the FFN typically adopts a bottleneck structure, which projects the input dimension $d$ into a smaller dimension $m$, reducing the number of parameters to achieve parameter-efficient adaptation.

**Context-aware adaptation.** At the data level, we augment the embedding of each entity (head or tail) by additional pre-trained contextual information from their related entities, as shown in Fig. 2(b). The contextual information enables more tailored adaptation to each distinct novel relation.

$$\begin{aligned} \mathbf{e}^c = \mu \cdot \texttt{Mean}(\{f(e_k) \mid e_k \in \mathcal{N}_e\}) \\ + (1-\mu) \cdot \texttt{emb}(e) \quad (5) \end{aligned}$$

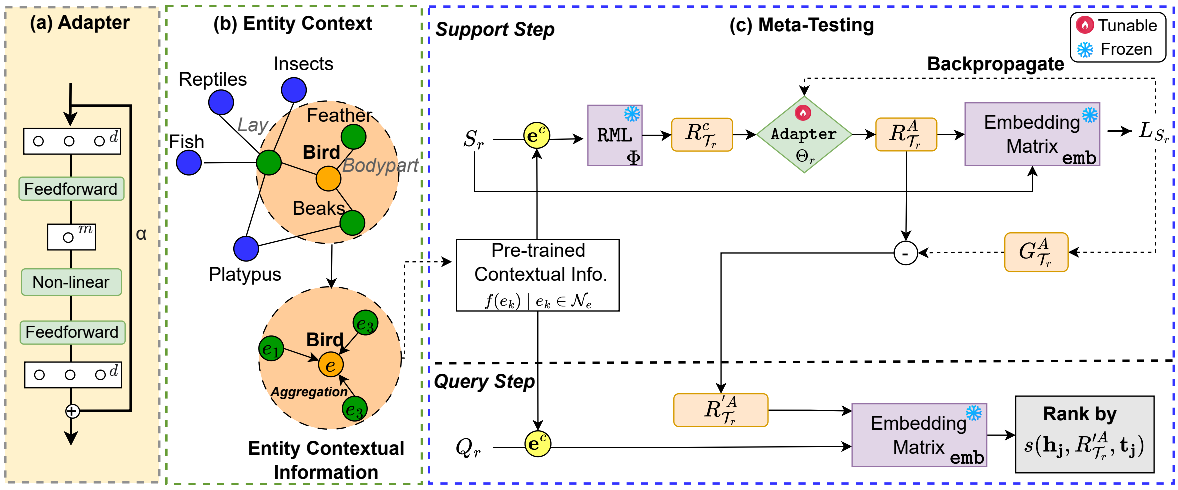



Figure 2: Illustration of key concepts in RelAdapter, hinging on an entity-aware adapter (a, b) in the meta-testing stage (c). Note that we omit the meta-training stage, which is similar to meta-testing but with backpropagation of the query loss to update the model parameters (emb and $\Phi$).

where $e \in \mathcal{V}$ is an entity, $N_e$ is the set of neighbors of $e$ in the knowledge graph (without the novel relations in meta-testing), and $\mu$ is a hyper-parameter. For each neighbor $e_k$ of $e$, we utilize the pre-trained encoder $f(\cdot)$ to extract its embedding. In summary, the input is the original entity embedding $\mathtt{emb}(e)$ and the mean contextual embedding aggregated from its neighbors $\mathcal{N}_e$, and the output is the augmented entity embedding $\mathbf{e}^c$. In this way, the augmented embedding $\mathbf{e}^c$ preserves the embedding trained via emb, while leveraging pre-trained graph contextual information.

Given the context-augmented entity embeddings $\mathbf{e}^c$ , we can derive the *context-aware* relation meta, $R^c_{\mathcal{T}_r}$, by rewriting Eq. (1) as follows.

$$R^c_{\mathcal{T}_r} = \mathtt{Mean}(\{\mathtt{RML}(\mathbf{h}^c, \mathbf{t}^c; \Phi) \mid \mathcal{S}_r\}) \quad (6)$$

Subsequently, we update the adapter in Eq. (4) to take in the context-aware relation meta, such that $R^A_{\mathcal{T}_r} = \mathtt{Adapter}(R^c_{\mathcal{T}_r}; \Theta_r)$.

### 4.2 Integration with Meta-learning

Following MetaR (Chen et al., 2019), our approach RelAdapter consists of the meta-training and meta-testing stages.

**Meta-training.** Our approach focuses on relation-specific adaption on novel relations in meta-testing, closing their gap from the seen relations due to divergence in distributions. Hence, our meta-training stage largely follows that of MetaR, as described in Sect. 3. The only difference is that, to have a consistent architecture to our meta-testing stage, we also include an adapter module in meta-training. Due to space constraint, we detail the description of the meta-training process in Appendix A.

