Graph Neural Stochastic Diffusion for Estimating Uncertainty in Node Classification

Xixun Lin [1] **Wenxiao Zhang** [2] **Fengzhao Shi** [1] **Chuan Zhou** [3 4] **Lixin Zou** [5] **Xiangyu Zhao** [6] **Dawei Yin** [7] **Shirui Pan** [8] **Yanan Cao** [1 4]

Abstract

Graph neural networks (GNNs) have advanced the state of the art in various domains. Despite their remarkable success, the uncertainty estimation of GNN predictions remains under-explored, which limits their practical applications especially in risk-sensitive areas. Current works suffer from either intractable posteriors or inflexible prior specifications, leading to sub-optimal empirical results. In this paper, we present graph neural stochastic diffusion (GNSD), a novel framework for estimating predictive uncertainty on graphs by establishing theoretical connections between GNNs and stochastic partial differential equation. GNSD represents a GNN-based parameterization of the proposed graph stochastic diffusion equation which includes a Q-Wiener process to model the stochastic evolution of node representations. GNSD introduces a drift network to guarantee accurate prediction and a stochastic forcing network to model the propagation of epistemic uncertainty among nodes. Extensive experiments are conducted on multiple detection tasks, demonstrating that GNSD yields the superior performance over existing strong approaches.

1. Introduction

Nowadays graph neural networks (GNNs) have become a standard model architecture of learning graph-structured data, which benefit a series of real-world applications, e.g., molecular prediction (Stärk et al., 2022), recommender system (Ying et al., 2018) and traffic forecasting (Wang et al., 2020b). There has been a fair amount of GNNs in recent years, and the core operation is to design a powerful graph propagation mechanism where each node representation is iteratively updated by aggregating neighbour information (Kipf & Welling, 2017; Hamilton et al., 2017; Veličković et al., 2018; Balcilar et al., 2021).

Despite their superior predictive performance, GNNs are poor at estimating uncertainty in their decision process (Stadler et al., 2021). Uncertainty estimation (Gal & Ghahramani, 2016b; Lakshminarayanan et al., 2017; Malinin & Gales, 2018), which quantifies prediction confidence for enabling neural networks to know what they do not know, is an important requirement for deploying models on safety-critical areas. According to different source types, uncertainty is usually divided into *aleatoric uncertainty* and *epistemic uncertainty* (Gal et al., 2016). Aleatoric uncertainty represents the natural randomness inherent in the data distribution, such as label noise and class overlap; while epistemic uncertainty refers to the uncertainty in model parameters caused by the lack of observation data, which can be explained away given enough data (Kendall & Gal, 2017; Kong et al., 2020; Charpentier et al., 2020).

Most works of uncertainty estimation concentrate on i.i.d. data (Van Amersfoort et al., 2020; Abdar et al., 2021; Gawlikowski et al., 2023). The problem of estimating uncertainty for graphs is more complex, since interdependent nodes may follow different input feature distributions, and their predictive uncertainties are also significantly influenced by the graph structure. Previous approaches of uncertainty estimation over graphs follow the ensemble-based or Bayesian-based learning manner (Lakshminarayanan et al., 2017; Ng et al., 2018; Zhang et al., 2019), which are high computationally demanding. Current focus has shifted to deterministic uncertainty estimation for improving model efficiency. Nonetheless, they have to pre-specify prior distributions for performing posterior updates which degrade model flexibility and empirical performance (Zhao et al., 2020; Stadler et al., 2021).

In this paper, we introduce graph neural stochastic diffusion (GNSD), a novel framework of estimating uncertainty on graphs from the perspective of stochastic dynamical system.

[1]Institute of Information Engineering, Chinese Academy of Sciences [2]Beijing Jiaotong University [3]Academy of Mathematics and Systems Science, Chinese Academy of Sciences [4]School of Cyber Security, University of Chinese Academy of Sciences [5]Wuhan University [6]City University of Hong Kong [7]Baidu Inc. [8]Griffith University. Correspondence to: Yanan Cao <caoyanan@iie.ac.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Compared with the above methods of deterministic uncertainty estimation, GNSD is a more flexible model without relying on any specific form of prior distributions. It approximates the proposed graph stochastic diffusion equation by GNNs, where the feature propagation of nodes can be treated as discretizations of an underlying stochastic partial differential equation (SPDE) (Carmona et al., 1986). Different from existing graph diffusion equations (Chamberlain et al., 2021; Choi et al., 2023), the graph stochastic diffusion equation includes a Q-*Wiener process* to model the stochastic evolution of node embeddings. With such a learnt equation, GNSD corresponds to a stochastic process on graphs, from which we can sample the node embeddings from the representation distribution at the final time to quantify aleatoric and epistemic uncertainties explicitly.

GNSD owns two core modules to approximate the graph stochastic diffusion equation: *drift network* and *stochastic forcing network*. Drift network is developed as a continuous-depth GNN to parameterize the drift term in the graph stochastic diffusion equation, which aims to guarantee the stochastic dynamical system with good prediction ability. Stochastic forcing network is leveraged to capture epistemic uncertainty by describing the variance of Q-Wiener process, which integrates the graph Laplacian with added self-connections into the learning process for modeling the propagation of epistemic uncertainty among nodes. We further provide a theoretical analysis of the existence and uniqueness of the mild solution to the proposed equation.

The main contributions of this paper are summarized here: (1) We introduce a new formulation of uncertainty estimation for semi-supervised node classification within SPDE paradigms. Building upon this, we present GNSD, a GNN-based framework to capture aleatoric uncertainty and epistemic uncertainty by approximating the proposed graph stochastic diffusion equation. (2) We provide a novel definition of Q-Wiener process on graph domain and prove its validity theoretically. Moreover, we introduce an effective approximation of the discretization sampling of Q-Wiener process for solving the graph stochastic diffusion equation via the explicit design of stochastic forcing network. (3) We have conducted comprehensive experiments across multiple benchmark datasets in three tasks: out-of-distribution detection, misclassification detection and graph structure shifts. Empirical results and detailed analyses demonstrate the superior performance of our model compared with previous methods of uncertainty estimation on graphs.

2. Related Work

2.1. Uncertainty Estimation on Graphs

Uncertainty estimation for graph-structured data has recently attracted a lot of attention. Most previous works follow the ensemble-based or Bayesian-based learning manner. The ensemble methods (Lakshminarayanan et al., 2017; Goyal et al., 2020; Bazhenov et al., 2022) train multiple models with different initializations and use their predictions for estimating uncertainty. Bayesian graph neural networks (Zhang et al., 2019; Hasanzadeh et al., 2020; Pal et al., 2020) are representative works in Bayesian-based methods. They typically require an exact inference of model posterior which is hard for scaling to large-scale graphs.

Deterministic uncertainty estimation is a promising branch of uncertainty estimation on graphs, where Dirichlet-based approaches are gradually becoming mainstream (Zhao et al., 2020). Particularly, GPN (Stadler et al., 2021) is the state-of-the-art (SOTA), adopting normalizing flows (Rezende & Mohamed, 2015) to learn the probability density per class in the latent space. But they pre-specify the fixed-form model priors for posterior updates, which compromise model performance. Notice that there are several works on confidence calibration of GNNs, such as scaling-based (Wang et al., 2021; Hsu et al., 2022) and conformal prediction-based methods (Zargarbashi et al., 2023; Huang et al., 2023; Zargarbashi & Bojchevski, 2024). The key idea of them is to design an effective post-hoc calibration function for GNNs, which is orthogonal to our work.

2.2. Differential Equations on Graphs

Neural ODEs (Chen et al., 2018) provide a new learning paradigm for generalizing discrete neural layers to continuous transformations parameterized by differential equations, allowing smooth feature evolution in a view of dynamical systems. Motivated by this, recent works (Poli et al., 2019; Thorpe et al., 2021; Choi et al., 2023) try to describe differential equations on graphs for addressing some inherent issues in GNNs, e.g., the over-smoothing issue (Li et al., 2018). Graph diffusion process (Belkin & Niyogi, 2003; Liao et al., 2019; Chamberlain et al., 2021) is a fundamental way of graph learning, which aims to use PDEs to analyze the information diffusion on graphs. From the view of diffusion equations, the GNN layer and graph topology correspond to different discretizations of temporal and spatial operators, and many GNN architectures can be derived (Li et al., 2023). Different from them, we are the first work that introduces a GNN-based parameterization of the proposed graph stochastic diffusion equation to quantify predictive uncertainty for semi-supervised node classification.

3. Preliminary

In this section, we introduce some basic notations and the backgrounds of uncertainty types and of graph diffusion equation. The explanations of differential operators on graphs, e.g., divergence operator and involved derivations in graph diffusion equation are provided in Appendix A.

Notations. Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be an undirected graph, and $\mathcal{V}$ and $\mathcal{E}$ represent the sets of nodes and edges. The input feature matrix of all nodes is denoted as $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$, where $|\mathcal{V}|$ and d respectively denote the numbers of nodes and of input features. $\mathbf{A}$ is the adjacency matrix and each element $\mathbf{A}_{ij} \in \{0, 1\}$ denotes the connectivity between nodes i and j. The normalized graph Laplacian $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathsf{T}}$, where $\mathbf{D}$ is the diagonal degree matrix, $\mathbf{U}$ is the matrix of eigenvectors of $\mathbf{L}$ and $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues. In the task of semi-supervised node classification, $\mathcal{V}$ is split into the sets of labelled nodes $\mathcal{T}$ and of unlabeled nodes $\mathcal{U}$, i.e., $\mathcal{V} = \mathcal{T} \cup \mathcal{U}$. The classification goal is to infer the label vector of $\mathcal{U}$, i.e., $\mathbf{y} \in \{1, \ldots, c\}^{|\mathcal{U}|}$.

Uncertainty source. Predictive uncertainty comes from two sources: aleatoric uncertainty and epistemic uncertainty (Gal et al., 2016). Concretely, given $\mathbf{X}$ and $\mathcal{G}$, we can estimate a classification probability as follows:

$$P(\mathbf{y}|\mathbf{X}, \mathcal{G}) = \int P(\mathbf{y}|\mathbf{X}, \theta, \mathcal{G}) P(\theta|\mathcal{G}) \mathrm{d}\theta. \tag{1}$$

Let $\mathcal{H}$ denote the Shannon's entropy of a probability distribution. The total uncertainty of Eq.(1) can be quantified as the entropy $\mathcal{H}\big[\mathbb{E}_{P(\theta|\mathcal{G})}[P(\mathbf{y}|\mathbf{X}, \theta, \mathcal{G})]\big]$. Assuming that θ has a specific value, $\mathcal{H}[P(\mathbf{y}|\mathbf{X}, \theta, \mathcal{G})]$ measures the uncertainty coming from labels, thus aleatoric uncertainty can be given as the expectation of this quantity under $P(\theta|\mathcal{G})$, i.e., $\mathbb{E}_{P(\theta|\mathcal{G})}\big[\mathcal{H}[P(\mathbf{y}|\mathbf{X}, \theta, \mathcal{G})]\big]$. Epistemic uncertainty is then defined as the difference between the total and aleatoric uncertainties (Depeweg et al., 2018; Zhao et al., 2020):

$$\begin{aligned} I(\mathbf{y}, \theta|\mathbf{X}, \mathcal{G}) = \mathcal{H}\big[\mathbb{E}_{P(\theta|\mathcal{G})}[P(\mathbf{y}|\mathbf{X}, \theta, \mathcal{G})]\big] - \\ \mathbb{E}_{P(\theta|\mathcal{G})}\big[\mathcal{H}[P(\mathbf{y}|\mathbf{X}, \theta, \mathcal{G})]\big], \end{aligned} \tag{2}$$

which is the mutual information between $\mathbf{y}$ and θ and referred as the uncertainty coming from the posterior distribution of model parameters.

Graph diffusion equation. It is an important branch of graph PDEs (Vol'pert, 1972), which originally describes the diffusion motion of particles and is recently extended to the graph (Chamberlain et al., 2021):

$$\frac{\partial \mathbf{X}(t)}{\partial t} = \mathrm{div}\big(\mathbf{G}(\mathbf{X}(t), t)\nabla \mathbf{X}(t)\big), \tag{3}$$

where $\mathbf{X}(t)$ describes the evolution of $\mathbf{X}$ over time, div and ∇ are the divergence and gradient operators. $\mathbf{G}(\mathbf{X}(t), t) = \mathrm{diag}\big(\tilde{a}(\mathbf{x}_i(t), \mathbf{x}_j(t), t)\big)$ is the diffusivity factor parameterized as an $|\mathcal{E}| \times |\mathcal{E}|$ diagonal matrix, where $\tilde{a}$ is a similarity function to calculate the similarity between nodes i and j. Substituting the expressions of div and ∇ into Eq.(3), we can get

$$\frac{\partial \mathbf{X}(t)}{\partial t} = \big(\tilde{\mathbf{A}}(\mathbf{X}(t)) - \mathbf{I}\big)\mathbf{X}(t). \tag{4}$$

Here $\tilde{\mathbf{A}}(\mathbf{X}(t)) \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ is regarded as the attention matrix. When $\tilde{\mathbf{A}}(\mathbf{X}(t))$ is time-independent and fixed throughout the diffusion process, i.e., $\tilde{a}(\mathbf{x}_i(t), \mathbf{x}_j(t), t) = \tilde{a}(\mathbf{x}_i, \mathbf{x}_j)$, this diffusion equation has a linear form with the analytical solution; otherwise it is nonlinear with the following solution:

$$\mathbf{X}(T) = \mathbf{X}(0) + \int_0^T \frac{\partial \mathbf{X}(t)}{\partial t} \mathrm{d}t. \tag{5}$$

It describes the initial feature $\mathbf{X}(0)$ is continuously evolved from $t = 0$ to T to obtain the final node embeddings $\mathbf{X}(T)$.

