Multi-Relational Graph Representation Learning with Bayesian Gaussian Process Network

Guanzheng Chen†,‡, Jinyuan Fang†,‡, Zaiqiao Meng§, Qiang Zhang$^{\triangledown,\triangle}$, Shangsong Liang†,‡,¶,*

† School of Computer Science and Engineering, Sun Yat-sen University, China
‡ Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, China
§ School of Computing Science, University of Glasgow, United Kingdom
$^{\triangledown}$ Hangzhou Innovation Center, Zhejiang University, China
$^{\triangle}$ College of Computer Science and Technology, Zhejiang University, China
¶ Department of Machine Learning, Mohamed bin Zayed University of Artificial Intelligence, United Arab Emirates
{guanzzh.chen, fangjy6}@gmail.com, zaiqiao.meng@glasgow.ac.uk, qiang.zhang.cs@zju.edu.cn,
liangshangsong@gmail.com

Abstract

Learning effective representations of entities and relations for knowledge graphs (KGs) is critical to the success of many multi-relational learning tasks. Existing methods based on graph neural networks learn a deterministic embedding function, which lacks sufficient flexibility to explore better choices when dealing with the imperfect and noisy KGs such as the scarce labeled nodes and noisy graph structure. To this end, we propose a novel multi-relational graph Gaussian Process network (GGPN), which aims to improve the flexibility of deterministic methods by simultaneously learning a family of embedding functions, i.e., a stochastic embedding function. Specifically, a Bayesian Gaussian Process (GP) is proposed to model the distribution of this stochastic function and the resulting representations are obtained by aggregating stochastic function values, i.e., messages, from neighboring entities. The two problems incurred when leveraging GP in GGPN are the proper choice of kernel function and the cubic computational complexity. To address the first problem, we further propose a novel kernel function that can explicitly take the diverse relations between each pair of entities into account and be adaptively learned in a data-driven way. We address the second problem by reformulating GP as a Bayesian linear model, resulting in a linear computational complexity. With these two solutions, our GGPN can be efficiently trained in an end-to-end manner. We evaluate our GGPN in link prediction and entity classification tasks, and the experimental results demonstrate the superiority of our method. Our code is available at https://github.com/sysu-gzchen/GGPN.

1 Introduction

Knowledge graphs (KGs), as high-profile multi-relational graphs, are composed of entities as nodes and relations as the types of edges to store plenty of factual knowledge. Analyzing KGs has become one of the most important topics in machine learning community, with a wide spectrum of applications such as question answering (Liang, Luo, and Meng 2021; Luo, Liang, and Meng 2019), personalized recommendation (Qian et al. 2013), and relation extraction (Zhou et al. 2005; Qu et al. 2020).

Recent researches on KGs have been focusing on learning representations for entities and relations that encode semantic and structural information inherent in KGs. Particularly, multi-relational graph neural networks (Schlichtkrull et al. 2018; Vashishth et al. 2020b; Nathani et al. 2019), which iteratively aggregate messages from neighboring entities, have achieved impressive performance on multi-relational learning tasks. However, these methods learn a deterministic embedding function, which makes them less effective to handle the imperfect and noisy KGs. For instance, the labeled entities in KGs are usually scarce due to the expensiveness and difficulty of human annotation, and missing or wrong edges are also common in KGs (Ji et al. 2021). Learning with these imperfect and noisy KGs may lead to the poor performance of deterministic embedding methods in entity classification task, due to the rigidness of deterministic methods and lacking sufficient supervision signals (Wang et al. 2020).

To address these challenges, we propose a novel multi-relational **G**raph **G**aussian **P**rocess **N**etwork (GGPN) for learning KG representations. GGPN aims to simultaneously learn a family of embedding functions, i.e., a stochastic embedding function (Ross et al. 1996), instead of learning a deterministic function. The learned stochastic function of GGPN offers more flexibility to handle the imperfections or noises in KGs, effectively improving the performance. Specifically, we model the stochastic embedding function with Gaussian Process (GP), which defines a distribution over functions in a continuous domain (Rasmussen 2003). The resulting representations are obtained through aggregating the stochastic function values, i.e., messages, from neighboring entities associated with different relations.

Despite using GP for representation learning is promising, there are two additional problems incurred by GP remain to be solved: (1) Existing kernel functions, e.g., RBF (Bergman 1970), which heavily affect the performance of GPs, are incapable of taking diverse relations between each pair of entities into account and difficult to adapt to the training data. (2)

[*]Corresponding author.

It has cubic computational complexity (Rasmussen 2003) with respect to the number of entities to learn the network parameters through GP. To address the first problem, we further propose a novel relation-aware kernel function. The proposed kernel function firstly extracts the relation-specific information of an entity using an *information extractor* and then uses a base kernel to obtain the final value. To make our kernel be adaptive to the data, we propose to define the base kernel based on the Bochner's theorem (Rudin 2017), whose parameters can be learned in a data-driven way. Moreover, we address the second problem by leveraging the equivalence between GP and Bayesian linear model. Specifically, we reformulate GP as a Bayesian linear model with linear computational complexity, bypassing the cubic computational complexity of GP. Consequently, the network parameters and kernel parameters can be jointly learned efficiently by optimizing the training objectives in an end-to-end manner. Our contributions can be summarized as:

- We propose a novel GGPN framework to learn a family of embedding functions, i.e., a stochastic embedding function, to handle the imperfect and noisy KGs.
- We propose a novel relation-aware kernel function in our GGPN framework, which can explicitly leverage the diverse relations between each pair of entities and be adaptive to KGs.
- We leverage the equivalence between GP and Bayesian linear model to reformulate our GGPN framework to reduce the computational complexity, making our method efficiently trainable in an end-to-end manner.
- The experimental results in both link prediction and entity classification tasks demonstrate the superior performance of the proposed GGPN.

2 Related Work

Knowledge Graph Embeddings. Recently, various KG embedding methods have been proposed. These methods mainly fall into three different categories: (1) Translation-based methods, such as TransE (Bordes et al. 2013). (2) Tensor factorization based methods, such as DistMult (Yang et al. 2015) and ComplEx (Trouillon et al. 2016). (3) Neural network based methods such as ConvE (Dettmers et al. 2018) and DBKGE (Liao et al. 2021). However, these methods do not leverage the local neighborhood information of an entity, which might potentially improve the quality of embeddings.

To this end, some multi-relational graph convolution networks (RGCNs) (Schlichtkrull et al. 2018; Ye et al. 2019; Bansal et al. 2019; Vashishth et al. 2020b) have been proposed. They extend graph convolution networks (GCNs) (Kipf and Welling 2017) to handle multi-relational graphs, which obtain embeddings of KGs based on the message passing framework. Furthermore, RGCNs based on the attention mechanism have also been proposed (Gilmer et al. 2017; Nathani et al. 2019). These attention-based methods aggregate messages from neighbors by assigning respective weights. However, aforementioned methods learn a single deterministic function for relational learning tasks, which may cause poor performance due to the imperfections or noises (e.g., scarce labels or wrong edges) in KGs (Arora 2020; Ji et al. 2021). In contrast, our GGPN learns a family of functions, i.e., a stochastic function, offering more flexibility to handle these imperfections or noises in KGs.