**Meta-testing.** The process is outlined in Fig. 2(c). The meta-training stage learns the embedding matrix emb and prior $\Phi$ from the tasks on seen relations. To transfer the prior $\Phi$ to novel relations in meta-testing, it undergoes adaptation from two perspectives. As in traditional meta-learning, one form of adaptation involves a rapid fine-tuning step on the support set. However, meta-learning assumes that the seen and unseen relations are drawn from an identical distribution, but the distribution of a novel relation might diverge significantly from those of the seen relations. Hence, we propose an additional form of adaptation based on the context-aware adapter in Sect. 4.1.

In the following, we elaborate on the adaption process and how the adapter is tuned, and outline the overall algorithm for meta-testing.

**Adaptation.** On one hand, our adapter module is used to transform the context-aware relation meta $R^c_{\mathcal{T}_r}$, which is generated from the global prior $\Phi$ via Eq. (6), into a locally adapted version, i.e., $R^A_{\mathcal{T}_r} = \mathtt{Adapter}(R^c_{\mathcal{T}_r}; \Theta_r)$. On the other hand, the local relation meta is further adapted by a quick gradient step on the support set, following conventional MAML-based adaptation.

$$G^A_{\mathcal{T}_r} = \nabla_{R^A_{\mathcal{T}_r}} L_{\mathcal{S}_r}, \quad (7)$$

$$R'^A_{\mathcal{T}_r} = R^A_{\mathcal{T}_r} - \beta G^A_{\mathcal{T}_r}. \quad (8)$$

**Adapter tuning.** In the meta-testing stage, the global prior $\Phi$ and embedding $\texttt{emb}$ learned from meta-training are frozen, while only the adapter, parameterized by $\Theta_r$, is optimized in a task-wise manner for $r$-specific adaptation. First, $\Theta_r$ is randomly initialized for each unseen task $\mathcal{T}_r \in \mathcal{D}^{\text{te}}$ without leveraging any meta-learned knowledge, to overcome the potential divergence from seen tasks in $\mathcal{T}_r \in \mathcal{D}^{\text{tr}}$. Next, $\Theta_r$ is optimized on the support set $\mathcal{S}_r$, using the same support loss in Eq. (2). In other words, gradients on the support loss is back-propagated to update $\Theta_r$ within each task $\mathcal{T}_r$.

**Scoring.** The final adapted relation meta from Eq. (8), $R'^{A}_{\mathcal{T}_r}$, is used on the query set, to score and rank the candidates for the missing tail entities. The scoring function follows that of MetaR in Sect. 3. That is, for a given head $h$ from the query set $\mathcal{Q}_r$, for each candidate tail $t \in \mathcal{C}_{h,r}$, we compute $s(\texttt{emb}(h), R'^{A}_{\mathcal{T}_r}, \texttt{emb}(t)) = \|\texttt{emb}(h) + R'^{A}_{\mathcal{T}_r} - \texttt{emb}(t))\|$, and rank the candidates in $\mathcal{C}_{h,r}$ by the computed score.

**Algorithm.** We outline the algorithm of our meta-testing procedure in Algorithm 1. Compared to MetaR, the novel components in RelAdapter constitute the injection of contextual information specific to each target relation (lines 2–3), and the insertion of the adapter module (lines 5–8), which are integrated into a context-aware adapter. Despite the additional components, the overall time complexity remains unchanged and is linear to the number of shots. The overhead of the adapter module is negligible due to its parameter-efficient design.

**Algorithm 1** Meta-testing for RelAdapter

**Require:** Few-shot tasks for novel relations $\mathcal{D}^{\text{te}}$, embedding matrix $\texttt{emb}$, prior $\Phi$
1: **for** each task $\mathcal{T}_r = (S_r, Q_r) \in \mathcal{D}^{\text{te}}$ **do**
2: Compute context-augmented entity embeddings, $\{\mathbf{e}^c\}$, from Eq. (5);
3: Compute context-aware relation meta, $R^{c}_{\mathcal{T}_r}$, from Eq. (6);
4: **while** $\Theta_r$ not converged **do**
5: Compute adapted relation meta, $R^{A}_{\mathcal{T}_r} \leftarrow \texttt{Adapter}(R^{c}_{\mathcal{T}_r}; \Theta_r)$;
6: Compute support loss $L_{S_r}$ by Eq. (2);
7: Compute $G^{A}_{\mathcal{T}_r}$, the gradient of $R^{A}_{\mathcal{T}_r}$ by Eq. (7);
8: Update adapted relation meta, $R'^{A}_{\mathcal{T}_r} \leftarrow R^{A}_{\mathcal{T}_r} - \beta G^{A}_{\mathcal{T}_r}$ by Eq. (8);
9: Update $\Theta_r$ w.r.t. $L_{S_r}$;
10: **end while**
11: **for** each $(h, r, ?) \in \mathcal{Q}_r$ **do**
12: Rank candidates in $\mathcal{C}_{h,r}$ by the scoring function;
13: **end for**
14: **end for**

Table 1: Statistics of datasets.

| | Entities | Triplets | Relations Total | Pre-train | Train | Valid | Test |
|---|---|---|---|---|---|---|---|
| WIKI | 4,838,244 | 5,859,240 | 639 | 456 | 133 | 16 | 34 |
| FB15K-237 | 14,541 | 281,624 | 237 | 118 | 75 | 11 | 33 |
| UMLS | 135 | 6529 | 25 | 5 | 10 | 5 | 5 |

## 5 Experiments

In this section, we conduct comprehensive experiments on our proposed approach RelAdapter.