4. Method

In this section, we first introduce the graph stochastic diffusion equation. Based on this graph SPDE, we present the graph neural stochastic diffusion (GNSD), a concrete GNN-based parameterization for uncertainty estimation on graphs. We then provide the discretization schemes of time, space and Q-Wiener process for solving this equation. The concrete procedure of uncertainty estimation and the model analysis are given in the end.

4.1. Graph Stochastic Diffusion Equation

The general form of the graph stochastic diffusion equation can be defined as[1]

$$\mathrm{d}\mathbf{H}(t) = \mathbf{f}(\mathbf{H}(t))\mathrm{d}t + \mathbf{g}(\mathbf{H}(t))\mathrm{d}\mathbf{W}(t). \tag{6}$$

We denote $\mathbf{H}(0) \in \mathbb{R}^{|\mathcal{V}| \times d'}$ as the initial node embeddings at $t = 0$ which is generated by an input encoder ϕ that embeds the input feature matrix $\mathbf{X}$ into the representation space. $\mathbf{H}(T) \in \mathbb{R}^{|\mathcal{V}| \times d'}$ represents the final node embeddings at $t = T$. $\mathbf{f}(\mathbf{H}(t))$ is the drift term that is determined by the Laplacian operator acting on the representation space, thereby facilitating a deterministic graph diffusion process, i.e., $\mathbf{f}(\mathbf{H}(t)) = (\tilde{\mathbf{A}}(\mathbf{H}(t)) - \mathbf{I})\mathbf{H}(t)$. The third part is the stochastic forcing term including a continuous operator $\mathbf{g}(\mathbf{H}(t))$ defined in Hilbert space and a high-dimensional Q-Wiener process $\mathbf{W}(t)$. The introduction of stochastic forcing term arms the original equation with stochasticity which is mainly driven by Q-Wiener process.

Definition 1 (Q**-Wiener process** (Lord et al., 2014)). *$\mathscr{H}$ is a separable Hilbert space and $(\Omega, \mathcal{F}, \mathcal{F}_t, \mathbb{P})$ is a filtered probability space. Let Q be a trace class non negative, symmetric operator on $\mathscr{H}$. A $\mathscr{H}$-valued stochastic process $W(t) : t \geq 0$ is called a Q-Wiener process if satisfies:(1) $W(0) = 0$ almost surely; (2) $W(t, \omega)$ is a continuous sample trajectory $\mathbb{R}^+ \mapsto \mathscr{H}$, for each $\omega \in \Omega$; (3) $W(t)$ is $\mathcal{F}_t$-adapted and has independent increments $W(t) - W(s)$ for $s < t$; (4) $W(t) - W(s) \sim \mathcal{N}(0, (t - s)Q)$ for all $0 \leq s \leq t$.*

[1] We have omitted the spatial symbol in $\mathbf{H}(t)$ and $\mathbf{W}(t)$ for simplifying notation.

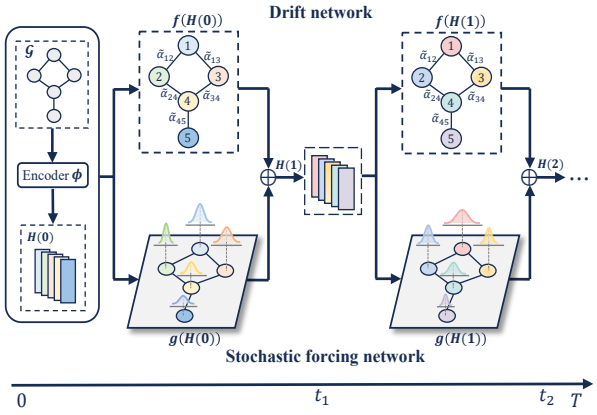


Figure 1. A simple overview of the proposed GNSD.

$W(t)$ can be extended to $\mathbf{W}(t) = [W_1(t), \ldots, W_{d'}(t)]$ used in Eq.(6) by sampling d' i.i.d. Q-Wiener processes.

$\mathbf{g}(\mathbf{H}(t))$ represents the variance of Q-Wiener process. Compared with the graph diffusion process in Eq.(5), adding the stochastic forcing term enables us to model epistemic uncertainty on graphs from the view of SPDE dynamical system. Specifically, for a region with the evident information, such as a specific community structure, the embedding evolution of the nodes belonging to this region should follow a relatively deterministic trajectory, and the variance of Q-Wiener process is small with low epistemic uncertainty. For a region with the ambiguous categorial or structural information, the evolution trajectory of the nodes in this region could be divergent, and the variance of Q-Wiener process is increased with the higher epistemic uncertainty. So we can capture epistemic uncertainty from $\mathbf{H}(t)$ which encodes the variance of Q-Wiener process.

4.2. Graph Neural Stochastic Diffusion

We present graph neural stochastic diffusion (GNSD), using GNNs to model the proposed graph stochastic diffusion equation. The core components of GNSD are the drift and stochastic forcing networks for respectively modeling $\mathbf{f}(\mathbf{H}(t))$ and $\mathbf{g}(\mathbf{H}(t))$, with the goal of achieving powerful model expressiveness and accurate uncertainty estimation. The whole model architecture is shown in Figure 1.

Drift network. Benefiting from the graph diffusion equation in Eq.(4), the drift network $\mathbf{f}_\theta$ can be developed as a continuous-depth GNN to learn node representations. Concretely, the diffusivity $\tilde{\mathbf{A}}(\mathbf{H}(t))$ is parameterized as a learnable multi-head self-attention matrix, i.e., $\tilde{\mathbf{A}}(\mathbf{H}(t)) = \frac{1}{l}\sum_l \tilde{\mathbf{A}}^l(\mathbf{H}(t))$. Each element $\tilde{a}^l_{i,j}(t)$ in the single-head attention matrix $\tilde{\mathbf{A}}^l(\mathbf{H}(t))$ is calculated as

$$\tilde{a}^l_{i,j}(t) = \frac{\exp\big(\sigma((\mathbf{a}^l)^\top[\mathbf{W}^l\mathbf{h}_i(t)||\mathbf{W}^l\mathbf{h}_j(t)])\big)}{\sum_{k\in\mathcal{N}_i}\exp\big(\sigma((\mathbf{a}^l)^\top[\mathbf{W}^l\mathbf{h}_i(t)||\mathbf{W}^l\mathbf{h}_k(t)])\big)}, \quad (7)$$

where $\mathbf{a}^l$ and $\mathbf{W}^l$ are two learnable parameters, σ is the LeakyReLU activation function, $\mathcal{N}_i$ denotes the neighborhood of node i in the graph, $(\cdot)^\top$ and $||$ represent the transposition and concatenation operations respectively. Through choosing different discretization schemes, $\mathbf{f}_\theta$ is a flexible framework to alleviate some inherent GNN issues, such as the over-smoothing issue.

Stochastic forcing network. The function of stochastic forcing network $\mathbf{g}_\theta$ is to learn the variance of Q-Wiener process for further capturing epistemic uncertainty. Since performing predictions on nodes relies on the feature information and the topological structure, $\mathbf{g}_\theta$ should take these two types of information into consideration. In light of this objective, the output $\mathbf{V}(\mathbf{H}(t))$ is given as

$$\mathbf{V}(\mathbf{H}(t)) = \mathbf{H}(t)\mathbf{W}^b\widehat{\mathbf{L}}, \quad (8)$$

where $\mathbf{W}^b$ denotes a linear transformation of node embeddings and $\widehat{\mathbf{L}}$ is the graph Laplacian with added self-connections, i.e., $\widehat{\mathbf{L}} = \mathbf{L} + \mathbf{I}$. Additionally, we apply the ReLU activation function on $\mathbf{V}(\mathbf{H}(t))$ to ensure that $\mathbf{g}_\theta$ is Lipschitz continuous (Kong et al., 2020; Zhang et al., 2024).

Highlights. We emphasize that the design of $\mathbf{g}_\theta$ has the following two important characteristics: (1) **Transitivity**. Following the graph homophily assumption that similar nodes tend to connect to each other more densely (McPherson et al., 2001), we arm $\mathbf{g}_\theta$ with $\widehat{\mathbf{L}}$ to model the propagation of epistemic uncertainty among nodes, i.e., connected nodes have similar epistemic uncertainty. (2) **Efficiency**. The solution of the graph stochastic diffusion equation involves the discretization sampling from Q-Wiener process on graph domain. In Appendix B.3, we show that $\mathbf{g}_\theta$ can serve as an effective approximation strategy to avoid complicated eigendecomposition and dense matrix multiplication.

4.3. Model Training

Discretization scheme. Similar to existing graph diffusion equations (Chamberlain et al., 2021; Choi et al., 2023), we discretize the spatial derivatives using the finite difference method, leaving a system of SDEs along the temporal axis. For graph-structured data, the spatial operator follows the input graph structure and is already discrete. For temporal discretization, we use the Euler method and the explicit Euler scheme for Eq.(6) is

$$\mathbf{H}_{n+1} = \mathbf{H}_n + \tau\big(\tilde{\mathbf{A}}(\mathbf{H}_n) - \mathbf{I}\big)\mathbf{H}_n + \mathbf{g}(\mathbf{H}_n)\Delta\mathbf{W}_\tau, \quad (9)$$

where $\tau = \frac{T}{N}$ is the discrete time step size, $\Delta\mathbf{W}_\tau := \mathbf{W}_{t+\tau} - \mathbf{W}_t$ is the increment of d' i.i.d. Q-Wiener processes at time t. This method has strong 0.5-order convergence and weak 1-order convergence. It is one-step numerical method since $\mathbf{H}_{n+1}$ is directly derived from $\mathbf{H}_n$.

Besides discretizitions of time and space, the solution of the graph stochastic diffusion equation requires a discrete sampling from Q-Wiener process where each increment is dependent on both time and the graph structure. To achieve this goal, we introduce a new definition of Q-Wiener process on graph domain.

Definition 2 (Q**-Wiener process on graph domain).** Let $\mathscr{G}$ denote a graph domain with a complete orthonormal basis $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_{|\mathcal{V}|})$ which corresponds to the eigenvectors of the graph Laplacian $\mathbf{L}$. $\mathbf{L}$ is non-negative definite and symmetric. λ_k is the eigenvalue of the eigenvector $\mathbf{u}_k$. Then, $W(t)$ is a Q-Wiener process on graph domain if and only if

$$W(t) = \sum_{k=1}^{|\mathcal{V}|} \sqrt{\lambda_k}\mathbf{u}_k\beta_k(t), \tag{10}$$

where $\{\beta_k(t)\}_{k=1}^{|\mathcal{V}|}$ are i.i.d. Brownian motions and this series converges in $L_2(\Omega, \mathscr{G})$.

Definition 2 is derived by using graph Fourier transform (Shuman et al., 2013). A more detailed description of Definition 2 and a corresponding validity proof are provided in Appendix B.1 and B.2 respectively.

However, Definition 2 requires the eigendecomposition of $\mathbf{L}$, and each discretization step would incur a dense matrix multiplication with eigenvectors. From the view of spectral graph theory (Chung, 1997), we adopt a first-order Taylor series on $\sqrt{\mathbf{\Lambda}}$ to derive an approximated sampling strategy for this Q-Wiener process:

$$\mathbf{g}(\mathbf{H}_n)\Delta W_\tau = \mathbf{H}_n\mathbf{W}^b\widehat{\mathbf{L}}\big(\boldsymbol{\beta}'(t+\tau) - \boldsymbol{\beta}'(t)\big), \tag{11}$$

where ΔW_τ is the increment of a single Q-Wiener process and $\boldsymbol{\beta}'(t)$ represents i.i.d. Brownian motions coming from graph domain. We can see that Eq.(11) has no requirements for eigendecomposition and eigenvector multiplications, so the above calculation overhead is alleviated. A detailed explanation of this part is given in Appendix B.3.