Gaussian Processes for Graphs. GP methods for graphs have been developed in the relational learning discipline. They have been proposed with different emphases on different relational learning tasks, such as object classification (Sindhwani, Ghahramani, and Keerthi 2007; Xu, Kersting, and Tresp 2009; Ng, Colombo, and Silva 2018), link prediction (Yu et al. 2007; Opolka and Liò 2020) and graph classification (Li et al. 2020). However, the kernel functions applied in these methods do not consider the relations among entities, which are vital indicators of their similarities. To address this problem, (Fang et al. 2021a) propose a graph convolutional kernel to incorporate graph structures in the kernel functions. However, it assumes a single relation among entities, i.e., edges belong to the same type, which makes it impractical in our multi-relational learning setting. In contrast, to effectively tackle multi-relational learning problems, our GGPN proposes a novel relation-aware kernel function, which can explicitly leverage the diverse relations among entities for calculating similarities.

3 Preliminary

Gaussian Process (GP) (Rasmussen 2003) defines a stochastic function over a set of inputs, which assumes that the marginal distribution over function values of any finite inputs is a Gaussian. Specifically, a GP places Gaussian prior on the function values, which is denoted as: $f(\mathbf{x}) \sim \mathcal{N}(\mathbf{0}, k_\theta(\mathbf{x}, \mathbf{x}'))$, where $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ denote the inputs, and $k_\theta(\mathbf{x}, \mathbf{x}')$ is the kernel function parameterized by θ. The kernel function is a key component of GP as it implicitly imposes assumptions on the statistical structure of data.

It's worth noting that any kernel function can be expressed as the inner product of feature maps $\langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{H}}$ in Hilbert space (Young 1988), where φ is a function that projects $\mathbf{x}$ into a high-dimensional (possibly infinite) Hilbert space $\mathcal{H}$. With this insight, researchers have found that GP is equivalent to a Bayesian linear model whose weights follow Gaussian distributions (Rasmussen 2003; Flam-Shepherd, Requeima, and Duvenaud 2017). Therefore, a GP can be reformulated as: $f(\mathbf{x}) = \varphi(\mathbf{x})^\top \mathbf{w}$, where $\mathbf{w}$ is assigned with a Gaussian prior.

4 Methodology

In this section, we introduce the technical details of the proposed method.

4.1 Notations and Task

We denote a knowledge graph as $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$, where $\mathcal{E}$ and $\mathcal{R}$ denote the set of entities (nodes) and relations (types of edges), respectively, and $\mathcal{T}$ denotes the set of facts with the form of triples: $\{(e, r, o) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}\}$. To enable the bidirectional flow of messages, as shown by the "**KG**" in Figure 1, we further extend $\mathcal{T}$ and $\mathcal{R}$ with corresponding inverse relations, i.e., $\mathcal{T}' = \mathcal{T} \cup \{(o, r^{-1}, e) \mid (e, r, o) \in$

$\mathcal{T}\} \cup \{(e, \tau, e) \mid e \in \mathcal{E}\}$ and $\mathcal{R}' = \mathcal{R} \cup \mathcal{R}_{inv} \cup \tau$, where $\mathcal{R}_{inv} = \{r^{-1} | r \in \mathcal{R}\}$ denotes the inverse relations and τ is the self-loop relation. Given the knowledge graph, the goal of our model is to learn representations of both entities and relations, i.e., $\mathbf{H} \in \mathbb{R}^{(N_e+N_r)\times d}$, where N_e, N_r and d denote the total number of entities $\mathcal{E}$, the total number of relations $\mathcal{R}'$, and the embedding dimension, respectively,

4.2 Overview of Our GGPN

The overall architecture of our method is show in Figure 1. The proposed method is a general framework for a variety of multi-relational learning tasks, and follows an encoder-decoder structure (Cho et al. 2014). The encoder is the proposed L-layered GGPN, which aims to learn the entity and relation embeddings. Subsequently, these embeddings are utilized in the decoder for practical tasks, such as link prediction and entity classification.

Specifically, in each layer of GGPN, GP with proposed relation-aware kernel is leveraged to model the stochastic embedding function over entities. Afterwards, the entity embeddings are obtained with a relation-aware weighted aggregation mechanism, which aggregates the stochastic function values, i.e., messages, from neighboring entities for a given entity. Additionally, the relation embeddings are updated with a linear transformation. The choice of decoder can be task-specific, which means that we can use different decoders based on the learning tasks. For instance, in link prediction task, the decoder can be chosen as TransE (Bordes et al. 2013), DistMult (Yang et al. 2015), or Conve (Dettmers et al. 2018), while the $\mathrm{softmax}$ decoder can be applied to the entity classification task. The pseudocode of our algorithm is provided in Section A of the Appendix.

4.3 Multi-Relational Gaussian Process Network

In this section, we introduce details of our GGPN. We firstly introduce a single layer of GGPN and then generalize it to a multi-layer setting. We denote the input features of entities and relations to the l-th layer as $\mathbf{H}^l \in \mathbb{R}^{(N_e+N_r)\times d}$. To learn a family of embedding functions, we define a vector-valued stochastic function f on entities, the prior of which is specified with GP. Denoting the function values as $\mathbf{F}^l$, i.e. $\mathbf{F}^l = f(\mathbf{H}^l)$, we can obtain its prior as $p(\mathbf{F}^l \mid \mathbf{H}^l, \mathcal{G}) = \mathcal{N}(\mathbf{0}, \mathbf{K}^l)$, where $\mathbf{K}^l$ is the covariance matrix with each component being the covariance between two entities, which is obtained with a kernel function.

One critical problem in GP is to choose a proper kernel to capture the underlying statistical structure of data. Off-the-shelf kernel functions, such as the Radial Basis Function (RBF) kernel (Bergman 1970) and deep kernel (Wilson et al. 2016), measure similarities based on the entity features only, and hence are incapable of taking diverse relations between each pair of entities into account. However, the relations between each pair of entities are vital indicators for their similarities. Intuitively, *Joe Biden* and *USA* should be more similar under the relation *President_of* than the relation *Born_in*. To explicitly consider the entity relationships, we propose a novel relation-aware kernel function:

$$k(e, r, o) = \tilde{k}_\theta\big(\mathbf{g}(\boldsymbol{h}_e^l, \boldsymbol{h}_r^l), \mathbf{g}(\boldsymbol{h}_o^l, \boldsymbol{h}_r^l)\big) , \qquad (1)$$

where $\boldsymbol{h}_e^l, \boldsymbol{h}_o^l, \boldsymbol{h}_r^l \in \mathbf{H}^l$ denote the features of corresponding entities and relation. Here, $\mathbf{g}$ is an *information extractor* to obtain the specific information of an entity associated with a relation, and is defined as:

$$\mathbf{g}(\boldsymbol{h}_e^l, \boldsymbol{h}_r^l) = \mathbf{W}(\boldsymbol{h}_e^l \odot \boldsymbol{h}_r^l) , \qquad (2)$$

where $\odot$ denotes the Hadamard (element-wise) multiplication and $\mathbf{W} \in \mathbb{R}^{d\times d}$ is a linear transformation matrix, which is shared across layers. After the relation-aware features are obtained with $\mathbf{g}$, we use a base kernel function $\tilde{k}_\theta$ to obtain the final kernel value. Since applying a deterministic transformation to kernel inputs results in a valid kernel (MacKay 1998), the function defined in (1) is a valid kernel function. After defining the relation-aware kernel function, we can use it to calculate the kernel matrix $\mathbf{K}^l$, and hence obtain the GP prior over the entity function values $\mathbf{F}^l$.