### 5.1 Experiment Setup

**Datasets.** We utilize three benchmark datasets, namely, WIKI, FB15K-237 and UMLS. Table 1 depicts the dataset details, including the pre-train/train/validation/test splits on the relations. The pre-trained encoder, $f(\cdot)$, which provides initial entity embeddings for the FSRL models, are based on the pre-train split. Note that the four splits are mutually exclusive to avoid information leakage (Zhang et al., 2020). Additional details for the dataset can be found in Appendix B.

**Metrics.** We employ two popular evaluation metrics, mean reciprocal rank (MRR) and hit ratio at top $N$ (Hit@$N$) to compare our approach against the baselines. Specifically, MRR reflects the absolute ranking of the first relevant item in the list and Hits@$N$ calculates the fraction of candidate lists in which the ground-truth entity falls within the first $N$ positions.

**Baselines.** RelAdapter is compared with a series of baselines in two major categories. (1) *Supervised relation learning*. They learn one model for all the relations in a supervised manner. We choose four classic and popular supervised methods: **TransE** (Bordes et al., 2013), **DistMult** (Yang et al., 2014), **ComplEx** (Trouillon et al., 2016) and **RGCN** (Schlichtkrull et al., 2018). We follow the same setup in GMatching (Xiong et al., 2018), which trains on the triplets combined from the pre-train and train splits, as well as the support sets of the test splits. (2) *Few-shot relation learning (FSRL)*. They are designed for few-shot relation prediction tasks. We choose several state-of-the-art FSRL methods, as follows: **GMatching** (Xiong et al., 2018), **FSKGC** (Zhang et al., 2020), **GANA** (Niu et al., 2021), **FAAN** (Sheng et al., 2020), **HiRe** (Wu et al., 2023), **MetaR** (Chen et al., 2019) and the details can be found in Appendix C.

**Implementation Details.** For a fair comparison,

Table 2: Performance comparison against baselines in the 3-shot setting. (Best: bolded, runners-up: underlined).

| | WIKI | | | FB15K-237 | | | UMLS | | |
|---|---|---|---|---|---|---|---|---|---|
| Models | MRR | Hit@10 | Hit@1 | MRR | Hit@10 | Hit@1 | MRR | Hit@10 | Hit@1 |
| TransE | .031±.007 | .043±.012 | .021±.014 | .294±.005 | .437±.011 | .204±.014 | .178±.036 | .310±.051 | .146±.068 |
| DistMult | .047±.003 | .082±.009 | .031±.011 | .234±.008 | .364±.007 | .208±.010 | .231±.035 | .337±.049 | .214±.067 |
| ComplEx | .093±.004 | .166±.011 | .071±.012 | .239±.007 | .359±.010 | .205±.013 | .251±.038 | .351±.041 | .227±.058 |
| RGCN | .217±.012 | .363±.023 | .188±.031 | .332±.011 | .495±.013 | .241±.031 | .409±.059 | .549±.072 | .389±.089 |
| GMatching | .133±.017 | .331±.013 | .114±.026 | .309±.019 | .441±.015 | .245±.019 | .296±.059 | .532±.040 | .257±.087 |
| FSKGC | .131±.003 | .267±.010 | .104±.016 | .355±.005 | .523±.004 | .217±.011 | .525±.031 | .682±.024 | .490±.038 |
| GANA | .291±.014 | .384±.012 | .272±.015 | <u>.388</u>±.004 | .553±.008 | **.301**±.017 | .541±.045 | .721±.076 | .502±.047 |
| FAAN | .278±.018 | .421±.020 | .275±.024 | .363±.009 | .542±.007 | .279±.013 | .545±.034 | .746±.120 | .505±.068 |
| HiRe | .300±.028 | <u>.444</u>±.012 | <u>.282</u>±.015 | .378±.013 | <u>.571</u>±.011 | .281±.015 | <u>.577</u>±.060 | <u>.752</u>±.066 | <u>.533</u>±.089 |
| MetaR | <u>.314</u>±.013 | .420±.016 | .274±.028 | .368±.007 | .536±.005 | .251±.012 | .435±.075 | .601±.095 | .417±.103 |
| RelAdapter | **.347**±.006 | **.454**±.012 | **.317**±.013 | **.405**±.012 | **.575**±.014 | <u>.297</u>±.019 | **.608**±.067 | **.780**±.044 | **.555**±.062 |

we initialize all FSRL models with pre-trained entity and relation embeddings, if needed. Other details can be found in Appendix B.

