Uncertainty Quantification on Graph Learning: A Survey

CHAO CHEN*, School of Computer Science, Harbin Institute of Technology (Shenzhen), China

CHENGHUA GUO*, Key Laboratory of Trustworthy Distributed Computing and Service (MoE), Beijing University of Posts and Telecommunications, Beijing, China

RUI XU*, Artificial Intelligence Thrust, The Hong Kong University of Science and Technology (Guangzhou), China

XIANGWEN LIAO, College of Computer and Data Science, Fuzhou University, China

XI ZHANG, Key Laboratory of Trustworthy Distributed Computing and Service (MoE), Beijing University of Posts and Telecommunications, Beijing, China

SIHONG XIE†, Artificial Intelligence Thrust, The Hong Kong University of Science and Technology (Guangzhou), China

HUI XIONG, Artificial Intelligence Thrust, The Hong Kong University of Science and Technology (Guangzhou), China

PHILIP YU, University of Illinois at Chicago, U.S.

Graphical models, including Graph Neural Networks (GNNs) and Probabilistic Graphical Models (PGMs), have demonstrated their exceptional capabilities across numerous fields. These models necessitate effective uncertainty quantification to ensure reliable decision-making amid the challenges posed by model training discrepancies and unpredictable testing scenarios. This survey examines recent works that address uncertainty quantification within the model architectures, training, and inference of GNNs and PGMs. We aim to provide an overview of the current landscape of uncertainty in graphical models by organizing the recent methods into uncertainty representation and handling. By summarizing state-of-the-art methods, this survey seeks to deepen the understanding of uncertainty quantification in graphical models, thereby increasing their effectiveness and safety in critical applications.

CCS Concepts: • **Do Not Use This Code → Generate the Correct Terms for Your Paper**; *Generate the Correct Terms for Your Paper*; Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper.

Additional Key Words and Phrases: Do, Not, Us, This, Code, Put, the, Correct, Terms, for, Your, Paper

ACM Reference Format:

Chao Chen, Chenghua Guo, Rui Xu, Xiangwen Liao, Xi Zhang, Sihong Xie, Hui Xiong, and Philip Yu. 2018. Uncertainty Quantification on Graph Learning: A Survey. In . ACM, New York, NY, USA, 30 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Graphical models have emerged as fundamental tools in machine learning, and are employed across various domains, such as spam detection [94] and recommendation systems [118], due to their capability to manage the complex relational

*Three authors are listed in alphabetical order with equal contributions to this work. Specifically, Chao primarily focused on Sec. 2, Sec. 5.1 and Sec. 5.2; Chenghua primarily worked on Sec. 4; and Rui primarily handled Sec. 3, Sec. 5.2, Sec. 5.3, and Sec. 6.

†Corresponding author: sihongxie@hkust-gz.edu.cn

Manuscript submitted to ACM

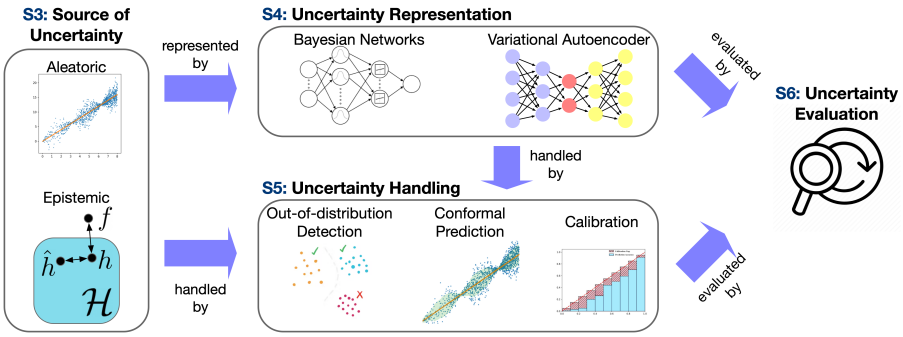

Fig. 1. Overview of the survey: the survey introduces the source of uncertainty in Sec. 3. Then, the survey will demonstrate the state-of-the-art methods used for uncertainty representation and handling in Sec. 4 and Sec. 5, respectively. Besides, the evaluation metrics used to assess the uncertainty will be discussed in Sec. 6.

dynamics among variables. Unlike traditional models that assume variables to be independent and identically distributed (I.I.D.), graphical models reveal intricate relationships and conditional dependencies, thereby enhancing predictive performance by considering both individual variables and their interconnections.

Handling uncertainty in graphical models is critical, as these models often encounter uncertainties stemming from inherent data randomness, model training errors, and unforeseen test data distributions [114]. Effective uncertainty quantification is especially essential for robust decision-making, especially in critical fields such as healthcare [97], autonomous driving, and [104], where precise risk assessment and decisions under uncertainty are paramount. By modeling and quantifying uncertainty, these systems provide confidence measures and explanations alongside predictions, enhancing their reliability and safety.

This survey explores uncertainties in two primary types of graphical models: Graph Neural Networks (GNNs) and Probabilistic Graphical Models (PGMs). GNNs extend the successes of deep learning to graph-structured data but face challenges such as noisy links, missing nodes, or mislabeled data. Recent developments have incorporated uncertainty quantification into the architecture and training of GNNs, improving their robustness and interpretability. PGMs, including Bayesian Networks and Markov Random Fields, utilize probability theory to effectively handle uncertainty and are particularly effective in scenarios that require joint probability modeling and causal relationship analysis.

Existing surveys on UQ can be categorized along two dimensions: the application scope, such as graph learning, and the scope of coverage, ranging from general UQ topics to specific UQ aspects. The survey most closely related to the present survey is [114], which also focuses on graph learning and covers general UQ topics. Similar to our survey, [114] investigates the sources of uncertainty, evaluation methods, and various techniques for handling uncertainty in graph learning. However, a key distinction between [114] and our survey is that the former does not include a discussion on the representation methods for uncertainty. In more general application scopes, there are surveys that cover general UQ topics. For example, [1, 32] explore uncertainty in deep learning, while [92] examines uncertainty in scientific machine learning. These surveys provide valuable insights into the broader landscape of UQ across different application domains. However, they do not delve deep into the uncertainty specifically in graph learning. Furthermore, several surveys offer an in-depth analysis of specific aspects of UQ, but not in the context of graph learning. These include surveys on sources of uncertainty [36, 53], optimization under uncertainty [85], and noise in machine learning [39].

Despite the valuable contributions of above works, they collectively lack a comprehensive overview of uncertainty quantification on graph learning.

This survey aims to review various strategies and methodologies to model, measure, and mitigate uncertainty in GNNs and PGMs. By examining state-of-the-art research, the survey provides a comprehensive understanding of how current models address uncertainty and forecast future enhancements to increase their efficacy. Specifically, this survey organizes the literature on uncertainty in graphs into four major parts, as demonstrated in Fig. 1. Specifically, we first provide the preliminary knowledge concerning PGMs and GNNs in Sec. 2, and introduce the source of uncertainty in Sec. 3. Then we discuss the methods used for uncertainty representation and handling in Sec. 4 and 5, respectively. Besides, we will demonstrate widely used metrics for uncertainty evaluation in Sec. 6. Finally, the conclusion of the survey will be made in Sec. 7.

2 PRELIMINARIES

In this section, we briefly introduce two key families of models for graphs, the Probabilistic Graphical Model and the Graph Neural Network.

2.1 Probabilistic Graphical Model

A probabilistic graphical model (PGM) [9, 55] is widely used to represent the probability distribution of random variables, whose relations can be depicted in a graph. PGMs aim to minimize the cost of establishing compatible dependency relationships among all the variables. There are two distinct categories of PGMs, depending on whether the graph's edges are directed or undirected. Specifically, a Bayesian Network is employed for modeling a directed graph, where edges represent causality. Conversely, a Markov Random Field (MRF) is utilized to model an undirected graph, where edges signify the correlation between nodes. In an MRF, each node is conditionally independent of all other nodes, except for its immediate neighbors.

Formally, a graph $G = (V, E)$ consists of a set of n nodes representing $V = \{X_1, \ldots, X_n\}$ random variables, and a set E of edges each of which is a relationship between two variables. Each variable $X_i \in V$ takes values from the discrete sample space $\{1, \ldots, K\}$. An MRF factorizes the joint distribution $P(V)$ as

$$P(V) = \frac{1}{Z} \prod_{X_i \in V} \phi(X_i = x_i) \prod_{X_j \in \mathcal{N}(X_i)} \psi(X_i = x_i, X_j = x_j), \tag{1}$$

where Z normalizes the product to a probability distribution. $\mathcal{N}(X_i)$ represents the neighbors of X_i within the graph G. $\phi(X_i = x_i)$ denotes the prior probability of X_i taking value x_i, independent of other variables. The compatibility $\psi(X_i = x_i, X_j = x_j)$ is the preference of the pair (X_i, X_j) jointly taking the value (x_i, x_j), and capture the dependencies between the two variables. The marginal distribution $b(X_i)$, also known as belief, for any $X_i \in V$ is given by

$$b(X_i = x_i) = \sum_{X_1} \cdots \sum_{X_{i-1}} \sum_{X_{i+1}} \cdots \sum_{X_n} P(X_1, X_2, \ldots, X_i = x_i, \ldots, X_n). \tag{2}$$

Computing $b(X_i = x_i)$ for any node $X_i \in V$ by Eq. (2) has exponential complexity in general and the Belief Propagation (BP) [9] algorithm is a dynamic programming algorithm that infers the beliefs much more efficiently. First, the message $m_{i \to j}(X_j = x_j)$ from X_i to X_j is defined by

$$\frac{1}{Z_j} \sum_{X_i = x_i} \left[\psi(X_i = x_i, X_j = x_j) \phi(X_i = x_i) \prod_{k \in \mathcal{N}(X_i) \setminus \{X_j\}} m_{k \to i}(X_i = x_i) \right], \tag{3}$$

where Z_j is a normalization factor so that $m_{i\to j}$ is a probability distribution of X_j. The messages in both directions on all edges $(X_i, X_j) \in E$ will be updated iteratively. As the message converges (guaranteed when G is acyclic [90]), belief $b(X_i = x_i)$ is given by

$$b(X_i = x_i) \propto \phi(X_i = x_i) \prod_{X_j \in \mathcal{N}(X_i)} m_{j\to i}(X_i = x_i). \tag{4}$$

For simplicity, we will omit X_i in $\phi(X_i = x_i)$ and use $\phi(x_i)$, and similarly for $\psi(x_i, x_j)$, $m_{j\to i}(x_i)$ and $b(x_i)$ if there is no ambiguity.

2.2 Graph Neural Networks

Graph Neural Network (GNN) is a deep learning technique designed for graphs. Unlike traditional neural networks that operate on grid-like data (such as images or sequences), GNN could learn from complex relationships and dependencies in graphs.

Formally, given a graph $G = (V, E)$, V is the set of nodes and E is the set of edges connecting the nodes. Let $\mathcal{N}(X_i)$ be the set of neighbors of node $X_i \in V$, $i = 1, \ldots, |V|$. Assume that a GNN of L layers is trained to predict class distributions of the nodes on the graph G. On the l-th layer, $l = 1, \ldots, L$, GNN calculates $\mathbf{h}_i^{(l)}$ for node X_i considering messages sent from its neighborhood $\mathcal{N}(X_i)$. Formally,

$$\mathbf{a}_i^{(l)} = \text{AGG}\left(\left\{\mathbf{m}_{ji}^{(l)} | X_j \in \mathcal{N}(X_i)\right\}\right), \tag{5}$$

$$\mathbf{h}_i^{(l)} = \text{UPDATE}\left(\mathbf{a}_i^{(l)}, \mathbf{h}_i^{(l-1)}\right). \tag{6}$$

The AGG function aggregates the messages sent from all $X_j \in \mathcal{N}(X_i)$ to X_i, which can be the element-wise average, maximum of the messages, or a parametric layer [121]. The UPDATE function combines representations from the $l-1$ and the current l layers. For example, it is defined as $W \cdot \left[\mathbf{h}_i^{(l-1)}, \mathbf{a}_i^{(l)}\right]$ with trainable matrix W in GraphSAGE [42]. $\mathbf{h}_i^{(0)}$ is the input node feature for node X_i, and the output of the GNN is $\mathbf{h}_i^{(L)}$, which can be softmaxed to a class distribution $\pi(X_i)$. The parameters of GNN, $\boldsymbol{\theta} = \{\theta^{(l)}, l = 1, \ldots, L\}$, are trained in an end-to-end style using labeled nodes on G.

3 SOURCES OF UNCERTAINTY

The total uncertainty in machine learning can be decomposed into **aleatoric uncertainty** and **epistemic uncertainty** based on their sources [21]. Aleatoric uncertainty, which is irreducible with increasing amounts of data, represents the randomness inherent to the data generation process [40]. However, epistemic uncertainty can be reduced by obtaining more knowledge or data of events to better estimate the ground truth [53, 57]. The techniques for the disentanglement of aleatoric and epistemic uncertainty are introduced by [107]. [79] formulated their mathematical expression in the context of GNNs.

The perfect separation between aleatoric uncertainty and epistemic uncertainty is almost impossible [47]. For example, flipping a coin is usually regarded as a purely random event, involving aleatoric uncertainty. However, if we know the whole physics process of flipping a coin, including the flipping force, the coin's mass, volume, initial position, etc, we can simulate the process and predict whether it would be head or tail. Based on the simulation, randomness will be reduced or even eliminated. Therefore, the identification of aleatoric and epistemic uncertainty must be built upon the context of the selected model and available data.

Under the problem setup of supervised machine learning, we further explore the source of different kinds of uncertainty and how they impact prediction reliability. Let $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ denote input feature variables and the

output label variable, which can take values x and y. The underlying ground truth data probability model is F and the data generation mechanism is f.

3.1 Aleatoric Uncertainty

Aleatoric uncertainty is only about data noises, so we do not need to consider the influence of model selection and training issues and can investigate this problem under the view of statistics. When f is conditioned on $X = x$, aleatoric uncertainty is the conditional probability distribution of Y, and its variance, $Var(Y|x)$, can be applied for quantification [36].

$$Y|x \sim F_{Y|X}(\cdot|X = x) \tag{7}$$

Four sources of aleatoric uncertainty are briefly introduced below. We refer the readers to [36] for a more comprehensive and detailed taxonomy.