Inspired by the success of graph neural networks, which iteratively aggregate messages from neighborhoods, we further leverage a relation-aware aggregation mechanism to obtain the layer output. The output embedding of an entity is obtained with the weighted aggregation of messages from its neighboring entities associated with diverse relations. Specifically, the output embedding of entity e is given by:

$$\boldsymbol{h}_e^{l+1} = \psi\left(\sum_{(r,o)\in\mathcal{N}(e)} \alpha_e^{r,o}\mathbf{W}_{\lambda(r)}^l \mathbf{F}_o^l\right) , \qquad (3)$$

where ψ is the activation function such as Relu, and $\mathcal{N}(e)$ denotes the neighbors of entity e, i.e., $\mathcal{N}(e) = \{(r, o)|(e, r, o) \in \mathcal{T}'\}$. $\mathbf{W}_{\lambda(r)}^l \in \mathbb{R}^{d\times d}$ is the aggregation parameters based on the relation type, where $\lambda(r)$ is the relation type function that returns the specific type of relation r. In this paper, we use three different relation types: real relation (T), inverse relation (I) and self-loop relation (L), and we define the function λ as: $\lambda(r) = \mathrm{T}$, if $r \in \mathcal{R}$; $\lambda(r) = \mathrm{I}$, if $r \in \mathcal{R}_{inv}$; and $\lambda(r) = \mathrm{L}$, otherwise.

Moreover, we also propose to attend differently to neighboring messages by assigning different attentions. The attention of a neighboring entity o is denoted as $\alpha_e^{r,o}$ in (3). We can readily use the kernel values in GP to define the attentions since the kernel values have already considered the similarities between two entities under specific relations. Specifically, the proposed kernel attention is defined as:

$$\alpha_e^{r,o} = \frac{\exp\big(k(e, r, o)\big)}{\sum_{(r',o')\in\mathcal{N}(e)} \exp\big(k(e, r', o')\big)} , \qquad (4)$$

where we use a softmax operation for normalization.

After obtaining the updated representations of entities at l-th layer, the relation embeddings are also updated as:

$$\boldsymbol{h}_r^{l+1} = \mathbf{W}_{rel}^l \boldsymbol{h}_r^l . \qquad (5)$$

Consequently, a layer of GGPN consists of two major components, where the entity embeddings are firstly updated with a GP followed by the relation-aware weighted aggregation, and the relation embeddings are updated with a linear transformation. It is straightforward to extend to the multi-layer setting, where we simply use the output embeddings of the previous layer as the input to the next layer. We use the outputs of the final layer as the desired embeddings, i.e., $\mathbf{H} = \mathbf{H}^L$, where L is the number of layers.

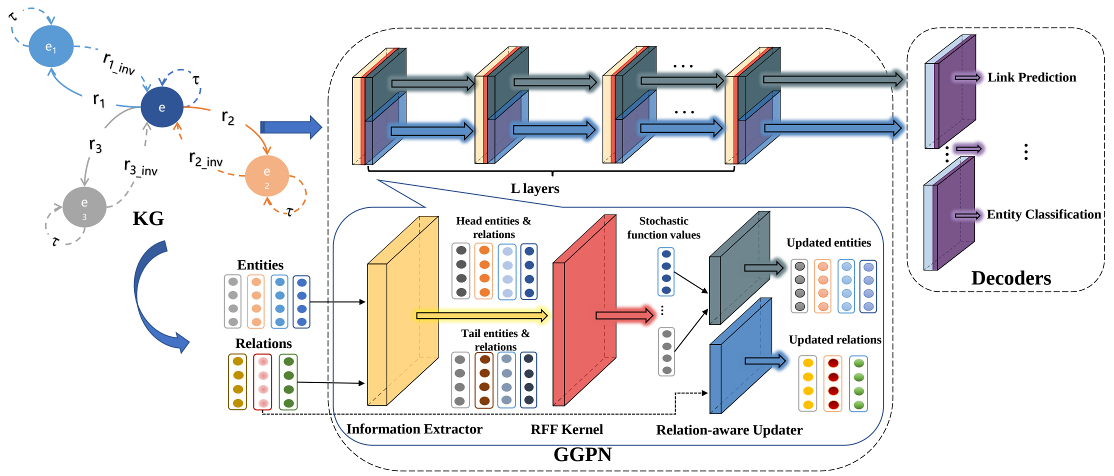

Figure 1: The overall architecture of GGPN. Given entity and relation embeddings, each layer uses three components: *Information Extractor*, *RFF kernel*, and *Relation-aware Updater*, to update the embeddings, where **head / tail entities & relations** generated by *Information Extractor* denotes the correlated information of head/tail entities associated with relations.

4.4 Adaptive and Scalable Kernel Learning

There are still two more problems in the proposed method: the first one is that we need to choose a proper base kernel $\tilde{k}_\theta$ in (1) to capture the similarities between relation-aware features. We can use off-the-shelf kernel functions, such as RBF kernel. However, the parameters of these kernel functions can hardly be learned (Rasmussen 2003) and hence they are difficult to adapt to the data, which can be problematic when modeling complicated structured data such as KGs. The second problem is that it is computationally expensive to learn the network parameters, since inferring through GP has a cubic computational complexity (Rasmussen 2003) with respect to the number of entities due to matrix inversion.

To address the first problem*, we propose a novel kernel function that can be adaptively learned in a data-driven way based on the Bochner's theorem.* The Bochner's theorem (Rudin 1962; Rahimi and Recht 2008) is given by:

Theorem 1 (Bochner's theorem) *A continuous, symmetric and translation-invariant functions $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{r})$ on $\mathbb{R}^d$, where $\mathbf{r} = \mathbf{x} - \mathbf{x}'$, is a positive definite kernel if and only if it is the Fourier transform of a positive finite measure.*

The Bochner's theorem states that we can define a kernel function through the Fourier transform of a probability distribution. Inspired by this insight, we firstly specify a probability distribution $p(\boldsymbol{\omega})$, where $\boldsymbol{\omega}$ is a random variable, and define a kernel function through the Fourier transform of this probability distribution:

$$\hat{k}_\theta(\mathbf{x}, \mathbf{x}') = \int_{\mathbb{R}} e^{i\boldsymbol{\omega}^\top(\mathbf{x}-\mathbf{x}')} p(\boldsymbol{\omega}) d\boldsymbol{\omega}. \qquad (6)$$

By extracting the real part of above equation, we can obtain:

$$\begin{aligned}\hat{k}_\theta(\mathbf{x}, \mathbf{x}') &= \mathbb{E}_{\boldsymbol{\omega}}\big[\cos(\boldsymbol{\omega}^\top(\mathbf{x}-\mathbf{x}'))\big] \qquad (7)\\ &= \mathbb{E}_{\boldsymbol{\omega}}\big[\cos(\boldsymbol{\omega}^\top\mathbf{x})\cos(\boldsymbol{\omega}^\top\mathbf{x}') + \sin(\boldsymbol{\omega}^\top\mathbf{x})\sin(\boldsymbol{\omega}^\top\mathbf{x}')\big].\end{aligned}$$

The expectation term suggests that we can use Monte Carlo method to approximate the kernel function by sampling from distribution $p(\boldsymbol{\omega})$. Specifically, the kernel function can be approximated as: $\hat{k}_\theta(\mathbf{x}, \mathbf{x}') \approx \phi(\mathbf{x})^\top\phi(\mathbf{x}')$, where $\phi(\cdot)$ is the feature mapping function defined as:

$$\phi(\mathbf{x}) = \sqrt{\frac{1}{M}}\Big[\cos(\boldsymbol{\omega}_m^\top\mathbf{x}), \sin(\boldsymbol{\omega}_m^\top\mathbf{x})\Big]_{m=1}^{M}, \qquad (8)$$

where M denotes the total number of samples from $p(\boldsymbol{\omega})$. The features obtained with ϕ is referred to as the random Fourier features (RFFs). We set the base kernel function $\tilde{k}_\theta$ in (1) as the proposed kernel function defined with the random Fourier features, i.e., $\hat{k}_\theta$.

