Two Birds with One Stone: Enhancing Uncertainty Quantification and Interpretability with Graph Functional Neural Process

Lingkai Kong* Haotian Sun* Yuchen Zhuang Haorui Wang
Wenhao Mu Chao Zhang
School of Computational Science and Engineering
Georgia Institute of Technology

Abstract

Graph neural networks (GNNs) are powerful tools on graph data. However, their predictions are mis-calibrated and lack interpretability, limiting their adoption in critical applications. To address this issue, we propose a new uncertainty-aware and interpretable graph classification model that combines graph functional neural process and graph generative model. The core of our method is to assume a set of latent rationales which can be mapped to a probabilistic embedding space; the predictive distribution of the classifier is conditioned on such rationale embeddings by learning a stochastic correlation matrix. The graph generator serves to decode the graph structure of the rationales from the embedding space for model interpretability. For efficient model training, we adopt an alternating optimization procedure which mimics the well known Expectation-Maximization (EM) algorithm. The proposed method is general and can be applied to any existing GNN architecture. Extensive experiments on five graph classification datasets demonstrate that our framework outperforms state-of-the-art methods in both uncertainty quantification and GNN interpretability. We also conduct case studies to show that the decoded rationale structure can provide meaningful explanations.

*Equal contribution. Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

1 Introduction

Graph neural networks (GNNs) [Kipf and Welling, 2016, Veličković et al., 2018, Hamilton et al., 2017, Li et al., 2016b] have been successful in various graph analytic tasks, such as graph classification, node classification and link prediction. GNNs provide a flexible framework to learn node representations with the message passing scheme, which aggregates vector representations from their topological neighborhoods. Compared with traditional graph mining techniques, GNNs transform the graph from the discrete graph space into the continuous embedding space that is easier to optimize for downstream tasks. Moreover, they can leverage the representation power of deep neural networks (DNNs) to learn complex input-output mapping functions and thus can achieve high accuracy across many tasks.

Many graph applications demand not only accurate but also *uncertainty-aware* and *explainable* predictions. These two features are crucial for understanding and building trust in GNN predictions. First, the absence of uncertainty estimates can result in unreliable probabilistic predictions and failure in practice. For example, in molecular property prediction [Wieder et al., 2020, Feinberg et al., 2018], highly parameterized GNNs are vulnerable to overfitting to training scaffolds and may produce incorrect and poorly calibrated predictions for new scaffolds without uncertainty quantification. Second, GNNs are commonly used as black-box predictors, lacking explanations for their predictions. For example, it is important to understand which chemical groups in a molecular graph contribute to the predictions in molecular property prediction [Amara et al., 2023]. However, the black-box nature of current GNNs makes it difficult to verify if their working mechanisms align with real-world chemical rules, reducing trust in their predictions and limiting their adoption in critical applications.

The challenge of providing uncertainty-aware and explainable predictions with GNNs remains unresolved. Recent studies [Wang et al., 2021b] have shown that GNNs are poor at quantifying predictive uncertainty and miscalibrated in their predictions. One natural idea to remedy this issue is to apply existing uncertainty quantification techniques for GNNs, such as model ensembling and Bayesian neural networks (BNNs). Model ensembling trains multiple deep neural networks (DNNs) with different initializations and ensembles their predictions for uncertainty quantification[Lakshminarayanan et al., 2017, Ganaie et al., 2021], but this approach incurs significant computation cost. BNN [Welling and Teh, 2011, Louizos and Welling, 2017, Ritter et al., 2018, Blundell et al., 2015, Li et al., 2015, Zhang et al., 2019] quantify uncertainty by imposing probability distributions over model parameters, but the exact inference of the posterior distribution and proper specification of prior distributions for uninterpretable GNN parameters remain difficult. With regard to interpretability, most existing graph interpretable methods [Ying et al., 2019, Luo et al., 2020] provide sample-level explanations, but these are too specific and difficult to generalize. Instead, model-level interpretations, which aim to explain the overall behavior of the model by uncovering the key patterns or substructures driving predictions, are more general and require less human supervision. However, generating model-level explanations for graphs remains an underexplored area.

We present a new uncertainty-aware and explainable graph classification approach that combines graph functional neural process and graph generative model. This framework has the following three capabilities: (1) quantifying predictive uncertainty directly from the functional space, (2) generating model-level rationales for interpretability, and (3) being applicable to any GNN architecture. The method assumes the existence of latent model-level rationales that can be mapped to distributions in the embedding space and predicts the class based on these stochastic rationale embeddings, providing a natural way to quantify uncertainty. The classifier, inspired by the functional neural process (FNP) [Louizos et al., 2019], models the relationship between rationales and training graphs in the shared latent embedding space by learning a stochastic correlation matrix and generates the final predictive distribution based on correlated rationales. Additionally, the framework includes an autoregressive graph generator that produces the graph structure of the rationale embeddings for interpretability.

We have conducted thorough experiments to evaluate our proposed model on five graph classification datasets. Our model consistently outperforms existing state-of-art (SOTA) methods in both uncertainty quantification and model interpretability. Specifically, GraphFNP outperforms the strongest baseline by up to 4.8% in terms of expected calibration error and meanwhile maintains competitive predictive performance. To quantitatively evaluate the learned rationales, we apply a KNN-based classifier by computing the distance between the rationales and input graphs in the embedding space; GraphFNP outperforms SOTA methods by up to 10.8% in terms of F1 score. We also qualitatively visualize the decoded rationale structure to show that they do contain critical substructure for the class and align with the real-world rules.

2 Notations and Problem Definition

We focus on the graph classification problem in this paper. Let $D = \{(G_i^D, y_i^D)\}_{i=1}^N$ be the set of labeled training data. $G = (\mathcal{V}, \mathcal{E})$ represents a graph with $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$ denoting the node set and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ denoting the edge set. The numbers of nodes and edges are denoted by n and m, respectively. The nodes in $\mathcal{V}$ are associated with the d-dimensional features, denoted as by $\mathbf{X} \in \mathcal{R}^{n \times d}$. $y = \{1, 2, \cdots, K\}$ denotes the category of the corresponding whole graph and K the total number of categories. Taking the molecular property prediction as an example, $\mathcal{V}$ is the set of atoms; $\mathcal{E}$ is the set of bonds; $\mathbf{X}$ is the one-hot encoding of the atom type.

The graph classification problem involves learning a model $\mathcal{M}_\theta$, parameterized by θ, that can categorize a graph G into different classes. The model outputs a predicted category $\hat{y}$ and its corresponding confidence $\hat{p}$, such that $\mathcal{M}_\theta(G) \to (\hat{y}, \hat{p})$. Traditional GNNs only provide point estimates and lack interpretability, making them unsuitable for safety-critical applications. Our goal is to train a model that, given an unseen graph G^* at test time, can provide both (1) a well-calibrated predictive distribution $p(y^*|G^*)$ and (2) a model-level rationale G^R that explains the crucial patterns that led to the prediction.

Calibrated Uncertainty Estimates: A well-calibrated predictive model should have confidence levels in its predictions that align with the actual accuracy of those predictions. For instance, if 100 data points are predicted with a confidence of 0.8, we expect 80 of them to be correctly classified. The calibration error of the predictive model, as defined by Guo et al. [2017], Kong et al. [2020a], measures the discrepancy between the model's confidence in its predictions and the actual accuracy of those predictions. Given a confidence level $p \in [0, 1]$, the calibration error is given by:

$$\mathcal{E}_p = |\mathcal{P}(\hat{y} = y|\hat{p} = p) - p|, \quad (1)$$

where $\mathcal{P}(\hat{y} = y|\hat{p} = p)$ is the probability that the predicted class $\hat{y}$ is the same as the true class y, given