Loss function. We adopt the distributional uncertainty loss (Biloš et al., 2019) for training GNSD:

$$\mathcal{L} = \mathbb{E}_{\hat{P}(\mathbf{H}(T))}\big[\mathcal{H}[P(\mathbf{y}^*|\mathbf{X}, \mathcal{G}), P(\mathbf{y}|\mathbf{H}(T))]\big], \tag{12}$$

where $\hat{P}(\mathbf{H}(T)) = P(\mathbf{H}(T)|\mathbf{H}(0), \theta, \mathcal{G})$ is the conditional node representation distribution at the final time T, $P(\mathbf{y}^*|\mathbf{X}, \mathcal{G})$ denotes the true class distribution and $P(\mathbf{y}|\mathbf{H}(T))$ is modeled by a output decoder $\boldsymbol{\eta}$ for making prediction. Instead of the point estimation of the class distribution, this loss function is the expected cross-entropy which takes predictive uncertainty into consideration.

4.4. Uncertainty Estimation

After the model training is completed, GNSD can generate multiple random samples from $\hat{P}(\mathbf{H}(T))$ to estimate aleatoric uncertainty and epistemic uncertainty explicitly. Each sampling can generate the separate node embeddings $\mathbf{H}(T)$, which corresponds to a final solution of the proposed graph stochastic diffusion equation. Based on this flexible framework, aleatoric uncertainty can be given as $\mathbb{E}_{\hat{P}(\mathbf{H}(T))}\big[\mathcal{H}[P(\mathbf{y}|\mathbf{H}(T))]\big]$, while epistemic uncertainty can be calculated by the variance of the final solution $\mathbf{H}(T)$. In our model, we integrate the input features and the graph structural information into the learning process of both drift network $\mathbf{f}_\theta$ and stochastic forcing network $\mathbf{g}_\theta$. So GNSD provides an accurate uncertainty estimation of interdependent nodes under the graph homophily assumption.

Notice that the above sampling-computing procedure is similar to the ensemble methods. The key advantage of our model lies in that the modeling of predictive uncertainty is mainly driven by Q-Wiener process so that GNSD is a single model without the need of training multiple submodels simultaneously, which is more efficient in terms of computation and storage costs.

4.5. Model Analysis

Existence and uniqueness. We analyze the existence and uniqueness of the mild solution of the proposed graph stochastic diffusion equation. This unique mild solution guarantees that each node representation can evolve along a specific trajectory with Brownian motions based on its initial representation and the graph structure, which is helpful to alleviate the over-smoothing issue.

Theorem 1. *If $\boldsymbol{\psi}$ is a linear operator and has a complete orthonormal basis set and eigenvalues $\lambda_k > 0$, the continuous operator $\mathbf{g}$ satisfies Lipschitz condition, the initial node representation $\mathbf{h}_i(0)$ is square integrable and $\mathcal{F}_0$-measurable, then there exists a unique mild solution $\mathbf{h}_i(t)$ on $[0, T]$ for any $T > 0$ and $i \in \mathcal{V}$ such that*

$$\mathbf{h}_i(t) = e^{t\boldsymbol{\psi}}\mathbf{h}_i(0) + \int_0^t e^{(t-s)\boldsymbol{\psi}}\mathbf{g}(\mathbf{h}_i(s))\mathrm{d}\mathbf{W}(s), \tag{13}$$

where $e^{t\boldsymbol{\psi}}$ is the semigroup[2] generated by $\boldsymbol{\psi}$. Furthermore, there exists a constant $C_T > 0$ such that

$$\sup_{t\in[0,T]} ||\mathbf{h}_i(t)|| \leq C_T(1 + ||\mathbf{h}_i(0)||). \tag{14}$$

According to (Da Prato & Zabczyk, 2014), the linear operator $\boldsymbol{\psi}$ in Theorem 1 can be relaxed to the condition that $\boldsymbol{\psi}$ can generate a semigroup. In Appendix C, we provide a detailed analysis to show that our model satisfies the condition of the existence and uniqueness of the mild solution.

[2] $S(t) : \mathcal{X} \to \mathcal{X}$ is a semigroup if: 1) $S(0) = \mathbf{I}$ (the identity operator on the vector space $\mathcal{X}$) and 2) $S(t+s) = S(t)S(s)$ for $s, t \geq 0$.

Table 1. OOD detection performance comparison on Amazon-Computers with different OOD constructions.

Model	Label leave-out						Feature perturbation					
	AUROC	AUPR in	AUPR out	FPR95	DET ACC	ID ACC	AUROC	AUPR in	AUPR out	FPR95	DET ACC	ID ACC
GCN	82.35	56.46	93.67	56.06	74.72	87.34	80.55	78.53	78.55	80.67	75.09	86.67
GAT	80.66	53.19	93.05	53.91	72.65	89.57	73.69	78.00	65.61	97.41	75.76	85.46
GRAND	80.27	52.51	92.84	54.81	71.99	89.13	84.93	81.29	87.33	<u>54.98</u>	65.34	<u>87.60</u>
GREAD	80.56	54.05	92.70	54.14	72.68	<u>89.60</u>	85.38	79.07	<u>87.60</u>	59.10	68.29	86.67
MSP	74.88	47.53	89.64	75.52	68.85	85.80	72.86	74.50	67.73	95.70	70.81	83.47
ODIN	71.78	37.70	89.83	70.54	50.21	84.54	79.13	80.09	77.09	83.09	66.75	86.68
Mahalanobis	71.87	37.76	89.87	70.24	50.18	84.54	74.47	67.54	76.28	82.48	50.04	55.44
GNNsafe	<u>90.50</u>	<u>77.20</u>	95.05	48.25	<u>84.47</u>	89.37	<u>89.46</u>	<u>91.31</u>	84.01	75.62	76.49	87.25
GCN-Ensemble	79.53	52.39	91.99	69.28	73.51	85.44	77.71	79.45	72.60	94.20	77.51	79.36
BGCN	82.19	57.52	93.30	57.43	73.55	87.90	83.60	82.93	81.50	72.49	75.78	85.59
GKDE	76.46	48.18	90.64	73.36	64.35	74.68	71.69	71.40	69.04	90.83	69.70	82.79
GPN	88.76	68.23	<u>96.45</u>	<u>42.08</u>	81.02	88.87	87.92	85.99	85.98	67.10	<u>81.24</u>	86.95
GNSD	**94.06**	**82.27**	**97.06**	**31.47**	**88.76**	**89.64**	**95.95**	**94.76**	**94.02**	**15.69**	**91.28**	**87.99**

Computational complexity. The computational complexity of GNSD includes three main parts: node encoding, the calculations in $\mathbf{f}_\theta$, and the calculations in $\mathbf{g}_\theta$. The complexity of the first part is $\mathcal{O}(|\mathcal{V}|dd')$. The second one is dominated by Eq.(7): $\mathcal{O}(|\mathcal{E}|d')$. The last part in Eq.(8) is also implemented in the scheme of information aggregation so that its complexity is $\mathcal{O}(|\mathcal{E}||\mathcal{V}|)$. Thus, GNSD keeps an acceptable computational complexity, i.e., $\mathcal{O}\big(|\mathcal{V}|(dd' + |\mathcal{E}|)\big)$. In addition, both the input encoder ϕ and the output decoder η are implemented as the simple multilayer perceptrons (MLPs) for model efficiency. In Section 5.7, we provide an empirical study of runtime comparison.

5. Experiment

In this section, we conduct extensive experiments in the scenario of semi-supervised node classification with the following tasks: out-of-distribution (OOD) detection, misclassification detection and graph structure shifts. For each task, uncertainty estimation plays a critical role. Additionally, we conduct a series of model analyses including model variants, visual study and runtime comparison for comprehensive evaluations.

5.1. Experimental Setup

Datasets. We evaluate our model on five benchmark graph datasets including one co-purchase graph (McAuley et al., 2015) (**Amazon-Computers**), three citation graphs (Sen et al., 2008) (**Cora**, **CiteSeer** and **Pubmed**) and one academic graph (Hu et al., 2020) (**OGBN-Arxiv**). The detailed descriptions of them are summarized in Appendix E.2.

Baselines. We compare GNSD with the following three model families. The first family includes four strong GNNs: **GCN** (Kipf & Welling, 2017), **GAT** (Veličković et al., 2018), **GRAND** (Chamberlain et al., 2021) and **GREAD** (Choi et al., 2023). The second family is OOD detection baselines: **MSP** (Hendrycks & Gimpel, 2016), **ODIN** (Liang et al., 2018), **Mahalanobis** (Lee et al., 2018) and **GNNsafe** (Wu et al., 2023). Among them, the first three models are standard OOD detection methods, we replace their original backbones with a GCN encoder to handle graph-structured data. GNNsafe is current SOTA OOD detection model for GNN-based learning on graphs. The third family is representative uncertainty estimation methods on graphs: ensemble-based GCN (**GCN-Ensemble**) (Lakshminarayanan et al., 2017), Bayesian-based GCN (**BGCN**) (Zhang et al., 2019), **GKDE** (Zhao et al., 2020) and **GPN** (Stadler et al., 2021).

For GNN baselines, we use aleatoric uncertainty for all detection tasks. For uncertainty estimation methods, we use epistemic uncertainty for OOD detection and aleatoric uncertainty for misclassification detection as suggested by previous works (Zhao et al., 2020; Stadler et al., 2021). Further baseline details are provided in Appendix E.3.

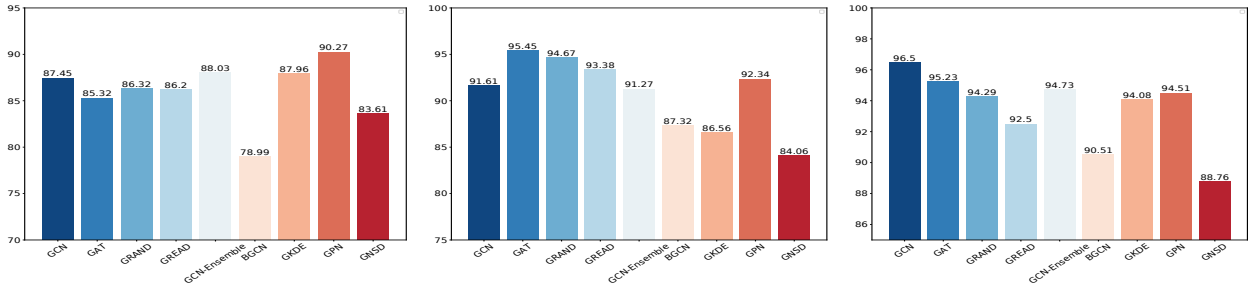
Evaluation metrics. For an ideal uncertainty estimation GNN, it should own the ability of detecting abnormal samples while keeping good predictive performance on normal samples. Based on this view, we use AUROC, AUPR, FPR95, detection accuracy (DET ACC) and in-distribution classification accuracy (ID ACC) as evaluation metrics. Larger values of AUROC, AUPR, DET ACC and ID ACC indicate better performance while a smaller value of FPR95 is more preferable. We show the best two results in bold (**first**) and underlined (<u>second</u>). '-' denotes the out-of-memory issue in reported results. The more information about these metrics is given in Appendix E.4.

5.2. OOD Detection

We evaluate our model in two OOD scenarios: (1) Label leave-out: We use nodes with partial classes as in-distribution data and leave out others as OOD data. Concretely, the categories which are greater than 3 are classified as ID classes for Cora; the categories which are greater than

Table 2. Misclassification detection performance comparison on three datasets.

Model	Amazon-Computers				Cora				CiteSeer			
	AUROC	AUPR succ	AUPR err	FPR95	AUROC	AUPR succ	AUPR err	FPR95	AUROC	AUPR succ	AUPR err	FPR95
GCN	79.78	95.98	34.04	74.87	76.04	92.54	42.12	79.81	71.03	81.22	47.50	85.52
GAT	81.45	95.81	36.55	73.35	80.88	94.80	48.90	69.35	74.27	82.18	52.93	82.67
GRAND	79.62	95.57	33.25	75.02	76.13	92.70	43.29	78.54	71.52	81.63	54.24	82.70
GREAD	80.90	94.40	39.67	75.64	81.49	94.31	49.77	72.80	70.90	80.92	50.80	84.05
GCN-Ensemble	80.79	95.78	39.26	74.59	76.09	92.53	44.17	75.24	72.44	72.51	60.90	87.40
BGCN	81.63	95.71	36.92	73.48	78.08	93.13	47.45	73.56	71.55	83.19	48.80	82.57
GKDE	79.39	95.94	32.77	73.72	71.55	85.60	48.63	81.29	70.01	82.10	50.66	84.01
GPN	80.23	96.09	36.37	72.69	76.14	89.06	52.11	77.82	75.69	81.98	61.70	**78.35**
GNSD	**83.08**	**96.34**	**47.09**	**66.48**	**88.04**	**97.86**	**58.43**	**51.22**	**76.04**	**83.68**	**61.98**	82.26



(a) Cora (b) CiteSeer (c) Pubmed

Figure 2. Performance (%) comparisons of the relative average aleatoric confidence on three citation graphs with structure shifts.

2 are classified as ID classes for CiteSeer; the categories which are greater than or equal to 1 are classified as ID classes for Pubmed; the categories which are greater than 5 are classified as ID classes for Amazon-Computers; the categories which are greater than 20 are classified as ID classes for OGBN-Arxiv.