By using the proposed kernel function, we convert the kernel learning problem as the distribution learning problem, i.e., learning the distribution $p(\boldsymbol{\omega})$ in (6). To enable efficient sampling and backpropagation, we set the distribution to be reparameterizable distributions such as Gaussian distributions, i.e., $p(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega}; \boldsymbol{\mu}, \boldsymbol{s})$, where $\boldsymbol{\mu}$ and $\boldsymbol{s}$ represent the mean and covariance, respectively. Since the parameters of this distribution can be inferred from data, we can learn an adaptive kernel function in a data-driven way, which is more effective in capturing the statistical structure of the KGs.

We now proceed ***to address the second problem*** *by our method.* Inspired by the insight that GP is equivalent to a Bayesian linear model whose weights follow Gaussian distributions (Rasmussen 2003), we also reformulate our GP module as a Bayesian linear model (Fang et al. 2021b). Specifically, the GP module in l-th layer is defined as $\mathbf{F}^l = \mathbf{A}^l\varphi(\mathbf{H}^l)$, where $\mathbf{A}^l \sim \mathcal{N}(\mathbf{0}, \delta\mathbf{I})$ is the weights of linear transformation with δ denoting the variance of noise, and φ is the feature mapping function which maps the input features into a Hilbert space. In this paper, we set the mapping function φ as the information extractor function $\mathbf{g}$ defined in (2) as we empirically found this can achieve the best performance.

Combining the above formulations with (3), the output embeddings of an entity e in the l-th layer is denoted as:

$$\boldsymbol{h}_e^{l+1} = \psi\left(\sum_{(r,o)\in\mathcal{N}(e)} \alpha_e^{r,o}\mathbf{W}_{\lambda(r)}^l \cdot \mathbf{A}^l \mathbf{g}(\mathbf{h}_o, \mathbf{h}_r)\right). \quad (9)$$

To make predictions, we need to infer the posterior distribution of $\mathbf{A}^l$. In this paper, to enable efficient learning, we approximate this posterior with a Dirac distribution. As a result, the parameters of our GGPN, including network parameters and kernel parameters, can be efficiently learned in an end-to-end fashion.

4.5 Training Objectives

We would like to emphasize that our method is a general framework for multi-relational learning tasks. It is straightforward to apply our method to different learning tasks by specifying proper decoders. For instance, we conduct experiments in link prediction and entity classification tasks, where the decoders can be chosen as any existing KG score functions and $\mathrm{softmax}$ function, respectively. Specifically, in link prediction task, we utilize the following three score functions (Bordes et al. 2013; Yang et al. 2015; Dettmers et al. 2018) as decoders:

- **TransE**: $s(e,r,o) = -\|\boldsymbol{h}_e + \boldsymbol{h}_r - \boldsymbol{h}_o\|_2^2$,
- **Distmult**: $s(e,r,o) = -\|\boldsymbol{h}_e \mathrm{diag}(\boldsymbol{h}_r)\, \boldsymbol{h}_o\|_2^2$,
- **ConvE**: $s(e,r,o) = \psi\left(\mathrm{vec}\big(\psi\left([\overline{\boldsymbol{h}}_e; \overline{\boldsymbol{h}}_r] * \mu\right)\big)\mathbf{W}_c\right) \boldsymbol{h}_o$,

where $(e,r,o) \in \mathcal{T}'$, $\boldsymbol{h}_e, \boldsymbol{h}_o, \boldsymbol{h}_r \in \mathbf{H}$, and $\overline{\boldsymbol{h}}_e, \overline{\boldsymbol{h}}_r \in \mathbb{R}^{d_1\times d_2}$ are 2D reshapings of $\boldsymbol{h}_e, \boldsymbol{h}_r$, where $d = d_1 \times d_2$ is the dimension of $\boldsymbol{h}_e, \boldsymbol{h}_r$, μ denotes a set of filters, * is the convolution operator, vec$(\cdot)$ is a vectorization function and $\mathbf{W}_c$ is a weight matrix. By the score function, our GGPN is trained with the cross-entropy loss as in (Vashishth et al. 2020b).

On the other hand, in entity classification, the decoder can be chosen as a full-connected layer followed by a $\mathrm{softmax}$ function, i.e., $s(e) = \mathrm{softmax}(\sigma(\mathbf{W}_s \boldsymbol{h}_e))$, where $\mathbf{W}_s$ is the linear weight matrix and σ is the sigmoid function. Similarly, we also use the cross-entropy loss to train our model.

5 Experiments

In this section, we evaluate the performance of our GGPN in two typical tasks: link prediction and entity classification.

5.1 Link Prediction

In this section, we mainly study the following four research questions: (**RQ1**) How does GGPN perform in link prediction task compared with baselines? (**RQ2**) How is the performance of GGPN with different decoders? (**RQ3**) How is the performance of the proposed kernel function in GGPN? (**RQ4**) What are the effects of the number of RFF features, i.e., M, on the performance of GGPN?

Datasets We adopt two widely used benchmark datasets in our link prediction experiments: FB15K-237 (Toutanova et al. 2015) and WN18RR (Dettmers et al. 2018), which are the subsets of popular KG datasets: FB15K (Bordes et al. 2013) and WN18 (Bordes et al. 2013), respectively. FB15K and WN18 are not adopted in our experiments, as they are observed a serious flaw that contains a large amount of inverse triples (Toutanova et al. 2015; Dettmers et al. 2018). Consequently a simple linear model trained to inverse triples can irrationally outperform lots of models on link prediction. FB15K-237 and WN18RR avoid this problem by removing the reverse relations. The details of these two datasets are provided in Section B.1 of the Appendix.

Baselines In our experiments, we compare our GGPN against a variety of baselines which can be categorized as:

- **Non-GNN methods:** Methods that use vector based operations for computing score, such as TransE (Bordes et al. 2013), DistMult (Yang et al. 2015), ComplEx (Trouillon et al. 2016), Conve (Dettmers et al. 2018), RotatE (Sun et al. 2019), and InteractE (Vashishth et al. 2020a).
- **GNN methods:** Methods that leverage graph neural network (GNN) based architecture as encoder like R-GCN (Schlichtkrull et al. 2018), A2N (Bansal et al. 2019), SACN (Shang et al. 2019), VR-GCN (Ye et al. 2019), KBAT (Nathani et al. 2019) and CompGCN (Vashishth et al. 2020b).

Evaluation Metrics Following (Bordes et al. 2013), we evaluate methods in the filtered setting, i.e., while evaluating on test triples, we build a candidate set either corrupting the head or tail entity of a triple and then filter out all the valid triples in it. The performance is reported on the standard evaluation metrics: Mean Reciprocal Rank (MRR), Mean Rank (MR) and Hits@N for N=1, 3, and 10.