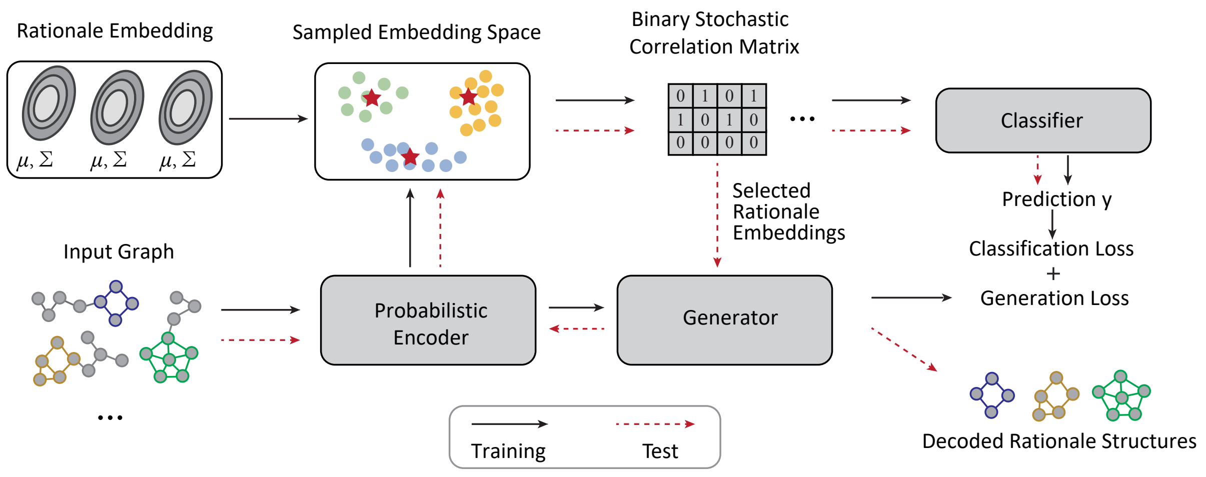


Figure 1: The overall generative process of the proposed framework. We assume a set of latent model-level rationales which can be mapped into a probabilistic embedding space; the graph classifier is conditioned on such rationale embeddings and graph embeddings through a stochastic correlation matrix. The graph generator is used to obtain the graph structure of the selected rationales from the latent embedding space.

the predicted confidence level $\hat{p} = p$. A model with perfect predictive uncertainty should satisfy $\mathcal{P}(\hat{y} = y|\hat{p} = p) = p$ for all $p \in [0, 1]$.

Model-Level Rationales for Interpretability: In graph classification, model-level rationales refer to a collection of subgraph structures, $(G_i^R, y_i^R)_{i=1}^{|R|}$, where each subgraph G_i^R represents a key pattern associated with its corresponding category y_i^R. For instance, in molecular property prediction, a molecule containing a NO_2 substructure often has mutagenic properties. By identifying such salient patterns, practitioners can verify if the model aligns with real-world rules and gain actionable insights. Note that, unlike sample-level rationales [Ying et al., 2019, Luo et al., 2020, Schlichtkrull et al., 2020], which are defined as subgraphs of individual input graphs, model-level rationales are not specific to any particular graph.

3 Methodology

3.1 Model Overview

Our objective is to learn a predictive probability distribution $p(y|G)$ from the training data D and provide model-level rationales to explain the predictions. To this end, we propose a non-parametric probabilistic generative process that jointly models the graph generation and classification procedures. This framework is capable of: (1) directly estimating predictive uncertainty in the functional space and (2) identifying the relationship between model predictions and generated rationales for better interpretability.

Our framework learns a set of rationale embeddings from the continuous embedding space. The rationale embeddings hold crucial information for the classification task, and so the predictive distribution $p(y|G)$ is based on them, which transforms the problem into learning the correlation between rationale and graph embeddings. Inspired by the functional neural process (FNP) [Louizos et al., 2019], we represent these correlations with a binary stochastic matrix. Furthermore, we include a graph generative model to derive the graph structure of rationales from their embeddings.

Our method has three key components (Fig. 1):

(1) **Probabilistic Rationale Embedding**: To learn the rationale embeddings $\mathbf{Z}^R = \{\mathbf{z}_i^R\}_{i=1}^{|R|}$ without having access to rationales and to make the learning process differentiable, we opt to learn a set of rationale embeddings from the latent embedding space. To capture the uncertainty in the embeddings, we represent them as high-dimensional Gaussian random variables. The input graphs are transformed into the same space through a GNN encoder.

(2) **Stochastic Correlation Matrix**: The correlation matrix $\mathbf{C}$ models the relationship between training graphs and rationales in the embedding space. It serves a similar role as a kernel function in Gaussian processes (GPs) for non-parametric uncertainty estimation. The correlated rationales reflect the crucial substructures emphasized by the model for prediction. The final predictive distribution is parameterized with two stochastic latent variables: (a) The local rationale embedding $\mathbf{U}^D = \{\mathbf{u}_i^D\}_{i=1}^N$, which summarize correlated rationale embeddings. (b) The graph embedding $\mathbf{Z}^D = \{\mathbf{z}_i^D\}_{i=1}^N$, which captures embedding uncertainty and provides novel information not present in the rationales.

(3) **Graph Generative Model for Model Interpretability**: This component is crucial for model interpretability as it allows us to obtain the rationale structure from the learned rationale embeddings using the decoder after training. We design the graph generator as a simple and flexible autoregressive model. To incorporate information from the latent embedding space, our generation procedure is dependent on the graph/rationale embeddings.

Combing all components, we arrive at the following generative process for the training data:

$$p(\mathbf{y}^D|\mathbf{G}^D) = \sum_{\mathbf{C}} \int \underbrace{p_\theta(\mathbf{Z}^R)p_\theta(\mathbf{Z}^D|\mathbf{G}^D)}_{\text{Latent embedding}} \underbrace{p(\mathbf{C}|\mathbf{Z}^D,\mathbf{Z}^R)}_{\text{Stochastic correlation matrix}} \underbrace{p_\theta(\mathbf{U}^D|\mathbf{C},\mathbf{Z}^R)p(\mathbf{y}^D|\mathbf{U}^D,\mathbf{Z}^D)}_{\text{Predictive distribution}} \underbrace{p(\mathbf{G}^D|\mathbf{Z}^D)}_{\text{Graph generator}} \; d\mathbf{Z}^R d\mathbf{Z}^D d\mathbf{U}^D d\mathbf{G}. \tag{2}$$

3.2 Graph Functional Neural Process With Learnable Rationales

Our approach differs from the traditional FNP [Louizos et al., 2019] , which uses a randomly selected subset of the dataset (known as the reference set) to base its predictive distribution. This method lacks interpretability as it fails to provide summarized graph patterns for each class. In contrast, our approach involves learning a set of rationales for each class and basing the predictive distribution on these correlated rationales. The learned rationales represent the crucial substructures for each class.

3.2.1 Learning rationales from probabilistic latent space.

Learning the rationales directly in the graph domain is difficult due to its discrete nature, making it non-differentiable. Previous approaches, such as the one in You et al. [2018a]'s work, have used reinforcement learning techniques to solve this issue through formulating the rationale generation as a Markov decision process. However, these methods often suffer from poor performance and instability during training [Wang and Shen, 2023]. Our proposed solution is to learn a set of rationale embeddings in the continuous embedding space, which is easier to optimize and leads to improved performance.

Specifically, we first randomly initialize a set of vectors $\{\mathbf{s}_i^R\}_{i=1}^{|R|}$ for the rationales. For a K-way classification problem, we assign $|R_k|$ rationales ($|R| = \sum_{k=1}^{K} |R_k|$) to the k-th class, where $|R_k|$ is a hyper-parameter that we can tune during the training process. To capture the embedding uncertainty, we propose to further project $\mathbf{s}_i^R$ to a high-dimensional Gaussian distribution space:

$$\mathbf{z}_i^R \sim \mathcal{N}(\mathrm{MLP}(\mathbf{s}_i^R), \exp(\mathrm{MLP}(\mathbf{s}_i^R))). \tag{3}$$

We denote the union of the parameters of $\{\mathbf{s}_i\}$ and the two MLPs as θ_r. The distribution of the rationale embeddings will be updated during training.

Each rationale embedding will encode one crucial predictive pattern for a specific class. We will base the predictive distribution of an input graph on such salient embeddings based on the correlations between the rationale embeddings and graph embeddings. These rationale embeddings share a similar spirit with inducing points in stochastic variational Gaussian Process (SVGP, [Hensman et al., 2013]). However, we will enhance this by utilizing a generative model to decode the corresponding graph structures for model-level interpretability 3.3.

Then we project the training graph into the same Gaussian distribution space through a GNN encoder. The GNN encoder computes node representations $\{\mathbf{h}_{i,v}|v \in \mathcal{V}_i\}$ as $\{\mathbf{h}_{i,v}\} = \mathrm{GNN}_e(G_i)$. The node vectors are aggregated to represent G_i^D as a single vector $\mathbf{h}_i^D = \mathrm{Aggregate}(\{\mathbf{h}_{i,v}\})$. Finally, the graph embedding follows: $\mathbf{z}_i^D \sim \mathcal{N}\left(\mathrm{MLP}(\mathbf{h}_i^D), \exp(\mathrm{MLP}(\mathbf{h}_i^D))\right)$. We denote the union of the parameters of the GNN encoder and the two MLPs as θ_e.

3.2.2 Constructing stochastic correlation matrix C

The stochastic correlation matrix $\mathbf{C}$ models the relationship between the graphs and rationales in the embedding space, serving two important purposes: (1) it captures the uncertainty in the correlations for non-parametric uncertainty estimation, and (2) it identifies which rationale can be used to explain the corresponding prediction, thus improving the interpretability of the model.