(2) Feature perturbation: We use the original graph as in-distribution data, and the linear interpolation of original features and random features is used for creating perturbed node features as OOD data.

The OOD detection results of Amazon-Computers are provided in Table 1 and more experimental results are provided in Appendix E.5 (Tables 4-8). From the reported results, we conclude that GNSD achieves the best performance for OOD detection. Compared with previous SOTA baselines, GNSD averagely increases AUROC by 5.6%, AUPR in by 5.2%, AUPR out by 4.1%, DET ACC by 8.7% and reduces FPR95 by 48.3% in two OOD scenarios. Meanwhile, it still keeps competitive classification results as shown in the ID ACC results. It demonstrates that GNSD can well recognize OOD nodes by modeling predictive uncertainty and has accurate prediction ability.

5.3. Misclassification Detection

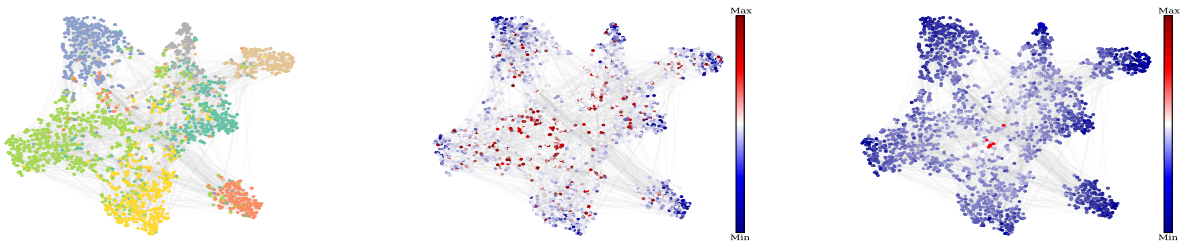
Besides OOD detection, we also report the results for the detection of misclassified samples. Misclassification detection is also an important task of testing whether the uncertainty estimation models could aware of prediction mistakes at test time (Hendrycks & Gimpel, 2016). The results of Amazon-Computers, Cora and CiteSeer are shown in Table 2, and remaining results are provided in Appendix E.5 (Table 9). From Table 2, we conclude that GNSD achieves the best detection performance. In particular, compared with the SOTA uncertainty estimation model GPN, GNSD can increase AUROC by 6.5%, AUPR succ by 4.1%, AUPR err by 14.0% and reduce FPR95 by 12.6% on three datasets averagely. Thus, GNSD is good at using aleatoric uncertainty to identify misclassified test samples.

5.4. Graph Structure Shifts

We test whether our model could be used for recognizing structure shifts via uncertainty estimation. Following GPN (Stadler et al., 2021), we adopt DICE (Waniek et al., 2018) as the perturbation strategy to generate shifted graphs. DICE is a classic global adversarial attack, which inserts edges between nodes from different classes and deletes edges between nodes from the same classes. We leverage DICE to perturb 30% edges in three citation graphs: Cora, CiteSeer and Pubmed. The relative average aleatoric confidence is used according to the consistent observations in GPN. Figure 2 shows the concrete results. We can see that the edge perturbation leads to more aleatoric uncertain predictions and GNSD can perceive the shift of graph structures

Table 3. OOD detection performance comparison (AUROC) of model variants on all datasets.

Model	Label leave-out					Feature perturbation				
	Amazon	Cora	CiteSeer	Pubmed	Arxiv	Amazon	Cora	CiteSeer	Pubmed	Arxiv
GNSD-L	92.53	94.69	**83.17**	75.48	77.81	94.72	93.74	**88.49**	95.68	99.24
GNSD-S	93.20	**94.93**	82.30	76.10	-	94.58	94.46	87.81	96.79	-
GNSD-E	**95.11**	94.87	83.02	78.44	-	**96.58**	**94.58**	88.06	96.60	-
GNSD	94.06	94.76	82.95	**78.81**	**78.82**	95.95	94.41	86.82	**97.09**	**99.55**



(a) Ground truth classes (b) Epistemic uncertainty at $t = 0$ (c) Epistemic uncertainty at $t = T$

Figure 3. Visual results of the propagation of epistemic uncertainty on Cora.

via the change of aleatoric confidence.

5.5. Variant Study

We consider three different model variants of GNSD to verify model effectiveness: **GNSD-L**, **GNSD-S** and **GNSD-E**. Following GRAND, we derive a linear version of GNSD, i.e., GNSD-L where the attention weights in $\mathbf{f}_\theta$ are fixed throughout the integration. GNSD-S represents that we use the Stochastic Runge-Kutta method (Kloeden et al., 1992) as the SDE solver. GNSD-E denotes that we perform the exact discretization sampling of Q-Wiener process instead of using the proposed approximation strategy. We conduct OOD detection on all datasets and report AUROC results in Table 3. From it, we have the following three conclusions: 1) By comparing GNSD-L and GNSD, we find that GNSD is more suitable to detection tasks; 2) Although GNSD-S adopts a more complex discretization scheme, the gap in outcomes is unnoticeable; 3) The approximation of the discretization sampling is an effective strategy, since GNSD keeps the comparable performance and without the out of memory exception in GNSD-E for large-scale graphs.

5.6. Visual Analysis

Figure 3 shows the visual results of epistemic uncertainty among nodes at $t = 0$ and $t = T$ on Cora. The whole epistemic uncertainty is high as shown in (b). Because in the beginning GNSD has learned the little graph information, especially for the nodes located at the middle region. With the evolution of node embeddings, epistemic uncertainty is propagated along the graph structure and is gradually decreasing, which is reasonable since these nodes have been observed in the training phase as shown in (c).

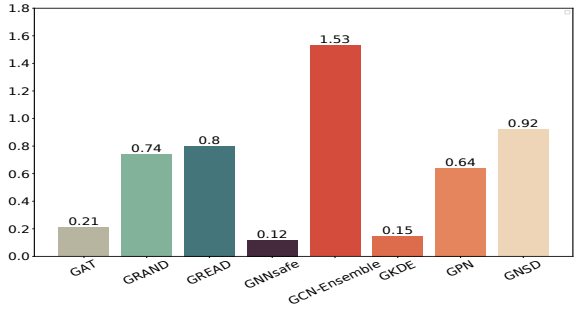


Figure 4. Runtime comparison of one epoch (in seconds).

5.7. Runtime Comparison

We provide an empirical runtime comparison between GNSD and seven strong baselines. The experiment is conducted on Amazon-Computers for OOD detection in the setting of label leave-out. Figure 4 demonstrates that the runtime complexity of GNSD is acceptable. Through the respective comparisons of GRAND and of GPN with our model, we can conclude that GNSD arms the continuous-depth GNNs with the ability of uncertainty estimation without incurring the extra high computational complexity.

6. Conclusion

We present GNSD, augmenting GNNs with the ability of uncertainty estimation through modeling the proposed graph stochastic diffusion equation. To solve this equation, we introduce a new Q-Wiener process on graph domain and provide an efficient approximation to support its discretization sampling. The distinctive network design enables GNSD to satisfy the condition of the existence and uniqueness of the mild solution of the stochastic dynamical system. Extensive experiments show that GNSD outperforms many strong baselines. There are two interesting directions we want to further explore: 1) studying the more advanced architectures of drift network and stochastic forcing network which are applicable to both homophily and heterophily settings; 2) investigating how to deploy GNSD on large-scale graphs.

Impact Statement

We develop a new GNN framework via modeling the graph SPDE. We hope that our study can facilitate a new methodology of uncertainty estimation on graphs. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgements

This work is supported by the National Key Research and Development Program of China (NO.2022YFB3102200), the Strategic Priority Research Program of the Chinese Academy of Sciences (XDB0680101), the National Natural Science Foundation of China (No.62302345, No.U23A20305), the Research Impact Fund (No.R1015-23), APRC-CityU New Research Initiatives (No.9610565, Start-up Grant for New Faculty of CityU), CityU-HKIDS Early Career Research Grant (No.9360163), Hong Kong ITC Innovation and Technology Fund Midstream Research Programme for Universities Project (No.ITS/034/22MS), Hong Kong Environmental and Conservation Fund (No.88/2022), SIRG-CityU Strategic Interdisciplinary Research Grant (No.7020046, No.7020074), Huawei Innovation Research Program, CCF-Tencent Open Fund, Tencent Rhino-Bird Focused Research Program, CCF-Ant Research Fund, Ant Group Research Fund, CCF-BaiChuan-Ebtech Foundation Model Fund, and Kuaishou.

References

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, 2021.

Balcilar, M., Héroux, P., Gauzere, B., Vasseur, P., Adam, S., and Honeine, P. Breaking the limits of message passing graph neural networks. In *International Conference on Machine Learning*, 2021.

Bazhenov, G., Kuznedelev, D., Malinin, A., Babenko, A., and Prokhorenkova, L. Revisiting uncertainty estimation for node classification: New benchmark and insights. 2022.

Belkin, M. and Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

Biloš, M., Charpentier, B., and Günnemann, S. Uncertainty on asynchronous time event prediction. *Advances in Neural Information Processing Systems*, 32, 2019.

Carmona, R., Kesten, H., Walsh, J. B., and Walsh, J. B. *An introduction to stochastic partial differential equations*. Springer, 1986.

Chamberlain, B., Rowbottom, J., Gorinova, M. I., Bronstein, M., Webb, S., and Rossi, E. Grand: Graph neural diffusion. In *International Conference on Machine Learning*, pp. 1407–1418. PMLR, 2021.

Charpentier, B., Zügner, D., and Günnemann, S. Posterior network: Uncertainty estimation without ood samples via density-based pseudo-counts. *Advances in Neural Information Processing Systems*, 33:1356–1367, 2020.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Chien, Y. and Fu, K.-S. On the generalized karhunen-loève expansion (corresp.). *IEEE Transactions on Information Theory*, 13(3):518–520, 1967.

Choi, J., Hong, S., Park, N., and Cho, S.-B. Gread: Graph neural reaction-diffusion networks. In *International Conference on Machine Learning*, 2023.

Chung, F. R. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

Da Prato, G. and Zabczyk, J. *Stochastic equations in infinite dimensions*. Cambridge university press, 2014.

Depeweg, S., Hernandez-Lobato, J.-M., Doshi-Velez, F., and Udluft, S. Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. In *International Conference on Machine Learning*, pp. 1184–1193. PMLR, 2018.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, 2016a.

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

Gal, Y. et al. Uncertainty in deep learning. *Ph.D. thesis*, 2016.

Gasteiger, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

Gau, H.-L., Wang, K.-Z., and Wu, P. Y. Numerical ranges of row stochastic matrices. *Linear Algebra and its Applications*, 506:478–505, 2016.

Gawlikowski, J., Tassi, C. R. N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., et al. A survey of uncertainty in deep neural networks. *Artificial Intelligence Review*, 56(Suppl 1):1513–1589, 2023.

Goyal, P., Raja, S., Huang, D., Chhetri, S. R., Canedo, A., Mondal, A., Shree, J., and Jawahar, C. Graph representation ensemble learning. In *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 24–31. IEEE, 2020.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Hasanzadeh, A., Hajiramezanali, E., Boluki, S., Zhou, M., Duffield, N., Narayanan, K., and Qian, X. Bayesian graph neural networks with adaptive connection sampling. In *International conference on machine learning*, pp. 4094–4104. PMLR, 2020.

Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

Hilt, D. E. and Seegrist, D. W. *Ridge, a computer program for calculating ridge regression estimates*. Department of Agriculture, Forest Service, Northeastern Forest Experiment . . . , 1977.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hsu, H. H.-H., Shen, Y., Tomani, C., and Cremers, D. What makes graph neural networks miscalibrated? *Advances in Neural Information Processing Systems*, 35:13775–13786, 2022.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 2020.

Huang, K., Jin, Y., Candes, E., and Leskovec, J. Uncertainty quantification over graph with conformalized graph neural networks. *Advances in Neural Information Processing Systems*, 2023.

Kendall, A. and Gal, Y. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Kittaneh, F. and Manasrah, Y. Improved young and heinz inequalities for matrices. *Journal of Mathematical Analysis and Applications*, 361(1):262–269, 2010.

Kloeden, P. E., Platen, E., Kloeden, P. E., and Platen, E. *Stochastic differential equations*. Springer, 1992.

Kong, L., Sun, J., and Zhang, C. Sde-net: Equipping deep neural networks with uncertainty estimates. In *International Conference on Machine Learning*, 2020. URL https://api.semanticscholar.org/CorpusID:221082095.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Lee, K., Lee, K., Lee, H., and Shin, J. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, 2018.

Li, Y., Wang, X., Liu, H., and Shi, C. A generalized neural diffusion framework on graphs. *arXiv preprint arXiv:2312.08616*, 2023.

Liang, S., Li, Y., and Srikant, R. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations*, 2018.

Liao, R., Zhao, Z., Urtasun, R., and Zemel, R. S. Lanczosnet: Multi-scale deep graph convolutional networks. In *International Conference on Machine Learning*, 2019.