Omitted Features: Omitted features, which are investigated in the context of causal models [16], management research [12], and model sensitivity analysis [17], are significant sources of aleatoric uncertainty. The omission may be due to the unobservability of data or overlooking during data collection. Let $Z \in \mathcal{Z}$ denote an unobserved variable that influences the value of Y, we can see the expectation of $Y|x$ is a weighted integration of $E_{Y|X,Z}$ on Z.

$$E_{Y|X}(Y|x) = \int_{y\in\mathcal{Y}} \int_{z\in\mathcal{Z}} yF_{Y|X,Z}(y|x,z)F_{Z|X}(z|x)dzdy = \int_{z\in\mathcal{Z}} E_{Y|X,Z}(Y|x,z)F_{Z|X}(z|x)dz. \tag{8}$$

[36] also claims $Var_{Y|X,Z}(Y|x,z) \leq Var_{Y|X}(Y|x)$ with at least one $z \in \mathcal{Z}$, indicating including the non-negligible variables will reduce aleatoric uncertainty.

Feature Errors: We will likely lose information with improper measurements, like low-resolution cameras, low-quality audio recorders, and uncalibrated gauges. These errors can cause biases and increased variance in feature collection. [39] reviewed 79 works on identifying and handling data noise. Here we let Z represent feature variables by faultless measurements and X denote those by inaccurate measurements, assuming Y is independent of $X|Z$. The change of aleatoric uncertainty caused by noisy feature values is illustrated by [36] below. The first term of the right-hand side is $Var_{Y|Z}(Y|Z)$ conditioned on $X = x$, and the second term is an increment.

$$Var_{Y|X}(Y|x) = E_{Z|X}(Var_{Y|Z}(Y|Z)|x) + Var_{Z|X}(E_{Y|Z}(Y|Z)|x). \tag{9}$$

Label Errors: Label errors can be introduced by human labeling with subjectivity. Label errors in training sets will deviate models from correct gradient descent directions, and those in test sets will improperly evaluate models' performance. [86] states that at least 3.3% error rate across the most commonly-used computer vision, natural language, and audio datasets. If Z is the correct label variable and Y is the error-prone one, Eq. (7) can be rewritten below.

$$Y|x \sim F_{Y|X}(\cdot|x) = \int_{z\in\mathcal{Z}} F_{Y|X,Z}(\cdot|x,z)F_{Z|X}(z|x)dz. \tag{10}$$

Missing Data: Missing data indicates some entries in the dataset are incomplete [70]. If (x_i, y_i) is the i-th sample of the dataset, M_i is its corresponding missing indicator. $M_i = 0$ means the sample is complete, and $M_i = 1$ otherwise. A bias factor is explained by [36] as $\frac{P(M=0|y,x)}{P(M=0|x)}$ to express the influence from incompleteness.

$$F_{Y|X,M}(y|x, M = 0) = \frac{P(M = 0|y, x)}{P(M = 0|x)} F_{Y|X}(y|x). \tag{11}$$

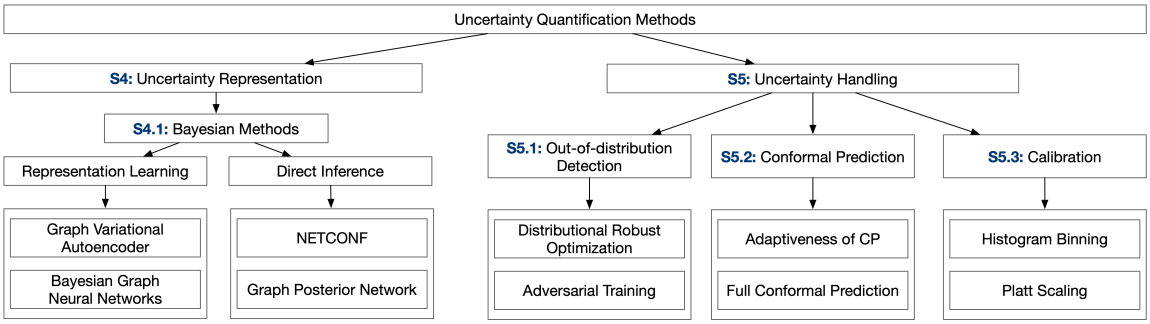

Fig. 2. Taxomony of methods for Uncertainty Quantification. Bayesian methods, including Bayesian representation learning and direct inference methods, are introduced in Sec. 4.1. The state-of-the-art works handling uncertainty are discussed in Sec. 5. Specifically, existing works related to out-of-distribution detection, conformal prediction, and calibration are discussed in Sec. 5.1, Sec. 5.2, and Sec. 5.3, respectively. The large boxes in the bottom row demonstrate significant examples for each method.

3.2 Epistemic Uncertainty

Epistemic uncertainty is introduced by model selection and training, rooted in the lack of data and knowledge. [53] clearly decomposed epistemic uncertainty into model and approximation uncertainty. Denote $f \in \mathcal{F}$ is the ground truth mapping relationship from X to Y, $h \in \mathcal{H} \subset \mathcal{F}$ is the best possible predictor we can get, and $\hat{h} \in \mathcal{H}$ is the induced predictor in practice.

Model Uncertainty measures the difference between f and h, because we can only explore a limited function space $\mathcal{H}$, which may not include the true f. Therefore, there can be a distance between the optimally trained predictor and the real mapping function. Taking l as the loss function during training, [53] explicitly defines f and h as follows.

$$f = \arg\min_{f \in \mathcal{F}} \int_{\mathcal{X}} \int_{\mathcal{Y}} l(y, f(x)) F_{X,Y}(x, y) dy dx. \tag{12}$$

$$h = \arg\min_{h \in \mathcal{H}} \int_{\mathcal{X}} \int_{\mathcal{Y}} l(y, h(x)) F_{X,Y}(x, y) dy dx. \tag{13}$$

Approximation Uncertainty counts for the uncertainty due to the limited number of training samples. Assuming we have N samples available , the definition of $\hat{h}$ is listed as

$$\hat{h} = \arg\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} l(y_i, h(x_i)). \tag{14}$$

It is obvious that, unlike aleatoric uncertainty, model uncertainty and approximation uncertainty are reducible by enlarging the function space H and increasing sample size N. We recommend [52, 64, 83] for the latest progress in predicting, sampling, and quantifying epistemic uncertainty.

4 METHODS FOR UNCERTAINTY REPRESENTATION

4.1 Bayesian Methods

In the realm of UQ, Bayesian methods stand as a pivotal approach, offering a powerful framework for representing uncertainty. At the heart of Bayesian UQ is the application of Bayes' Theorem, which mathematically expresses how our belief or knowledge about a certain parameter or model is updated with evidence. The theorem is elegantly encapsulated

in the formula:

$$P(\boldsymbol{\theta}|D) = \frac{P(D|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(D)}, \tag{15}$$

where $P(\boldsymbol{\theta}|D)$ is the posterior probability of the parameter $\boldsymbol{\theta}$ given the data $D = \{X, Y\}$, $P(D|\boldsymbol{\theta})$ is the likelihood of observing the data given the parameter, $P(\boldsymbol{\theta})$ is the prior probability of the parameter, and $P(D)$ is the marginal likelihood of the data. This Bayesian paradigm shifts the focus from a single 'true' model to a probabilistic distribution of models, reflecting all possible values that the parameters can take, weighted by their likelihood. By incorporating prior knowledge and continuously updating it with new data, Bayesian methods provide a dynamic and adaptive approach to UQ, allowing for a more comprehensive understanding of uncertainty in models.

Advancing from the foundational Bayesian principles, two categories of uncertainty representation methods have emerged. The first category directly applies Bayesian principles to specific tasks, inferring posterior probability distributions to represent posterior uncertainty. In this survey, we refer to these methods as *direct inference*. The second category is *Bayesian representation learning*, where methods use the Bayesian approach to discover useful representations that inherently encode the uncertainty associated with them. In this context, the learned representations are characterized not only by their ability to capture the underlying patterns in the data but also by their probabilistic nature which quantifies the uncertainty in these representations.

4.1.1 *Direct Inference for Graphs.*

In the task of semi-supervised node classification on graphs, one approach that directly applies Bayesian methods to infer the posterior probability distribution of nodes is as follows: First, a prior probability distribution is assigned to each node, which is then updated with evidence propagated from other nodes to obtain the posterior distribution. Yamaguchi et al. [124] assumed the node label is a categorical random variable as $P(\hat{y}_i = k|\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the parameter of the categorical distribution. Based on the smoothness hypothesis, they believed that a neighbor of node i shares the same parameter $\boldsymbol{\theta}$ as i. This leads to the multinomial likelihood function of labels of neighbors of i $P(\hat{N}_i|\boldsymbol{\theta}) \propto \Pi_{k=1}^{K} \theta_k^{n_{ik}}$ and the conjugate Dirichlet prior $P(\boldsymbol{\theta}) \propto \Pi_{k=1}^{K} \theta_k^{\alpha_k - 1}$, where n_{ik} is the number of i's neighbors whose label is k, and $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_K)^T$ is the parameter of Dirichlet distribution. Combining these, the posterior distribution of $\boldsymbol{\theta}$ is $P(\boldsymbol{\theta}|\hat{N}_i) \propto \Pi_{k=1}^{K} \theta_k^{\alpha_k + n_{ik} - 1}$. Beyond the smoothness hypothesis and label propagation, Eswaran et al. [26] used compatibility matrices to represent the strength of connections between nodes and propagates multinomial messages to support both homophily and heterophily network effects. To handle high-dimensional features of nodes, Stadler et al. [100] employed MLP encoding and computes node-level pseudo-counts, then propagates them via PPR-based message passing. All three methods utilize Dirichlet priors and posteriors, with the main difference being in the form of multinomial distribution evidence propagated.

4.1.2 *Bayesian Representation Learning Overview.*

Representation learning constitutes a foundational component in the domain of machine learning, where the primary aim is to transform raw data into a format that is more amenable for algorithms to process and extract meaningful patterns. This transformation is crucial because it captures the underlying structures and features inherent in the data, which are often not immediately apparent. The essence of representation learning is to discover such representations that facilitate improved performance on various tasks such as classification, regression, or even generation of new data instances. Shifting the focus towards Bayesian representation learning, this approach integrates the principles of Bayesian inference with the process of learning data representations. Unlike traditional deterministic methods, Bayesian representation learning seeks to model uncertainty in aspects such as model parameters and the learned representations

by treating them as random variables with associated probability distributions, rather than as fixed points. In this survey, we focus on two prominent classes of Bayesian representation learning methods: variational autoencoders (VAEs) and Bayesian neural networks (BNNs). VAEs are generative models that learn a probabilistic mapping between the input data and a latent representation space. BNNs are discriminative models that extend traditional neural networks by treating the model parameters as random variables.

Autoencoders typically consist of two main components: an encoder and a decoder. The encoder, denoted by the function $q_\phi(\mathbf{z}|\mathbf{x})$, maps the input data $\mathbf{x}$ into a latent representation $\mathbf{z}$, while the decoder, denoted by $p_\theta(\mathbf{x}|\mathbf{z})$, attempts to reconstruct the input data from this latent representation. The aim is to minimize the reconstruction error, typically measured by a loss function such as mean squared error for continuous input data or cross-entropy loss for binary input data. VAEs build upon this structure but introduce a critical twist by incorporating a probabilistic model. The encoder in a VAE becomes a variational encoder that learns the parameters (μ and σ^2) of a probability distribution, typically a Gaussian, which represents the latent space. Thus, instead of directly outputting a fixed point $\mathbf{z}$, the encoder outputs parameters to a distribution from which $\mathbf{z}$ is sampled.

The basic principle of VAEs involves the estimation of the posterior distribution $p_\theta(\mathbf{x}|\mathbf{z})$, which is typically intractable. The intractability stems from the integral over all possible values of the latent variables $\mathbf{z}$ when trying to marginalize them out to get the likelihood of the observed data $\mathbf{x}$:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})d\mathbf{z}. \tag{16}$$

To overcome this, VAEs maximize the Evidence Lower Bound (ELBO) on the marginal likelihood of the data. The ELBO is given by the equation:

$$\mathcal{L}_{\text{ELBO}}(\theta, \phi; \mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log\, p_\theta(\mathbf{x}|\mathbf{z})] - \text{KL}[q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})]. \tag{17}$$

The first term is the expected log-likelihood, which encourages the decoder to reconstruct the data accurately. The second term is the Kullback-Leibler (KL) divergence between the encoder's distribution $q_\phi(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{z})$, the prior distribution over the latent variables, which acts as a regularizer and enforces the latent space to resemble the prior. The parameters θ and ϕ of the model are optimized to maximize the ELBO, leading to a balance between accurate reconstruction and a well-formed probabilistic latent space. By optimizing this lower bound, VAEs are effectively trained to produce representations that encapsulate the essential characteristics of the data, including its underlying uncertainty.