Specifically, we first model the correlations in the latent embedding space using kernel similarity: $\kappa(\mathbf{z}_i^D, \mathbf{z}_j^R)$, *e.g.*, we can use the radial basis function (RBF) kernel: $\kappa(\mathbf{z}_i^D, \mathbf{z}_j^R) = \exp(-\gamma||\mathbf{z}_i^D - \mathbf{z}_j^R||)$. Instead of directly using raw kernel similarity to parameterize, we further use Bernoulli sampling to generate a binary symmetric correlation matrix:

$$\mathbf{C}_{i,j} \sim \mathrm{Bern}(\mathbf{C}_{i,j}|\kappa(z_i^D, z_j^R)). \tag{4}$$

This sampling process leads to sparse correlations for each sampled matrix and enjoys two benefits: (a) it can capture uncertainty from the data correlation perspective; (b) it can speed up model training by virtue of sparsity.

When $C_{i,j} = 1$, it means that the j-th rationale z_j^R is correlated with the i-th graph G_i. Through analyzing such a binary correlation matrix, we can identify which rationale represents a crucial pattern for the input graph.

3.2.3 Constructing the predictive distribution

With the binary correlation matrix, we summarize the information of the correlated rationales for each graph into the local rationale embedding $\mathbf{u}_i^D$:

$$\mathbf{u}_i^D \sim \mathcal{N}(C_i \sum_{j:\mathbf{C}_{j,i}=1} \mathrm{MLP}(\mathbf{z}_j^R), \exp(C_i \sum_{j:\mathbf{C}_{j,i}=1} \mathrm{MLP}(\mathbf{z}_j^R))), \tag{5}$$

where $C_i = \sum_j \mathbf{C}_{i,j}$ is for normalization.

As we can see, Equation 5 encodes the inductive bias that predictions on points that are “far away,” i.e., have very small probability of being connected to the rationales, will default to an uninformative standard Gaussian prior. This is similar to the behavior that Gaussian processes (GPs) with RBF kernels exhibit.

The local rationale embeddings are derived solely from the rationale embedding set and may not capture novel information present in the graphs. To address this issue, we include the graph embedding $\mathbf{z}_i$ in the final prediction. This allows the neural network to extrapolate beyond the distribution of the rationale embeddings, which is important when the unseen test graphs contain novel predictive patterns that are not present in the learned rationale set. Therefore, we concatenate the graph embedding and local rationale embedding into a single vector and obtain the final predictive distribution:

$$y_i = \mathrm{MLP}(\mathrm{concat}(\mathbf{z}_i, \mathbf{u}_i)). \tag{6}$$

We denote the parameters of the MLPs in Eq. 5 and Eq. 6 as θ_{cls}.

3.3 Graph Structure Decoder

The goal of learning rationales in the embedding space is to understand the graph structure that drives the model's predictions. To achieve this, we design a graph generator that can decode the rationales from the embedding space and produce graph structures.

To ensure that our decoded rationale structures contain critical patterns for classification, we propose a variant of the GraphRNN model [You et al., 2018b]. This variant incorporates latent embedding information into the graph generation process. The graph is generated in breadth-first order, with each step involving the generation of a node and its edges, taking into account all previously generated nodes. Unlike standard GraphRNN models, which are only used for random graph generation, our proposed model ensures that the decoded rationales retain the critical patterns necessary for accurate classification.

Specifically, in t-th step, the decoder first runs a decoder GNN over current graph G_t to compute node representations:$\{\mathbf{h}_v^t\} = \mathrm{GNN}_d(G_t)$. The current graph G_t is represented as an aggregation of its node vectors $\mathbf{h}_{G_t} = \mathrm{Aggregate}(\{\mathbf{h}_v^t\})$. With the current graph representation, we predict the probability of the node type of v_t as:

$$\mathbf{p}_{v_t} = \mathrm{softmax}(\mathrm{MLP}(\mathbf{h}_{v_t}^t, \mathbf{h}_{G_t}, \mathbf{z})). \tag{7}$$

Note that it is easy to generalize to continuous node features by assuming $\mathbf{p}_{v_t}$ follows a Gaussian distribution. To fully capture edge dependencies, we predict the edge type between v_t and all the previously generated node $\{v_j\}_{j=1}^{t-1}$ sequentially and update the representation of v_t when a new edge is added to G_t. In the j-th step, we predict the edge type between v_t and v_j as:

$$\mathbf{p}_{e_{v_t,v_j}} = \mathrm{softmax}(\mathrm{MLP}(\mathbf{h}_{v_t}^{t,j}, \mathbf{h}_{v_j}^t, \mathbf{h}_{G_t}, \mathbf{z})), \tag{8}$$

where $\mathbf{h}_{v_t}^{t,j}$ is the new representation of v_t after the edge e_{v_t,v_j} has been added. We denote the parameters of the graph decoder as θ_{d}.

As seen from Eq. 7 and Eq. 8, the generation procedure of nodes and edges both depend on the latent graph embedding $\mathbf{z}$. This latent embedding holds crucial information about graph properties, and the learned rationale embeddings are salient points within the same space. Hence, the decoded rationale structure should reflect the critical subgraph patterns of its corresponding class.

3.4 Model Training

We now describe how to learn the model parameters during training. The rationales and training graphs are intertwined in the same space, making direct optimization of the overall data likelihood (Eq. 1) difficult. To solve this, we employ an alternating optimization strategy, similar to the Expectation-Maximization (EM) approach.

E-step In this step, we aim to update the rationale embeddings and graph decoder given a fixed graph encoder and classifier. Canceling out the constant terms, we have the following loss:

$$\begin{aligned}
&\mathrm{argmin}_{\theta_{\mathrm{d}},\theta_{\mathrm{r}}}\mathcal{L}_{\mathrm{E}} \\
&= \underbrace{-\mathrm{E}_{p_\theta(\mathbf{Z}^R)p_\theta(\mathbf{Z}^D,\mathbf{U}^D,\mathbf{C}|\mathbf{G}^D,\mathbf{Z}^R)} \log p(y^R|\mathbf{Z}^R, \mathbf{U}^R)}_{\text{Classification loss of training data}} \\
&- \underbrace{\mathrm{E}_{p_\theta(\mathbf{Z}^D|\mathbf{G}^D)} \log p_{\theta_d}(\mathbf{G}^D|\mathbf{Z}^D)}_{\text{Generation loss}} \\
&- \underbrace{\mathrm{E}_{p(\mathbf{Z}^R)p(\mathbf{U}^R|\mathbf{Z}^R)} \log p_\theta(\mathbf{y}^D|\mathbf{Z}^D, \mathbf{U}^D)}_{\text{Classification loss of rationales}}
\end{aligned} \tag{9}$$

Table 1: ECE and predictive performance on all the five datasets. We report the average performance and standard deviation for 5 random initializations. For predictive performance, Graph-SST2 and MUTAG use accuracy as the metric, while BBBP and BACE use AU-ROC.