Lord, G. J., Powell, C. E., and Shardlow, T. *An introduction to computational stochastic PDEs*, volume 50. Cambridge University Press, 2014.

Malinin, A. and Gales, M. Predictive uncertainty estimation via prior networks. *Advances in neural information processing systems*, 31, 2018.

McAuley, J., Targett, C., Shi, Q., and Van Den Hengel, A. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 2015.

McPherson, M., Smith-Lovin, L., and Cook, J. M. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 2001.

Ng, Y. C., Colombo, N., and Silva, R. Bayesian semi-supervised learning with graph gaussian processes. *Advances in Neural Information Processing Systems*, 31, 2018.

Pal, S., Malekmohammadi, S., Regol, F., Zhang, Y., Xu, Y., and Coates, M. Non parametric graph learning for bayesian graph neural networks. In *Conference on uncertainty in artificial intelligence*, pp. 1318–1327. PMLR, 2020.

Poli, M., Massaroli, S., Park, J., Yamashita, A., Asama, H., and Park, J. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532*, 2019.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1530–1538, 2015.

Salvi, C., Lemercier, M., and Gerasimovics, A. Neural stochastic pdes: Resolution-invariant learning of continuous spatiotemporal dynamics. *Advances in Neural Information Processing Systems*, 35:1333–1344, 2022.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 2008.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

Stadler, M., Charpentier, B., Geisler, S., Zügner, D., and Günnemann, S. Graph posterior network: Bayesian predictive uncertainty for node classification. *Advances in Neural Information Processing Systems*, 34:18033–18048, 2021.

Stärk, H., Beaini, D., Corso, G., Tossou, P., Dallago, C., Günnemann, S., and Liò, P. 3d infomax improves gnns for molecular property prediction. In *International Conference on Machine Learning*, pp. 20479–20502. PMLR, 2022.

Thorpe, M., Nguyen, T. M., Xia, H., Strohmer, T., Bertozzi, A., Osher, S., and Wang, B. Grand++: Graph neural diffusion with a source term. In *International Conference on Learning Representations*, 2021.

Van Amersfoort, J., Smith, L., Teh, Y. W., and Gal, Y. Uncertainty estimation using a single deep deterministic neural network. In *International conference on machine learning*, pp. 9690–9700. PMLR, 2020.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Vol'pert, A. I. Differential equations on graphs. *Mathematics of the USSR-Sbornik*, 17(4):571, 1972.

Wang, K., Shen, Z., Huang, C., Wu, C.-H., Dong, Y., and Kanakia, A. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413, 2020a.

Wang, X., Ma, Y., Wang, Y., Jin, W., Wang, X., Tang, J., Jia, C., and Yu, J. Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of the web conference 2020*, pp. 1082–1092, 2020b.

Wang, X., Liu, H., Shi, C., and Yang, C. Be confident! towards trustworthy graph neural networks via confidence calibration. *Advances in Neural Information Processing Systems*, 34:23768–23779, 2021.

Waniek, M., Michalak, T. P., Wooldridge, M. J., and Rahwan, T. Hiding individuals and communities in a social network. *Nature Human Behaviour*, 2(2):139–147, 2018.

Wu, Q., Chen, Y., Yang, C., and Yan, J. Energy-based out-of-distribution detection for graph neural networks. In *International Conference on Learning Representations*, 2023.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48. PMLR, 2016.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 974–983, 2018.

Zargarbashi, S. H. and Bojchevski, A. Conformal inductive graph neural networks. In *The Twelfth International Conference on Learning Representations*, 2024.

Zargarbashi, S. H., Antonelli, S., and Bojchevski, A. Conformal prediction sets for graph neural networks. In *International Conference on Machine Learning*, pp. 12292–12318. PMLR, 2023.

Zhang, S., Zhou, C., Liu, Y., Zhang, P., Lin, X., and Ma, Z.-M. Neural jump-diffusion temporal point processes. In *International Conference on Machine Learning*, 2024.

Zhang, Y., Pal, S., Coates, M., and Ustebay, D. Bayesian graph convolutional neural networks for semi-supervised classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 5829–5836, 2019.

Zhao, X., Chen, F., Hu, S., and Cho, J.-H. Uncertainty aware semi-supervised learning on graph data. *Advances in Neural Information Processing Systems*, 33:12827–12836, 2020.

Appendix

A. Differential Operators on Graphs

In diffusion processes, the definition of spatial differential operators in diffusion PDEs actually reflects the underlying domain's structure (Lord et al., 2014). Some fundamental concepts that are well-defined on Riemannian manifolds, such as vector inner product, scalar inner product, gradient and divergence, play a pivotal role in shaping these spatial differential operators. When transitioning to graphs, the analogous definitions of these concepts should be established to accommodate the specific characteristics of graphs (Chamberlain et al., 2021; Thorpe et al., 2021).

For an undirected graph with the row feature vectors of all input nodes $\mathbf{X} = (\mathbf{x}_1^\mathsf{T}, \ldots, \mathbf{x}_{|\mathcal{V}|}^\mathsf{T})^\mathsf{T} \in \mathbb{R}^{|\mathcal{V}| \times d}$ and the adjacency matrix $\mathbf{A}$, the functions defined on nodes and edges can be expressed as node fields $\mathbf{P} = (\mathbf{p}_1^\mathsf{T}, \ldots, \mathbf{p}_{|\mathcal{V}|}^\mathsf{T})^\mathsf{T} \in \mathbb{R}^{|\mathcal{V}| \times k_1}$ and edge fields $\mathcal{P} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}| \times k_2}$ by analogy to scalar and vector fields on Riemannian manifolds. The edge fields are alternating, i.e., $\mathcal{P}_{ij} = -\mathcal{P}_{ji}$. For convenience, let $k_1 = k_2 = d$, and we define the inner products of node fields and edge fields as

$$\langle \mathbf{P}, \mathbf{M} \rangle = \sum_{i=1}^{|\mathcal{V}|} \mathbf{p}_i \cdot \mathbf{m}_i, \qquad \langle \mathcal{P}, \mathcal{M} \rangle = \frac{1}{2} \sum_{i,j=1}^{|\mathcal{V}|} \mathbf{A}_{ij} \mathcal{P}_{ij} \cdot \mathcal{M}_{ij}. \tag{15}$$

Gradient describes the changes in space of physical quantities, while divergence is used for measuring the strength of the sources or sinks in a vector field. Since graphs usually propagate information through edges, the gradient operator of node fields $\mathbf{P} = (\mathbf{p}_1^\mathsf{T}, \ldots, \mathbf{p}_{|\mathcal{V}|}^\mathsf{T})^\mathsf{T} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is defined as edge fields $(\nabla \mathbf{P}) \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}| \times d}$, where $(\nabla \mathbf{P})_{ij} = \mathbf{p}_j - \mathbf{p}_i$. The divergence operator of node i, i.e., $(\mathrm{div}(\mathcal{M}))_i$ is defined by edge fields $\mathcal{M} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}| \times d}$ as

$$(\mathrm{div}(\mathcal{M}))_i = \sum_{j=1}^{|\mathcal{V}|} \mathbf{A}_{ij} \mathcal{M}_{ij}. \tag{16}$$

The gradient and divergence operators are adjoint, i.e., $\langle \nabla \mathbf{P}, \mathcal{M} \rangle = \langle \mathbf{P}, \mathrm{div}(\mathcal{M}) \rangle$.

When $\mathbf{X}(t) = (\mathbf{x}_1(t)^\mathsf{T}, \ldots, \mathbf{x}_{|\mathcal{V}|}(t)^\mathsf{T})^\mathsf{T}$ describes the evolution of $\mathbf{X}$ over time, the attention matrix $\tilde{\mathbf{A}}(\mathbf{X}(t)) \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ also changes over time, where $\tilde{\mathbf{A}}_{ij}(\mathbf{X}(t)) = \tilde{a}(\mathbf{x}_i(t), \mathbf{x}_j(t), t)$ represents the similarity of the time-evolving features between nodes i and j. Following GRAND (Chamberlain et al., 2021), we assume $\tilde{a}(\mathbf{x}_i(t), \mathbf{x}_j(t), t) = \tilde{a}(\mathbf{x}_i(t), \mathbf{x}_j(t))$ in $\mathbf{G}(\mathbf{X}(t), t)$ for the sake of simplicity. According to the above definitions of gradient and divergence operators, the graph diffusion equation in Eq.(3) is introduced as

$$\begin{pmatrix} \frac{\partial \mathbf{x}_1(t)}{\partial t} \\ \vdots \\ \frac{\partial \mathbf{x}_{|\mathcal{V}|}(t)}{\partial t} \end{pmatrix} = \begin{pmatrix} \sum_{j \neq 1} \tilde{a}(\mathbf{x}_1(t), \mathbf{x}_j(t)) \mathbf{x}_j(t) - \mathbf{x}_1(t) \\ \vdots \\ \sum_{j \neq |\mathcal{V}|} \tilde{a}(\mathbf{x}_{|\mathcal{V}|}(t), \mathbf{x}_j(t)) \mathbf{x}_j(t) - \mathbf{x}_{|\mathcal{V}|(t)} \end{pmatrix} \tag{17}$$

where $\sum_{j \neq i} \tilde{a}(\mathbf{x}_i(t), \mathbf{x}_j(t)) = 1$ for all $i = 1, \ldots, |\mathcal{V}|$, and $\tilde{a}(\mathbf{x}_i(t), \mathbf{x}_j(t)) = 0$ if $(i, j) \notin \mathcal{E}$. We convert this matrix into the form of matrix product as used in Eq.(4):

$$\begin{pmatrix} \frac{\partial \mathbf{x}_1(t)}{\partial t} \\ \vdots \\ \frac{\partial \mathbf{x}_{|\mathcal{V}|}(t)}{\partial t} \end{pmatrix} = \begin{pmatrix} -1 & \cdots & \tilde{a}(\mathbf{x}_1(t), \mathbf{x}_{|\mathcal{V}|}(t)) \\ \vdots & \vdots & \vdots \\ \tilde{a}(\mathbf{x}_{|\mathcal{V}|}(t), \mathbf{x}_1(t)) & \cdots & -1 \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{|\mathcal{V}|} \end{pmatrix} = \big(\tilde{\mathbf{A}}(\mathbf{X}(t)) - \mathbf{I}\big)\mathbf{X}(t). \tag{18}$$

When $\tilde{\mathbf{A}}(\mathbf{X}(t))$ is only dependent on the initial feature matrix $\mathbf{X}$, i.e., $\tilde{\mathbf{A}}(\mathbf{X}(t)) = \tilde{\mathbf{A}}(\mathbf{X})$, $\tilde{\mathbf{A}}(\mathbf{X})$ is right-stochastic and the equation is linear with an analytical solution $\mathbf{X}(t) = e^{t(\tilde{\mathbf{A}}(\mathbf{X}) - \mathbf{I})}\mathbf{X}(0)$. Otherwise, we get a general form of the solution in Eq.(5).

The numerical methods of PDEs require the discretizations of space and of time respectively. It is critical to choose appropriate numerical methods. Commonly used methods include finite difference method (FDM), finite element method (FEM), finite volume method (FVM) and spectral method (Lord et al., 2014). For graph-structured data, employing FDM for spatial derivative discretization and the Euler or Runge-Kutta technique for temporal derivative discretization has been proved to be an effective approach (Chamberlain et al., 2021).

B. Q-Wiener Process on Graph Domain

Q-Wiener process is originally defined in a separable Hilbert space $\mathscr{H}$, where both space and time are continuous (Lord et al., 2014). $Q : \mathscr{H} \to \mathscr{H}$ is an trace class non negative, symmetric operator on $\mathscr{H}$. In addition, Q-Wiener process can be expressed as a combination of spatially orthonormal basis and i.i.d. Brownian motions. Typically, the Fourier basis is the most commonly used set of orthonormal basis e.g., $\omega_k(x) = \frac{1}{C_1 C_2} e^{2i\pi(k_1 x_1/C_1 + k_2 x_2/C_2)}$ (Salvi et al., 2022).

Motivated by this observation, we first employ the theory of graph Fourier transform (Shuman et al., 2013) to derive the definition of Q-Wiener process on graph domain, and further prove that it still keeps a valid form of Q-Wiener process. Afterwards, we provide an effective approximation strategy of the discretization sampling from the proposed Q-Wiener process, which largely reduces the original computational complexity.