Bayesian Neural Networks (BNNs) offer an intricate method that integrates the flexibility of neural networks with the probabilistic rigor of Bayesian inference. BNNs are predicated on the notion of assigning probability distributions to the neural network's parameters $\boldsymbol{\theta}$, which encapsulates the model's uncertainty. This is a departure from traditional neural networks that often yield a single point estimate for parameters upon training completion. During the training phase, BNNs aim to ascertain the posterior distribution of the parameters $P(\boldsymbol{\theta}|D)$. This is a result of updating the prior distribution $P(\boldsymbol{\theta})$ with the likelihood of the observed data $P(D|\boldsymbol{\theta})$, as Eq. (15). In the context of predictions, BNNs leverage Bayesian model averaging during the test phase. This is not merely based on a singular set of parameters but rather an aggregation of multiple parameter sets sampled from the posterior distribution. For a given test sample $\mathbf{x}^*$, the prediction thereby incorporates the model uncertainty, yielding the predictive distribution:

$$p(y^*|\mathbf{x}^*, D) = \int p(y^*|\mathbf{x}^*, \boldsymbol{\theta})p(\boldsymbol{\theta}|D)d\boldsymbol{\theta}. \tag{18}$$

Due to the general intractability of the integral, various techniques for inferring $p(\boldsymbol{\theta}|D)$ have been proposed. Gal et al. [30] demonstrated that Monte Carlo dropout is equivalent to sampling from an approximate posterior of $\boldsymbol{\theta}$, allowing Eq. (18) to be simplified to the following Monte Carlo integral:

$$p(y^*|\mathbf{x}^*, D) \approx \frac{1}{S}\sum_{i=1}^{S} p(y^*|\mathbf{x}^*, \boldsymbol{\theta}_i), \tag{19}$$

where dropout is used to obtain S weights $\boldsymbol{\theta}_i$. Hasanzadeh et al. [44] extended the concept of dropout to graphs, introducing Graph DropConnect (GDC) to more effectively leverage the topological structure of graphs. Another technique for approximating the posterior distribution $p(\boldsymbol{\theta}|D)$ is variational inference (VI). The objective is to approximate a distribution $q_\phi(\boldsymbol{\theta})$ that is close to the posterior $p(\boldsymbol{\theta}|D)$. This can be achieved by minimizing the Kullback-Leibler divergence $\mathrm{KL}(q_\phi(\boldsymbol{\theta})\|p(\boldsymbol{\theta}|D))$, and in practice, by maximizing the following evidence lower bound (ELBO) [10]:

$$\mathcal{L}_{\mathrm{ELBO}}(\phi) = \int q_\phi(\boldsymbol{\theta}) \log p(Y|X, \boldsymbol{\theta}) d\boldsymbol{\theta} - \mathrm{KL}(q_\phi(\boldsymbol{\theta})\|p(\boldsymbol{\theta}))) \tag{20}$$

Once $P(\boldsymbol{\theta}|D)$ is obtained, the uncertainty quantified by the above Bayesian framework can be elegantly decomposed by the following relationship:[20]:

$$\underbrace{I(y^*, \boldsymbol{\theta}|\mathbf{x}^*, D)}_{Epistemic} = \underbrace{\mathcal{H}\left[\mathbb{E}_{P(\boldsymbol{\theta}|D)}\left[P(y^*|\mathbf{x}^*, \boldsymbol{\theta})\right]\right]}_{Entropy} - \underbrace{\mathbb{E}_{P(\boldsymbol{\theta}|D)}\left[\mathcal{H}\left[P(y^*|\mathbf{x}^*, \boldsymbol{\theta})\right]\right]}_{Aleatoric}, \tag{21}$$

where $\mathcal{H}(\cdot)$ is Shannon's entropy of a probability distribution. The *entropy* term represents the total uncertainty in the prediction while the *aleatoric* term reflects the inherent noise in the data that cannot be reduced with more information. The subtraction of aleatoric uncertainty from total uncertainty provides the measure of epistemic uncertainty, which denotes the uncertainty in the model's parameters and reduces as more data becomes available. This mathematical demarcation between the uncertainties empowers BNNs to provide a nuanced understanding of prediction reliability and decision-making under uncertainty.

4.1.3 Bayesian Representation Learning for Graphs.

The endeavor to encode the complexity of graph-structured data within a framework that acknowledges and quantifies uncertainty has led to the evolution of Bayesian graph representation learning. This field extends beyond deterministic embeddings, acknowledging that the inherent stochasticity of graphs—arising from irregular structures, noisy features, and other factors—must be captured for a more faithful representation. GNNs lay the groundwork by leveraging local neighborhood information, but their deterministic nature often falls short in representing the ambiguity and unpredictability inherent in real-world graphs. To bridge this gap, Bayesian extensions to GNNs have emerged. One route lies in integrating the principles of variational inference to explicitly model the uncertainties in graph data, thus allowing for more nuanced and informative graph embeddings that facilitate enhanced downstream task performance in the presence of uncertainty.

Kipf and Welling [59] introduced a pioneering framework for unsupervised learning on graph-structured data using variational autoencoders. They proposed a GCN based encoder that effectively captures the probabilistic distribution of latent variables representing the graph. The encoder transforms the input graph G with nodes V into a latent space $\mathbf{Z}$, mapping node i to a latent vector $\mathbf{z}_i$ with the variational posterior $q(\mathbf{Z}|\mathbf{X}, \mathbf{A})$ approximated as a product of Gaussian distributions for each node. The model defines the posterior as $q(\mathbf{z}_i|\mathbf{x}_i, \mathbf{A}) = \mathcal{N}(\mathbf{z}_i|\boldsymbol{\mu}_i, \mathrm{diag}(\boldsymbol{\sigma}_i^2))$, where $\boldsymbol{\mu}$ and $\log\ \boldsymbol{\sigma}^2$ are the outputs of a GCN applied to the graph's adjacency matrix $\mathbf{A}$ and node features $\mathbf{X}$. The objective function derived

from the ELBO (Eq. 17) to be optimized is

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}[\log\, p(\mathbf{A}|\mathbf{Z})] - \mathrm{KL}[q(\mathbf{Z}|\mathbf{X},\mathbf{A})\|p(\mathbf{Z})], \tag{22}$$

where $\log\, p(\mathbf{A}|\mathbf{Z})$ is the reconstruction loss that encourages the model to reconstruct the graph adjacency matrix, and $\mathrm{KL}[q(\mathbf{Z}|\mathbf{X},\mathbf{A})\|p(\mathbf{Z})]$ is the KL divergence acting as a regularizer, pushing the variational distribution towards a prior $p(\mathbf{Z})$, typically chosen to be a standard normal distribution.

To enhance the flexibility of Kipf and Welling's work or to overcome some of its limitations, various methods have been introduced. Hasanzadeh et al. [45] employed a hierarchical variational framework that enabled neighboring nodes to share information, thereby facilitating improved generative modeling of the graph's dependency structure. This approach allowed the propagation of graph structural information and distribution uncertainty, which was essential for capturing complex posterior distributions. To clearly explain what latent factors are and why they perform well, Li et al. [66] introduced the Dirichlet Graph Variational Autoencoder (DGVAE) with graph cluster memberships as latent factors. This study connects VAE-based graph generation with balanced graph cut and provides a new perspective for understanding and enhancing the internal mechanisms of VAE-based graph generation. Ahn and Kim [3] observed that existing autoencoders produce embeddings for isolated nodes that are close to zero, regardless of their content features. To address this issue, they proposed a novel Variational Graph Normalized AutoEncoder (VGNAE) that utilizes L_2-normalization to derive improved embeddings for isolated nodes. Grover et al. [35] proposed a method that augments the decoder with an "iterative" GNN-based decoder to address the limitation of generating only node embeddings, which is not conducive to sampling new graphs. Differing from the node-level VGAE mentioned above, Simonovsky and Komodakis [99] proposed a graph-level VAE [43], named GraphVAE, which modifies the encoder and decoder functions to work with graph-level latent representations, consequently facilitating the direct modeling of uncertainty within graphs.

Leveraging the strength of graph variational autoencoders in representation and uncertainty modeling, some studies have integrated them into GNNs for application in downstream tasks. Hajiramezanali et al. [41] developed a novel hierarchical variational model that introduces additional latent random variables. This model jointly captures the hidden states of a Graph Recurrent Neural Network (GRNN) to effectively represent both topological changes and node attribute variations in dynamic graphs. To learn dynamic graph representations in hyperbolic space while also modeling uncertainty, Sun et al. [103] devised a hyperbolic graph variational autoencoder based on the proposed TGNN. This framework generates stochastic node representations from hyperbolic normal distributions.

Another route lies in transferring general Bayesian neural networks to GNNs. These works consider model parameters, passed messages, etc., as distributions rather than fixed values, and propose a series of Bayesian Graph Neural Networks (BGNNs). To address limitations in GCNs' ability to handle uncertain graph structures, Zhang et al. [134] treated the observed graph G_{obs} as a sample from a random graph family. As a result, the posterior probability of node (or graph) labels can be computed by:

$$p(\mathbf{Z}|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, G_{obs}) = \int p(\mathbf{Z}|\boldsymbol{\theta}, G, \mathbf{X}) p(\boldsymbol{\theta}|\mathbf{Y}_{\mathcal{L}}, G, \mathbf{X}) p(G|\lambda) p(\lambda|G_{obs})\, d\boldsymbol{\theta}\, dG\, d\lambda\,. \tag{23}$$

Here $\boldsymbol{\theta}$ is a random variable representing the parameters of a Bayesian GCN over graph G, and λ denotes the parameters that characterize a family of random graphs. $\mathbf{Y}_{\mathcal{L}}$ and $\mathbf{X}$ represent the training label set and the node feature matrix, respectively. $p(\mathbf{Z}|\boldsymbol{\theta}, G, \mathbf{X})$ can be modelled using a categorical distribution by applying a softmax function to the output of the L-layer GCN $\mathbf{Z} = \mathbf{H}^{(L)}$. Due to the intractability of the integral in Eq. (23), the authors employ the following

Monte Carlo approximation:

$$p(\mathbf{Z}|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, G_{obs}) \approx \frac{1}{N_G S} \sum_{i=1}^{N_G} \sum_{s=1}^{S} p(\mathbf{Z}|\boldsymbol{\theta}_{s,i}, G_i, \mathbf{X}) \,. \tag{24}$$

In this approximation, N_G graphs G_i are sampled from $p(G|\hat{\lambda})$, where $\hat{\lambda}$ is the parameter of the random graph model obtained through maximum a posteriori estimation $\hat{\lambda} = \arg\max_{\lambda} p(\lambda|G_{obs})$. S weights $\boldsymbol{\theta}$ are drawn from $p(\boldsymbol{\theta}|\mathbf{Y}_{\mathcal{L}}, G_i, \mathbf{X})$ using Monte Carlo dropout across the Bayesian GCN corresponding to G_i.

Regarding the specific choice of the random graph model in Eq. (23), Zhang et al. [134] utilized an assortative mixed membership stochastic block model (a-MMSBM) [34, 68]. To more effectively harness the information offered by node features and training labels for the inference of graph topology, Pal et al. [88] introduced a generative model based on node copying as an alternative to the a-MMSBM. Motivated by comparable considerations, Pal et al. [87] proposed a non-parametric posterior distribution of the graph G to infer the graph topology.

In addition to considering the observed graph as sampled from a random graph family, some studies also treat the inputs and other elements within the GNN as random variables. To model the propagation of uncertainty in message-passing mechanisms, Xu et al. [122] treated messages as multivariate Gaussian variables and employed GNNs to predict their means. To align with the intuition that more neighboring nodes provide more evidence, they defined an uncertainty propagation mechanism in GNNs for predicting the covariance of Gaussian distributions, which differs from traditional message passing. The covariance ultimately provides the uncertainty of predictions. Zhao et al. [137] viewed model parameters as variables and then obtained both epistemic uncertainty and aleatoric uncertainty based on Eq. (21). Additionally, this work incorporates uncertainty from evidence theory, namely *vacuity* due to lack of evidence and *dissonance* due to conflicting evidence, offering a richer representation of uncertainty. Elinas et al. [24]considered the adjacency matrix $\mathbf{A}$ as a random variable. The prior distribution of the adjacency matrix $\mathbf{A}$ is given by:

$$p(\mathbf{A}) = \Pi_{ij} p(A_{ij}), \text{ with } p(A_{ij}) = \text{Bern}(A_{ij}|\rho_{ij}^{o}), \tag{25}$$

where $\text{Bern}(A_{ij}|\rho_{ij}^{o})$ is a Bernoulli distribution over ρ_{ij}^{o}. The variational posterior, which takes a form similar to the prior, is given by:

$$q_{\phi}(\mathbf{A}) = \Pi_{ij} q_{\phi}(A_{ij}), \text{ with } q_{\phi}(A_{ij}) = \text{Bern}(A_{ij}|\rho_{ij}), \ \rho_{ij} > 0, \tag{26}$$

where ρ_{ij} are free parameters then $\phi = \{\rho_{ij}\}$. By relaxing the discrete distribution to a continuous Concrete distribution [54, 74] and maximizing the ELBO, they are able to estimate the parameters ϕ of the posterior $q_{\phi}(\mathbf{A})$. Munikoti et al. [79] considered probabilistic links, noise in node features, and modeling errors, treating node features, links between nodes, and model parameters as variables. They propagated aleatoric uncertainty to the output by defining mechanisms for the propagation of means and variances of the node embeddings in GNN, while also estimating epistemic uncertainty through MC dropout.

5 METHODS FOR UNCERTAINTY HANDLING

5.1 Out of Distribution

Out-of-distribution (OOD) pertains to instances when the data encountered by a model substantially deviates from the data upon which it was originally trained. It is especially significant for models deployed in real-world contexts, where they are subjected to a diverse array of inputs that are unpredictable and uncertain, in stark contrast to the controlled and uniform datasets used for training. Despite concerted efforts to encompass a wide range of data, these efforts often fail to encapsulate the full extent of variability present in real-world environments. Consequently, when

models are exposed to OOD data, they may exhibit underperformance or unpredictable behavior, underscoring the imperative for the development of models that are both robust and reliable. To address this problem, there has been a surge of research focused on out-of-distribution (OOD) generalization. The strategies developed to mitigate the issues posed by OOD generalization can be broadly classified into several categories [65], including distributionally robust optimization (DRO), adversarial training (AT), self-supervised learning (SSL), and so on. Most of them primarily aim to improve model performance under OOD conditions during *test* time, for example, recommendation systems make better predictions on new item domains. However, this survey concentrates on examining the measurement, calibration, and conformity of predictive uncertainty, areas often overlooked in OOD research. Thus, rather than delving into specific model architectures, this survey will emphasize the way to integrate OOD and uncertainty studies, especially how to measure and estimate the disagreement between training data and test data in OOD studies.

5.1.1 *Distributionally Robust Optimization Overview.*

Distributionally Robust Optimization (DRO) aims to enhance the model's performance across a range of potential data distributions, instead of the observed data distribution, e.g., training data. Specifically, DRO optimizes the objective function under the "worst case" of the distributions where the loss will be maximized. Formally, DRO is formulated by a bi-level optimization problem:

$$\min_{f\in\mathcal{F}} \sup_{P\in\mathcal{P}} \mathbb{E}_{x\sim P}[\mathcal{L}_0(f)], \qquad \text{s.t. } \mathcal{P} := \{P|D(P,P_0) \le \rho\}. \tag{27}$$

where $\mathcal{L}_0$ is a classification loss, such as cross-entropy. P_0 is the observed data distribution, which could be estimated by the empirical data distribution $\hat{P}_0$. The uncertainty set $\mathcal{P}$ is the set of distributions within the robust radius distance ρ from P_0. $D(\cdot,\cdot)$ measures the distance between two distributions. Many measures D of the distribution distance are used in DRO.

f-divergence, also known as phi-divergence, is formally represented by the formula

$$D_f(P||P_0) = \int f\left(\frac{dP}{dP_0}\right) dP_0. \tag{28}$$

This metric is widely utilized for quantifying the divergence between distributions in DRO [7, 8, 22]. It is noted that the domain delineated by f-divergence can be viewed as a statistical confidence region, and the inner maximization problem is tractable for several choices of f [8, 89].