Settings The input features of entities and relations to our GGPN are randomly initialized with a size of 100, and the output embedding size of both entities and relations are set to 200. We use Adam (Kingma and Ba 2015) as the optimizer and perform grid search to select the hyperparameters that have the best performance on validation sets. More details about hyperparameters are provided in section C.1 of the Appendix.

5.2 Results and Analyses

Link Prediction Results We firstly address (**RQ1**) by examining the performance of GGPN in link prediction. Table 1 reports the link prediction results of GGPN and baselines on FB15K-237 and WN18RR benchmark datasets. The results demonstrate that our GGPN outperforms all the baselines in three out of five metrics (MRR, Hits@1 and Hits@3) on FB15K-237 and in all the metrics on WN18RR. These results suggest that GGPN is effective for link prediction task. Particularly, when compared with the second best model, i.e., CompGCN, on FB15K-237, GGPN increases the performance by 1.69%, 2.65% and 1.54% on MRR, Hits@1 and Hits@3 respectively. Similar improvements can also be observed on WN18RR. The improvements on these metrics indicate that the embeddings learned with GGPN can better capture the similarities among entities and relations, leading to the overall lower rankings, i.e., higher MRR and Hits@N. The higher MR on FB15K-237 is due the fact that some triples might have abnormal high rankings, thus increasing the overall average ranking.

	FB15K-237					WN18RR				
	MR	MRR	Hits@1	Hits@3	Hits@10	MR	MRR	Hits@1	Hits@3	Hits@10
TransE ¶	357	0.294	-	-	0.465	3384	0.226	-	-	0.501
DistMult ¶	254	0.241	0.155	0.263	0.419	5110	0.430	0.390	0.440	0.490
ComplEx ¶	339	0.247	0.158	0.275	0.428	5261	0.440	0.410	0.460	0.510
ConvE ¶	244	0.325	0.237	0.356	0.501	4187	0.430	0.400	0.440	0.520
RotatE ¶	<u>177</u>	0.338	0.241	0.375	0.533	<u>3340</u>	0.476	0.428	0.492	**0.571**
InteractE ¶	**172**	0.354	0.263	-	0.535	5202	0.463	0.430	-	0.528
R-GCN ¶	-	0.248	0.153	0.258	0.414	-	-	-	-	-
A2N ¶	-	0.317	0.232	0.348	0.486	-	0.450	0.420	0.460	0.510
SACN ¶		0.350	0.260	0.390	**0.540**	-	0.470	0.430	0.480	0.540
VR-GCN ¶	-	0.248	0.159	0.272	0.432	-	-	-	-	-
KBAT §	251	0.318	0.231	0.362	0.499	3958	0.410	0.423	0.451	0.501
CompGCN ¶	197	<u>0.355</u>	<u>0.264</u>	<u>0.390</u>	0.535	3533	<u>0.479</u>	<u>0.443</u>	<u>0.494</u>	0.546
Ours	189	**0.361**†	**0.271**†	**0.396**†	**0.540**	**2937**†	**0.481**	**0.447**†	**0.499**†	<u>0.548</u>

Table 1: Link prediction performance of GGPN and baselines on FB15K-237 and WN18RR datasets. We use ConvE as decoder here. ¶ indicates the results are taken from origin papers, and § indicates that the results are reproduced by ourselves. The best scores per dataset per metric are marked in **boldface** and the second best scores are <u>underlined</u>. † denotes the significant improvements over the comparative methods (paired t-test, $p<0.05$).

	TransE			DistMult			ConvE		
	MRR	MR	Hits@10	MRR	MR	Hits@10	MRR	MR	Hits@10
No-encoder ¶	0.294	357	0.465	0.241	354	0.419	0.325	244	0.501
R-GCN $\star$	0.281	325	0.443	0.324	230	0.497	0.344	200	0.524
KBAT §	0.244	547	0.413	0.281	397	0.433	0.318	251	0.499
CompGCN $\star$	<u>0.336</u>	<u>214</u>	<u>0.518</u>	<u>0.335</u>	<u>227</u>	<u>0.514</u>	<u>0.355</u>	<u>197</u>	<u>0.535</u>
Ours	**0.340**†	**188**†	**0.527**†	**0.342**†	**197**†	**0.530**†	**0.361**†	**189**	**0.540**†

Table 2: Link prediction performance of GGPN and baselines under different decoders on FB15K-237 dataset. ¶ indicates the results are taken from origin papers, § indicates the results are reproduced by ourselves, and $\star$ indicates the results are taken from (Vashishth et al. 2020b). No-encoder denotes using corresponding decoder as embedding model directly. The best scores per dataset per metric are marked in **boldface** and second best scores are <u>underlined</u>. † denotes the significant improvements over the comparative methods (paired t-test, $p<0.05$).

Performance Comparison on Decoders Next, we address (**RQ2**) by examining the performance of GGPN with different decoders. Recall that the proposed method is a general encoder-decoder framework with GGPN as the encoder and the decoder can be chosen based on the learning tasks. Here, we study the performance of GGPN under three typical decoders for link prediction: TransE, DistMult, and ConvE. We compare the results against some GNN-based encoders, e.g., R-GCN, KBAT and CompGCN. The experimental results in Table 2 show that GGPN can outperform all the baselines under all decoders. This result indicates that our GGPN, as an encoder, provides a more effective and powerful way to learn high quality entity and relation embeddings, which can improve the performance of downstream task regardless of the decoders.

Kernel Comparison Subsequently, we turn to (**RQ3**) by examining the performance of GGPN with different kernel functions. Recall that we propose a novel adaptive kernel function in GGPN based on the Bochner's theorem, such that the kernel function can be learned in a data-driven way. We study the performance of the proposed kernel function by comparing it with other kernel functions. Specifically, we evaluate the performance of GGPN with other kernel functions such as linear, RBF, Laplacian, and sigmoid kernels (Bergman 1970) in link prediction task on FB15K-237. The experimental results are provided in Table 3, which show that the proposed kernel function can outperform all the other kernels in all the metrics. This is due to the fact that the proposed kernel function can be effectively learned from data and better adapt to the statistical structure of data, resulting in an improved performance.

Influence of the Number of RFF Features Finally, we answer (**RQ4**) by studying the effects of the number of RFF features, i.e., M. Specifically, we conduct experiments on FB15K-237 by varying M from 100 to 700 and report the results of GGPN. In addition to the evaluation metrics, i.e., MRR and Hits@10, we also report the results of training time and memory usages. The experimental results are shown in Figure 2. We can find that the performance of GGPN increases with M at first and then gradually decreases with M. In contrast, the training time and memory usage keep increasing with M. The results suggest that we can strike a balance between the performance and training efficiency of our GGPN by choosing a proper M.

	MRR	MR	Hits@1	Hits@3	Hits@10
Linear	0.341	217	0.253	0.373	0.512
RBF	<u>0.344</u>	218	<u>0.256</u>	<u>0.377</u>	<u>0.520</u>
Laplacian	0.334	228	0.249	0.372	0.497
Sigmoid	0.341	<u>214</u>	0.252	0.374	0.518
Ours	**0.361**†	**189**†	**0.271**†	**0.396**†	**0.540**†

Table 3: Link prediction performance of GGPN with different kernel functions on FB15K-237. The best scores per dataset per metric are marked in **boldface** and second best scores are <u>underlined</u>. † denotes the significant improvements over the comparative methods (paired t-test, $p<0.05$).