B.1. Derivation of Q-Wiener Process on Graph Domain

In this work, we extend Q-Wiener process to graph domain. The concrete derivation is here: For a graph, the graph Laplacian $\mathbf{L}$ is a semi-positive definite symmetric matrix with its eigenvalues being non-negative. $\mathbf{U}$ is an orthogonal matrix where each column eigenvector $\mathbf{u}_k$ is mutually orthogonal to others. These eigenvectors constitute a set of orthonormal basis on the graph:

$$\mathbf{L} = \mathbf{U} \begin{pmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & \lambda_{|\mathcal{V}|} \end{pmatrix} \mathbf{U}^\mathsf{T} \tag{19}$$

where $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_{|\mathcal{V}|})$. In analogy to the Karhunen–Loève expansion (Chien & Fu, 1967), Q-Wiener process can be represented as a linear combination of the eigenvectors. Instead of using uncorrelated random variables, we employ i.i.d. Brownian motions $\beta_k(t)$. Let $W(t)$ be a Q-Wiener process on graph domain and suppose that $\lambda_k > 0$ for all k without loss of generality. $W(t)$ can be expressed as the sum of its projection under each orthogonal basis $\mathbf{u}_k$:

$$W(t) = \sum_{k=1}^{|\mathcal{V}|} \langle W(t), \mathbf{u}_k \rangle \, \mathbf{u}_k = \sum_{k=1}^{|\mathcal{V}|} \sqrt{\lambda_k} \mathbf{u}_k \beta_k(t), \tag{20}$$

where $\beta_k(t) := \frac{1}{\sqrt{\lambda_k}} \langle W(t), \mathbf{u}_k \rangle$ and $\{\mathbf{u}_k\}_{k=1}^{|\mathcal{V}|}$ is a set of orthonormal bases. Until now we obtain the form of Eq.(10) in Definition 2, where $\beta_k(0) = 0$ a.s. $\beta_k(t)$ is $\mathcal{F}_t$-adapted and has continuous sample paths. The increment of $\beta_k(t) - \beta_k(s)$ is independent of $\mathcal{F}_s$ and has the form

$$\beta_k(t) - \beta_k(s) = \frac{1}{\sqrt{\lambda_k}} \langle W(t) - W(s), \mathbf{u}_k \rangle, \qquad 0 \le s \le t. \tag{21}$$

As $W(t) - W(s) \sim \mathcal{N}(0, (t-s)\mathbf{L})$, we can get the covariance between the increments of Brownian motions of different components:

$$\begin{aligned}
\mathrm{Cov}(\beta_j(t) - \beta_j(s), \beta_k(t) - \beta_k(s)) &= \mathbb{E}[(\beta_j(t) - \beta_j(s))(\beta_k(t) - \beta_k(s))] - \mathbb{E}[\beta_j(t) - \beta_j(s)]\mathbb{E}[\beta_k(t) - \beta_k(s)] \\
&= \frac{1}{\sqrt{\lambda_j \lambda_k}} \mathbb{E}[\langle W(t) - W(s), \mathbf{u}_j \rangle \langle W(t) - W(s), \mathbf{u}_k \rangle] - \\
&\frac{1}{\sqrt{\lambda_j \lambda_k}} \mathbb{E}[\langle W(t) - W(s), \mathbf{u}_j \rangle] \mathbb{E}[\langle W(t) - W(s), \mathbf{u}_k \rangle] \\
&= \frac{1}{\sqrt{\lambda_j \lambda_k}} \mathbb{E}[\mathbf{u}_j^\mathsf{T} (W(t) - W(s))(W(t) - W(s))^\mathsf{T} \mathbf{u}_k] - \\
&\frac{1}{\sqrt{\lambda_j \lambda_k}} \left(\mathbf{u}_j^\mathsf{T} \mathbb{E}[W(t) - W(s)]\right) \left(\mathbf{u}_k^\mathsf{T} \mathbb{E}[W(t) - W(s)]\right) \\
&= \frac{1}{\sqrt{\lambda_j \lambda_k}} \mathbf{u}_j^\mathsf{T} \mathbb{E}[(W(t) - W(s))(W(t) - W(s))^\mathsf{T}] \mathbf{u}_k \\
&= \frac{1}{\sqrt{\lambda_j \lambda_k}} \mathbf{u}_j^\mathsf{T} (t-s) \mathbf{L} \mathbf{u}_k = (t-s) \delta_{jk}.
\end{aligned} \tag{22}$$

So any pair of increments forms a multivariate Gaussian. Hence, β_j and β_k are independent for $j \neq k$, $\beta_k(t) - \beta_k(s) \sim \mathcal{N}(0, t-s)$ for $j = k$, and $\beta_k(t)$ is a $\mathcal{F}_t$-Brownian motion.

B.2. Validity Proof of Q-Wiener Process on Graph Domain

In this section, we prove that $W(t)$ in eq.(20) is a valid Q-Wiener process according to Definition 1. We first consider the convergence of this series in $L_2(\Omega, \mathscr{G})$ for any fixed $t \geq 0$. In this case, Q-Wiener process can be viewed as the sum of a finite number of elements. By the orthonormality of the eigenvectors and Parseval's identity, we can get

$$\begin{aligned} \| W(t) \|^2 &= \sum_{k=1}^{|\mathcal{V}|} | \langle W(t), \mathbf{u}_k \rangle |^2 = \sum_{k=1}^{|\mathcal{V}|} (\mathbf{u}_k^\mathsf{T} W(t))^2 \\ &= \sum_{k=1}^{|\mathcal{V}|} (\mathbf{u}_k^\mathsf{T} \sum_{j=1}^{|\mathcal{V}|} \sqrt{\lambda_j} \mathbf{u}_j \beta_j(t))^2 \\ &= \sum_{k=1}^{|\mathcal{V}|} (\sqrt{\lambda_k} \beta_k(t))^2 \\ &= \sum_{k=1}^{|\mathcal{V}|} \lambda_k \beta_k(t)^2. \end{aligned} \tag{23}$$

As $\sum_{k=1}^{|\mathcal{V}|} \lambda_k < \infty$ and each $\beta_k(t)$ is a Brownian motion so that $\mathbb{E}[\beta_k(t)^2] = t$. We take the expectation as

$$\mathbb{E}[\| W(t) \|^2] = \sum_{k=1}^{|\mathcal{V}|} \lambda_k \mathbb{E}[\beta_k(t)^2] = t \sum_{k=1}^{|\mathcal{V}|} \lambda_k < \infty. \tag{24}$$

Then the convergence of this series in $L_2(\Omega, \mathscr{G})$ has been proved. Next we prove that the right side of Eq.(20) satisfies the definition of Q-Wiener process. We can obtain the covariance between the different components of the Q-Wiener process:

$$\begin{aligned} \mathrm{Cov}(\langle W(t), \mathbf{u}_j \rangle, \langle W(t), \mathbf{u}_k \rangle) &= \mathbb{E}[\langle W(t), \mathbf{u}_j \rangle \langle W(t), \mathbf{u}_k \rangle] - \mathbb{E}[\langle W(t), \mathbf{u}_j \rangle] \mathbb{E}[\langle W(t), \mathbf{u}_k \rangle] \\ &= \mathbb{E}[\langle W(t), \mathbf{u}_j \rangle \langle W(t), \mathbf{u}_k \rangle] - \mathbf{u}_j^\mathsf{T} \mathbb{E}[W(t)] \mathbf{u}_k^\mathsf{T} \mathbb{E}[W(t)] \\ &= \mathbb{E}[\langle W(t), \mathbf{u}_j \rangle \langle W(t), \mathbf{u}_k \rangle] \\ &= \mathbf{u}_j^\mathsf{T} \mathbb{E}[W(t) W(t)^\mathsf{T}] \mathbf{u}_k \\ &= t \mathbf{u}_j^\mathsf{T} \mathbf{L} \mathbf{u}_k \\ &= t \lambda_j \delta_{jk}. \end{aligned} \tag{25}$$

From this formula we can conclude that $W(t) \sim \mathcal{N}(0, t\mathbf{L})$. This observation naturally extends to the increments, i.e., $W(t) - W(s) \sim \mathcal{N}(0, (t-s)\mathbf{L})$ for all $0 \leq s \leq t$, and we see that $W(t)$ satisfies Definition 1.

B.3. Approximate Discretization Sampling

We first represent the right-hand side of Eq.(20) in a more compact matrix-vector form:

$$W(t) = \mathbf{S}\boldsymbol{\beta}(t), \tag{26}$$

where $\mathbf{S} = \mathbf{U}\sqrt{\mathbf{\Lambda}}$. $\boldsymbol{\beta}(t)$ is a vector containing only i.i.d. Brownian motions:

$$\boldsymbol{\beta}(t) = \begin{pmatrix} \beta_1(t) \\ \vdots \\ \beta_{|\mathcal{V}|}(t) \end{pmatrix} \tag{27}$$

Utilizing the above formula, we can sample a Q-Wiener process by sampling from i.i.d. Brownian motions. But the discretization sampling of this Q-Wiener process requires the eigendecomposition and dense matrix multiplication in the

high-dimensional case, which is computation unaffordable. Here we propose an approximation strategy of the discretization sampling from this Q-Wiener process to alleviate this issue.

Note that $\mathbf{U}$ is the inverse graph Fourier transform, so $W(t) = \mathbf{U}\sqrt{\mathbf{\Lambda}}\boldsymbol{\beta}(t)$ represents scaled Brownian motions followed by $\mathbf{U}$. From this spectral graph view, $W(t)$ is actually from graph domain, whereas $\boldsymbol{\beta}(t)$ is originated in frequency domain. We can regard $\boldsymbol{\beta}(t)$ as a Fourier transformation $\mathbf{U}^\mathsf{T}$ on Brownian motions $\boldsymbol{\beta}'(t)$ coming from graph domain:

$$W(t) = \mathbf{U}\sqrt{\mathbf{\Lambda}}\mathbf{U}^\mathsf{T}\boldsymbol{\beta}'(t), \tag{28}$$

where $\boldsymbol{\beta}'(t)$ are i.i.d. Brownian motions because i.i.d. normal distributions are still i.i.d. normal distributions after orthogonal transformation.

Lemma 1. *Eigenvalues of* $\widehat{\mathbf{L}} \in (0, 2]$. *Eigenvalues of* $\widehat{\mathbf{L}} - \mathbf{I} \in (-1, 1]$.

Following this intuitive interpretation, we then perform a first-order Taylor series on $\sqrt{\mathbf{\Lambda}}$ as:

$$2\sqrt{\mathbf{\Lambda}} = \mathbf{I} + \mathbf{\Lambda} + \mathcal{O}((\mathbf{\Lambda} - \mathbf{I})^2). \tag{29}$$

Due to the following Lemma 1, we can obtain the approximation as $\sqrt{\mathbf{\Lambda}} \approx \mathbf{I} + \mathbf{\Lambda}$ which omits $\mathcal{O}((\mathbf{\Lambda} - \mathbf{I})^2)$ and the constant factor. Integrating this approximated term into Eq.(28) and we get

$$\begin{aligned} W(t) &= \mathbf{U}\sqrt{\mathbf{\Lambda}}\mathbf{U}^\mathsf{T}\boldsymbol{\beta}'(t) \\ &\approx \mathbf{U}\big(\mathbf{I} + \mathbf{\Lambda}\big)\mathbf{U}^\mathsf{T}\boldsymbol{\beta}'(t) \\ &= \big(\mathbf{U}\mathbf{U}^\mathsf{T} + \mathbf{U}\Lambda\mathbf{U}^\mathsf{T}\big)\boldsymbol{\beta}'(t) \\ &= \big(\mathbf{I} + \mathbf{L}\big)\boldsymbol{\beta}'(t) \\ &= \widehat{\mathbf{L}}\boldsymbol{\beta}'(t). \end{aligned} \tag{30}$$

Therefore, we can incorporate the graph Laplacian with added self- connections, i.e., $\widehat{\mathbf{L}}$ into the design of stochastic forcing network for the discretization sampling of Q-Wiener process in Eq.(9) without any complicated calculations. The concrete form is given in Eq.(11).

C. Analysis of Existence and Uniqueness

Theorem 1 is a special case of the existence and uniqueness theorem of a general stochastic differential equation in Thm.10.26 (Lord et al., 2014), where the restriction of the linear operator $\boldsymbol{\psi}$ can be relaxed to that $\boldsymbol{\psi}$ can generate a semigroup (Da Prato & Zabczyk, 2014). Here we prove that our model has the following two properties for satisfying the condition of the existence and uniqueness theorem of the mild solution:

- $\tilde{\mathbf{A}}(\mathbf{H}(t)) - \mathbf{I}$ is linear and can generate a semigroup.
- $\mathbf{g}$ satisfies Lipschitz condition in Assump.10.23 (Lord et al., 2014): For constants $\xi \in (0, 2]$ and $C > 0$, $\mathbf{g}$ satisfies

$$\begin{aligned} ||\boldsymbol{\psi}^{(\xi-1)/2}\mathbf{g}(\mathbf{H}(t))|| &\leq C(1 + ||\mathbf{H}(t)||), \\ ||\boldsymbol{\psi}^{(\xi-1)/2}(\mathbf{g}(\mathbf{H}^{(1)}(t)) - \mathbf{g}(\mathbf{H}^{(2)}(t)))|| &\leq C||\mathbf{H}^{(1)}(t) - \mathbf{H}^{(2)}(t)||, \end{aligned} \tag{31}$$

  where $\mathbf{H}^{(1)}(t)$ and $\mathbf{H}^{(2)}(t)$ are two elements from the space of $\mathbf{H}(t)$.