KL-divergence, as one special case of f-divergence, is also widely used as the distance metric in DRO, such as the works [51, 117]. Especially, authors in [117] propose that DRO adopting KL-divergence as D is equivalent to the contrastive learning in some cases:

$$\begin{aligned}\mathcal{L}_{CL-DRO} =& -\mathbb{E}_{P_0}\left[\mathbb{E}_{(x,y^+)}[f(x,y^+)] - \min_{\alpha\ge 0,\beta}\max_{P\in\mathcal{P}}\{\alpha[D_{KL}(P||P_0)-\eta] + \beta(\mathbb{E}_{P_0}\left[\frac{P}{P_0}\right]-1)\}\right] \\ =& \alpha^*(\eta)\mathcal{L}_{InfoNCE} + Constant.\end{aligned} \tag{29}$$

The possible limitation of metrics based on f-divergence is that they may not be comprehensive enough to include some relevant distributions, or they may fail to precisely measure the distance for extreme distributions. Alternatively, some works adopt the Wasserstein distance to evaluate the distribution distance [31, 76, 136]. Formally, the Wasserstein distance is defined by

$$D_W(P,P_0) := \inf_{\gamma\in T(P,P_0)} \mathbb{E}_{p,p_0\sim\gamma}[c(p,p_0)], \tag{30}$$

where $T(p, p_0)$ contains all possible couplings of p and p_0. $c(p, p_0)$ is some cost functions transferring from p to p_0 and $c(p, p_0) \geq 0$.

Besides that, authors in [101] study DRO with the uncertainty sets measured by maximum mean discrepancy (MMD). The authors in [25] propose to define the uncertainty sets using the Prohorov metric.

5.1.2 *Distributionally Robust Optimization for Graphs.*

Many works have studied DRO with various distribution metrics on graphs. Authors in [15] assume that node embeddings share the same label m adhere to an underlying distribution ($e_i \sim P_m \in \mathcal{P}_m, \forall i : y_i = m$), within an uncertainty set $\mathcal{P}_m$ encompassing all potential distributions P_m. Correspondingly, the objective function is given by

$$\min_{f \in \mathcal{F}} \max_{P_m \in \mathcal{P}_m, 1 \leq m \leq M} \mathcal{L}_0(f; P_1, \ldots, P_M) \tag{31}$$

$\mathcal{L}_0 = \sum_{m=1}^{M} \mathbb{E}_{e \sim P_m}[1 - \pi_m(\xi)]$ is the risk function. The uncertainty set $\mathcal{P}_m$ for each label m is defined by the Wasserstein distance:

$$\mathcal{P}_m := \{P_m | D(P_m, \hat{P}_m) \leq \rho\}, \tag{32}$$

and the empirical distribution $\hat{P}_m$ is defined by the Dirac point mass:

$$\hat{P}_m := \frac{1}{|\{i : y_i = m\}|} \sum_{i=1}^{n} \delta_{\xi_i} \mathbb{1}\{y_i = m\},\ m = 1, \ldots, M, \tag{33}$$

where δ is the Dirac delta function, and $|\cdot|$ is the cardinality of a set.

Authors in [96, 132] also utilize Wasserstein distance as distribution metric D. Especially, authors in [132] address data uncertainty by establishing an uncertainty set for the distributions of the observed data. They propose a graph learning framework that employs Wasserstein DRO and two models based on two assumptions on the uncertainty of the data, respectively. One presumes that every distribution within the uncertainty set is Gaussian, while the second one does not rely on prior distributional assumptions.

Furthermore, authors in [113] study DRO in the field of recommendation systems, where the KL-divergence, D is used to measure the distribution difference. The authors prove that LightGCN, $E^{(k)} = \tilde{A}E^{(k-1)}$, conduct as a Graph smoothness regularizer in the case of DRO and propose a regularization-based method to promote model robustness/generalization.

5.1.3 *AT and SSL for Graphs.*

Besides DRO, it is noticeable that many other methods, such as adversarial training (AT) and self-supervised learning (SSL) are proposed to mitigate OOD issues. However, most of these works adopt L_p-norm or predefined perturbations to quantify or mimic the uncertainty.

In essence, adversarial training on graphs addresses the following min-max optimization problem:

$$\min_{\Theta} \max_{\|\epsilon_\theta\| \leq \rho_\theta, \|\epsilon_x\| \leq \rho_x, \|\epsilon_a\| \leq \rho_a} \mathcal{L}_0(\Theta + \epsilon_\theta, X + \epsilon_x, A + \epsilon_a), \tag{34}$$

where ϵ_θ, ϵ_x, ϵ_a represent the perturbations applied to model parameters, input node features, and adjacency matrix, respectively. In the most recent works, some focus on modifying the graph structure [28], other apply perturbations to model parameters [119], and a few consider perturbations on both node feature and model parameters [123]. Regardless of whether the input data or the model parameters are being perturbed, it is worth noting that perturbation magnitude (uncertainty) is typically measured using the L_p-norm, such as the L_1, L_2, or L_∞ norm.

In the field of self-supervised learning (SSL) for graph analysis, the core idea entails the deliberate perturbation or modification of graphs to construct a new, controlled dataset wherein the intrinsic information of the graphs remains unchanged. These modifications are typically predefined and encompass augmentations at the node, edge, and subgraph levels, facilitating the generation of positive pair instances. Such augmentations include, but are not limited to, node masking [50], edge perturbation, attribute masking, subgraph sampling [127], and the utilization of random walks for graph sampling [93]. Some work proposes the augmentation of graph data at both structural and attribute levels [139]. Furthermore, the ambit of SSL objectives is diverse: authors in [126] propose a novel SSL task, centered on the prediction of the d-pattern tree descriptor. Authors in [71] propose a collaborative framework between teacher and student networks aimed at discovering and leveraging high-confidence, high-quality labeled data. However, these works refrain from explicitly formulating the modifications, and thus cannot provide precise measurement of the uncertainty for the changes.

5.2 Conformal Prediction

5.2.1 *Conformal Prediction Overview.*

Conformal Prediction (CP), or Conformal Inference, is an uncertainty quantification framework that can be applied to almost any model. With a trained model and a pre-designed significant level α, CP takes advantage of the experience of the calibration data to provide the estimated confidence intervals of predicted outcomes based on the scoring equations, which quantify the similarity, or conformity in other words, between the test data and calibration data. The theoretical essentials of CP are illustrated in "Algorithmic Learning in a Random World" [111] by Vladimir Vovk and coauthors in 2005. In this section, we will offer readers a generalized introduction to CP. First, we would like to help readers develop a basic understanding of the frameworks of CP on classification and regression models. Then, we talk about how to generate adaptive CP, which is a trending topic in this domain. We will also cover some content about how to improve the stability of CP and the combination of CP and Bayesian inference.

For CP on classification models, assume we have a trained classifier $\hat{f}$ which can output the estimated probability $\hat{y}_i = \hat{f}(X_i) \in \Delta^K$, such as softmax. Another calibration set of size n, D_{cal}, including (X_i, Y_i), $i = 1, .., n$, is applied to quantify how uncertainty the model $\hat{f}$ would be.

Definition 5.1 (Exchangeability). For any $z_1, \ldots, z_n$ and any permutation ζ of $\{1, \ldots, n\}$, $\mathbb{P}\left((Z_{\zeta(1)}, \ldots, Z_{\zeta(n)}) = (z_1, \ldots, z_n)\right) = \mathbb{P}\left((Z_1, \ldots, Z_n) = (z_1, \ldots, z_n)\right)$ holds.

Theorem 5.2. *With the model $\hat{f}$ and the calibration dataset D_{cal}, whose elements (X_i, Y_i) and (X_{test}, Y_{test}) are **exchangeable**, the conformal score is defined as $s_i = 1 - \hat{f}(X_i)$. The empirical quantile of the conformal score set is below, where S is the score set including s_i, $i = 1...n$.*

$$Q_{1-\alpha}(S, D_{cal}) := (1-\alpha)(1 + 1/|D_{cal}|)\text{-th empirical quantile of } S. \tag{35}$$

The coverage of the prediction set, $C(X_{test}) = \{y | \hat{f}(X_{test})_y \geq 1 - Q_{1-\alpha}\}$, is at least $1 - \alpha$.

$$1 - \alpha \leq P\left(Y_{test} \in C(X_{test})\right) \leq 1 - \alpha + \frac{1}{(n+1)} \tag{36}$$

The score function $s(X, y)$ measures how y "conforms" to the prediction at X. Selecting a suitable score function is one of the challenges in conformal prediction, especially for conformal prediction in the graphs. One popular choice is adaptive prediction sets (APS). APS first sorts the predicted distribution into descending order, such that $\pi_{\zeta(1)}(X) \geq \pi_{\zeta(2)}(X) \geq \cdots \geq \pi_{\zeta(K)}(X)$. Then score function is defined as $s(X, y) = \sum_j \pi_{\zeta(j)}(X)$, and the corresponding prediction

set is constructed as

$$C(X) = \{\zeta(1), \ldots, \zeta(k^*)\}, \quad \text{where } k^* = \inf\{k : \sum_{j=1}^{k} \pi_{\zeta(j)}(X) \geq Q_{1-\alpha}\}. \tag{37}$$

One may utilize a uniform random value to break potential ties between scores [102].

The procedure of applying CP on a regression model is similar to the process mentioned above. However, the prediction sets in this case become intervals instead of assembles of predicted labels. With a trained regressor $\hat{\mu}$ and a calibration dataset D_{cal}, to construct a prediction set that satisfies Eq. (36), we also need to calculate the conformal score of D_{cal}, but this time it is the residual between the predicted values and the true value. $r_i = |\hat{\mu}(X_i) - Y_i|$. The empirical quantile on R, including r_i, $i = 1, .., n$, is

$$Q_{1-\alpha}(R, D_{cal}) := (1-\alpha)(1 + 1/|D_{cal}|)\text{-th empirical quantile of } R. \tag{38}$$

Finally, the predicted interval of i.d.d. X_{test} with given α can be written as $C(X_{test}) = [\hat{\mu}(X_{test}) - Q_{1-\alpha}(R, D_{cal}), \hat{\mu}(X_{test}) + Q_{1-\alpha}(R, D_{cal})]$.

One weakness of traditional CP is that it takes all elements within the calibration set to quantify the uncertainty of the trained model. This may cause the model to be overconfident in hard examples but underconfident in easy examples. However, since we know the value of X_{test}, we can make use of the information to make the prediction set $C(X_{test})$ more adaptive to the test input.

The goal of improving the adaptiveness of CP is to reduce the size of prediction sets under the guarantee of prediction coverage. Designing novel conformal score equations is a widely applied method to improve the adaptiveness of prediction sets. In [95], the authors revised the traditional conformal score equation to generate adaptive prediction sets for classification models. We follow the setup of the model $\hat{f}$ discussed above. For each element X_i in D_{cal}, the trained model $\hat{f}$ can output the softmax result for all potential K classes, $\hat{f}(X_i)_{y_k} = \pi_{y_k}(X_i), k = 1, 2, .., K$. During this process, we assign the indices $\{1, 2, .., K\}$ to elements within the set $(\pi_{y_k}(X_i))_{k \in [K]}$ from high to low, which means $\pi_{y_1}(X_i)$ is the possibility of the most likely class of X_i, whereas $\pi_{y_K}(X_i)$ is the least possible output. Then, the conformal score is defined as the summation of the first M elements where $y_M = Y_i$, which is the true class of X_i.

$$s_i = \sum_{j=1}^{M} \pi_{y_j}(X_i), \text{ where } y_M = Y_i. \tag{39}$$

In other words, the score equation will keep including the softmax results from high to low until meets the true class. We still define S as the set of s_i and then calculate the quantile of S via Eq. (35). With this quantile value, we will form the prediction set $C(X_{test}) = \{y_1, .., y_M, y_{M+1} | M : \sup(\sum_{j=1}^{M} \pi_{y_j}(X_i)) < Q_{1-\alpha}\}$. y_{M+1} is included in $C(X_{test})$ to avoid empty sets. [4] is a well-cited publication using this adaptive prediction method for image classification tasks.

Applying localizers involved with the neighbor information of test data is also an effective way to make predicted intervals more adaptive. For instance, in [37], the localizer for regression problems, with score r, is defined as the following, which can give conformal scores closer to the test input higher weights.

$$H(X_{test}, X_i) = e^{-5|X_i - Xtest|}. \tag{40}$$

Let δ_r denote the point mass at r. The weighted conformal score distribution according to the test input X_{test} is

$$\mathcal{F}_{test} := (\sum_{i=1}^{n} p^H_{test,i} \delta_{r_i}) \text{ for i = 1...n , where } p^H_{test,i} = \frac{H_{test,i}}{\sum_{j=1}^{n} H_{test,j}}. \tag{41}$$

Some other work considers improving the adaptiveness of CP during the training step. For instance, [102] apply conformalization on mini-batches during training to reduce prediction size.

Notice that all the algorithms above are built on the assumption that test data comes from the same distribution of the calibration set, which means (X_i, Y_i) and (X_{test}, Y_{test}) are exchangeable. However, in the real world, the probability distributions between calibration data D_{cal} and test data D_{test} can be different, i.e., **non-exchangeable**.