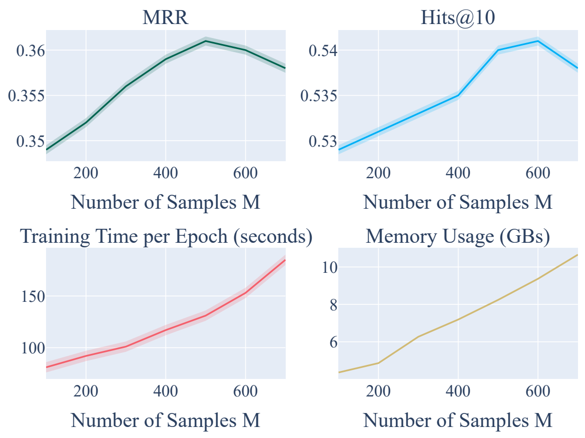

Figure 2: Influence of the number of RFF features for *MRR*, *Hits@10*, *Training time* and *Memory usage*. The solid lines and shadow areas denote the mean values and standard deviations over 5 runs, respectively.

5.3 Entity Classification

Entity classification aims to correctly classify the types of entities in a KG, depending on the way of how entities associate with other entities through corresponding relations. In this section, we mainly study the performance of GGPN in the entity classification task.

Datasets We conduct experiments on four RDF-format datasets (Ristoski, De Vries, and Paulheim 2016): AIFB, MUTAG, BGS, and AM. In these datasets, entities are represented as nodes with specific features and relations represent the dependencies from subject nodes to object nodes. The details of datasets are provided in Section B.2 of Appendix.

Baselines We compare GGPN against some state-of-the-art entity classification methods including RDF2Vec (Ristoski and Paulheim 2016), R-GCN, CompGCN, and handle-designed feature exactors (Feat) (Paulheim and Fümkranz 2012). RDF2Vec is a skip-gram based embedding method, Feat is an aggregation method for in-relation and out-relation of every labeled entity, and R-GCN and CompGCN are GNN-based methods to learn entity and relation embeddings for relational graphs.

Settings We report *accuracy* as evaluation metric based on the test splits provided by (Ristoski and Paulheim 2016).

Datasets	AIFB	MUTAG	BGS	AM
Feat $^\flat$	55.55	<u>77.94</u>	72.41	66.66
RDF2Vec $^\flat$	88.88	67.20	87.24	<u>88.33</u>
R-GCN $^\flat$	**95.83**	73.23	83.10	**89.29**
CompGCN §	<u>94.44</u>	76.47	<u>89.27</u>	78.81
Ours	**95.83**	**80.88**†	**93.10**†	87.37

Table 4: Accuracies (%) of GGPN and baselines in entity classification task. $\flat$ indicates the results are taken from (Schlichtkrull et al. 2018) and $\S$ indicates the results are reproduced by ourselves. The best scores per dataset are marked in **boldface** and the second best scores are <u>underlined</u>. † denotes the significant improvements over the comparative methods (paired t-test, $p<0.05$).

Furthermore, following (Vashishth et al. 2020b), we randomly divide 20% of training data as the validation dataset for hyperparameters tuning. We also use grid search to find the hyperparameters based on their performance on validation sets. Further details for hyperparameters are provided in Section C.2 of Appendix.

Results and Analyses Table 4 reports the experimental results of entity classification task on AIFB, MUTAG, BGS, and AM datasets. We can observe that our model achieves state-of-the-art performance on MUTAG and BGS by large margins of 4.41% and 3.83% respectively, and has competitive performance on AIFB and AM. The results validate that our GGPN is effective in entity classification task. Moreover, it is worth noting that on AIFB, MUTAT and BGS datasets, where the labels are scarce (only a few hundreds of labels), our GGPN can outperform deterministic baselines (Feat, RDF2Vec, R-GCN and CompGCN) in most cases.

This supports our original hypothesis that scarce and limited labeled data cannot provide sufficient supervision signals for the effective learning of a deterministic model, leading to a suboptimal performance. In contrast, GGPN addresses this problem by introducing GP, i.e., learning a family of classification functions, which offers much more flexibility to handle the insufficient supervision signals such that it can outperform deterministic methods.

6 Conclusion

This paper deals with the representation learning problem in knowledge graphs. To effectively handle the imperfections and noises in KGs, we propose GGPN, a Bayesian representation learning algorithm which simultaneously learn a family of embedding functions, i.e., a stochastic embedding function, through Gaussian Process. In particular, we employ a novel relation-aware kernel function built on the Bochner's theorem in our GGPN, which can explicitly take the diverse relations among entities into account and be adaptive to the training data. Furthermore, we reformulate GP as Bayesian linear model to address the issue of high computational cost. Extensive experiments on popular benchmark datasets of link prediction and entity classification demonstrate the superior performance of our GGPN.

Acknowledgments

This research was partially supported by the National Natural Science Foundation of China (Grant No. 61906219), and the Mohamed bin Zayed University of Artificial Intelligence, United Arab Emirates.

References

Arora, S. 2020. A Survey on Graph Neural Networks for Knowledge Graph Completion. arXiv:2007.12374.

Bansal, T.; Juan, D.-C.; Ravi, S.; and McCallum, A. 2019. A2n: Attending to neighbors for knowledge graph inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4387–4392.

Bergman, S. 1970. *The kernel function and conformal mapping*, volume 5. American Mathematical Soc.

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.

Cho, K.; van Merriënboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 103–111. Doha, Qatar: Association for Computational Linguistics.

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. Convolutional 2D Knowledge Graph Embeddings. In McIlraith, S. A.; and Weinberger, K. Q., eds., *Thirty-second AAAI conference on artificial intelligence*, 1811–1818. AAAI Press.

Fang, J.; Liang, S.; Meng, Z.; and Zhang, Q. 2021a. Gaussian Process with Graph Convolutional Kernel for Relational Learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 353–363.

Fang, J.; Zhang, Q.; Meng, Z.; and Liang, S. 2021b. Structure-Aware Random Fourier Kernel for Graphs. *Advances in Neural Information Processing Systems*, 34.

Flam-Shepherd, D.; Requeima, J.; and Duvenaud, D. 2017. Mapping Gaussian process priors to Bayesian neural networks. In *NIPS Bayesian deep learning workshop*, volume 13.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*, 1263–1272. PMLR.

Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; and Philip, S. Y. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017*.

Li, N.; Li, W.; Sun, J.; Gao, Y.; Jiang, Y.; and Xia, S.-T. 2020. Stochastic Deep Gaussian Processes over Graphs. *Advances in Neural Information Processing Systems*, 33.

Liang, S.; Luo, Y.; and Meng, Z. 2021. Profiling users for question answering communities via Flow-based constrained Co-embedding model. *ACM Transactions on Information Systems (TOIS)*, 40(2): 1–38.

Liao, S.; Liang, S.; Meng, Z.; and Zhang, Q. 2021. Learning Dynamic Embeddings for Temporal Knowledge Graphs. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 535–543.

Luo, Y.; Liang, S.; and Meng, Z. 2019. Constrained co-embedding model for user profiling in question answering communities. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 439–448.

MacKay, D. J. 1998. Introduction to Gaussian processes. *NATO ASI Series F Computer and Systems Sciences*, 168: 133–166.

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.

Ng, Y. C.; Colombo, N.; and Silva, R. 2018. Bayesian semi-supervised learning with graph Gaussian processes. In *Advances in Neural Information Processing Systems*, 1690–1701.

Opolka, F. L.; and Liò, P. 2020. Graph Convolutional Gaussian Processes For Link Prediction. arXiv:2002.04337.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32: 8026–8037.