### 5.2 Comparison with Baselines

Table 2 reports the quantitative comparison of RelAdapter against other baselines in the 3-shot setting, i.e., the size of the support set is 3. (We study the effect of the number of shots in Sect. 5.5).

Overall, our model RelAdapter outperforms other baselines across all the three datasets, which demonstrates the benefit of incorporating context-aware adapter for FSRL. In particular, RelAdapter outperforms the most competitive baseline HiRe by 9.84% in terms of average MRR and 2.22% in terms of average Hit@10. Furthermore, we also draw the following observations.

First, *supervised relation learning* methods tend to perform worse as compared to *FSRL* methods as they are not designed to handle novel relations in few-shot setting. Meanwhile, as RGCN considers the neighborhood aggregated information, it consistently outperforms other supervised relation learning models across all the three datasets.

Second, among the *FSRL* methods, GMatching is designed for one-shot setting, and a simple mean pooling is applied to handle multiple shots, resulting in unsatisfactory performance. FSKGC generally performs better than GMatching as it extends the one-shot setting to more shots and explores new ways to encode neighbors with an attention mechanism. Although both GMatching and FSKGC consider neighborhood information, the simple neighborhood aggregation design is not expressive enough to capture complex relations. On the other hand, GANA and FAAN outperform GMatching and FSKGC as they consider the neighborhood information via more expressive neighborhood aggregators, employing a gated attentive neighborhood aggregator and a task-specific entity adaptive neighbor encoder, respectively. On top of the neighborhood information, HiRe considers triplet-level contextual information to generalize to few-shot relations, thus outperforming GANA and FAAN in most cases.

Table 3: Ablation study under the 3-shot setting. A: adapters in both meta-train and meta-test stage; C: using contextual information; $A^{tr}$: adapter in meta-training; $A^{te}$-Trf: adapter in meta-testing by transferring from meta-trained adapter. MetaR is considered a special variant without any of these components.

| | MRR | | |
|---|---|---|---|
| | WIKI | FB15K-237 | UMLS |
| MetaR | .314±.013 | .368±.007 | .435±.075 |
| W/o A | .312±.004 | .385±.007 | .389±.036 |
| W/o C | .332±.009 | .395±.007 | .590±.043 |
| W/o $A^{tr}$ | .343±.015 | .395±.012 | .583±.047 |
| $A^{te}$-Trf | .341±.008 | .394±.014 | .586±.046 |
| RelAdapter | **.347**±.006 | **.405**±.012 | **.608**±.067 |

Third, MetaR is the closest to RelAdapter in terms of the model architecture. In particular, the addition of context-aware adapter in the meta-learning framework enables more precise and relation-specific adaptation to each novel relation, mitigating the distribution shift between different relations. Hence, compared against MetaR, RelAdapter achieves an average improvement of 20.1% in MRR and 15.1% in Hit@10.

### 5.3 Ablation Study

To investigate the impact of various modules, we study four variants of our model, as shown in Table 3. (1) **W/o A:** We remove the adapter module entirely, while retaining the contextual information for entities. It shows a pronounced drop in per-

Table 4: Comparison of our adapter and MetaR in terms of number of parameters.

| | WIKI | FB15K-237 | UMLS |
|---|---|---|---|
| MetaR | 241,967,556 | 1,650,206 | 234,306 |
| Our adapter | 5,125 | 5,125 | 5,125 |
| % of MetaR | 0.002 | 0.311 | 2.187 |

formance and can be worse than MetaR, implying that simply adding contexts without further adaptation may introduce additional noises that harm the performance. In particular, the effect is the most significant on the UMLS dataset, which could be attributed to its smaller size. Specifically, the meta-learned model is more likely to overfit to the smaller meta-training data, and thus it becomes more important to have the adapter to deal with the distribution shift from meta-training. (2) **W/o C:** we remove the contextual information, while retaining the adapter. The lower performance shows that leveraging contexts can enhance model performance, if adapter is also present. (3) **W/o $\mathbf{A}^{\text{tr}}$:** We remove the adapter only from meta-training (see Appendix A). The decrease in performance justifies the need for a consistent architecture in both meta-training and -testing. (4) **$\mathbf{A}^{\text{te}}$-Trf:** we transfer the meta-trained adapter parameters to meta-testing, serving as the initialization for the adapter in meta-testing. In contrast, in RelAdapter, the adapter is randomly initialized in meta-testing. This variants suffer from a notable drop in performance, which is consistent with our earlier hypothesis on the divergence between relations. Specifically, due to the distribution shift in novel relations encountered in meta-testing, using prior adapter parameters from meta-training may be counterproductive.

### 5.4 Efficiency Analysis

We analyze the parameter and runtime efficiency of our adapter module.