Covariant shift is a typical situation in which the distribution of input values changes between D_{cal} and D_{test}, i.e., $P_{D_{cal}}(X) \neq P_{D_{test}}(X)$, but their conditional probabilities of obtaining the same output are identical, i.e., $P_{D_{cal}}(Y|X) = P_{D_{test}}(Y|X)$. [106] introduced using the likelihood ratio between $P_{D_{test}}(X)$ and $P_{D_{cal}}(X)$ to calculate the corresponding weight of each conformal score from the calibration set. To be specific, we define $w(x) = \frac{dP_{D_{test}}(X)}{dP_{D_{cal}}(X)}$, then the weights are in Eq. (42). These weights will assign higher priority to conformal scores which are more likely to appear in the test probability distribution. The quantile can be rewritten from Eq. (38) to illustrate the influence of p_{cal}, as Eq. (43).

$$p^i_{cal}(x) = \frac{w(X_i)}{\sum_{j=1}^{n} w(X_j) + w(x)}, \text{ and } p_{test}(x) = \frac{w(x)}{\sum_{j=1}^{n} w(X_j) + w(x)}. \tag{42}$$

$$Q(x) = inf\{R_j, \sum_{i=1}^{j} p^i_{cal}(x)\mathbb{1}\{R_i < R_j\} \leq 1 - \alpha\}. \tag{43}$$

Concept shift is another kind of probability distribution difference where the conditional probabilities between calibration distribution and test distribution are inconsistent,i.e.,$P_{D_{cal}}(Y|X) \neq P_{D_{test}}(Y|X)$. For online prediction on time series data, [33] proposed an adaptive conformal inference method that can adjust the miscoverage value α and prediction size according to the historical data. [120] takes a different method by aggregating bootstrap estimators to calculate prediction sets based on the recent new samples. For the domain adaptation case, [13] defines that the joint distribution of test data belongs to the f-divergence ball centered on the calibration distribution, and $(1 - \alpha)$ can be guaranteed by calculating the worst-case prediction set. [6] summarized two sources of non-exchangeability, data is not i.i.d and the training algorithm does not treat data symmetrically. It employed a fixed, rather than data-dependent, weighted quantile to introduce robustness to the first source and designed a new randomization technique to swap orders of samples for nonsymmetric algorithms.

In the previous discussion, we begin with a well-trained model $\hat{f}$, which is called **inductive** case, but in fact, how to wisely split the known data D_{known} into training data D_{train} to train $\hat{f}$ and calibration data D_{cal} to calculate conformal score is a long-standing concern for CP. Below we briefly introduce the works for data processing and CP implementation to improve CP's reliability.

Full conformal prediction (FCP), also referred to as **transductive** case, is the first proposed method for conducting CP. With training data D_{train} of (X_i, Y_i) and a test input X_{test}, the trained model f^z is based on $D_{train} \cup (X_{test}, z)$, where z is all possible Y_i labels appear in D_{train}, so multiple models need to be trained. For each f^z, the conformal scores are $r_i = |Y_i - f^z(X_i)|$ and $r_{test} = |z - f^z(x_{test})|$. If r_{test} is within the $1 - \alpha$ quantile of conformal scores, we put the corresponding z to the prediction set of X_test. FCP is computationally expensive and split conformal prediction (SCP) is developed to reduce the cost. As we discussed, SCP divides the known data D_{known} into training data D_{train} and calibration data D_{cal} and calculates the prediction set based on the conformal scores from D_{cal}. However, such division may lead to randomness and unreliability of prediction output due to the reduced sample size for training and calibration phases. To solve this problem, [98] suggested that using between 70% and 90% of the data for training often achieves a good balance between minimizing the size of the prediction intervals and the variability of practical coverage. Some cross-validation-related works are also designed to improve prediction robustness, such as cross-conformal

prediction [110, 112], and CP with jackkneif+ [5, 23]. Stable CP [82] states that the robustness of CP can be characterized by the trained model's stability τ.

A combination of CP and Bayesian method is proposed in [46]. With a trained Bayesian model $\hat{f}(y|X_{test})$, which estimates the posterior distribution of the true label Y_{test} with given input X_{test}. Theoretically, the optimal prediction set should be

$$C(X_{test}) = \{y : \hat{f}(y|X_{test}) > t\}, \text{ where t is chosen so } \int_{y \in C(X_{test})} \hat{f}(y|X_{test})dy = 1 - \alpha. \tag{44}$$

However, Eq. (44) is hard to calculate due to the limitation of sample numbers and approximated posterior distributions, such as by Monte Carlo Dropout, so we can define the conformal score as $s(X_{test}, y) = -\hat{f}(y|X_{test})$. After calculating the quantile value $Q_{1-\alpha}$ on the calibration set, the prediction set of the input feature X_i is

$$C(X_{test}) = \{y : \hat{f}(y|X_{test}) > -Q_{1-\alpha}\}. \tag{45}$$

5.2.2 *Conformal Prediction for Graphs.*

For node classification tasks on graphs, the exchangeability may be violated for some specific settings [130]. First, in the *transductive* settings, the model has access to the entire graph structure and all the node features during training, calibration, and testing, but the labels of calibration and testing set are not available during training. In this case, the union of calibration and the unlabeled sample is exchangeable. However, in the *inductive* settings, the model only has access to the subgraph induced by the nodes in the training set $\mathcal{D}_{train}$ during training. The assumption of exchangeability does not hold in this case.

In the study [56], the uncertainty of a node X_i inferred a GNN is defined by:

$$\mathbb{U}(X_i) = \mathbb{C}^+(X_i) - \mathbb{C}^-(X_i), \tag{46}$$

where $\mathbb{C}^+(X_i)$ and $\mathbb{C}^-(X_i)$ are the lower and upper bound of the predictive confidence interval of node X_i by the jackknife method, respectively:

$$\begin{aligned} \mathbb{C}^+(X_i) &= \text{Quantile}(1-\alpha; \|f(X_i; \Theta^*_{\epsilon,j})\|_2 + err_j | \forall j \in \mathcal{D}_{train} \backslash \{X_j\}), \\ \mathbb{C}^-(X_i) &= \text{Quantile}(\alpha; \|f(X_i; \Theta^*_{\epsilon,j})\|_2 - err_j | \forall j \in \mathcal{D}_{train} \backslash \{X_j\}), \end{aligned} \tag{47}$$

where $err_j = \|y_j, f(j, \Theta^*_{\epsilon,j})\|_2$ is the leave-one-out (LOO) error of node X_j. $\Theta^*_{\epsilon,j}$ is the loss when upweighting the importance of a node X_j with small constant ϵ:

$$\Theta^*_{\epsilon,j} = \arg\min_{\Theta} \epsilon \ell(X_j, y_j) + \frac{1}{|\mathcal{D}_{train}|} \sum_{X_i \in \mathcal{D}_{train}} \ell(X_i, y_i), \tag{48}$$

ℓ is a node-specific loss, such as cross-entropy. The influence function is considered to estimate the LOO loss and $\Theta^*_{\epsilon,j}$, without retraining the model. Specifically, the influence function is defined as

$$\Theta^*_{\epsilon,j} \approx \Theta^* + \epsilon H^{-1}_{\Theta^*} \nabla_{\Theta} \ell(X_j, y_j), \tag{49}$$

where H_{Θ^*} is the Hessian matrix with respect to the model parameters, and Eq. (49) is effectively estimated by the vectorization and the Hessian-vector product.

In the study [130], a diffused score function, namely Diffused Adaptive Prediction Sets (DAPS), is introduced. DAPS exploits the graph structure based on neighborhood diffusion, and the diffused score function is defined as

$$\hat{s}(X_i, y) = (1 - \lambda)s(X_i, y) + \frac{\lambda}{|\mathcal{N}_i|} \sum_{X_j \in \mathcal{N}_i} s(X_j, y), \tag{50}$$

where λ signifies the diffusion parameter. From a matrix perspective, given the original score matrix $H \in \mathbb{R}^{|V| \times K}$, where each row is the conformal score for a node, the (1-hop) diffused score matrix is defined as

$$\hat{H} = (1 - \lambda)H + \lambda D^{-1}AH, \tag{51}$$

where D is the degree matrix. Extending it to a general k-hop diffused version, the formulation is given by

$$\hat{H} = \lambda_0 H + \sum_{i=1}^{k} \lambda_i (D^{-1}A)^i \times H. \tag{52}$$

Note that the diffused score function preserves exchangeability if the original scores are exchangeable. This diffusion process proves particularly advantageous in homophilous graphs, where connected nodes exhibit similar ground truth distributions. The rationale behind this is that for a node surrounded by mostly unperturbed neighbors, its probability can be better estimated by using its neighbors' vectors (information), as long as these vectors are sufficiently similar.

For the non-exchangeable conformal prediction cases, the procedure assumes a choice of deterministic fixed weights $w_1, \ldots, w_n \in [0, 1]$ [6]. The prediction set concerning the weighted quantiles of the score distribution

$$\hat{C}(X) = \left\{ y \in \mathcal{Y} : s(X, y) \leq \hat{q}_{1-\alpha} \left(\sum_{i=1}^{n} w_i \delta_{s_i} \right) \right\} \tag{53}$$

The inductive cases are studied in [18], where the principle of exchangeability is not applicable. They integrate non-exchangeable conformal prediction with the concept of homophily in graphs, leading to the development of three distinct variants of Neighbourhood Adaptive Prediction Sets (NAPS) for the prediction set construction. The primary variant, NAPS, assigns the weights in Eq. (53) to $w_i = 1$ if $X_i \in \mathcal{N}_{n+1}^T$, where $\mathcal{N}_{n+1}^T$ includes all nodes within T-hop neighborhood of node X_{n+1}. The second one, NAPS-H, adopts a hyperbolic decay rate for $w_i = t^{-1}$ where t is the t-hop neighbor of node X_{n+1}. The third one, NAPS-G, uses a geometric decay rate for $w_i = 2^{-t}$.

5.3 Calibration

5.3.1 *Calibration Overview.*

Classification tasks by machine learning are preferred to output both the predicted label and the confidence of the prediction, which can be regarded as the probability that the predicted label is correct. For example, with a trained model f, the estimation of correctness, or accuracy, of its predictions with confidence = 0.8 is expected to be 80%. The calibration performance of modern neural networks has been proven poor due to a series of factors, including network width, depth, batch normalization, and weight decay [38].

Calibration Formulation:

We begin with a classifier f trained by feature set X and corresponding label set Y whose label space is $\mathcal{Y} = \{1, ..., K\}$. With n inputs $x_i \in X$ having true label y_i, f applies softmax operation to output a probability vector, $\mathbf{v}(x_i) = \mathbf{v}_i = (v_i^1(x), ..., v_i^K(x))$, $\sum_{j=1}^{K} v_i^j = 1$. The predicted label of x_i, denoted by $\hat{y}_i$, is the element with the highest value, which is

the confidence, $\hat{p}_i$, of the prediction.

$$\hat{y}_i = \arg\max_{j \in K} v_i^j, \ \hat{p}_i = \max_{j \in K} v_i^j. \tag{54}$$

A model is well-calibrated if the probability of correct prediction equals the prediction confidence for all labels and all confidence levels [38].

$$\mathbf{P}(y_i = j | v_i^j = p) = p, \ \forall j \in \mathcal{Y}, \ \forall p \in [0, 1]. \tag{55}$$

Calibration Method:

The processing of calibrating a model's prediction can be post-processed (post hoc), or incorporated during the model's training. Based on this, we categorize calibration methods and introduce them as below.

(i) Post-processed Calibration: Post-processed calibration requires an unseen calibration set, which should not be the same as the training set, as a model loss on training data would be biased compared with the loss on unseen data.

Histogram binning was initially proposed in [128] as a non-parametric calibration method of decision trees and naive Bayesian classifiers, and this method can be generalized to almost all models. Assume we have a binary classification task with label space $\mathcal{Y} = \{0, 1\}$ and $\hat{p}_{i,1}$ as the confidence of sample i in label 1. Here we slightly revise the definition of B_m. Instead of being the set of samples whose prediction confidences, $\hat{p}_i$, fall into $(\frac{m-1}{M}, \frac{m}{M}]$, now B_m includes samples whose $\hat{p}_{i,1}$ is in the interval. Then, we try to minimize the equation below, where θ_m is the estimated number of positive-class samples in the B_m set. For a test sample x_t whose $\hat{p}_{t,1}$ belong to I_m, $\hat{p}_{t,1}$ should be replaced by θ_m as a calibrated confidence.

$$\min_{\theta_1, \ldots, \theta_M} \sum_{m=1}^{M} \sum_{i \in B_m} (\theta_m - y_i)^2 \tag{56}$$

I_m in histogram binning is usually assigned according to equal-width or equal-frequency methods with pre-defined total bin number M. Isotonic regression by [129] insists the number and size of I_m should be optimized as well. We write I_m as an interval of $(a_{m-1}, a_m]$, and isotonic regression optimizes the following objective function.

$$\min_{\substack{\theta_1 \leq, \ldots, \leq \theta_M \\ a_0 \leq, \ldots \leq a_M = 1}} \sum_{m=1}^{M} \sum_{i=1}^{n} \mathbb{1}(a_{m-1} \leq \hat{p}_{i,1} \leq a_m)(\theta_m - y_i)^2. \tag{57}$$

[61] applied isotonic regression to deep learning models for regression tasks with promising performance. Both histogram binning and isotonic regression can be applied to multi-class classification tasks using the one-vs-rest strategy.

As we mentioned previously, $\mathbf{v}(x) = (v_1(x), ..., v_K(x))$ is the output vector where $v_i(x)$ indicates the probability of x's label $y = i$. This $\mathbf{v}$ is transformed from a logit vector $\mathbf{z}$ via softmax function $\sigma(\cdot)$.

$$\mathbf{v} = \sigma(\mathbf{z}) = \frac{(e^{z_1}, ..., e^{z_K})}{\sum_{i=1}^{K} e^{z_i}} \tag{58}$$

Platt scaling is introduced by [91] as a parametric calibration method for support vector machines, and it can be generalized by adding scale and location parameters as below, which can be optimized by NLL as discussed above.

$$\mathbf{v} = \sigma(\mathbf{W} \cdot \mathbf{z} + \mathbf{b}) \tag{59}$$

When $\mathbf{W}$ is a scaler as $\frac{1}{T}$ and $\mathbf{b} = 0$, it is temperature scaling. We are using the same scale factor for different z_i and the predicted class remains unchanged. When $\mathbf{W}$ is a diagonal matrix $\in \mathbf{R}^{K \times K}$, it becomes vector scaling, assigning different scale factors to classes. $\mathbf{W}$ can be an ordinary matrix as well, which is the most generalized form, but this form may lead to overfitting by over-parameterization.