Paulheim, H.; and Fümkranz, J. 2012. Unsupervised generation of data mining features from linked open data. In *Proceedings of the 2nd international conference on web intelligence, mining and semantics*, 1–12.

Qian, X.; Feng, H.; Zhao, G.; and Mei, T. 2013. Personalized recommendation combining user interest and social circle. *IEEE transactions on knowledge and data engineering*, 26(7): 1763–1777.

Qu, M.; Gao, T.; Xhonneux, L.-P.; and Tang, J. 2020. Few-shot relation extraction via bayesian meta-learning on relation graphs. In *International Conference on Machine Learning*, 7867–7876. PMLR.

Rahimi, A.; and Recht, B. 2008. Random Features for Large-Scale Kernel Machines. In Platt, J.; Koller, D.; Singer, Y.; and Roweis, S., eds., *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc.

Rasmussen, C. E. 2003. Gaussian processes in machine learning. In *Summer school on machine learning*, 63–71. Springer.

Ristoski, P.; De Vries, G. K. D.; and Paulheim, H. 2016. A collection of benchmark datasets for systematic evaluations of machine learning on the semantic web. In *International Semantic Web Conference*, 186–194. Springer.

Ristoski, P.; and Paulheim, H. 2016. Rdf2vec: Rdf graph embeddings for data mining. In *International Semantic Web Conference*, 498–514. Springer.

Ross, S. M.; Kelly, J. J.; Sullivan, R. J.; Perry, W. J.; Mercer, D.; Davis, R. M.; Washburn, T. D.; Sager, E. V.; Boyce, J. B.; and Bristow, V. L. 1996. *Stochastic processes*, volume 2. Wiley New York.

Rudin, W. 1962. *Fourier analysis on groups*, volume 121967. Wiley Online Library.

Rudin, W. 2017. *Fourier analysis on groups*. Courier Dover Publications.

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.

Shang, C.; Tang, Y.; Huang, J.; Bi, J.; He, X.; and Zhou, B. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 3060–3067.

Sindhwani, W. C. V.; Ghahramani, Z.; and Keerthi, S. S. 2007. Relational learning with gaussian processes. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, volume 19, 289. MIT Press.

Sun, Z.; Deng, Z.; Nie, J.; and Tang, J. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *7th International Conference on Learning Representations, ICLR 2019*.

Toutanova, K.; Chen, D.; Pantel, P.; Poon, H.; Choudhury, P.; and Gamon, M. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, 1499–1509.

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.; Agrawal, N.; and Talukdar, P. 2020a. Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 3009–3016.

Vashishth, S.; Sanyal, S.; Nitin, V.; and Talukdar, P. P. 2020b. Composition-based Multi-Relational Graph Convolutional Networks. In *8th International Conference on Learning Representations, ICLR 2020*.

Wang, H.; Zhou, C.; Chen, X.; Wu, J.; Pan, S.; and Wang, J. 2020. Graph stochastic neural networks for semi-supervised learning. *Advances in Neural Information Processing Systems*.

Wilson, A. G.; Hu, Z.; Salakhutdinov, R.; and Xing, E. P. 2016. Deep kernel learning. In *Artificial intelligence and statistics*, 370–378. PMLR.

Xu, Z.; Kersting, K.; and Tresp, V. 2009. Multi-Relational Learning with Gaussian Processes. In *the 21st International Joint Conference on Artificial Intelligence*, 1309–1314.

Yang, B.; Yih, W.; He, X.; Gao, J.; and Deng, L. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *3rd International Conference on Learning Representations, ICLR 2015*.

Ye, R.; Li, X.; Fang, Y.; Zang, H.; and Wang, M. 2019. A Vectorized Relational Graph Convolutional Network for Multi-Relational Network Alignment. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 4135–4141. International Joint Conferences on Artificial Intelligence Organization.

Young, N. 1988. *An Introduction to Hilbert Space*. Cambridge university press.

Yu, K.; Chu, W.; Yu, S.; Tresp, V.; and Xu, Z. 2007. Stochastic Relational Models for Discriminative Link Prediction. In Schölkopf, B.; Platt, J.; and Hoffman, T., eds., *Advances in Neural Information Processing Systems*, volume 19. MIT Press.

Zhou, G.; Su, J.; Zhang, J.; and Zhang, M. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd annual meeting of the association for computational linguistics (acl'05)*, 427–434.

A Pseudo Codes of GGPN

We further provide an overview of our proposed GGPN model with pseudo codes in Algorithm 1.

Algorithm 1: Overview of GGPN.

```
Input: A KG $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$, the number of GGPN layers $L$, the distribution for RFF sampling $p(\boldsymbol{\omega})$.
Output: The representations of entities and relations.
1: Extend $\mathcal{T}$ and $\mathcal{R}$ with corresponding inverse relations into $\mathcal{T}'$ and $\mathcal{R}'$, respectively;
2: Initialize the entity and relation embeddings;
3: for $l$ = 1 to $L$ do
4:     Calculate the correlation information of entities associated with relations by information extractor according to Equation 2;
5:     Sample $\boldsymbol{\omega}$ from $p(\boldsymbol{\omega})$ to build the RFF kernel;
6:     Calculate the kernel function values for GP according to Equation 1;
7:     Update the entity and relation embeddings in layer $l$ according to Equation 9 and 5, respectively.
8: end for
9: return Updated entity and relation embeddings.
```

B Dataset Details

In this section, we introduce more details about the datasets used in our experiments.

B.1 Link prediction

In link prediction, we use FB15K-237 and WN18RR as our experimental datasets, the statistics of which are provided in Table 5. As mentioned in the main text, FB15K-237 and WN18RR are built from FB15K and WN18 respectively, by removing the reverse relations. FB15K-237 has 237 types of relations, such that it has a high structure heterogenity, which is low in WN18RR with 11 types of relations. We use the two datasets with different structure heterogenity to illustrate the general performance of our GGPN. For fair comparison, we use the standard train/validation/test splits as in (Bordes et al. 2013).

B.2 Entity classification

In entity classification, we use AIFB, MUTAG, BGS and AM as our experimental datasets, the statistics of which are provided in Table 6. Relations in these datasets need not necessarily encode directed subject-object relations, but are also used to encode the presence, or absence, of a specific feature for a given entity (Schlichtkrull et al. 2018). Following (Schlichtkrull et al. 2018), we remove the relations : *employs* and *affiliation* for AIFB, *isMutagenic* for MUTAG, *hasLithogenesi* for BGS, and *objectCategory* and *material* for AM, which are used to create entity labels to build the same strategy for evaluation.

In each dataset, the targets to be classified are properties of a group of entities represented as nodes. In our experiments, to make a fair comparison, we use the same train/validation/test splits as in (Ristoski and Paulheim 2016; Schlichtkrull et al. 2018).

Datasets	FB15K-237	WN18RR
#E	14,541	40,943
#R	237	11
#Train edges	272,115	86,835
#Val. edges	17,535	3,034
#Test edges	20,466	3,134

Table 5: Statistics of datasets in link prediction. #E denotes the number of entities, and #R denotes the number of relations. #Train / Val. / Test. edges denote the number of train/-validation/test triples respectively.