For the first property, $\tilde{\mathbf{A}}(\mathbf{H}(t))$ is right-stochastic, and $\boldsymbol{\psi} = \tilde{\mathbf{A}}(\mathbf{H}(t)) - \mathbf{I}$ can generate a semigroup $S(t) = e^{t(\tilde{\mathbf{A}}(\mathbf{H}(t))-\mathbf{I})}$ for the drift term $\mathbf{f}(\mathbf{H}(t))$ and satisfies

$$\begin{aligned} S(t+s) &= e^{(t+s)(\tilde{\mathbf{A}}(\mathbf{H}(t))-\mathbf{I})} \\ &= e^{t(\tilde{\mathbf{A}}(\mathbf{H}(t))-\mathbf{I})}e^{s(\tilde{\mathbf{A}}(\mathbf{H}(t))-\mathbf{I})} \\ &= S(t)S(s), \end{aligned} \tag{32}$$

where $S(0) = \mathbf{I}$.

Lemma 2. *For two matrices* $\mathbf{A},\mathbf{B} \in \mathbf{R}^{n\times n}$, $||\mathbf{A}+\mathbf{B}|| \leq ||\mathbf{A}|| + ||\mathbf{B}||$, $||\mathbf{AB}|| \leq ||\mathbf{A}|| \cdot ||\mathbf{B}||$, $||\alpha\mathbf{A}|| = |\alpha|||\mathbf{A}||$.

Lemma 3. *(Gau et al., 2016) Let* $\mathbf{A} \in \mathbf{R}^{n\times n}$ *be a right stochastic matrix, then* $1 \leq ||\mathbf{A}|| \leq \sqrt{n}$.

Lemma 4. *(Kittaneh & Manasrah, 2010) if* $\mathbf{A},\mathbf{B} \in \mathbf{R}^{n\times n}$ *are positive semidefinite matrices and* $0 \leq \alpha \leq 1$, $||\mathbf{A}^{\alpha}\mathbf{B}^{1-\alpha}|| \leq ||\alpha\mathbf{A} + (1-\alpha)\mathbf{B}||$.

For the second property, in the case $\xi \in (1,2]$ we have

$$
\begin{aligned}
||\tilde{\mathbf{A}}(\mathbf{H}(t)) - \mathbf{I})^{(\xi-1)/2}|| &= ||(\tilde{\mathbf{A}}(\mathbf{H}(t)) - \mathbf{I})^{(\xi-1)/2}\mathbf{I}^{(3-\xi)/2}|| && (33)\\
&\leq ||\frac{\xi-1}{2}(\tilde{\mathbf{A}}(\mathbf{H}(t)) - \mathbf{I}) + \frac{3-\xi}{2}\mathbf{I}|| && (34)\\
&\leq \frac{\xi-1}{2}||\tilde{\mathbf{A}}(\mathbf{H}(t)) - \mathbf{I}|| + \frac{3-\xi}{2}||\mathbf{I}|| && (35)\\
&\leq \frac{\xi-1}{2}(||\tilde{\mathbf{A}}(\mathbf{H}(t))|| + ||\mathbf{I}||) + \frac{3-\xi}{2}||\mathbf{I}|| && (36)\\
&= \frac{\xi-1}{2}||\tilde{\mathbf{A}}(\mathbf{H}(t))|| + ||\mathbf{I}|| && (37)\\
&\leq C_1, && (38)
\end{aligned}
$$

where C_1 is a constant. In the above equation, Eq.(34) is derived by Lemma 4. Eq.(35) and Eq.(36) are derived by Lemma 2. Eq.(38) is derived by Lemma 3.

In the another case $\xi \in (0,1]$, the derivation procedure is similar to the case $\xi \in (1,2]$, but the matrix inversion has been involved. In practical applications, if $\tilde{\mathbf{A}}(\mathbf{H}(t)) - \mathbf{I}$ is irreversible, we can use Tikhonov regularization (Hilt & Seegrist, 1977) which adds an appropriately small positive value λ to the diagonal of $\tilde{\mathbf{A}}(\mathbf{H}(t)) - \mathbf{I}$ for making it invertible and keeping its norm within a reasonable range. Combining these two cases together, we see that $||(\tilde{\mathbf{A}}(\mathbf{H}(t)) - \mathbf{I})^{(\xi-1)/2}||$ for $\xi \in (0,2]$ is bounded.

In addition, due to the fact that $\mathbf{g}$ adopts the ReLU activation function, so we have $||\mathbf{g}(\mathbf{H}(t))|| \leq C_2(1 + ||\mathbf{H}(t)||)$ and

$$
\begin{aligned}
||(\tilde{\mathbf{A}}(\mathbf{H}(t)) - \mathbf{I})^{(\xi-1)/2}\mathbf{g}(\mathbf{H}(t))|| &\leq ||(\tilde{\mathbf{A}}(\mathbf{H}(t)) - \mathbf{I})^{(\xi-1)/2}|| \cdot ||\mathbf{g}(\mathbf{H}(t))||\\
&\leq C_1C_2(1 + ||\mathbf{H}(t)||)\\
&\leq C(1 + ||\mathbf{H}(t)||), && (39)
\end{aligned}
$$

where C_1, C_2, C are constants. According to the above equation, we can also get

$$
\begin{aligned}
||(\tilde{\mathbf{A}}(\mathbf{H}(t)) - \mathbf{I})^{(\xi-1)/2}(\mathbf{g}(\mathbf{H}^{(1)}(t)) - \mathbf{g}(\mathbf{H}^{(2)}(t)))|| &\leq ||(\tilde{\mathbf{A}}(\mathbf{H}(t)) - \mathbf{I})^{(\xi-1)/2}|| \cdot ||\mathbf{g}(\mathbf{H}^{(1)}(t)) - \mathbf{g}(\mathbf{H}^{(2)}(t))||\\
&\leq C_1C_2||\mathbf{H}^{(1)}(t) - \mathbf{H}^{(2)}(t)||\\
&\leq C||\mathbf{H}^{(1)}(t) - \mathbf{H}^{(2)}(t)||. && (40)
\end{aligned}
$$

The proof that $\mathbf{g}$ satisfies Lipschitz condition in Assump.10.23 has been completed. In general, our model satisfies the condition of the existence and uniqueness of the mild solution of the graph stochastic diffusion equation.

D. Training Algorithm

The pseudo-code of training algorithm of GNSD is provided in Algorithm 1. In each epoch, the input encoder first embeds the input feature matrix into the representation space for generating initial node embeddings. In the discretization phase, we use the Euler method to update node embeddings iteratively. Through N discretizations, we obtain the final node embeddings $\mathbf{H}(T)$. The output encoder takes $\mathbf{H}(T)$ as input and ouputs $P(\mathbf{y}|\mathbf{H}(T))$ to calculate the distributional uncertainty loss.

E. Experimental Details

E.1. Experimental Environment

The experiments are conducted on a Linux server with Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz, 128G RAM and NVIDIA Tesla V100. The operating system is Ubuntu 18.04. We implement our model and all baselines with the deep

Algorithm 1 The training algorithm of GNSD.

Input: Undirected graph $\mathcal{G}$ with the input feature matrix $\mathbf{X}$.
Output: Learned input encoder ϕ, drift network $\mathbf{f}_\theta$, stochastic forcing network $\mathbf{g}_\theta$ and output decoder $\boldsymbol{\eta}$.

```
Initialize model parameters.
for # training epochs do
    Embed X into the representation space as initial node embeddings, i.e., φ(X) = H(0) = H_0.
    for n = 0 to N − 1 do
        H_{n+1} = H_n + τ(Ã(H_n) − I)H_n + g(H_n)ΔW_τ.
    end for
    Use η to model P(y|H(T)) where H(T) = H_{N−1}.
    Calculate the distributional uncertainty loss of Eq.(12).
    Update model parameters by Adam optimizer.
end for
```

learning library PyTorch (version 1.11) in Python 3.8.

E.2. Dataset Descriptions

- **Amazon-Computers** (McAuley et al., 2015) is a segment an Amazon co-purchase graph, where each node represents a product and an edge exists when these two products are frequently bought together. The feature vector of each node is the bag-of-words representation of product reviews and the product category is the class label. This dataset has 13,752 nodes, 245,861 edges, 767 features and 10 classes.

- **Cora**, **CiteSeer** and **Pubmed** (Sen et al., 2008) are three citation graph datasets. Each graph is directed, where nodes are documents and edges are citation links. When document A cites document B, there exists a directed edge from A to B, or vice-versa. The feature vector of each node is also the bag-of-words representation. Cora contains 2,708 nodes, 5,429 edges, 1,433 features and 7 classes. CiteSeer has 3,327 nodes, 9,104 edges, 3,703 features and 6 classes. Pubmed owns 19,717 nodes, 88,648 edges, 500 features and 3 classes. Following previous works, we treat citation links as undirected edges and use the provided training/validation/testing splits on in-distribution data. The goal is to predict each paper's topic as the class label.

- **OGBN-Arxiv** (Hu et al., 2020) is a large-scale academic graph based on Microsoft academic (Wang et al., 2020a), where each node is a paper with a subject area as the label to predict, and the edge represents the citation relationship. Each node owns a 128-dimensional feature vector obtained by the skip-gram embeddings of its title and abstract. This dataset has 169,343 nodes, 2,315,598 edges, 128 features and 40 classes.

For amazon-computers, the training/validation/testing split is 2:1:1. For three citation graphs and OGBN-Arxiv, we use the public data split as suggested in the original papers (Yang et al., 2016; Hu et al., 2020).

E.3. Baseline Descriptions

- **GCN** (Kipf & Welling, 2017) is a standard GNN framework that introduces a localized first-order approximation of spectral graph convolutions.

- **GAT** (Veličković et al., 2018) is the first work that uses self-attention layers to learn neighbour weights in the process of node feature aggregation.

- **GRAND** (Chamberlain et al., 2021) introduces the discretization of diffusion PDEs on graphs that allows the training of very deep GNNs. We choose the nonlinear version of GRAND where the attention weights are updated at each step of the numerical integration.

- **GREAD** (Choi et al., 2023) is also a GNN model based on reaction-diffusion equation. Different from GRAND, GREAD adds a reaction term to the diffusion equation for solving the over-smoothing problem. GREAD gives multiple forms of reaction terms. We use the proposed Blurring-sharpening term which is designed by the authors.

- **MSP** (Hendrycks & Gimpel, 2016) proposes a simple method that utilizes probabilities from softmax distributions for OOD detection with the observation that OOD examples tend to be lower than the prediction probability for correct examples.
- **ODIN** (Liang et al., 2018) uses the temperature scaling in the softmax function (Hinton et al., 2015) and adds small controlled perturbations to inputs for further enlarging the gap between in-distribution and out-of-distribution data.
- **Mahalanobis** (Lee et al., 2018) derives a generative classifier to obtain a confidence score based on the Mahalanobis distance w.r.t the closest class conditional distribution. Using this generative classifier can effectively detect OOD and adversarial samples.
- **GNNsafe** (Wu et al., 2023) proposes an effective OOD discriminator based on an energy function directly extracted from GNNs trained with standard classification loss. It includes a process of energy propagation to improve model generalization ability. GNNsafe includes two model variants. Since the ideal OOD samples are not available, we choose the first variant without the requirement of OOD samples.
- **GCN-Ensemble** (Lakshminarayanan et al., 2017) combines GCN with ensemble methods. The ensemble of models has 10 different random initializations of model parameters.
- **BGCN** (Zhang et al., 2019) is a representative Bayesian-based graph learning method, which views the observed graph as a realization from a parametric family of stochastic block models (SBMs). It performs a model inference of the joint posterior of SBM parameters and GCN parameters. For the approximate posterior of GCN parameters, BGCN uses Monte Carlo dropout (Gal & Ghahramani, 2016a) to finish parameter sampling.
- **GKDE** (Zhao et al., 2020) provides a graph-based kernel Dirichlet distribution estimation for predicting node-level Dirichlet distributions. The Dirichlet estimation requires the shortest path between nodes for kernel computation, which has the quadratic complexity w.r.t the number of nodes. We use GCN as the backbone in GKDE.
- **GPN** (Stadler et al., 2021) is the SOTA uncertainty estimation method on graphs. It extends posterior networks (Charpentier et al., 2020) to graph domain, which derives the posterior Dirichlet distributions of nodes by using the density estimation model to predict class pseudo-counts. GPN belongs to deterministic uncertainty estimation for quantifying predictive uncertainty and performs the personalized pagerank (Gasteiger et al., 2018) between nodes to model uncertainty propagation on graphs.

For GCN and GAT, we use their implementations in the GNN library PyG (Fey & Lenssen, 2019). For OOD detection baselines, we use the implementation in GNNsafe[3]. GCN-Ensemble is implemented by ourselves. For GRAND[4], GREAD[5], BGCN[6], GKDE[7] and GPN[8], we refer to their public implementations and adapt them to different tasks.