(ii) Calibration during Training: The true label of input sample x_i can be represented as a one-hot vector $\mathbf{y}_i$ where the element corresponding to correct label y_i is 1 and the rest elements are 0. The cross-entropy (CE) of predicted vector $\mathbf{v}_i$ and $\mathbf{y}_i$ is listed below, where y_i^j and v_i^j denote j-th elements of vector $\mathbf{y}_i$ and $\mathbf{v}_i$.

$$CE(\mathbf{y}_i, \mathbf{v}_i) = -\sum_{j=1}^{K} y_i^j \log v_i^j \tag{60}$$

Label smoothing can be applied to calibration as stated in [78] by preventing over-confidence. Tuning parameter $\alpha \in [0, 1]$, the one-hot vector $\mathbf{y}_i$ will be smoother. When $\alpha = 1$, $\mathbf{y'}_i$ will represent a uniform distribution. As we train $CE(\mathbf{y'}_i, \mathbf{v}_i)$, label smoothing encourages the model to be less confident, as it penalizes the model for assigning full probability to the ground truth label.

$$\mathbf{y}_i = (y_i^1, ..., y_i^K) \rightarrow \mathbf{y'}_i = ((1-\alpha)y_i^1 + \frac{\alpha}{K}, ..., (1-\alpha)y_i^K + \frac{\alpha}{K}) \tag{61}$$

Focal loss (FL) was introduced by [69] to address the problem of class imbalance and this loss function, as shown below, was applied to calibration in [77].

$$FL(x_i) = -(1 - v_i^{\hat{y}_i})^\gamma \log(v_i^{\hat{y}_i}), \gamma \geq 0 \tag{62}$$

Focal loss only considers confidence in the true label and the loss value will decrease faster as we increase γ. Thus, it can prevent being over-confident. If $\gamma = 0$, FL goes back to CE.

5.3.2 *Calibration for Graphs.*

Calibration on graph neural networks has started to draw people's attention recently. GNNs for classification tasks can generate softmax outputs that can be regarded as probabilities that the predicted results are correct. Accuracy alone is not enough for high-stake decision-making. This section mainly focuses on GNNs' confidence calibration for node classification tasks. Some works concentrating on the calibration of fair graph learning and link prediction are also mentioned at the end.

[105] is one of the first works investigating the calibration of GNNs. As it states, traditional calibration methods are used for models with i.i.d. data. However, when dealing with graph data (relational data), the features or state of a node can depend on the features or states of its neighbors. This is inherently a dependent structure because the properties of one node are not solely determined by that node alone but also by its context within the graph. This dependence makes the calibration of GNNs challenging. Besides, the class imbalance will cause collapsed predictions, so it is necessary to rebalance classes during training by weighting loss function. This work trained Graph Convolutional Networks [58], Graph Attention Networks [108], and Graph Graph Isomorphism Networks [121] on multiple datasets, concluding that GNNs can be miscalibrated and existing methods can not recalibrate them. Compared with histogram binning and isotonic regression, temperature scaling usually outputs the best calibration results in the experiment. Moreover, a new ECE-derived metric, $\text{ECE}_{\geq 50}$, is proposed as an ECE computed only over examples with confidence higher than 50%, because people care more about the calibration on predicted labels.

CaGCN is a topology-aware post-hoc calibration GNN model developed by [116] under a validated assumption that neighboured nodes share similar ground truth confidence levels. It observes that GNNs tend to be under-confident in their predictions. This paper focuses on an undirected attributed graph, $G = (V, E)$, for semi-supervised node classification tasks, with adjacent matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ and the node feature matrix $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]^T$. The output of a l-layer GCN before the softmax layer can be calculated below, where $\mathbf{W}^{(l)}$ is the weight matrix of l-th layer and $\sigma(\cdot)$ is

the activation function.

$$\mathbf{V} = \mathbf{A}\sigma(...\mathbf{A}\sigma(\mathbf{AXW}^{(1)})\mathbf{W}^{(2)}...)\mathbf{W}^{(l)} = [\mathbf{v}_1, ..., \mathbf{v}_N]^T. \tag{63}$$

CaGCN is a non-linear calibration model that takes topology information into account for relational data and can preserve the accuracy of the original classification GCN. With $\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_N]^T$, where $\mathbf{v}_i = [v_{i,1}, ..., v_{i,K}]$ is the logit output of node i with K potential labels. CaGCN works as another GCN on $\mathbf{V}$ to calibrate predictions as follows:

$$\mathbf{t} = \sigma^+(\mathbf{A}\sigma(...\mathbf{A}\sigma(\mathbf{AVW}^{(1)})\mathbf{W}^{(2)}...)\mathbf{W}^{(l)}) = [t_1, ..., t_N]^T \tag{64}$$

$$\mathbf{v'}_i = [v_{i,1}/t_i, ..., v_{i,K}/t_i]^T,\ \mathbf{z}_i = [\sigma_{SM}(v'_{i,1}), ..., \sigma_{SM}(v'_{i,K})] \tag{65}$$

Temperature t_i for each node i is a non-negative scalar, obtained by the final layer of CaGCN by $\sigma^+(x) = 1/(1+log(e^x))$, which is an element-wise softplus activation function. With σ_{SM} as a softmax function of $\mathbf{v'}_i$, $\hat{p}_i = \max_k z_{i,k}$ is the confidence of prediction of node i. As t_I is only a scaler, temperature scaling on $\mathbf{v'}_i$ does not influence the original prediction accuracy. The optimization objective function of CaGCN is a combination of NLL loss and a customized calibration loss on validation nodes, balanced by a hyperparameter.

Graph Attention Temperature Scaling (GATS) is developed by [49], introducing the attention mechanism in the calibration model. By the experiment of GCN [58] and GAT [108] on seven datasets. It validated the general under-confident tendency previously claimed by [116]. Also, relative confidence level, $\delta\hat{p}_i$, and node homophily are proposed as below to measure the confidence similarity and label similarity among neighbor nodes.

$$\text{Relative confidence} = \delta\hat{p}_i = \hat{p}_i - \frac{1}{|n(i)|}\sum_{j\in n(i)}\hat{p}_j \tag{66}$$

$$\text{Node homophily} = \log(\frac{n_a+1}{n_d+1}) \tag{67}$$

$n(i)$ indicates the neighbor nodes of node i, n_a is the number of agreeing neighbors, and n_d is the number of disagreeing ones. Experimental results show calibration error decreases as node homophily increases and absolute relative confidence, $|\delta\hat{p}_i|$ decreases, proving the insight from [116] as well. Based on the above discussion of influential factors, GATS produce node-wise temperature, T_i, as follows.

$$\forall i \in V, T_i = \frac{1}{H}\sum_{h=1}^{H}\text{softplus}(w\delta\hat{p}_i + \sum_{j\in n(i)}\alpha_{i,j}\gamma_j\tau_j^h) + T_0 \tag{68}$$

Global bias T_0 accounts for the general under-confident tendency. Relative confidence,$\delta\hat{p}_i$, is incorporated with a learnable parameter w as an additional contribution term. It also aggregates neighborhood influences τ_j with attention coefficient $\alpha_{i,j}$. Distance to training nodes is represented as a node-wise scaling factor γ_j. Finally, the above expression works on a multi-head formulation, and h stands for h-th attention head. Experiments demonstrate the overall performance of GATS is better than CaGCN on eight datasets since CaGCN tends to have an overly complex architecture similar to the original classification GCN.

Graph Calibration Loss (GCL) is introduced by [115] as an end-to-end calibration method for GNNs. DNNs are vulnerable to overfitting during training and cause overconfidence due to their high framework capacity. However, to prevent vanishing gradient issues as indicated by [67], GNNs prefer "shallow" architectures (usually less than 4 layers). Inspired by this phenomenon, the authors point out that their low framework capacity may cause the under-confidence of GNNs. As the number of network layers increases, GNNs change from being under-confident to overconfident. The GCL is designed to assign up-weight loss to examples with high confidence as below, where K is the number of classes,

p_y and $\hat{p}_y$ are the ground truth probability and predicted probability of the y-th class.

$$\mathcal{L}_{GCL} = -\sum_{y=1}^{K}(1+\gamma\hat{p}_y)p_y \log \hat{p}_y \tag{69}$$

GCL is the lower bound of the summation of KL divergence and predicted entropy. Minimizing the predicted distribution entropy $\hat{p}_y$ will help calibrate the GNNs. Several representative GNNs and various datasets proved the effectiveness of GCL in terms of ECE with M=20.

$$\mathcal{L}_{GCL} = \mathrm{KL}(p||\hat{p}_y) + \gamma p\mathbb{H}[\hat{p}] \leq \mathrm{KL}(p||\hat{p}_y) + \mathbb{H}[\hat{p}] + \mathbb{H}[p] \tag{70}$$

To investigate the effectiveness of multiple calibration techniques, [109] conducted experiments on Platt Scaling (PS) and its extension, Temperature Scaling (TS) [63, 91], Monte-Carlo dropout (MC-dropout) [30], Model Ensemble [29], Accuracy-versus-Uncertainty (AvU) [75], Graph Calibration Loss (GCL) [115], on GNN models with medical image data. The primary result is that models generated by multi-class node classification tasks will be better calibrated and more discriminative than those trained by binary tasks. Also, the overall performance of PS, TS, AvU, and GCL is better than MC-dropout and model ensemble, which are based on the average prediction of multiple models. The limited architectural diversity of GCNs may cause their underperformance.

Ratio-binned scaling (RBS), provided by [72], is also a topology-aware calibration method based on the similarity of neighbor nodes' labels. First, its experimental study shows that GNNs with wider layers tend to be better calibrated and confirms [115]'s observation of the non-monotonic relationship between network depth and ECE. Moreover, it finds out that GNNs are under-confident with early-stopping and will be overconfident with too many training epochs, as they start to overfit. The same-class-neighbor ratio, the proportion of neighbors that have the same class as the central node, is applied to quantify neighbors' similarity to the central node. However, the conclusion is conflict to the observations in [49]. The authors find that well-calibration is more likely to occur when the ratio is around 0.4 to 0.5. Too low a same-class-neighbor ratio will cause over-confidence and too high a ratio will cause under-confidence. RBS will first estimate the same-class-neighbor ratio of each node, group them into M bins, and finally learn a temperature for each bin to re-calibrate them.

Automated graph learning [135] also draws much attention as the prediction accuracy of traditional GNNs can be easily influenced by hyperparameter choices. Hyperparameter optimization is a mainstream direction of automated GNNs, which can be formulated as a bilevel optimization problem [73] as the following, where h and θ denotes hyperparameters and parameters, $\mathcal{L}_T$ and $\mathcal{L}_V$ represent training loss and validation loss.

$$h^* = \underset{h}{\operatorname{argmin}}\mathcal{L}_V(h,\theta^*) \text{ s.t. } \theta^* = \underset{\theta}{\operatorname{argmin}}\mathcal{L}_T(h,\theta) \tag{71}$$

HyperU-GCN is proposed by [125] as a bilevel uncertainty quantification model. The upper-level problem explains uncertainties by developing probabilistic hypernetworks through a variational Bayesian lens. The lower-level problem learns how hyperparameter distribution is propagated to the GCN weights. The optimization function of HyperU-GCN is shown below, where $p(h)$ is a pre-determined hyperparameter prior and $q_\phi(h)$ is the variational distribution parameterized by ϕ, learning the variance of h.

$$\min_{h,\phi}\mathcal{L}_V(q_\phi(h),\theta(h)) + KL(q_\phi(h)||p(h)) \text{ s.t. } \theta(h) = \underset{\theta}{\operatorname{argmin}}\mathcal{L}_T(\theta(h),q_\phi(h)) \tag{72}$$

To make GNNs robust to adversarial attacks and discriminatory biases, FairGNN [19], RobustGNN [138], and NIFTY [2] are proposed to enhance the stability of GNNs. However, these methods will make GNNs either overconfident

in incorrect predictions or biased to global graph noises. To address the problem, Multi-view Confidence-calibrated NIFTY (MCCNIFTY) is introduced by [133]. Based on evidential theory, a novel node embedding learning module is developed. It includes an intra-view evidence calibration, an inter-view evidence fusion, and an uncertainty-aware message-passing process, optimizing for counterfactual fairness and stability at the sub-graph level.

Link prediction [131] is also an important downstream task for GNN node classification, whereas the calibration of link prediction has not been explored extensively. [81] studies the calibration performance of link prediction and finds that GNNs tend to be overconfident in negative link predictions but are underconfident in positive ones. To address the issue, IN-N-OUT is proposed to parameterize the temperature for calibration. To be specific, two embeddings of an edge will be obtained when we train the GNNs with and without the edge, and calculating the discrepancy between the embeddings of the edge will help determine the value of temperature scaling.

6 UNCERTAINTY QUANTIFICATION EVALUATION

6.1 Calibration

The evaluation of calibration measures the difference between a model's prediction confidence and the exact probability that the predicted output is true. This section introduces two widely applied metrics.

Expected Calibration Error (ECE): We divide the entire confidence space [0,1] into M equal-size intervals $I_m = (\frac{m-1}{M}, \frac{m}{M}]$. B_m represents the set of samples whose confidences fall into I_m. The accuracy and confidence of B_m are illustrated below, where y_i and $\hat{y}_i$ are the true and predicted label of input x_i, and $\hat{p}_i$ is the confidence of the prediction.

$$acc(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}(\hat{y}_i = y_i), \tag{73}$$

$$conf(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i. \tag{74}$$

Expected Calibration Error (ECE) is designed to approximate the absolute difference between the two sides of Eq. (55) using the accuracy and confidence of B_m as below. We can also plot $acc(B_m)$ and $conf(B_m)$ called a reliability diagram [84] to visualize a model's calibration performance. The plot of a well-calibrated model should follow a diagonal, indicating $acc(B_m) = conf(B_m)$.

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)|. \tag{75}$$

Some works are extending the concept of ECE to make it suitable for different scenarios. [80] introduces Maximum Calibration Error(MCE), $MCE = \max_m |acc(B_m) - conf(B_m)|$, to compute the maximum gap between accuracy and confidence. [62] designed classwise ECE (cwECE), $cwECE = \frac{1}{K} \sum_{i=1}^{K} \sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)|$, to see averaged ECE on the class level.

Edge-wise ECE is designed by [48] as the authors state that node-wise ECE, used in [72, 105, 115, 116] is only suitable for i.i.d test samples and ignores graph structure. With graph $G = (\mathcal{N}, \mathcal{E})$, denote $\mathcal{E}_U = \{(i, j) \in \mathcal{E} | i, j \in U\}$ the subset of edges $\mathcal{E}$ constrained to test nodes U. The confidence of edge (i, j) is $\hat{c}_{(i,j)} = \max_{l,m} \hat{p}(y_i = l, y_j = m | x, y_L)$, where $y_L = \{y_i\}_{i \in L = \mathcal{N} \setminus U}$ and $x = \{x_i\}_{i \in \mathcal{N}}$. The calculation of edge-wise ECE is similar to node-wise ones. Each $\hat{c}_{(i,j)}$ is computed and assigned to corresponding bins. For accuracy, if the predicted l and m are correct, the edge prediction is

accurate. The gap between confidence bins and accuracy is edge-wise ECE. Besides, ECE for agreeing node pairs and disagreeing node pairs are also proposed.