Datasets	AIFB	MUTAG	BGS	AM
#Entities	8,285	23,644	333,845	1,666,764
#Relations	45	23	103	133
#Edges	29,043	74,227	916,199	5,988,321
#Labeled	176	340	146	1,000
#Classes	4	2	2	11

Table 6: Statistics of datasets in entity classification. #Entities, #Relations, #Edges denote the number of entities, relations, and triples. #Labeled denotes the number of labeled entities and #classes denotes the number of classes.

C Hyperparameters

In this section, we illustrate additional implementation details of our GGPN for each task in our experiments. Moreover, all of our experiments are implemented on Pytorch framework (Paszke et al. 2019).

C.1 Link prediction

As mentioned in the main text, the number of layers (L) is chosen from $\{1, 2\}$, the number of RFF features (M) is chosen from $\{100, 200, ..., 700\}$, the number of hidden units in each layer is chosen from$\{100, 150, 200\}$, the learning rate is chosen from $\{0.001, 0.0005, 0.0001\}$, and the dropout rate and batch size are chosen from $\{0.1, 0.3, 0.5\}$ and $\{128, 256\}$, respectively. We use the grid search method to obtain the best hyperparameters for each datasets, which are provided in Table 8.

C.2 Entity classification

Similar to link prediction task, we also randomly initialize the entity and relation embeddings, and initial embeddings, hidden embeddings, and final embeddings of entities and relations have the same size which is chosen from $\{32, 64, 128\}$, In this task, the number of layers (L) is chosen from$\{1, 2, 3\}$, the number of RFF features (M) is chosen from $\{50, 100, 200\}$, the dropout rate is chosen from $\{0.1, 0.2, 0.3\}$, and the learning rate is chosen from $\{0.0001, 0.001, 0.01\}$. The best hyperparameters for each dataset are provided in Table 9.

	Memory complexity	Memory complexity$^*_{\text{w/ bases}}$	Time complexity
RGCN	$\mathcal{O}(\mathcal{L}\lvert\mathcal{R}\rvert d^2 + \mathcal{L}\lvert\mathcal{E}\rvert d + \mathcal{L}\lvert\mathcal{R}\rvert d)$	$\mathcal{O}(\mathcal{L}\mathcal{B}d^2 + \mathcal{L}\lvert\mathcal{E}\rvert d + \mathcal{L}\mathcal{B}\lvert\mathcal{R}\rvert)$	$\mathcal{O}(\mathcal{L}\lVert\mathcal{A}\rVert_0 d^2)$
CompGCN	$\mathcal{O}(\mathcal{L}d^2 + \mathcal{L}\lvert\mathcal{E}\rvert d + \mathcal{L}\lvert\mathcal{R}\rvert d)$	$\mathcal{O}(\mathcal{L}d^2 + \mathcal{L}\lvert\mathcal{E}\rvert d + \mathcal{B}d + \mathcal{B}\lvert\mathcal{R}\rvert)$	$\mathcal{O}(\mathcal{L}\lVert\mathcal{A}\rVert_0 d^2)$
Ours	$\mathcal{O}(\mathcal{L}d^2 + \mathcal{L}\lvert\mathcal{E}\rvert d + \mathcal{L}\lvert\mathcal{R}\rvert d)$	$\mathcal{O}(\mathcal{L}d^2 + \mathcal{L}\lvert\mathcal{E}\rvert d + \mathcal{B}d + \mathcal{B}\lvert\mathcal{R}\rvert)$	$\mathcal{O}(\mathcal{L}\lVert\mathcal{A}\rVert_0 d^2)$

Table 7: Memory and time complexity. $\mathcal{L}$ denotes the number of layers, d is the embedding dimension, $\lVert\mathcal{A}\rVert_0$ is the number of nonzeros in the adjacency matrix, and $\lvert\mathcal{R}\rvert, \lvert\mathcal{E}\rvert$ denote the number of relations and entities respectively. $\mathcal{B}$ is the number of relation bases introduced in RGCN and CompGCN to improve the memory complexity (marked in $*$) which can also be applied to our model.

Hyperparameters	FB15K-237	WN18RR
Number of layers (L)	1	1
Number of RFF features (M)	500	500
Number of hidden units	200	200
Learning rate	0.001	0.0005
Dropout	0.3	0.1
Batch size	256	256

Table 8: Best values of hyperparameters for link prediction.

Hyperparameters	AIFB	MUTAG	BGS	AM
Number of layers(L)	3	4	4	4
Number of RFF features (M)	100	100	200	100
Size of embeddings	64	64	128	64
Learning rate	0.001	0.001	0.01	0.01
Dropout	0.1	0.2	0.3	0.3

Table 9: Best values of hyperparameters for entity classification.

D Supplementary Experiments and Analyses

D.1 Complexity Analyses

In Table 7, we provide the analysis of memory and time complexity of our GGPN. We compare it with two baselines that are closely related to our model, i.e., RGCN and CompGCN. These methods have the same time complexity because they share a similar message-passing architecture. However, our GGPN has a smaller memory complexity compared with RGCN.

D.2 Experimental results for different relation categories

In link prediction task, we also compare our GGPN with CompGCN in a fine-grained way. We compare against CompGCN only as it is the baseline that has the best performance in our experiments. Specifically, we report the *MRR* scores of both GGPN and CompGCN on different relation categories: *1-1*, *1-N*, *N-1*, *N-N*. In detail, we calculate the average number of head entities γ_h and tail entities γ_t for each relation r in FB15K-237 dataset, following (Bordes et al. 2013), if γ_h<1.5 and γ_t<1.5, r is categorized *1-1*; if γ_h<1.5 and $\gamma_t \geq 1.5$, r is categorized *1-N*; if $\gamma_h \geq 1.5$ and γ_t<1.5, r is categorized *N-1*; if $\gamma_h \geq 1.5$ and $\gamma_t \geq 1.5$, r is categorized *N-N*. The experimental results of GGPN and CompGCN on different relation categories are shown in Figure 3. The results show that our GGPN can consistently outperform CompGCN on all the relation categories, which demonstrates the superiority of our GGPN in link prediction task.

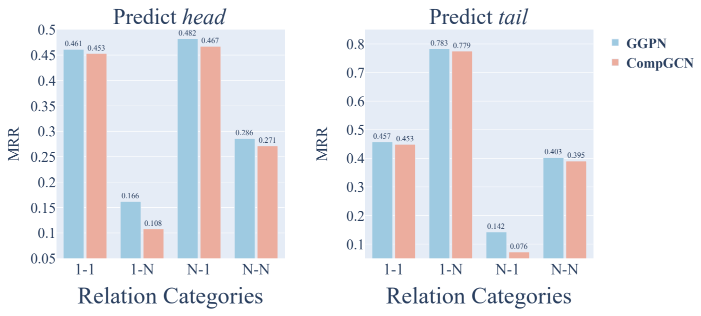

Figure 3: Link Prediction performance (*MRR*) of our GGPN and CompGCN on different relation categories.

	MRR	MR	Hits@1	Hits@3	Hits@10
CompGCN	0.296	**215**	0.207	0.326	0.476
Ours	**0.309**	225	**0.218**	**0.340**	**0.497**

Table 10: Link prediction performance of GGPN compared with CompGCN on FB15K-237 in noisy setting.

D.3 Experiments on Noisy KGs

We further conduct experiments on noisy KGs to study the performance of our GGPN. Specifically, we randomly sample 10% of edges in the training set of FB15K-237, and corrupt these edges by either removing them with a 80% probability or changing their relations to other random relations with a 20% probability. We compare our model with CompGCN, and the results in Table 10 support our model's ability for noisy KGs