**Parameter efficiency.** We first study the parameter overhead from the addition of adapter module. As shown in Table 4, the number of parameters in the adapter module is negligible w.r.t. MetaR. The parameter-efficient design implies that our adapter tuning is less likely to overfit to the few-shot examples. Also note that, compared to MetaR, the only new parameters of RelAdapter belong to the adapter module.

**Runtime efficiency.** As reported in Table 5, our approach RelAdapter generally incurs a lower or comparable total runtime for meta-training in comparison to various baselines. More specifically, compared to the most efficient approach MetaR, RelAdapter only takes slightly more time to complete the training. This observation shows that the parameter-efficient design of RelAdapter is able to achieve significant performance improvement with a decent runtime efficiency.

Table 5: Runtime (in seconds) for meta-training and meta-testing.

| Stage | Model | WIKI | FB15K-237 | UMLS |
|---|---|---|---|---|
| Meta-train (total) | GMatching | 28,941 | 19,678 | 12,061 |
| | FSKGC | 28,742 | 20,765 | 13,869 |
| | GANA | 35,374 | 28,167 | 15,081 |
| | FAAN | 32,036 | 23,675 | 11,302 |
| | HiRe | 34,257 | 27,736 | 12,213 |
| | MetaR | 22,691 | 16,504 | 8,802 |
| | RelAdapter | 24,085 | 17,529 | 9,656 |
| Meta-test (per instance) | GMatching | 0.009 | 0.004 | 0.039 |
| | FSKGC | 0.013 | 0.004 | 0.042 |
| | GANA | 0.017 | 0.006 | 0.064 |
| | FAAN | 0.016 | 0.005 | 0.053 |
| | HiRe | 0.036 | 0.007 | 0.051 |
| | MetaR | 0.012 | 0.005 | 0.043 |
| | RelAdapter | 0.045 | 0.008 | 0.053 |

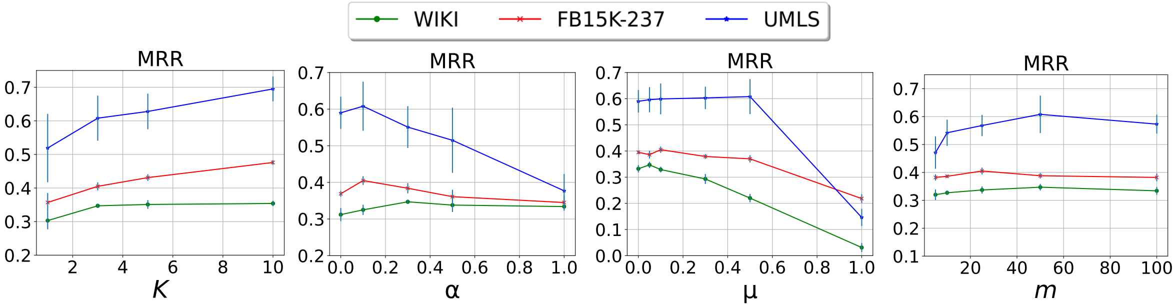
On the other hand, during meta-testing, we observe a notable increase in the average runtime per instance due to the relation-specific adaptation in RelAdapter. However, it remains on a similar order of magnitude as the baselines, with only a marginal absolute difference, especially when considering the much longer meta-training stage.

### 5.5 Sensitivity Analysis

Lastly, we conduct a sensitivity analysis for various settings and hyperparameters in Fig. 3. In particular, we vary the number of shots $K$ while fixing the hyperparamters, as well as several key hyperparameters while fixing $K = 3$. In the figures, the $x$-axis refers to the range of $K$ or parameters against the MRR metric in $y$-axis. The error bars represent the spread of standard deviation for each data point.

**Few-shot size $K$.** The few-shot size $K$ refers to the number of triplets in the support set of each relation. As shown in Fig. 3(a), when $K$ increases, we consistently observe performance improvement as more data becomes available for training.

**Adapter ratio $\alpha$.** As given in Eq. (4), Sect. 4.1, RelAdapter employs a hyperparameter $\alpha$ which controls the weight of the residual in the context-aware adapter. A bigger $\alpha$ means less residual, giving the adapter more transformative power to adapt to each relation. As shown in Fig. 3(b), the




(a) Few-shot size (b) Adapter ratio (c) Context ratio (d) Adapter neurons

Figure 3: Sensitivity analysis for the number of shots and hyperparameters.

performance peaks when $\alpha$ = 0.1 on FB15K-237 and UMLS, and $\alpha$ = 0.3 on WIKI. In general, [0.1, 0.3] appears as a robust range. Beyond the range, the performance starts to decrease, indicating that excessive adapter transformations are not beneficial. In particular, UMLS is more sensitive to the changes in $\alpha$. A potential reason is that UMLS is a relatively small dataset in a focused domain with less distribution shifts (see Fig. 1). Hence, it requires less adaptation across relations.