Proper Scoring Rules: The constraint of ECE is the loss of accuracy due to the finite number of I_m. Thus, the continuous proper scoring rule can act as an alternative to ECE. For example, with training n samples, we can apply negative log-likelihood (NLL) as below, where $\hat{p}_{i,y_i}$ is the x_i's confidence of the correct label y_i.

$$NLL = \frac{1}{n}\sum_{i=1}^{n}\log(\hat{p}_{i,y_i}) \tag{76}$$

NLL is minimized when the model assigns a high probability to the correct class for each instance, which means the model is rewarded for being confident in its correct predictions. However, this does not necessarily mean that the model is well-calibrated. A well-calibrated model not only makes accurate predictions but also outputs probability estimates that reflect the true likelihood of the prediction being correct. As [60, 77] stated, miscalibration in machine learning models has been linked to the overfitting of NLL.

Brier Score (BS) [11] is a more sophisticated metric as it considers a sample's confidence in all labels. $\hat{p}_{i,j}$ is the confidence of sample i in label j.

$$BS = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{K}(\hat{p}_{i,j} - \mathbb{1}(y_i = j)) \tag{77}$$

6.2 Conformal Prediction

The key contribution of conformal prediction is stated in Eq. (36), requiring the coverage of prediction sets. However, it is not explicitly claimed but desired that a conformal prediction model can be adaptive for different conditions. As [4] says, we want the CP procedure can distinguish the difficulty of quantifying the prediction uncertainty of each input, and adaptively generate a larger prediction set as the difficulty increases. In other words, the $(1-\alpha)$ coverage can be robust and stable no matter how the test domain shifts. Let $E_1, ..., E_n$ denote the n sets of test inputs coming from the corresponding test domains, and we want the coverage guarantee to always hold as below.

$$P(Y_{test} \in C(X_{test})) \geq 1-\alpha, X_{test} \in e, \forall e \in \{E_1, ..., E_n\} \tag{78}$$

This test domain can be characterized by an explicit feature in input data, an underlying environment variable related to the data generation process, or just the sizes of prediction sets.

Conditional Coverage Metric calculate the worst coverage among multiple validation domains. Let $\mathcal{I}_e$ represent the sample index set of set e and $|\mathcal{I}_e|$ denote the number of samples in set e [14, 27].

$$\min_{e\in\{E_1,...,E_n\}} \frac{1}{|\mathcal{I}_e|}\sum_{i\in|\mathcal{I}_e|}\mathbb{1}\{Y_i \in C(X_i)\} \tag{79}$$

If the metric value is much lower than the expected coverage $(1-\alpha)$, we say the CP procedure performs poorly on conditional coverage.

Coverage Correctness: The empirical coverage by split conformal prediction involves randomness due to the limited number of available samples and possibly biased data splitting. Running conformal prediction multiple times can help us determine the correctness and stability of prediction coverage. With n labeled samples available, we can randomly split them into n_{cal} samples for calibration and n_{val} samples for validation, and calculate the empirical coverage for T times as below. $Y_{i,t}$ and $X_{i,t}$ are the label and feature of the i-th element of n_{val} validation samples at

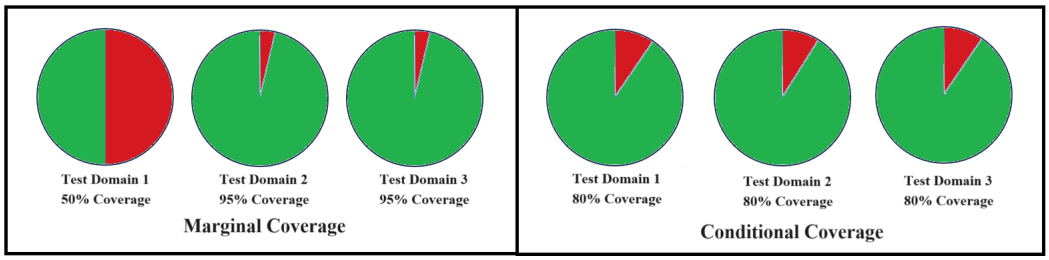

Fig. 3. Marginal Coverage vs Conditional Coverage with (1-α)=0.8. Marginal coverage can only guarantee the coverage on the overall sample set, but conditional coverage can ensure at least 1-α coverage on different sub-domians

t-th time splitting.

$$C_t = \frac{1}{n_{val}} \sum_{i=1}^{n_{val}} \mathbb{1}\{Y_{i,t} \in C(X_{i,t})\}. \tag{80}$$

The final averaged empirical coverage, $\bar{C}$, will be

$$\bar{C} = \frac{1}{T} \sum_{t=1}^{T} C_t. \tag{81}$$

The variance of these empirical coverage values indicates the stability as $\frac{1}{T} \sum_{t=1}^{T} (C_t - \bar{C})^2$

7 CONCLUSION

In this survey, we carefully reviewed the uncertainty quantification within graphical models, specifically focusing on Graph Neural Networks and Probabilistic Graphical Models. We examed existing methods for representing and handling uncertainty, including Bayesian representation learning, conformal prediction, calibration and so on. The domain of uncertainty in graphical models presents substantial opportunities for further innovation and advancement. Advancing our understanding of uncertainty modeling in graph-based systems will empower machine learning systems to make more reliable decisions in the presence of real-world noise. Such progress will not only improve the accuracy and safety of AI applications but also promote trust and encourage the broader adoption of these technologies in practical settings.

REFERENCES

[1] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. 2021. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion* 76 (2021), 243–297.

[2] Chirag Agarwal, Himabindu Lakkaraju, and Marinka Zitnik. 2021. Towards a unified framework for fair and stable graph representation learning. In *Uncertainty in Artificial Intelligence*. PMLR, 2114–2124.

[3] Seong Jin Ahn and MyoungHo Kim. 2021. Variational graph normalized autoencoders. In *Proceedings of the 30th ACM international conference on information & knowledge management*. 2827–2831.

[4] Anastasios Angelopoulos, Stephen Bates, Jitendra Malik, and Michael I Jordan. 2020. Uncertainty sets for image classifiers using conformal prediction. *arXiv preprint arXiv:2009.14193* (2020).

[5] Rina Foygel Barber, Emmanuel J. Candès, Aaditya Ramdas, and Ryan J. Tibshirani. 2019. Predictive inference with the jackknife+. *The Annals of Statistics* (2019). https://api.semanticscholar.org/CorpusID:147704029

[6] Rina Foygel Barber, Emmanuel J Candes, Aaditya Ramdas, and Ryan J Tibshirani. 2023. Conformal prediction beyond exchangeability. *The Annals of Statistics* 51, 2 (2023), 816–845.

[7] Güzin Bayraksan and David K Love. 2015. Data-driven stochastic programming using phi-divergences. In *The operations research revolution*. Informs, 1–19.

[8] Aharon Ben-Tal, Dick Den Hertog, Anja De Waegenaere, Bertrand Melenberg, and Gijs Rennen. 2013. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science* 59, 2 (2013), 341–357.

[9] Christopher M Bishop and Nasser M Nasrabadi. 2006. *Pattern recognition and machine learning*. Vol. 4. Springer.

[10] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. 2017. Variational inference: A review for statisticians. *Journal of the American statistical Association* 112, 518 (2017), 859–877.

[11] Glenn W Brier. 1950. Verification of forecasts expressed in terms of probability. *Monthly weather review* 78, 1 (1950), 1–3.

[12] John R Busenbark, Hyunjung Yoon, Daniel L Gamache, and Michael C Withers. 2022. Omitted variable bias: Examining management research with the impact threshold of a confounding variable (ITCV). *Journal of Management* 48, 1 (2022), 17–48.

[13] Maxime Cauchois, Suyash Gupta, Alnur Ali, and John C Duchi. 2024. Robust validation: Confident predictions even when distributions shift. *J. Amer. Statist. Assoc.* (2024), 1–66.

[14] Maxime Cauchois, Suyash Gupta, and John C Duchi. 2021. Knowing what you know: valid and validated confidence sets in multiclass and multilabel prediction. *Journal of machine learning research* 22, 81 (2021), 1–42.

[15] Shuyi Chen, Kaize Ding, and Shixiang Zhu. 2023. Uncertainty-Aware Robust Learning on Noisy Graphs. *arXiv preprint arXiv:2306.08210* (2023).

[16] Victor Chernozhukov, Carlos Cinelli, Whitney K Newey, Amit Sharma, and Vasilis Syrgkanis. 2021. *Omitted variable bias in machine learned causal models*. Technical Report. cemmap working paper.

[17] Carlos Cinelli and Chad Hazlett. 2020. Making sense of sensitivity: Extending omitted variable bias. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 82, 1 (2020), 39–67.

[18] Jase Clarkson. 2023. Distribution free prediction sets for node classification. In *International Conference on Machine Learning*. PMLR, 6268–6278.

[19] Enyan Dai and Suhang Wang. 2021. Say no to the discrimination: Learning fair graph neural networks with limited sensitive attribute information. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 680–688.

[20] Stefan Depeweg, Jose-Miguel Hernandez-Lobato, Finale Doshi-Velez, and Steffen Udluft. 2018. Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning. In *International Conference on Machine Learning*. PMLR, 1184–1193.

[21] Armen Der Kiureghian and Ove Ditlevsen. 2009. Aleatory or epistemic? Does it matter? *Structural safety* 31, 2 (2009), 105–112.

[22] John C Duchi and Hongseok Namkoong. 2021. Learning models with uniform performance via distributionally robust optimization. *The Annals of Statistics* 49, 3 (2021), 1378–1406.

[23] Bradley Efron. 1982. *The jackknife, the bootstrap and other resampling plans*. SIAM.

[24] Pantelis Elinas, Edwin V Bonilla, and Louis Tiao. 2020. Variational inference for graph convolutional networks in the absence of graph data and adversarial settings. *Advances in Neural Information Processing Systems* 33 (2020), 18648–18660.

[25] Emre Erdoğan and Garud Iyengar. 2006. Ambiguous chance constrained problems and robust optimization. *Mathematical Programming* 107 (2006), 37–61.

[26] Dhivya Eswaran, Stephan Günnemann, and Christos Faloutsos. 2017. The power of certainty: A dirichlet-multinomial model for belief propagation. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 144–152.

[27] Shai Feldman, Stephen Bates, and Yaniv Romano. 2021. Improving conditional coverage via orthogonal quantile regression. *Advances in neural information processing systems* 34 (2021), 2060–2071.

[28] Fuli Feng, Xiangnan He, Jie Tang, and Tat-Seng Chua. 2019. Graph adversarial training: Dynamically regularizing based on graph structure. *IEEE Transactions on Knowledge and Data Engineering* 33, 6 (2019), 2493–2504.

[29] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. 2019. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757* (2019).

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

[31] Rui Gao and Anton Kleywegt. 2023. Distributionally robust stochastic optimization with Wasserstein distance. *Mathematics of Operations Research* 48, 2 (2023), 603–655.

[32] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. 2023. A survey of uncertainty in deep neural networks. *Artificial Intelligence Review* 56, Suppl 1 (2023), 1513–1589.

[33] Isaac Gibbs and Emmanuel J. Candès. 2021. Adaptive Conformal Inference Under Distribution Shift. In *Neural Information Processing Systems*. https://api.semanticscholar.org/CorpusID:235266057

[34] Prem K Gopalan, Sean Gerrish, Michael Freedman, David Blei, and David Mimno. 2012. Scalable inference of overlapping communities. *Advances in Neural Information Processing Systems* 25 (2012).

[35] Aditya Grover, Aaron Zweig, and Stefano Ermon. 2019. Graphite: Iterative generative modeling of graphs. In *International conference on machine learning*. PMLR, 2434–2444.

[36] Cornelia Gruber, Patrick Oliver Schenk, Malte Schierholz, Frauke Kreuter, and Göran Kauermann. 2023. Sources of Uncertainty in Machine Learning–A Statisticians' View. *arXiv preprint arXiv:2305.16703* (2023).

[37] Leying Guan. 2023. Localized conformal prediction: A generalized inference framework for conformal prediction. *Biometrika* 110, 1 (2023), 33–50.

[38] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International conference on machine learning*. PMLR, 1321–1330.

[39] Shivani Gupta and Atul Gupta. 2019. Dealing with noise problem in machine learning data-sets: A systematic review. *Procedia Computer Science* 161 (2019), 466–474.

[40] Ian Hacking. 2006. *The emergence of probability: A philosophical study of early ideas about probability, induction and statistical inference*. Cambridge University Press.

[41] Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. 2019. Variational graph recurrent neural networks. *Advances in neural information processing systems* 32 (2019).

[42] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *NeurIPS* (2017).

[43] William L Hamilton. 2020. *Graph representation learning*. Morgan & Claypool Publishers.

[44] Arman Hasanzadeh, Ehsan Hajiramezanali, Shahin Boluki, Mingyuan Zhou, Nick Duffield, Krishna Narayanan, and Xiaoning Qian. 2020. Bayesian graph neural networks with adaptive connection sampling. In *International conference on machine learning*. PMLR, 4094–4104.

[45] Arman Hasanzadeh, Ehsan Hajiramezanali, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. 2019. Semi-implicit graph variational auto-encoders. *Advances in neural information processing systems* 32 (2019).

[46] Peter Hoff. 2023. Bayes-optimal prediction with frequentist coverage control. *Bernoulli* 29, 2 (2023), 901–928.

[47] Stephen C Hora. 1996. Aleatory and epistemic uncertainty in probability elicitation with an example from hazardous waste management. *Reliability Engineering & System Safety* 54, 2-3 (1996), 217–223.

[48] Hans Hao-Hsun Hsu, Yuesong Shen, and Daniel Cremers. 2022. A graph is more than its nodes: Towards structured uncertainty-aware learning on graphs. *arXiv preprint arXiv:2210.15575* (2022).

[49] Hans Hao-Hsun Hsu, Yuesong Shen, Christian Tomani, and Daniel Cremers. 2022. What Makes Graph Neural Networks Miscalibrated? *Advances in Neural Information Processing Systems* 35 (2022), 13775–13786.

[50] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2019. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265* (2019).