		ECE					Accuracy/AU-ROC				
Backbone	Model	Graph-SST2	BBBP	BACE	MUTAG	Github	Graph-SST2	BBBP	BACE	MUTAG	Github
GCN	Vanilla	15.07±0.60	18.30±1.56	19.52±2.87	3.54±1.05	6.73±0.99	82.83±0.58	69.34 ±1.29	77.52 ±0.66	79.38±0.87	61.26±0.39
	DropEdge	13.89 ±3.48	16.78±2.11	16.23±3.33	4.57±0.79	7.15±0.67	82.89±1.31	70.02±0.69	76.07± 2.40	76.5±1.63	61.09±0.64
	MC-Dropout	13.82±0.73	15.53±1.38	18.76±2.63	3.49±0.62	6.28±0.69	83.21±0.64	69.62± 1.13	78.40± 0.65	78.34±0.81	63.29±0.84
	SGLD	10.71± 0.59	15.66± 2.71	14.47± 3.61	5.76± 0.74	3.96±1.84	82.01± 0.62	67.83± 2.33	76.74±1.87	77.89±0.59	59.49±0.21
	BBP	14.57± 0.81	14.04± 1.30	13.94± 1.22	7.26± 1.24	4.35±0.93	82.72± 0.67	69.54± 2.26	75.34± 1.37	76.82±0.59	61.06±0.62
	Graph-GP	12.52±0.76	14.36± 1.98	14.31±2.87	4.96±1.32	5.32±0.87	81.01± 0.87	68.98± 1.76	76.96± 0.97	78.16±1.03	60.75±1.26
	GSAT	16.51±2.11	37.33± 0.71	31.32± 1.66	8.64± 1.86	4.04±0.50	81.09±1.04	71.55± 1.56	77.91± 0.78	78.89±0.58	60.42±0.11
	DeepEnsemble	9.81±NA	15.00±NA	16.04±NA	3.22±NA	6.21±0.15	84.71±NA	69.75±NA	78.17±NA	79.03±NA	61.97±0.65
	GraphFNP	**9.02**± 0.58	**10.95**± 0.92	**9.11**± 2.96	3.82±1.07	**3.25**±0.42	83.08± 0.26	70.25± 0.70	80.36± 0.89	79.80±0.71	61.34±0.85
GAT	Vanilla	13.73± 0.60	23.40± 5.30	17.84± 2.45	4.18±0.71	5.72±1.05	83.63± 0.82	69.82± 2.17	77.40± 3.85	72.88±1.34	62.19±1.75
	DropEdge	10.33± 0.62	20.34± 1.68	15.39± 3.07	5.32±1.67	9.65±1.03	83.76± 0.60	66.85± 0.74	80.10± 5.76	70.04±2.61	61.67±0.89
	MC-Dropout	12.32±0.73	19.85±3.93	15.20± 2.85	**3.74**±0.65	5.03±0.78	83.46±0.42	70.03± 1.98	77.82± 3.71	72.72±0.63	64.26±0.75
	SGLD	10.28± 0.31	17.66± 1.95	20.81± 4.94	4.66±0.57	6.21±1.48	83.67± 0.41	67.87±2.34	74.25± 3.67	72.02±1.33	60.75±0.57
	BBP	9.26± 1.94	16.45± 1.47	14.24± 6.25	6.09±1.20	4.95±0.89	80.82± 0.59	68.89± 1.62	74.78± 5.63	72.63±1.75	60.56±1.08
	Graph-GP	11.96±1.86	18.25 ±2.01	15.86± 1.75	4.75±1.54	4.32±0.86	82.75±1.86	68.78± 2.01	76.41± 2.76	72.26±1.59	61.94±1.03
	GSAT	13.54±2.04	34.17± 0.85	32.08±0.97	7.91±2.10	3.90±2.79	81.98±1.87	68.65± 1.06	72.23±1.54	72.02±1.65	62.22±1.50
	DeepEnsemble	8.43±NA	16.80±NA	17.73±NA	4.35±NA	4.92±0.29	85.69±NA	71.98±NA	79.20 ±NA	73.27±NA	63.89±0.72
	GraphFNP	**7.89** ±1.61	**15.76**±2.34	**13.58**±3.43	3.99±0.23	**3.74**±0.85	83.45±1.56	70.07 ±0.63	79.13 ±1.96	74.19±0.33	62.75±1.03

Note that, in Eq. 9, we also add the classification loss of the rationales. For the rationales, as their rationale embedding already represents crucial predictive patterns, we directly project them into the local rationale embedding space with the MLP used in Eq. 7, *i.e.*,$\mathbf{u}_i^R \sim \mathcal{N}(\mathrm{MLP}(\mathbf{z}_i^R), \exp(\mathrm{MLP}(\mathbf{z}_i^R)))$.

M-step In this step, we optimize the parameters of the probabilistic encoder and classifier by fixing the distributions of the rationale embeddings and graph decoder. Directly maximizing the data likelihood is intractable and we choose to use the amortized variational inference. Following the work of Louizos et al. [2019], we assume that the variational posterior distribution $q_\phi(\mathbf{U}^D, \mathbf{C}, \mathbf{Z}^D|\mathbf{G}^D)$ factorizes as $p_\theta(\mathbf{Z}^D|\mathbf{G}^D)p(\mathbf{C}|\mathbf{Z}^R, \mathbf{Z}^D)q_\phi(\mathbf{U}^D|\mathbf{G}^D)$. This leads to the following loss:

$$\begin{aligned}
&\mathrm{argmin}_{\theta_{\mathrm{e}}, \theta_{\mathrm{cls}}, \phi}\mathcal{L}_{\mathrm{M}} \\
&= \underbrace{-\mathrm{E}_{p_\theta(\mathbf{Z}^R)q_\phi(\mathbf{U}^D|\mathbf{G}^D)} \log p_\theta(\mathbf{y}^D|\mathbf{Z}^D, \mathbf{U}^D)}_{\text{Classification loss of training data}} \\
&\underbrace{-\mathrm{E}_{p_\theta(\mathbf{Z}^R)p_\theta(\mathbf{C},\mathbf{Z}^D|\mathbf{G}^D,\mathbf{Z}^R)q(\mathbf{U}^D|\mathbf{G}^D)} \log \frac{p_\theta(\mathbf{U}^D|\mathbf{C}, \mathbf{Z}^R, \mathbf{Z}^D)}{q_\phi(\mathbf{U}^D|\mathbf{G}^D)}}_{\text{Prior regularization}} \\
&\underbrace{-\mathrm{E}_{p_\theta(\mathbf{Z}^R)p(\mathbf{U}^R|\mathbf{Z}^R)} p_\theta(y^R|\mathbf{Z}^R, \mathbf{U}^R)}_{\text{Classification loss of rationales}} \qquad (10)
\end{aligned}$$

As sampling from the Bernoulli distribution in Equation 4 leads to discrete correlated data points, we make use of the Gumbel softmax trick [Jang et al., 2016] to make the model differentiable.

4 Experiments

In this section, we evaluate the empirical performance of our model GraphFNP. Our experiments are designed to answer the following questions: Q1: Can GraphFNP provide calibrated predictive uncertainty? and Q2: How is the predictive power of the learned rationales? Q3: Does the decoded rationale graph structure provide meaningful explanations? Q4: How important are the different components of GraphFNP (Appendix A.2)?

4.1 Q1: Can GraphFNP provide calibrated predictive uncertainty?

We first examine if GraphFNP can simultaneously provide calibrated uncertainty estimates and maintain high predictive accuracy.

4.1.1 Experimental Setup

We use **Expected calibration error (ECE)** to evaluate the model calibration [Guo et al., 2017, Naeini et al., 2015]. We use the following five graph classification datasets: **•BBBP** is designed for the modeling and prediction of barrier permeability. This dataset includes binary labels for 2039 compounds on their permeability properties. **•BACE** is a collection of 1522 compounds with their binary labels for a set of inhibitors of human β-secretase 1. **•Graph-SST2** is a sentiment analysis dataset, where each text sequence in SST2 is converted to a graph. Following the splits in the study of Wu et al. [2022b], this dataset contains degree shifts during test. **•MUTAG** consists of 4,337 molecule graphs. Each graph is assigned to one of 2 classes based on its mutagenic effect. **•Github** has 12,725 social networks and the task is to predict whether a network belongs to web or machine learning developers.

Table 2: RF1 of all the methods. K is the number of neighbors in the KNN classifier. We report the average performance and standard deviation for 5 random seeds. Our method consistently outperforms all the baselines in RF1 across different datasets and GNN architectures. XGNN cannot be applied on Graph-SST2 because it can only generate graphs with discrete node features.

		$K=1$					$K=3$				
Backbone	Model	Graph-SST2	BBBP	BACE	MUTAG	Github	Graph-SST2	BBBP	BACE	MUTAG	Github
GCN	GNNExplainer	80.45±0.30	63.93±2.66	67.38±0.59	64.36±0.87	54.37±0.16	80.46±0.17	64.98±1.49	68.61±2.17	65.46±1.64	58.78±0.07
	PGExplainer	80.16±0.13	59.75±2.46	61.53±3.64	71.52±0.61	45.25±0.54	80.69±0.17	63.32±0.30	66.70±0.99	74.79±2.57	52.79±0.36
	XGNN	NA	56.23±4.25	53.62±3.72	68.67±3.01	51.92±1.92	NA	52.28±6.29	58.96±2.82	65.86±1.96	50.29±1.85
	GstarX	81.60±0.14	58.55±2.40	65.84±1.62	69.47±2.84	56.97±1.21	81.64±0.17	61.00±6.40	65.00±0.43	68.32±1.17	60.12±2.76
	GSAT	79.13±1.35	62.62±2.46	63.76±5.96	53.96±14.59	58.19±2.96	78.45±2.73	64.26±4.65	64.98±1.27	65.24±7.35	56.20±2.49
	SubgraphX	82.56±0.09	63.70±0.49	68.24±2.45	**75.19**±0.73	57.28±0.86	82.64±0.19	64.62±0.90	44.35±4.38	68.17±7.78	60.93±1.29
	GraphFNP	**85.85**±0.67	**67.42**±1.85	**71.52**±2.53	**74.47**±0.48	**64.62**±1.07	**85.84**±0.64	**67.74**±2.20	**71.52**±2.52	**74.42**±0.45	**64.46**±1.38
GAT	GNNExplainer	77.23±1.42	60.52±5.52	47.75±5.57	59.20±1.42	51.71±0.80	78.49±1.76	48.76±14.81	44.86±3.75	66.28±2.19	58.40±0.28
	PGExplainer	79.59±1.87	55.38±6.61	54.56±3.65	58.81±2.34	48.76±1.45	80.22±2.54	66.44±2.37	48.31±10.88	59.51±1.74	49.09±1.21
	XGNN	NA	56.26±2.76	49.21±3.28	64.54±3.06	52.86±3.76	NA	50.75±5.06	60.76±10.23	62.07±4.28	50.20±6.08
	GstarX	79.82±2.21	58.06±2.02	38.28±6.52	61.14±3.90	56.12±2.18	80.48±1.42	63.24±1.09	55.26±2.84	66.87±0.52	59.21±1.46
	GSAT	77.43±2.09	61.23±24.10	62.95±6.20	63.72±3.47	55.25±3.52	78.04±1.87	63.37±18.70	63.67±5.86	63.96±4.32	59.86±2.76
	SubgraphX	82.29±1.56	64.16±2.99	58.46±1.47	66.73±1.70	54.26±1.72	81.96±1.79	64.85±1.91	57.91±2.26	63.83±0.35	56.34±1.29
	GraphFNP	**85.67**±2.21	**67.99**±3.00	**70.00**±6.28	**68.75**±0.27	**61.38**±1.26	**85.66**±1.90	**68.23**±3.01	**70.57**±5.85	**69.08**±0.32	**63.27**±1.19