**Context ratio** $\mu$. Similar to $\alpha$, *RelAdapter* controls the weight of contextual information via the hyper-parameter $\mu$ in Eq. 5, Sect. 4.1. As illustrated in Fig. 3(c), a smaller contextual ratio generally helps to improve the model performance across all datasets, while an excessively large ratio could bring in more noises and hurt the performance. In general, [0.1,0.3] appears to be a good range.

**Adapter neurons** $m$**.** We analyze how the number of neurons $m$ in the hidden layer of the adapter network affects model performance. Results in Fig. 3(d) show that the optimal $m$ is around 25 for FB15K-237, and 50 for WIKI and UMLS. Increasing $m$ further leads to a plateau in performance, suggesting the effectiveness of the bottleneck structure and ensuring a parameter-efficient design.

**Number of hops for contexts.** We investigate the impact of the number of hops considered in neighborhood contexts, $\mathcal{N}_e$, in Eq. (5). Due to space limit, we present the results in Appendix E.

## 6 Conclusion

In this paper, we proposed RelAdapter, a context-aware adapter for few-shot relation learning (FSRL). We investigated the limitation of FSRL methods in prevailing meta-learning frameworks, which rely on the i.i.d. assumption. This assumption may not hold for novel relations with distribution shifts from the seen relations. Based on this insight, we introduced a context-aware adapter module, enabling relation-specific, tunable and fine-grained adaptation for each distinct relation. Extensive experiments were conducted on three benchmark datasets, demonstrating the superior performance of RelAdapter.

## Limitations

In RelAdapter, one potential limitation is that the context-aware adapter module is currently only integrated into the MetaR framework. Despite the significant improvement in performance, it would be ideal to explore the integration of RelAdapter with other meta-learning frameworks in general.

## Acknowledgement

This research was supported by the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant (22-SIS-SMU-054) and National Research Foundation, Singapore under its AI Singapore Programme (AISG2-RP-2021-027). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education or National Research Foundation, Singapore. Ran Liu is also supported by the A*Star Graduate Scholarship offered by the Agency for Science, Technology and Research, Singapore for his PhD study.

## References

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

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

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS*.

Mingyang Chen, Wen Zhang, Wei Zhang, Qiang Chen, and Huajun Chen. 2019. Meta relational learning for few-shot link prediction in knowledge graphs. In *EMNLP*, pages 4208–4217.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pages 1126–1135.

Anchun Gui, Jinqiang Ye, and Han Xiao. 2024. G-Adapter: Towards structure-aware parameter-efficient transfer learning for graph transformer networks. In *AAAI*, pages 12226–12234.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *ICML*, pages 2790–2799.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-rank adaptation of large language models. In *ICLR*.

Qian Huang, Hongyu Ren, and Jure Leskovec. 2022. Few-shot relational reasoning via connection subgraph pretraining. In *NeurIPS*, pages 6397–6409.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL/IJCNLP*, pages 687–696.

Diederik P Kingma et al. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Haoyang Li, Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2022. Ood-gnn: Out-of-distribution generalized graph neural network. *TKDE*, 35(7):7328–7340.

Shengrui Li, Xueting Han, and Jing Bai. 2024. AdapterGNN: Efficient delta tuning improves generalization ability in graph neural networks. In *AAAI*, pages 13600–13608.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL/IJCNLP*, pages 4582–4597.

Haochen Liu, Song Wang, Chen Chen, and Jundong Li. 2024. Few-shot knowledge graph relational reasoning via subgraph adaptation. In *NAACL*, pages 3346–3356.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2023. Normalizing flow-based neural process for few-shot knowledge graph completion. In *SIGIR*, pages 900–910.

Lingyuan Meng, Ke Liang, Bin Xiao, Sihang Zhou, Yue Liu, Meng Liu, Xihong Yang, Xinwang Liu, and Jinyan Li. 2024. SARF: Aliasing relation–assisted self-supervised learning for few-shot relation reasoning. *TNNLS*.

Guanglin Niu, Yang Li, Chengguang Tang, Ruiying Geng, Jian Dai, Qiao Liu, Hao Wang, Jian Sun, Fei Huang, and Luo Si. 2021. Relational learning with gated and attentive neighbor aggregator for few-shot knowledge graph completion. In *SIGIR*, pages 213–222.

Byungkook Oh, Seungmin Seo, and Kyong-Ho Lee. 2018. Knowledge graph completion by context-aware convolutional learning with multi-hop neighborhoods. In *CIKM*, pages 257–266.

Wessel Radstok, Mel Chekol, Mirko Schaefer, et al. 2021. Are knowledge graph embedding models biased, or is it the data that they are trained on? In *Wikidata Workshop, ISWC*.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*, pages 593–607.

Jiawei Sheng, Shu Guo, Zhenyu Chen, Juwei Yue, Lihong Wang, Tingwen Liu, and Hongbo Xu. 2020. Adaptive attentional network for few-shot knowledge graph completion. In *EMNLP*, pages 1681–1691.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW*, pages 697–706.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge graph embedding by relational rotation in complex space. In *ICLR*.