E.4. Evaluation Metrics

To evaluate detection and prediction results, we use the following metrics as suggested by previous works (Kong et al., 2020; Stadler et al., 2021; Wu et al., 2023):

- **AUROC** refers to Area Under the Receiver Operating Characteristic curve, which summarizes the ROC curve into a single value to demonstrate the model performance with different thresholds. It tells how much the model is capable of distinguishing between positive (in-distribution) and negative (out-of-distribution) samples.
- **AUPR** refers to Area Under the Precision-Recall curve. Compared with AUROC, AUPR can adjust different positive/negative class rates to alleviate the problem of imbalanced class, which is also a commonly used metric in OOD detection. For ood detection, AUPR is divided into **AUPR in** (for in-distribution samples) and **AUPR out** (for out-of-distribution samples). For misclassification detection, AUPR is divided into **AUPR succ** (for correctly classified samples) and **AUPR err** (for wrongly classified samples).

[3]https://github.com/qitianwu/GraphOOD-GNNSafe
[4]https://github.com/twitter-research/graph-neural-pde
[5]https://github.com/jeongwhanchoi/GREAD
[6]https://github.com/huawei-noah/BGCN
[7]https://github.com/zxj32/uncertainty-GNN
[8]https://github.com/stadlmax/Graph-Posterior-Network

Table 4. OOD detection performance comparison on Cora with different OOD constructions.

Model	Label leave-out						Feature perturbation					
	AUROC	AUPR in	AUPR out	FPR95	DET ACC	ID ACC	AUROC	AUPR in	AUPR out	FPR95	DET ACC	ID ACC
GCN	88.67	74.52	95.87	52.40	78.80	89.25	75.40	54.22	86.69	80.98	69.94	79.20
GAT	90.81	77.97	96.50	44.91	83.65	91.19	89.39	80.19	92.80	70.37	78.16	76.10
GRAND	88.58	75.72	94.28	63.59	82.34	90.19	87.04	74.81	93.42	55.35	76.43	84.40
GREAD	88.40	73.80	95.01	57.35	80.74	90.63	86.49	70.59	93.01	57.10	75.30	83.00
MSP	91.58	80.59	96.72	44.03	84.51	90.51	90.58	82.31	92.02	50.33	80.45	80.50
ODIN	49.24	24.27	75.45	100.00	49.95	88.92	49.80	26.92	72.95	100.00	49.98	74.80
Mahalanobis	66.83	36.69	85.92	82.96	59.62	86.71	60.46	40.65	74.52	99.59	58.52	75.30
GNNsafe	92.68	82.03	97.48	31.54	83.48	88.92	93.28	88.16	96.35	43.43	83.71	76.20
GCN-Ensemble	90.97	80.35	97.37	29.92	85.73	91.24	89.01	79.71	94.57	61.74	74.57	81.60
BGCN	91.16	79.41	96.60	46.30	84.40	90.51	84.82	75.40	91.20	75.20	78.09	77.50
GKDE	82.80	65.12	92.62	70.59	71.64	87.66	80.22	63.05	90.01	75.01	64.74	70.45
GPN	90.10	79.98	96.13	50.71	82.81	91.46	91.89	82.62	96.01	42.32	81.56	80.20
GNSD	**94.76**	**88.45**	**97.73**	**27.38**	**89.74**	**91.77**	**94.41**	**88.23**	**97.22**	**26.27**	**86.92**	**87.97**

Table 5. OOD detection performance comparison on CiteSeer with different OOD constructions.

Model	Label leave-out						Feature perturbation					
	AUROC	AUPR in	AUPR out	FPR95	DET ACC	ID ACC	AUROC	AUPR in	AUPR out	FPR95	DET ACC	ID ACC
GCN	81.28	70.51	86.30	63.23	74.64	84.36	82.84	60.78	93.61	65.70	74.55	65.81
GAT	80.31	70.00	84.72	65.44	75.07	85.37	81.03	58.68	93.20	61.74	72.72	66.60
GRAND	78.04	69.08	81.50	83.49	72.17	86.07	82.60	63.18	92.15	73.91	74.06	66.80
GREAD	79.20	69.05	83.01	76.57	73.40	87.20	82.78	59.38	93.22	62.79	74.45	66.35
MSP	76.10	68.35	75.49	74.50	75.48	83.57	83.06	61.34	93.67	61.62	75.14	64.70
ODIN	50.15	39.82	60.47	100.00	50.15	85.36	49.61	23.07	76.75	100.00	49.98	64.40
Mahalanobis	56.34	44.17	67.15	90.05	51.56	75.71	50.65	29.79	72.86	99.70	54.56	60.30
GNNsafe	82.14	70.26	84.63	62.76	68.49	85.36	84.31	68.30	92.38	65.92	73.40	66.30
GCN-Ensemble	80.70	68.74	86.47	64.51	73.82	84.79	84.82	64.23	93.92	62.12	74.00	**67.90**
BGCN	79.09	67.07	87.05	62.18	74.19	84.12	84.50	62.37	94.17	61.71	64.98	66.93
GKDE	64.13	56.31	69.92	92.15	60.23	72.38	77.90	51.85	91.51	71.06	71.41	66.30
GPN	76.15	67.51	80.53	78.57	70.76	84.64	82.48	57.01	93.95	61.85	74.14	62.30
GNSD	**82.95**	**75.28**	**87.44**	**60.19**	**75.66**	**87.68**	**86.82**	**68.58**	**94.46**	**60.26**	**79.80**	67.10

- **FPR95** is False Positive Rate at 95% true positive rate. It represents the probability that a out-of-distribution sample is misclassified as the in-distribution one when the true positive rate is 95%, so a smaller value of FPR95 reflects better detection performance.
- **DET ACC** is the ratio between the test samples that are correctly detected and all test samples at the optimal threshold.
- **ID ACC** is in-distribution classification accuracy which is the standard metric used in semi-supervised node classification. We use it to evaluate the model performance on in-distribution test nodes.

Table 6. Search intervals of hyper-parameters.

Hyper-parameter	Search Space
learning rate	{0.001, 0.005, 0.01, 0.1}
weight decay	{0.01, 0.001, 0.0005, 0.0001}
training sample times	{1, 3, 5, 7}
dropout	{0.0, 0.1, 0.3, 0.5}
input dropout	{0.0, 0.1, 0.3, 0.5}
step size	{0.01, 0.05, 0.1, 0.2}

E.5. More Empirical Results

Tables 4, 5, 7, 8 demonstrate the OOD detection performances on Cora, CiteSeer, Pubmed and OGBN-Arxiv. Each dataset contains two OOD settings: label leave-out and feature perturbation. From these results, we can conclude that GNSD still keeps the best OOD detection results on these datasets compared with current baselines.

Table 7. OOD detection performance comparison on Pubmed with different OOD constructions.

Model	Label leave-out						Feature perturbation					
	AUROC	AUPR in	AUPR out	FPR95	DET ACC	ID ACC	AUROC	AUPR in	AUPR out	FPR95	DET ACC	ID ACC
GCN	73.58	32.39	93.33	70.12	67.20	78.70	87.71	42.89	99.07	53.44	78.37	74.90
GAT	74.42	30.71	89.50	77.70	68.73	80.14	85.42	32.98	99.00	53.13	77.62	75.60
GRAND	74.64	30.49	93.48	72.58	71.08	80.82	80.13	31.44	98.45	74.35	71.82	**77.10**
GREAD	75.60	31.10	93.45	70.93	69.25	80.84	85.65	31.26	99.05	54.90	80.16	76.20
MSP	70.96	30.78	92.29	75.90	65.17	78.66	87.81	45.26	99.18	53.90	79.35	75.00
ODIN	48.55	15.62	82.20	96.42	49.95	79.49	49.52	4.82	95.13	100.00	49.99	74.40
Mahalanobis	43.03	14.50	79.66	97.90	50.01	79.02	68.61	15.83	97.18	89.82	56.43	69.80
GNNsafe	76.72	**39.57**	94.07	69.73	69.50	80.12	94.44	71.80	99.59	30.09	85.97	75.40
GCN-Ensemble	73.79	32.71	93.28	69.97	67.72	79.27	82.94	37.59	98.66	71.77	75.58	73.40
BGCN	74.48	35.11	93.32	71.27	69.14	78.04	87.66	43.53	99.14	52.22	79.30	75.20
GKDE	57.99	20.91	86.41	93.54	60.73	76.87	78.05	29.63	97.84	89.28	72.97	54.30
GPN	75.18	32.67	93.50	71.12	69.32	80.23	88.62	46.90	99.08	53.82	79.39	75.10
GNSD	**78.81**	38.70	**94.95**	**59.49**	**73.39**	**81.23**	**97.09**	**73.55**	**99.81**	**12.21**	**92.39**	75.40

Table 8. OOD detection performance comparison on OGBN-Arxiv with different OOD constructions.

Model	Label leave-out						Feature perturbation					
	AUROC	AUPR in	AUPR out	FPR95	DET ACC	ID ACC	AUROC	AUPR in	AUPR out	FPR95	DET ACC	ID ACC
GCN	66.71	39.04	83.55	84.82	50.00	81.26	91.46	78.87	96.98	41.25	64.65	66.99
GAT	73.71	46.39	88.12	73.80	68.06	79.64	97.65	90.66	99.32	8.09	66.65	62.80
GRAND	70.09	40.13	86.97	72.51	65.71	81.09	96.05	87.62	98.84	17.82	77.77	69.47
GREAD	71.10	41.60	87.93	73.91	55.69	81.30	96.70	88.90	99.02	14.57	85.90	69.33
MSP	69.19	43.72	84.62	83.17	64.11	80.93	93.62	82.55	97.87	27.61	86.68	65.13
ODIN	52.06	27.00	76.05	88.31	52.41	80.46	63.77	33.13	84.38	90.75	57.16	67.53
Mahalanobis	40.19	21.65	69.43	94.19	50.00	80.55	64.03	36.84	85.46	83.13	50.01	66.23
GNNsafe	69.73	39.54	86.91	72.43	64.52	80.46	95.86	88.09	98.70	19.22	81.80	67.42
GCN-Ensemble	76.93	53.54	89.35	70.22	68.19	80.36	96.27	90.17	98.81	18.14	78.29	68.35
BGCN	-	-	-	-	-	-	-	-	-	-	-	-
GKDE	67.65	37.08	85.93	73.21	63.89	80.11	95.42	82.56	98.72	16.88	88.60	67.99
GPN	76.46	48.86	89.73	**69.00**	55.57	77.74	98.42	93.62	99.57	6.22	88.97	67.99
GNSD	**78.82**	**54.14**	**90.11**	71.64	**70.25**	**81.51**	**99.55**	**98.25**	**99.87**	**1.78**	**97.17**	**69.68**

Table 9. Misclassification detection performance comparison on Pubmed and OGBN-Arxiv.

Model	Pubmed				OGBN-Arxiv			
	AUROC	AUPR succ	AUPR err	FPR95	AUROC	AUPR succ	AUPR err	FPR95
GCN	67.45	86.87	43.31	88.14	75.31	86.60	55.98	80.14
GAT	71.00	87.30	38.27	84.62	76.11	86.86	57.04	79.60
GRAND	71.16	77.15	52.75	86.55	76.28	87.43	55.35	80.00
GREAD	70.55	78.66	46.90	86.17	76.15	88.03	55.95	78.62
GCN-Ensemble	67.27	83.27	40.22	88.72	76.54	88.50	54.69	79.03
BGCN	69.05	84.57	44.64	86.47	-	-	-	-
GKDE	68.15	76.79	54.06	85.75	76.68	88.43	55.35	78.40
GPN	72.00	87.44	42.94	85.19	75.12	86.26	56.12	80.79
GNSD	**75.85**	**87.81**	**54.26**	**84.01**	**76.75**	**88.60**	**57.76**	**78.04**

Table 9 shows the results of misclassification detection on pubmed and OGBN-Arxiv. The results align with the above performances of misclassification detection where our model achieves better detection results compared with multiple GNN and uncertainty estimation baselines. In addition, BGCN does not finish the experiments on OGBN-Arxiv, since BGCN requires a posterior inference of SBM and GCN with high model complexity. As such, it cannot scale well to large-scale graphs.

E.6. Hyper-parameter Setting

We fix the hidden dimensional size, the number of training epochs and time T as 64, 200 and 1.0. For other hyper-parameters, we use grid search to find their optimal values within the hyper-parameter search space shown in Table 6. The best

hyper-parameter configurations of each datasets are list here: $\{0.01, 0.0005, 1, 0.0, 0.3, 0.1\}$ for Amazon-Computers, $\{0.01, 0.01, 5, 0.0, 0.0, 0.1\}$ for Cora, $\{0.01, 0.0005, 3, 0.0, 0.0, 0.1\}$ for CiteSeer, $\{0.1, 0.0005, 3, 0.0, 0.0, 0.1\}$ for Pubmed and $\{0.01, 0.01, 1, 0.0, 0.0, 0.1\}$ for OGBN-Arxiv. To make fair comparisons, all baselines and our model have the same hidden dimensional size and the number of training epochs. The number of hidden layers used in all baselines are also fixed. The number of hidden layers in GCN-based encoders is 2, the number of hidden layers in GAT-based encoders is 2 with 8 attention heads.