[51] Zhaolin Hu and L Jeff Hong. 2013. Kullback-Leibler divergence constrained distributionally robust optimization. *Available at Optimization Online* 1, 2 (2013), 9.

[52] Ziyi Huang, Henry Lam, and Haofeng Zhang. 2021. Quantifying epistemic uncertainty in deep learning. *arXiv preprint arXiv:2110.12122* (2021).

[53] Eyke Hüllermeier and Willem Waegeman. 2021. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine learning* 110, 3 (2021), 457–506.

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

[55] Michael I Jordan. 2004. Graphical models. (2004).

[56] Jian Kang, Qinghai Zhou, and Hanghang Tong. 2022. JuryGCN: quantifying jackknife uncertainty on graph convolutional networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 742–752.

[57] Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems* 30 (2017).

[58] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[59] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[60] Ranganath Krishnan and Omesh Tickoo. 2020. Improving model calibration with accuracy versus uncertainty optimization. *Advances in Neural Information Processing Systems* 33 (2020), 18237–18248.

[61] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. 2018. Accurate uncertainties for deep learning using calibrated regression. In *International conference on machine learning*. PMLR, 2796–2804.

[62] Meelis Kull, Miquel Perello Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach. 2019. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. *Advances in neural information processing systems* 32 (2019).

[63] Fabian Kuppers, Jan Kronenberger, Amirhossein Shantia, and Anselm Haselhoff. 2020. Multivariate confidence calibration for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 326–327.

[64] Salem Lahlou, Moksh Jain, Hadi Nekoei, Victor Ion Butoi, Paul Bertin, Jarrid Rector-Brooks, Maksym Korablyov, and Yoshua Bengio. 2021. Deup: Direct epistemic uncertainty prediction. *arXiv preprint arXiv:2102.08501* (2021).

[65] Haoyang Li, Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2022. Out-of-distribution generalization on graphs: A survey. *arXiv preprint arXiv:2202.07987* (2022).

[66] Jia Li, Jianwei Yu, Jiajin Li, Honglei Zhang, Kangfei Zhao, Yu Rong, Hong Cheng, and Junzhou Huang. 2020. Dirichlet graph variational autoencoder. *Advances in Neural Information Processing Systems* 33 (2020), 5274–5283.

[67] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[68] Wenzhe Li, Sungjin Ahn, and Max Welling. 2016. Scalable MCMC for mixed membership stochastic blockmodels. In *Artificial Intelligence and Statistics*. PMLR, 723–731.

[69] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*. 2980–2988.

[70] Roderick JA Little and Donald B Rubin. 2019. *Statistical analysis with missing data*. Vol. 793. John Wiley & Sons.

[71] Hongrui Liu, Binbin Hu, Xiao Wang, Chuan Shi, Zhiqiang Zhang, and Jun Zhou. 2022. Confidence may cheat: Self-training on graph neural networks under distribution shift. In *Proceedings of the ACM Web Conference 2022*. 1248–1258.

[72] Tong Liu, Yushan Liu, Marcel Hildebrandt, Mitchell Joblin, Hang Li, and Volker Tresp. 2022. On calibration of graph neural networks for node classification. In *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[73] Matthew MacKay, Paul Vicol, Jon Lorraine, David Duvenaud, and Roger Grosse. 2019. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. *arXiv preprint arXiv:1903.03088* (2019).

[74] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712* (2016).

[75] Prerak Mody, Nicolas F Chaves-de Plaza, Klaus Hildebrandt, and Marius Staring. 2022. Improving error detection in deep learning based radiotherapy autocontouring using bayesian uncertainty. In *International Workshop on Uncertainty for Safe Utilization of Machine Learning in Medical Imaging*. Springer, 70–79.

[76] Peyman Mohajerin Esfahani and Daniel Kuhn. 2018. Data-driven distributionally robust optimization using the Wasserstein metric: Performance guarantees and tractable reformulations. *Mathematical Programming* 171, 1 (2018), 115–166.

[77] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip Torr, and Puneet Dokania. 2020. Calibrating deep neural networks using focal loss. *Advances in Neural Information Processing Systems* 33 (2020), 15288–15299.

[78] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When does label smoothing help? *Advances in neural information processing systems* 32 (2019).

[79] Sai Munikoti, Deepesh Agarwal, Laya Das, and Balasubramaniam Natarajan. 2023. A general framework for quantifying aleatoric and epistemic uncertainty in graph neural networks. *Neurocomputing* 521 (2023), 1–10.

[80] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. 2015. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 29.

[81] Erik Nascimento, Diego Mesquita, Samuel Kaskio, and Amauri H Souza. 2024. In-n-Out: Calibrating Graph Neural Networks for Link Prediction. *arXiv preprint arXiv:2403.04605* (2024).

[82] Eugene Ndiaye. 2022. Stable conformal prediction sets. In *International Conference on Machine Learning*. PMLR, 16462–16479.

[83] Vu-Linh Nguyen, Sébastien Destercke, and Eyke Hüllermeier. 2019. Epistemic uncertainty sampling. In *Discovery Science: 22nd International Conference, DS 2019, Split, Croatia, October 28–30, 2019, Proceedings 22*. Springer, 72–86.

[84] Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*. 625–632.

[85] Chao Ning and Fengqi You. 2019. Optimization under uncertainty in the era of big data and deep learning: When machine learning meets mathematical programming. *Computers & Chemical Engineering* 125 (2019), 434–448.

[86] Curtis G Northcutt, Anish Athalye, and Jonas Mueller. 2021. Pervasive label errors in test sets destabilize machine learning benchmarks. *arXiv preprint arXiv:2103.14749* (2021).

[87] Soumyasundar Pal, Saber Malekmohammadi, Florence Regol, Yingxue Zhang, Yishi Xu, and Mark Coates. 2020. Non parametric graph learning for bayesian graph neural networks. In *Conference on uncertainty in artificial intelligence*. PMLR, 1318–1327.

[88] Soumyasundar Pal, Florence Regol, and Mark Coates. 2019. Bayesian graph convolutional neural networks using node copying. In *Proceedings of the 36th International Conference on Machine Learning, Workshop on Learning and Reasoning with Graph-Structured Representations*. ICML.

[89] Leandro Pardo. 2018. *Statistical inference based on divergence measures*. Chapman and Hall/CRC.

[90] Judea Pearl. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

[91] John Platt et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers* 10, 3 (1999), 61–74.

[92] Apostolos F Psaros, Xuhui Meng, Zongren Zou, Ling Guo, and George Em Karniadakis. 2023. Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *J. Comput. Phys.* 477 (2023), 111902.

[93] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 1150–1160.

[94] Shebuti Rayana and Leman Akoglu. 2015. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining*. 985–994.

[95] Yaniv Romano, Matteo Sesia, and Emmanuel Candes. 2020. Classification with valid and adaptive coverage. *Advances in Neural Information Processing Systems* 33 (2020), 3581–3591.

[96] Alireza Sadeghi, Meng Ma, Bingcong Li, and Georgios B Giannakis. 2021. Distributionally robust semi-supervised learning over graphs. *arXiv preprint arXiv:2110.10582* (2021).

[97] Silvia Seoni, Vicnesh Jahmunah, Massimo Salvi, Prabal Datta Barua, Filippo Molinari, and U Rajendra Acharya. 2023. Application of uncertainty quantification to artificial intelligence in healthcare: A review of last decade (2013–2023). *Computers in Biology and Medicine* (2023), 107441.

[98] Matteo Sesia and Emmanuel J. Candès. 2019. A comparison of some conformal quantile regression methods. *Stat* 9 (2019). https://api.semanticscholar.org/CorpusID:202565594

[99] Martin Simonovsky and Nikos Komodakis. 2018. Graphvae: Towards generation of small graphs using variational autoencoders. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27*. Springer, 412–422.

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

[101] Matthew Staib and Stefanie Jegelka. 2019. Distributionally robust optimization and generalization in kernel methods. *Advances in Neural Information Processing Systems* 32 (2019).

[102] David Stutz, Ali Taylan Cemgil, Arnaud Doucet, et al. 2021. Learning optimal conformal classifiers. *arXiv preprint arXiv:2110.09192* (2021).

[103] Li Sun, Zhongbao Zhang, Jiawei Zhang, Feiyang Wang, Hao Peng, Sen Su, and S Yu Philip. 2021. Hyperbolic variational graph neural network for modeling dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4375–4383.

[104] Xiaolin Tang, Guichuan Zhong, Shen Li, Kai Yang, Keqi Shu, Dongpu Cao, and Xianke Lin. 2023. Uncertainty-aware decision-making for autonomous driving at uncontrolled intersections. *IEEE Transactions on Intelligent Transportation Systems* (2023).

[105] Leonardo Teixeira, Brian Jalaian, and Bruno Ribeiro. 2019. Are graph neural networks miscalibrated? *arXiv preprint arXiv:1905.02296* (2019).

[106] Ryan J Tibshirani, Rina Foygel Barber, Emmanuel Candes, and Aaditya Ramdas. 2019. Conformal prediction under covariate shift. *Advances in neural information processing systems* 32 (2019).

[107] Matias Valdenegro-Toro and Daniel Saromo. 2022. A Deeper Look into Aleatoric and Epistemic Uncertainty Disentanglement. arXiv:2204.09308 [cs.LG]

[108] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[109] Iris Vos, Ishaan Bhat, Birgitta Velthuis, Ynte Ruigrok, and Hugo Kuijf. 2024. Calibration techniques for node classification using graph neural networks on medical image data. In *Medical Imaging with Deep Learning*. PMLR, 1211–1224.

[110] Vladimir Vovk. 2012. Cross-conformal predictors. *Annals of Mathematics and Artificial Intelligence* 74 (2012), 9–28. https://api.semanticscholar.org/CorpusID:51973287

[111] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. 2005. *Algorithmic learning in a random world*. Vol. 29. Springer.

[112] Vladimir Vovk, Ilia Nouretdinov, Valery Manokhin, and Alexander Gammerman. 2018. Cross-conformal predictive distributions. In *conformal and probabilistic prediction and applications*. PMLR, 37–51.

[113] Bohao Wang, Jiawei Chen, Changdong Li, Sheng Zhou, Qihao Shi, Yang Gao, Yan Feng, Chun Chen, and Can Wang. 2024. Distributionally Robust Graph-based Recommendation System. *arXiv preprint arXiv:2402.12994* (2024).

[114] Fangxin Wang, Yuqing Liu, Kay Liu, Yibo Wang, Sourav Medya, and Philip S Yu. 2024. Uncertainty in Graph Neural Networks: A Survey. *arXiv preprint arXiv:2403.07185* (2024).

[115] Min Wang, Hao Yang, and Qing Cheng. 2022. GCL: Graph calibration loss for trustworthy graph neural network. In *Proceedings of the 30th ACM International Conference on Multimedia*. 988–996.

[116] Xiao Wang, Hongrui Liu, Chuan Shi, and Cheng Yang. 2021. Be confident! towards trustworthy graph neural networks via confidence calibration. *Advances in Neural Information Processing Systems* 34 (2021), 23768–23779.

[117] Junkang Wu, Jiawei Chen, Jiancan Wu, Wentao Shi, Xiang Wang, and Xiangnan He. 2024. Understanding contrastive learning via distributionally robust optimization. *Advances in Neural Information Processing Systems* 36 (2024).

[118] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2022. Graph neural networks in recommender systems: a survey. *Comput. Surveys* 55, 5 (2022), 1–37.

[119] Yihan Wu, Aleksandar Bojchevski, and Heng Huang. 2023. Adversarial weight perturbation improves generalization in graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 10417–10425.

[120] Chen Xu and Yao Xie. 2021. Conformal prediction interval for dynamic time-series. In *International Conference on Machine Learning*. PMLR, 11559–11569.

[121] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? *ICLR* (2019).

[122] Zhao Xu, Carolin Lawrence, Ammar Shaker, and Raman Siarheyeu. 2022. Uncertainty Propagation in Node Classification. In *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1275–1280.

[123] Haotian Xue, Kaixiong Zhou, Tianlong Chen, Kai Guo, Xia Hu, Yi Chang, and Xin Wang. 2021. Cap: Co-adversarial perturbation on weights and features for improving generalization of graph neural networks. *arXiv preprint arXiv:2110.14855* (2021).

[124] Yuto Yamaguchi, Christos Faloutsos, and Hiroyuki Kitagawa. 2015. Socnl: Bayesian label propagation with confidence. In *Advances in Knowledge Discovery and Data Mining: 19th Pacific-Asia Conference, PAKDD 2015, Ho Chi Minh City, Vietnam, May 19-22, 2015, Proceedings, Part I 19*. Springer, 633–645.

[125] Xueying Yang, Jiamian Wang, Xujiang Zhao, Sheng Li, and Zhiqiang Tao. 2022. Calibrate automated graph neural network via hyperparameter uncertainty. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 4640–4644.

[126] Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. 2021. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning*. PMLR, 11975–11986.

[127] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in neural information processing systems* 33 (2020), 5812–5823.

[128] Bianca Zadrozny and Charles Elkan. 2001. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Icml*, Vol. 1. 609–616.

[129] Bianca Zadrozny and Charles Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 694–699.

[130] Soroush H Zargarbashi, Simone Antonelli, and Aleksandar Bojchevski. 2023. Conformal prediction sets for graph neural networks. In *International Conference on Machine Learning*. PMLR, 12292–12318.

[131] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *Advances in neural information processing systems* 31 (2018).

[132] Xiang Zhang, Yinfei Xu, Qinghe Liu, Zhicheng Liu, Jian Lu, and Qiao Wang. 2021. Robust graph learning under Wasserstein uncertainty. *arXiv preprint arXiv:2105.04210* (2021).

[133] Xu Zhang, Liang Zhang, Bo Jin, and Xinjiang Lu. 2021. A multi-view confidence-calibrated framework for fair and stable graph representation learning. In *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1493–1498.

[134] Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Ustebay. 2019. Bayesian graph convolutional neural networks for semi-supervised classification. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 5829–5836.

[135] Ziwei Zhang, Xin Wang, and Wenwu Zhu. 2021. Automated machine learning on graphs: A survey. *arXiv preprint arXiv:2103.00742* (2021).

[136] Chaoyue Zhao and Yongpei Guan. 2018. Data-driven risk-averse stochastic optimization with Wasserstein metric. *Operations Research Letters* 46, 2 (2018), 262–267.

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

[138] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1399–1407.

[139] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131* (2020).