Zhaoxuan Tan, Zilong Chen, Shangbin Feng, Qingyue Zhang, Qinghua Zheng, Jundong Li, and Minnan Luo. 2023. KRACL: Contrastive learning with graph context modeling for sparse knowledge graph completion. In *WWW*, pages 2548–2559.

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

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

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

Song Wang, Xiao Huang, Chen Chen, Liang Wu, and Jundong Li. 2021. REFORM: Error-aware few-shot knowledge graph completion. In *CIKM*, pages 1979–1988.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119.

Han Wu, Jie Yin, Bala Rajaratnam, and Jianyuan Guo. 2023. Hierarchical relational learning for few-shot knowledge graph completion. In *ICLR*.

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2018. One-shot relational learning for knowledge graphs. In *EMNLP*, pages 1980–1990.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*.

Chuxu Zhang, Huaxiu Yao, Chao Huang, Meng Jiang, Zhenhui Li, and Nitesh V Chawla. 2020. Few-shot knowledge graph completion. In *AAAI*, pages 3041–3048.

# Appendices

## A Meta-training stage

In the meta-train stage as illustrated in Fig. 4, we update the model with the query loss $\mathcal{L}_{\mathrm{Qr}}$ in the same way as MetaR. In summary, we formulate the trainable parameters of the meta-train stage as $\Phi$, $\Theta$, $\texttt{emb}$ , where $\Phi$ is the set of trainable parameters in relation-meta learner $\texttt{RML}$, $\Theta$ is the set of trainable parameters of the adapter module and $\texttt{emb}$ is the embedding learner. The output of the meta-training stage includes the set of meta-trained entity embedding $\texttt{emb}$ as well as the relation-meta learner $\texttt{RML}$, which will be further used to adapt to downstream tasks in the meta-testing stage. The purpose of introducing the adapter network into the meta-training stage is to maintain consistency with the architecture of the meta-testing stage, which can improve performance.

## B Datasets

We utilize three benchmark datasets, namely, WIKI, FB15K-237 and UMLS. On each dataset, the relations are divided into four subsets: pre-training, training, validation and test, as shown in Table 1. For the smaller dataset UMLS, all relations with less than 50 triplets are removed.

For supervised models not designed for FSRL (TransE, DistMult and ComplEx from OpenKE[3], and RGCN[4]), we follow the same settings in GMatching (Xiong et al., 2018) by using all triplets in the pre-train and train splits, as well as the support sets from the valid/test splits to train the models. For other FSRL models (GMatching[5], FSKGC[6], MetaR[7], GANA[8], FAAN[9], HiRE[10]), we follow the same FSRL splits as RelAdapter. All results are averaged among 5 runs.

Note that, as FB15K contain many inverse triplets which can cause leakage during training, it has been omitted and replaced with its subset FB15K-237 (Toutanova and Chen, 2015) which has all the inverse triplets removed. In addition, the popular WN18 and WN18RR datasets have also been omitted, as they contain insufficient relations and therefore are not suitable to be used in our experiments.

## C Baselines

Few-shot relation learning (FSRL) are designed for few-shot relation prediction tasks, where the testing relations are previously unseen in pre-training or training. We choose several state-of-the-art FSRL methods, as follows: **GMatching** (Xiong et al., 2018) uses a neighbor encoder and a matching network, assuming that all neighbors contribute equally. **FSKGC** (Zhang et al., 2020) encodes neighbors with a fixed attention mechanism, and applies a recurrent autoencoder to aggregate the few-shot instances in the support set. **GANA** (Niu et al., 2021) improves on FSKGC by having a gated and attentive neighbor aggregator to capture valuable contextual semantics of each few-shot relation. **FAAN** (Sheng et al., 2020) introduces an adaptive neighbor encoder for different relation tasks. **HiRe** (Wu et al., 2023) brings in a hierarchical relational learning framework which considers triplet-level contextual information in contrastive learning. **MetaR** (Chen et al., 2019) utilizes a MAML-based framework, which aims to learn a good initialization for the unseen relations, followed by an optimization-based adaptation.

[3]https://github.com/thunlp/OpenKE
[4]https://github.com/JinheonBaek/RGCN
[5]https://github.com/xwhan/One-shot-Relational-Learning
[6]https://github.com/chuxuzhang/AAAI2020_FSRL
[7]https://github.com/AnselCmy/MetaR
[8]https://github.com/ngl567/GANA-FewShotKGC
[9]https://github.com/JiaweiSheng/FAAN
[10]https://github.com/alexhw15/HiRe