4.1.2 Baselines

We consider the following baselines: **Vanilla**[Hendrycks and Gimpel, 2016] directly uses the model's softmax score of the predicted category as the uncertainty estimate. **Monte Carlo dropout (MC-dropout)** [Gal and Ghahramani, 2016] applies dropout at test time for multiple times and then average the outputs. **Stochastic Gradient Langevin Dynamics (SGLD)** [Welling and Teh, 2011] is the most popular Markov Chain Monte Carlo (MCMC) based Bayesian neural network (BNN). **Bayes by Backprop (BBP)** [Blundell et al., 2015] fits a variational approximation to the true posterior of BNN with a fully factorised Gaussian assumption. [Kingma et al., 2015] as a variance reduction technique. **GSAT** [Miao et al., 2022] learns a stochastic attention to select task relevant subgraphs. **DropEdge** [Rong et al., 2020] randomly removes a certain number of edges from the input graph during training. **Graph-GP** [Ng et al., 2018] adopts GNN as the feature extractor and replaces the linear classifier layer with a Gaussian process. **Deep-Ensemble**[Lakshminarayanan et al., 2017] trains multiple GNNs with different initializations and aggregates their predictions at inference time. We use 5 as the ensemble size.

4.1.3 Main Results

Table 1 reports the ECE and predictive performance of all the methods. GraphFNP outperforms all the baselines on all the datasets except for MUTAG. On MUTAG, the Vanilla GNN has already achieved a good calibration score, and thus the room for further improvement is small. We can also see that the ECE of some baselines, such as BBP, are even worse than the Vanilla GNN on this dataset. GraphFNP can improve ECE up to 4.83% with GCN as the backbone and 1.62% with GAT as the backbone compared with the strongest baseline. The consistent improvement among GCN and GAT verifies that GraphFNP is a general method that can be compatible with existing GNN architecture.

Though with better calibration scores, existing Bayesian uncertainty estimation methods often suffer from decreased predictive performance. For example, on the BACE dataset, the predictive accuracy of SGLD drops by 3.15% compared with Vanilla GAT. However, GraphFNP can consistently boost the performance of Vanilla GNN. For example, on BACE, we improve the AU-ROC by 2.84% compared with Vanilla GCN. This is because GraphFNP quantifies uncertainty from the functional space and does not need to specify the uninterpretable prior distribution of model parameters as existing BNNs.

4.2 Q2: How is the predictive power of the learned rationales?

We have verified that GraphFNP can provide accurate and calibrated predictions. In this subsection, we further study whether GraphFNP can learn high-quality rationales.

4.2.1 Experimental Setup

Ideally, the rationale set should contain critical class information and achieve high predictive performance. We measure the predictive power of the discovered rationales by using a KNN classifier. To apply the KNN classifier, we compute the cosine distance between the test graph and the rationales in the embedding space. We refer to this metric as "**Rationale F1 (RF1)**". In the experiments, we set K to 1 and 3.

We compare with the following baselines: **GNNExplainer** [Ying et al., 2019] learns sample-level rationales by minimizing the mutual information between the graph prediction and distribution of possible subgraphs. **PGExplainer** [Luo et al., 2020] uses a deep neural network to parameterize the generative process

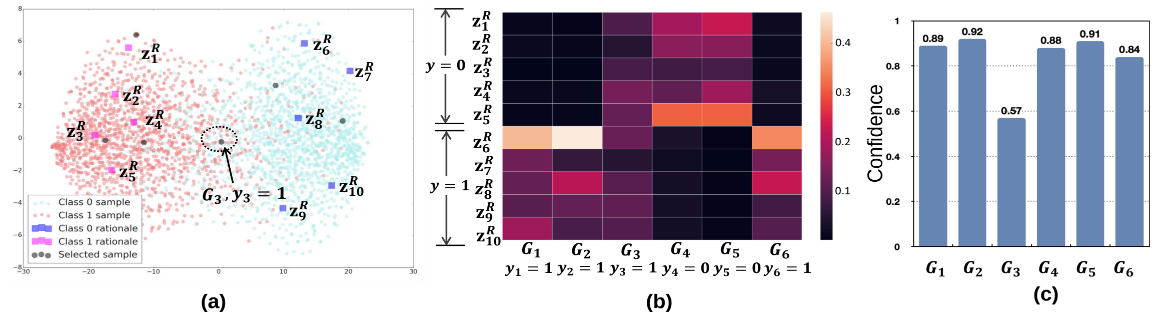


Figure 2: (a) visualizes the sampled rationale embeddings and graph embeddings. (b) plots the averaged stochastic correlations matrix for the selected six samples. (c) shows the predictive confidence of sample 1-3.

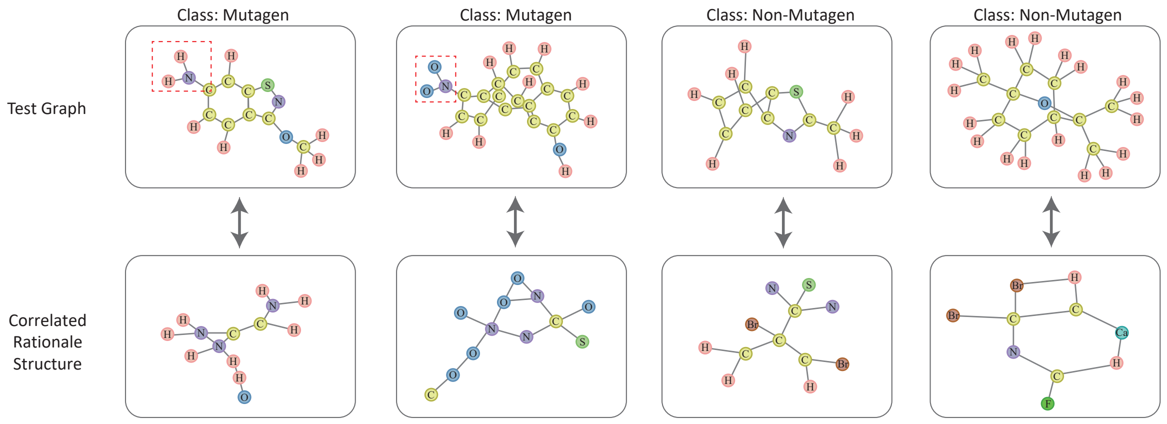


Figure 3: The decoded graph structure of the correlated rationale. The first row is the input test graphs, while the second shows the corresponding correlated model-level rationale structures. The rationales for the mutagen class contain repeated patterns of NH2 and NO2. While the rationales for the non-mutagen class have no particular patterns.

of the underlying graph structure as edge distributions, where the explanatory graph is sampled. **SubgraphX** [Yuan et al., 2020] searches representative subgraphs using Monte Carlo tree search with Shapley values as the importance score. **GStarX** [Zhang et al., 2022] proposes to use cooperative game theory to extract important subgraphs. **GSAT** [Miao et al., 2022] learns a stochastic attention weight to select task-relevant subgraphs. • **XGNN** [Yuan et al., 2020] trains a deep reinforcement learning model to generate the model-level explanation graphs.

All the baselines except for XGNN are originally designed to obtain sample-level rationales. To obtain model-level rationales, we first apply them to the training data and then perform centroid clustering for each class. The centroid of each cluster is used as the model-level rationale for each class.

4.2.2 Experimental Results

Table 2 reports the RF1 of all the methods. As shown, our method consistently outperforms other methods on all the datasets for both $K = 1$ and $K = 3$. The stable improvements of our method across GCN and GGNN demonstrates that our proposed the framework is general and can be applied to any GNN architectures.