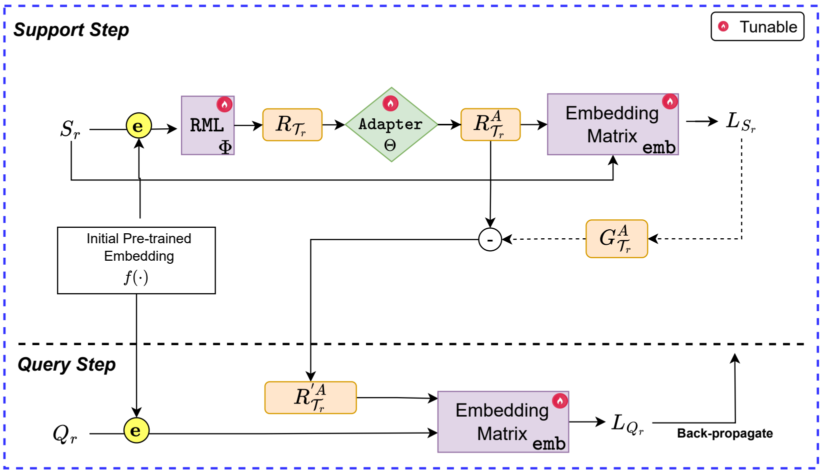



Figure 4: Illustration of the meta-training stage.

Table 6: Tuned hyperparameter settings based on validation data.

| | Hyperparameters | Range of values | Wiki | FB15K-237 | UMLS |
|---|---|---|---|---|---|
| TransE | norm | 1,2 | 1 | 2 | 1 |
| DistMult | norm | 1,2 | 1 | 1 | 1 |
| ComplEx | norm | 1,2 | 1 | 1 | 1 |
| RGCN | dropout | 0.1,0.2,0.5 | 0.2 | 0.2 | 0.1 |
| Gmatching | aggregate | mean, max, sum | max | mean | max |
| MetaR | beta | 1,3,5,10 | 5 | 3 | 5 |
| FSRL | aggregate | mean, max, sum | max | max | max |
| GANA | beta | 1,3,5,10 | 5 | 5 | 5 |
| FAAN | dropout_input | 0.1,0.3,0.5,0.8 | 0.5 | 0.3 | 0.5 |
| HIRE | beta | 1,3,5,10 | 5 | 5 | 10 |
| RelAdapter | $\alpha$, $\mu$ | $0.05 \sim 1.0$ | $\alpha = 0.5$, $\mu = 0.05$ | $\alpha = 0.1$, $\mu = 0.1$ | $\alpha = 0.1$, $\mu = 0.3$ |

## D Implementation details

For WIKI and FB15K-237, we directly use the pre-trained embeddings provided in (Xiong et al., 2018; Wang et al., 2021). For UMLS, we obtain the pre-trained embedding using the popular TransE-pytorch implementation by Mklimasz[11]. Throughout all the experiments, the embedding dimension is set to 100 for FB15K-237 and UMLS, and 50 for WIKI. Where applicable, the maximum number of neighbors of one given entity is set to 50. All results reported are on the candidate set after removing relations with less than 10 candidates. For each model, some settings are tuned using the validation set, while the others follow their respective original papers. More details on the hyperparameter settings can be found in Table 6.

[11] https://github.com/mklimasz/TransE-PyTorch

We train RelAdapter for 100,000 epochs, and select the most optimal model based on the validation relations every 1,000 epochs with early stopping for a patience setting of 30. The mini-batch gradient descent is applied with batch size set as 64 for FB15K-237 and UMLS, and 128 for WIKI. The number of hidden neurons is set as 50 for all datasets. We use Adam (Kingma et al., 2015) with the initial learning rate of 0.001 to update parameters. The intensity of gradient update is fixed at 5. The number of positive and negative triplets in each query set is 3 in FB15K-237 and UMLS, and 10 in WIKI. All experiments are conducted on an RTX3090 GPU server in Linux.

## E Number of hops in $\mathcal{N}_e$

As observed in Table 7, increasing the number of hops considered in neighborhood contexts $\mathcal{N}_e$ for

Table 7: Impact of the number of hops in neighborhood contexts $\mathcal{N}_e$ in the 3-shot setting.

| Dataset | No. of hops | MRR | Hit@10 |
|---|---|---|---|
| Wiki | 1-hop | 0.347±.006 | 0.454±.012 |
| | 2-hop | 0.353±.009 | 0.457±.010 |
| | 3-hop | 0.321±.005 | 0.415±.009 |
| FB15K-237 | 1-hop | 0.405±.012 | 0.575±.014 |
| | 2-hop | 0.402±.010 | 0.568±.016 |
| | 3-hop | 0.379±.014 | 0.531±.014 |
| UMLS | 1-hop | 0.608±.067 | 0.780±.044 |
| | 2-hop | 0.543±.051 | 0.697±.049 |
| | 3-hop | 0.454±.048 | 0.612±.046 |

data augmentation generally lead to performance degradation. For instance, on the UMLS dataset, the performance at the 3-hop setting is reduced by 25.3% in MRR compared to the original 1-hop setting in RelAdapter. This could be due to the additional noise introduced when considering a broader neighborhood, which can adversely affect model performance, especially for smaller datasets like UMLS, as they are more vulnerable to noise during model training.