Fig. 2 shows the sampled graph and rationale embeddings. As shown in Fig. 2(a), the rationale embeddings for each class are evenly distributed within their respective class cluster, demonstrating that each rationale encodes a critical predictive pattern. Fig. 3(b) depicts the average stochastic correlation matrix for six selected samples. It can be seen that the most correlated rationale for each sample is from the same class. The average values for rationales from other classes are close to zero for all samples except for sample 3, which is located near the boundary of two classes and has similar distances to the closest rationales of both classes, resulting in high uncertainty in model predictions. On the other hand, other samples are closely positioned to the rationales within their class cluster, leading to more confident predictions by the model.

4.3 Q3: Does the decoded graph structure of the rationale provide meaningful explanations?

In this subsection, we study whether the decoded graph structure of the correlated rationale can provide meaningful explanations for the model prediction.

We start by visualizing the synthetic BA2Motifs dataset, which consists of graphs built on an arabasi-Albert (BA) structure. Half of the graphs are attached with a house motif, and the other half with a five-node cycle motif. As seen in Fig. 4(b), the decoded graph structure of the selected rationale is a five-node cycle, aligning with the data generation process.

We further validate our method on the real-world MUTAG dataset. As reported by Debnath et al. [1991], the presence of chemical groups NH2 and NO2 in molecules is associated with mutagenicity. Fig. 3 shows the results on this dataset. The first two graphs in the first row are from the mutagen class and have either one NH2 or one NO2 group. The decoded rationales of these graphs, shown in the second row, reveal repeated patterns of NH2 and NO2, respectively. Conversely, the decoded rationale structures for the non-mutagen class do not display any specific patterns, which aligns with our prior knowledge.

The above results demonstrate that our decoded rationale structures contain crucial substructures for each class. These model-level rationales offer practitioners valuable insights into domain knowledge and help confirm if the model's workings align with real-world rules.

5 Related Works

Uncertainty quantification on GNNs. Ng et al. [2018] propose Graph Gaussian Process (Graph-GP), which stacks a Bayesian linear model on the feature representation of GNNs. Liu et al. [2020] further, extend Graph-GP by considering the uncertainty in the input graph. Zhao et al. [2020] proposes subjective GNN considering multi-source uncertainty. Hasanzadeh et al. [2020] adaptively drops some edges during training and shows that it approximates the variational inference of BNNs. Stadler et al. [2021] generalize the Posterior Network [Charpentier et al., 2020] to graph data. Along another line, Kang et al. [2022], Wang et al. [2021b] propose two post-hoc methods to improve GNN's calibration. However, these works are all designed for node classification and semi-supervised learning, while our work focuses on the graph-level property classification. Localized Neural Kernel (LNK, [Wollschläger et al., 2023]) primarily concentrates on energy prediction and a molecule's energy is additive concerning its atoms. Therefore, it initially defines a Gaussian process at the atom level, followed by summing up the energy predictions from each node for the final regression output. In contrast, our paper tackles graph-level classification tasks. We opt to aggregate node-level representations into a global one for classification purposes.

Explaining GNNs. There have been a large number of works in generating sample-level rationales for GNNs. These methods aim to provide the salient patterns for a specific input graph. According to Yuan et al. [2022]'s work, most existing works can be categorized into four directions [Wu et al., 2022a]: gradient-based methods [Baldassarre and Azizpour, 2019, Pope et al., 2019], pertubation-based methods [Ying et al., 2019, Luo et al., 2020, Zhang et al., 2022, Funke et al., 2021, Schlichtkrull et al., 2021], decomposition methods [Baldassarre and Azizpour, 2019, Schwarzenberg et al., 2019, Schnake et al., 2021, Wang et al., 2021a] and surrogate methods [Huang et al., 2022, Vu and Thai, 2020, Zhang et al., 2021]. In contrast, model-level rationale generation remains underexplored. It aims to explain the overall behavior of the model and is not specific to any particular graph. XGNN [Yuan et al., 2020] proposes to formulate the model-level rationale generation as a Markov decision process and use reinforcement learning to for optimization. DAG-Explainer [Lv and Chen, 2023] proposes a data-aware global explainer based on randomized greedy algorithm. Recently, Wang and Shen [2023] proposes a numerical optimization method to obtain the explanation graph via continuous relaxation. Both of these two works are post-hoc methods while we develop an inherently interpretable model. Existing work [Miao et al., 2022] has shown that the post-hoc interpretability methods are suboptimal from the information bottleneck perspective and can be sensitive to the pre-trained models.

6 Conclusion

We have proposed a new graph classification framework based on graph functional neural process (FNP) and graph generative model. The proposed framework quantifies the predictive uncertainty directly from the functional space and generates the model-level rationales for model interpretability. The core of our method is that we assume a set of latent rationales which can be mapped into a probabilistic latent space. Motivated by FNP, we design a stochastic correlation matrix to learn the correlations between rationale and graph embeddings. The predictive distribution of the graph classifier is conditioned on correlated rationale embeddings and graph embedding. To obtain the graph structures of the rationales, we further incorporate a graph generator into our framework. Extensive experiments on five graph classification datasets demonstrate that our method outperforms state-of-the-art uncertainty quantification and graph interpretability methods.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. This work was supported in part by NSF (IIS-2008334, IIS-2106961, CAREER IIS-2144338), ONR (MURI N00014-17-1-2656).

References

Kenza Amara, Raquel Rodríguez-Pérez, and José Jiménez-Luna. Explaining compound activity predictions with a substructure-aware loss for graph neural networks. *Journal of cheminformatics*, 15(1): 67, 2023.

Federico Baldassarre and Hossein Azizpour. Explainability techniques for graph convolutional networks. In *International Conference on Machine Learning (ICML) Workshops, 2019 Workshop on Learning and Reasoning with Graph-Structured Representations*, 2019.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. Posterior network: Uncertainty estimation without ood samples via density-based pseudo-counts. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Asim Kumar Debnath, Rosa L. Lopez de Compadre, Gargi Debnath, Alan J. Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34 (2):786–797, 1991. doi: 10.1021/jm00106a046. URL `https://doi.org/10.1021/jm00106a046`.

Evan N Feinberg, Debnil Sur, Zhenqin Wu, Brooke E Husic, Huanghao Mai, Yang Li, Saisai Sun, Jianyi Yang, Bharath Ramsundar, and Vijay S Pande. Potentialnet for molecular property prediction. *ACS central science*, 4(11):1520–1530, 2018.

Thorben Funke, Megha Khosla, and Avishek Anand. Hard masking for explaining graph neural networks, 2021. URL `https://openreview.net/forum?id=uDN8pRAdsoC`.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

MA Ganaie, Minghui Hu, et al. Ensemble deep learning: A review. *arXiv preprint arXiv:2104.02395*, 2021.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Arman Hasanzadeh, Ehsan Hajiramezanali, Shahin Boluki, Mingyuan Zhou, Nick Duffield, Krishna Narayanan, and Xiaoning Qian. Bayesian graph neural networks with adaptive connection sampling. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. 2016.

James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence*, page 282. Citeseer, 2013.

Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International Conference on Machine Learning*, pages 10362–10383. PMLR, 2022.

Jian Kang, Qinghai Zhou, and Hanghang Tong. Jurygcn: Quantifying jackknife uncertainty on graph convolutional networks. KDD '22, page 742–752, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3539286. URL `https://doi.org/10.1145/3534678.3539286`.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28:2575–2583, 2015.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. 2016.

Lingkai Kong, Haoming Jiang, Yuchen Zhuang, Jie Lyu, Tuo Zhao, and Chao Zhang. Calibrated language model fine-tuning for in-and out-of-distribution data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1326–1340, 2020a.

Lingkai Kong, Jimeng Sun, and Chao Zhang. Sde-net: Equipping deep neural networks with uncertainty estimates. *arXiv preprint arXiv:2008.10546*, 2020b.

Lingkai Kong, Jiaming Cui, Haotian Sun, Yuchen Zhuang, B Aditya Prakash, and Chao Zhang. Autoregressive diffusion model for graph generation. In *International conference on machine learning*, pages 17391–17408. PMLR, 2023.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Chunyuan Li, Changyou Chen, David Carlson, and Lawrence Carin. Preconditioned stochastic gradient langevin dynamics for deep neural networks. arxiv e-prints, page. *arXiv preprint arXiv:1512.07666*, 2015.

Chunyuan Li, Changyou Chen, David Carlson, and Lawrence Carin. Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016a.

Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. Gated graph sequence neural networks. In *Proceedings of ICLR'16*, 2016b.

Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. Graph normalizing flows. *Advances in Neural Information Processing Systems*, 32, 2019.

Zhao-Yang Liu, Shao-Yuan Li, Songcan Chen, Yao Hu, and Sheng-Jun Huang. Uncertainty aware graph gaussian process for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4957–4964, 2020.

Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. In *International Conference on Machine Learning*, pages 2218–2227. PMLR, 2017.

Christos Louizos, Xiahan Shi, Klamer Schutte, and Max Welling. The functional neural process. *arXiv preprint arXiv:1906.08324*, 2019.

Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 33: 19620–19631, 2020.

Ge Lv and Lei Chen. On data-aware global explainability of graph neural networks. *Proceedings of the VLDB Endowment*, 16(11):3447–3460, 2023.

David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4 (3):448–472, 1992.

Kaushalya Madhawa, Katushiko Ishiguro, Kosuke Nakago, and Motoki Abe. Graphnvp: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019.

Siqi Miao, Mia Liu, and Pan Li. Interpretable and generalizable graph learning via stochastic attention mechanism. In *International Conference on Machine Learning*, pages 15524–15543. PMLR, 2022.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Yin Cheng Ng, Nicolò Colombo, and Ricardo Silva. Bayesian semi-supervised learning with graph gaussian processes. *Advances in Neural Information Processing Systems*, 31, 2018.

Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. Explainability methods for graph convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10772–10781, 2019.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=Skdvd2xAZ`.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=Hkx1qkrKPr`.

Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting graph neural networks for nlp with differentiable edge masking. In *International Conference on Learning Representations*, 2020.

Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting graph neural networks for {nlp} with differentiable edge masking. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=WznmQa42ZAx`.

Thomas Schnake, Oliver Eberle, Jonas Lederer, Shinichi Nakajima, Kristof T Schütt, Klaus-Robert Müller, and Grégoire Montavon. Higher-order explanations of graph neural networks via relevant walks. *IEEE transactions on pattern analysis and machine intelligence*, 44(11):7581–7596, 2021.

Robert Schwarzenberg, Marc Hübner, David Harbecke, Christoph Alt, and Leonhard Hennig. Layerwise relevance visualization in convolutional text graph classifiers. *arXiv preprint arXiv:1909.10911*, 2019.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations*, 2019.

Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International conference on artificial neural networks*, pages 412–422. Springer, 2018.

Maximilian Stadler, Bertrand Charpentier, Simon Geisler, Daniel Zügner, and Stephan Günnemann. Graph posterior network: Bayesian predictive uncertainty for node classification. *Advances in Neural Information Processing Systems*, 34:18033–18048, 2021.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=UaAD-Nu86WX`.

Minh Vu and My T Thai. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. *Advances in neural information processing systems*, 33:12225–12235, 2020.

Xiang Wang, Yingxin Wu, An Zhang, Xiangnan He, and Tat seng Chua. Causal screening to interpret graph neural networks, 2021a. URL `https://openreview.net/forum?id=nzKv5vxZfge`.

Xiao Wang, Hongrui Liu, Chuan Shi, and Cheng Yang. Be confident! towards trustworthy graph neural networks via confidence calibration. *Advances in Neural Information Processing Systems*, 34, 2021b.

Xiaoqi Wang and Han Wei Shen. GNNInterpreter: A probabilistic generative model-level explanation for graph neural networks. In *International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=rqq6Dh8t4d`.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

Oliver Wieder, Stefan Kohlbacher, Mélaine Kuenemann, Arthur Garon, Pierre Ducrot, Thomas Seidel, and Thierry Langer. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies*, 37:1–12, 2020.

Tom Wollschläger, Nicholas Gao, Bertrand Charpentier, Mohamed Amine Ketata, and Stephan Günnemann. Uncertainty estimation for molecules: Desiderata and methods. In *International Conference on Machine Learning*, pages 37133–37156. PMLR, 2023.

Bingzhe Wu, Jintang Li, Junchi Yu, Yatao Bian, Hengtong Zhang, CHaochao Chen, Chengbin Hou, Guoji Fu, Liang Chen, Tingyang Xu, et al. A survey of trustworthy graph learning: Reliability, explainability, and privacy protection. *arXiv preprint arXiv:2205.10014*, 2022a.

Yingxin Wu, Xiang Wang, An Zhang, Xiangnan He, and Tat-Seng Chua. Discovering invariant rationales for graph neural networks. In *International Conference on Learning Representations*, 2022b. URL `https://openreview.net/forum?id=hGXij5rfiHw`.

Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32:9240, 2019.

Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems*, 31, 2018a.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR, 2018b.

Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 430–438, 2020.

Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient mcmc for bayesian deep learning. *arXiv preprint arXiv:1902.03932*, 2019.

Shichang Zhang, Yozen Liu, Neil Shah, and Yizhou Sun. Explaining graph neural networks with structure-aware cooperative games. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=Qry8exovcNA`.

Yue Zhang, David Defazio, and Arti Ramesh. Relex: A model-agnostic relational model explainer. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 1042–1049, 2021.

Xujiang Zhao, Feng Chen, Shu Hu, and Jin-Hee Cho. Uncertainty aware semi-supervised learning on graph data. *Advances in Neural Information Processing Systems*, 33:12827–12836, 2020.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]

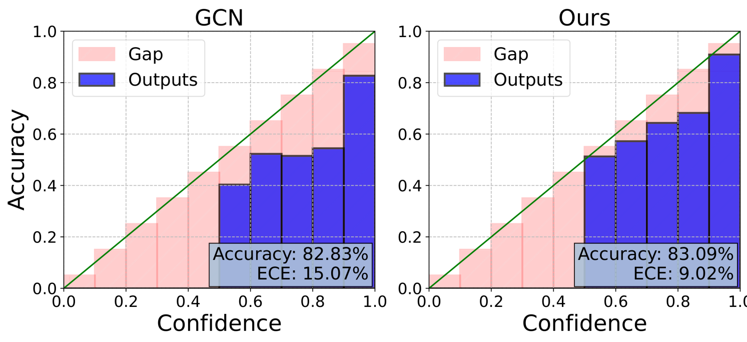
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

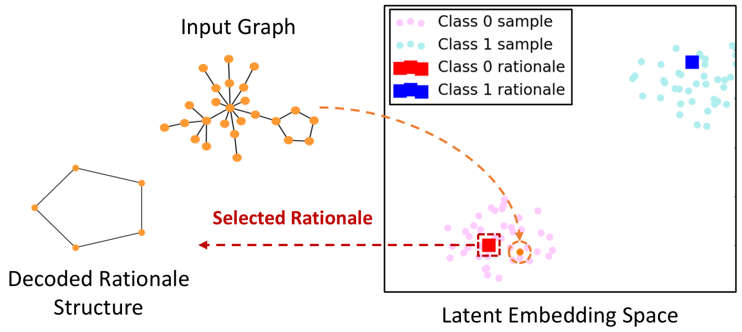
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A Additional Experiments

A.1 Fig. 4



(a) Calibration plot on the Graph-SST2 dataset.



(b) Visualization on the BA2Motifs dataset.

Figure 4: (a) GraphFNP improves model calibration significantly while maintaining high predictive performance. (b) GraphFNP learns a set of model-level rationales and decodes the structure of the correlated one for model interpretability.

A.2 Ablation Study

Table 3: Ablation Study on the BBBP and BACE datasets. We use GCN as the backbone. We set $K = 1$ when computing RF1.

	ECE		Acc/ROC-AUC		RF1	
	SST2	BACE	SST2	BACE	SST2	BACE
w/o $\mathbf{z}^D$	9.93±0.92	9.65±2.64	79.83±1.92	77.29±3.02	83.94±0.83	63.12±4.92
w/o $\mathbf{C}$	11.78±0.63	13.82±3.19	82.05±1.37	79.23±2.19	81.25±1.52	61.58±5.36
w/o $\mathbf{z}^R$	15.29±1.53	16.03±3.74	82.83±0.53	79.78±1.37	NA	NA
w/o EM	14.65±1.32	12.18±2.06	74.12±3.62	73.13±1.92	75.83±2.39	54.13±4.47
GraphFNP	**9.02**±0.58	**9.11**±2.96	**83.08**±0.26	**80.36**±0.89	**85.85**±0.67	**66.11**±3.75

We evaluate the impact of individual components in our method through an ablation study. The results obtained using the GCN architecture on the Graph-SST2 and BACE datasets are summarized in Table 3.

• The removal of the graph embedding $\mathbf{z}^D$ from the final predictive distribution leads to a significant decline in classification performance. This is due to the fact that without the graph embedding, the model may miss important information unique to the test graph and therefore have a poorer ability to generalize.

• Removing the binary stochastic correlation matrix $\mathbf{C}$ and using the pairwise kernel distance directly to obtain the local rationale embedding results in a decrease in the model's calibration score. This is because the sampling process of the binary stochastic matrix captures the uncertainty in data correlation, thereby improving the expected calibration error (ECE).

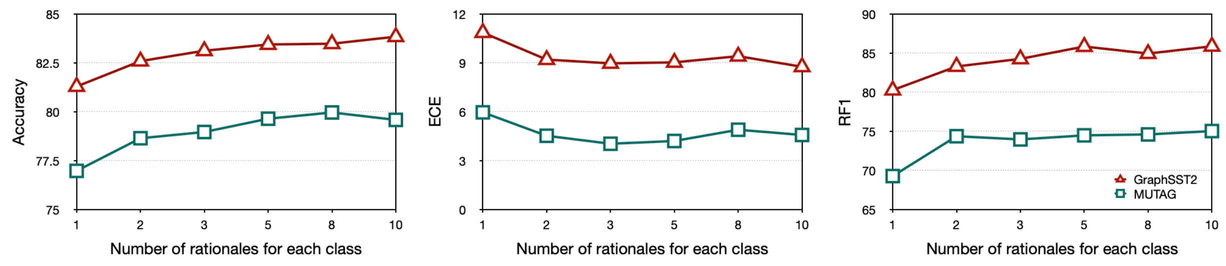


Figure 5: Hyper parameter study on the Graph-SST2 and MUTAG datasets. We use GCN as the backbone and set $K = 1$ when computing RF1.

- The removal of the rationale embedding ($\mathbf{z}^R$) results in a degradation of the Expected Calibration Error (ECE) to the same level as a basic Graph Convolutional Network (GCN) model. This highlights the crucial role the rationale embedding plays in the uncertainty quantification of our method. The RF1 metric cannot be evaluated in this scenario since the rationale embedding is not learned.
- By eliminating the EM algorithm from the training process, we optimize all model parameters together. We observe a marked decline in performance across all metrics. This is due to the coupling of the rationale and graph embeddings in the same embedding space, making simultaneous optimization difficult.

A.3 Compatibability with Post-hoc Calibration Method

In addition, post-hoc calibration techniques aim to enhance model calibration without altering the training procedure of standard neural networks. We evaluate the potential of such methods to improve the performance of existing uncertainty estimation techniques. Specifically, we apply temperature scaling, a widely-used post-hoc calibration method, to each method and evaluate their Expected Calibration Error (ECE) on the validation set. Table 4 shows the results on the Graph-SST2 and BACE datasets. As can be seen, with post-hoc calibration, GraphFNP significantly improves the ECE and remains the best among all methods in terms of uncertainty quantification performance.

Table 4: ECE with temperature scaling. We report the average performance and standard deviation for 5 random initializations

Backbone	GCN		GAT	
Model	Graph-SST2	BBBP	Graph-SST2	BBBP
Vanilla	13.34±1.04	15.72±1.8	11.49±0.51	18.35 ±1.84
DropEdge	11.95±1.29	12.25±2.01	8.02±0.82	16.02±1.96
MC-Dropout	11.35±1.24	13.51±1.73	10.74±1.21	15.51±3.46
SGLD	9.74±0.93	13.03±2.57	10.33±1.05	14.22±2.06
BBP	11.95±0.84	11.12±3.03	8.05±1.57	14.73±2.27
Graph-GP	10.76±1.13	13.12±2.13	9.27±1.25	15.01±1.67
GSAT	9.69±1.89	29.95±2.68	9.28±3.07	27.70±1.57
DeepEnsemble	8.26±NA	11.03±NA	7.43±6.96	13.74±NA
GraphFNP	7.25±0.74	**8.72**±1.26	**6.72**±1.38	**11.94**±2.97

A.4 Hyper-parameter Study

We conduct experiments to study the effect of the number of rationales on the Graph-SST2 and MUTAG datasets. As we can see from Fig. 5, the performance of GraphFNP is stable when R_k is larger than 2. In our experiments, we did not do extensive hyper-parameter tuning and simply set $R_k = 5$ for all the datasets.

B Additional Related Work

Uncertainty Quantification Bayesian Neural Networks (BNNs) [Blundell et al., 2015, Louizos and Welling, 2017, MacKay, 1992] are realized by first imposing prior distributions over NN parameters, next inferring parameter posteriors and integrating over them to make predictions. However, due to the intractability of posterior inference, approximation methods have been proposed, including variational inference [Blundell et al., 2015, Louizos and Welling, 2017], Monte Carlo dropout [Gal and Ghahramani, 2016] and stochastic gradient Markov chain Monte Carlo (SG-MCMC) [Li et al., 2016a, Zhang et al., 2019]. Neural Process (NP), a recently introduced framework [Garnelo et al., 2018] attempts to combine the stochastic processes and DNNs. It defines a distribution over a global latent variable to capture the functional uncertainty, while the Functional neural process (FNP) [Louizos et al., 2019] uses a dependency graph to encode the data correlation uncertainty. However, they are both designed for non-graph data. Besides the Bayesian methods, model ensembling [Lakshminarayanan et al., 2017] trains multiple DNNs with different initializations and uses their predictions for uncertainty estimation. However, training an ensemble of DNNs can be prohibitively expensive in practice. Kong et al. [2020b] proposes SDE-Net which quantifies uncertainty from a stochastic dynamical system perspective.

Graph Generation. The initial deep generation models for graphs were autoregressive, such as GraphRNN [You et al., 2018b] and GRAN [Liu et al., 2019] where nodes and edges were generated in a sequential manner. Later, Variational Autoencoder (VAE) based graph generation models were proposed[Simonovsky and Komodakis, 2018, Liu et al., 2019]. Normalizing flows have also been used for graph generation, with the first application introduced by Liu et al. [2019], where a flow models the node representations of a pre-trained autoencoder. Recently, GraphNVP [Madhawa et al., 2019], and GraphAF [Shi et al., 2019] have been proposed for molecular graph generation. GraphNVP consists of two flows, one for the adjacency matrix and another for the node types. GraphAF [Shi et al., 2019] is an autoregressive normalizing flow models that samples nodes and edges in sequence. Recently, diffusion models [Jo et al., 2022, Vignac et al., 2023, Kong et al., 2023] have been used for graph generation and achieved state-of-the-art performance.

C Datasets

We provide the dataset statistics in Table. 5. All datasets are publicly available:

1. BBBP and BACE: `https://moleculenet.org/`.
2. Graph-SST2: `https://github.com/Graph-COM/GSAT`
3. MUTAG: `https://github.com/flyingdoog/PGExplainer`
4. Github: `https://chrsmrrs.github.io/datasets/`

Table 5: Dataset statistics of BBBP, BACE, Graph-SST2, MUTAG, and Github

Datasets	BBBP	BACE	Graph-SST2	MUTAG	Github
# of Graphs	2039	505	70042	188	12725
# of Nodes (avg)	24.06	36.46	10.19	17.93	52.33
# of Edges (avg)	25.95	39.28	9.20	19.79	86.41
# of Nodes (min-max)	2-132	10-97	1-56	4-417	10-957
# of Edges (min-max)	1-145	10-101	0-55	3-112	9-1340

D Implementation details

We adopt ADAM Kingma and Ba [2014] as the optimizer for all the methods and select the learning rate from $\{1 \times 10^{-3}, 5 \times 10^{-4}, 1 \times 10^{-4}\}$ based on the predictive performance on the validation set. In the generator fine-tuning stage, we adopt a learning rate of 10^{-5} for all the datasets. For a fair comparison, we adopt the same graph convolutional network (GCN) and graph attention network (GAT) architectures for all the methods as the backbones. We found that the performance of GraphFNP is quite stable with regard to the hyper-parameter R_k, *i.e.*,the number of rationales for each class, and simply set it as 5 for all the datasets. For all the datasets, we employ three layers of GCNs with output dimensions equal to 256. We average all node representations as the whole graph representation. The final classifier contains three fully-connected layers in which the hidden dimension is set to 256. The MLPs used in GraphFNP are all one-hidden layers with ReLU activation with hidden dimension 256.

For all the baselines, we select the hyper-parameters from their recommended ranges in the original papers based on the validation predictive performance. For the GNN interpretability baseline, GNNExplainer, PGExplainer, and SubgraphX use the implementations provided by Dive into Graphs (DIG) library. GstarX and GSAT use their original code.

1. GNNExplainer, PGExplainer, and SubgraphX (based on DIG): `https://github.com/divelab/DIG`.
2. GstarX: `https://github.com/ShichangZh/GStarX`
3. GSAT: `https://github.com/Graph-COM/GSAT`

For XGNN, their original code cannot be used in the Pytorch-geometric framework. We reimplement it using the PyTorch-Geometric framework under the guidance of their paper and code in DIG library `https://github.com/divelab/DIG/blob/main/dig/xgraph/XGNN/gcn.py`.

All experiments are conducted on CPU: Intel(R) Core(TM) i7-5930K CPU @ 3.50GHz and GPU: NVIDIA GeForce RTX A5000 GPUs using python 3.8 and PyTorch 1.12.