Calibrate Automated Graph Neural Network via Hyperparameter Uncertainty

Xueying Yang
Santa Clara University
Santa Clara, CA, United States
xyang9@scu.edu

Jiamian Wang
Santa Clara University
Santa Clara, CA, United States
jwang16@scu.edu

Xujiang Zhao
NEC Laboratories America
Princeton, NJ, United States
xuzhao@nec-labs.com

Sheng Li
University of Virginia
Charlottesville, VA, United States
shengli@virginia.edu

Zhiqiang Tao
Rochester Institute of Technology
Rochester, NY, United States
zhiqiang.tao@rit.edu

ABSTRACT

Automated graph learning has drawn widespread research attention due to its great potential to reduce human efforts when dealing with graph data, among which hyperparameter optimization (HPO) is one of the mainstream directions and has made promising progress. However, how to obtain reliable and trustworthy prediction results with automated graph neural networks (GNN) is still quite underexplored. To this end, we investigate automated GNN calibration by marrying uncertainty estimation to the HPO problem. Specifically, we propose a hyperparameter uncertainty-induced graph convolutional network (HyperU-GCN) with a bilevel formulation, where the *upper-level* problem explicitly reasons uncertainties by developing a probabilistic hypernetworks through a variational Bayesian lens, while the *lower-level* problem learns how the GCN weights respond to a hyperparameter distribution. By squeezing model uncertainty into the hyperparameter space, the proposed HyperU-GCN could achieve calibrated predictions in a similar way to Bayesian model averaging over hyperparameters. Extensive experimental results on six public datasets were provided in terms of node classification accuracy and expected calibration error (ECE), demonstrating the effectiveness of our approach compared with several state-of-the-art uncertainty-aware and calibrated GCN methods.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning algorithms**;
• **Information systems** → *Data mining*.

KEYWORDS

Graph Neural Networks, Uncertainty Estimation, Hypernetworks

ACM Reference Format:
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 and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3511808.3557556

CIKM '22, October 17–21, 2022, Atlanta, GA, USA.

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557556

1 INTRODUCTION

Various graph neural networks (GNN) have been designed to handle complex graph data among different fields, such as graph convolutional networks (GCN) [17], graph attention networks [35], and still many more. While all these GNN models have achieved great success, their prediction results are largely impacted by hyperparameter choices. For example, the improper network depth and edge dropout rate [28] choices will lead to an over-smoothing issue [19] and badly degrade the prediction performance for GNNs. To this end, the automated graph learning [37], *e.g.*, applying hyperparameter optimization (HPO) on GNNs, has received increasing research attention in recent years. Along with its rapid development, the paper focuses on the following question – *how to obtain reliable and trustworthy model predictions in the automated GNN context?*

The above research problem is well explored for traditional GNNs by designing calibration functions [12, 36] or utilizing uncertainty estimation [10, 42]. On the one hand, Wang *et al.* [36] proposed a topology-aware post-hoc calibration method to alleviate the under-confident predictions for GCN, by using another GCN as the calibration function. On the other hand, Zhao *et al.* [42] introduced the evidential deep learning [31] to GCNs, and developed a graph-based kernel Dirichlet distribution estimation (GKDE-GCN) model to separately analyze different kinds of uncertainties in graph network predictions. Moreover, Bayesian approximation could be another way to calibrate the predictive uncertainty, such as applying the Monte-Carlo dropout [10] in GCNs. Yet, to date, it is still quite underexplored to study calibration in automated graph learning.

In this study, we calibrate automated GNNs by marrying uncertainty estimation to the hyperparameter optimization problem [23, 34, 44]. Particularly, we propose a hyperparameter uncertainty-induced GCN (HyperU-GCN) model by developing a new probabilistic hypernetwork. We provide a bilevel formulation for HyperU-GCN to jointly reason uncertainties and tune multiple hyperparameters for GCN (see Fig. 1). Notably, the proposed method enables optimizing a dynamic hyperparameter distribution rather than finding a single configuration, and thus achieves calibrated predictions in a similar way to Bayesian model averaging over hyperparameters. The contributions of this work are summarized as follows.

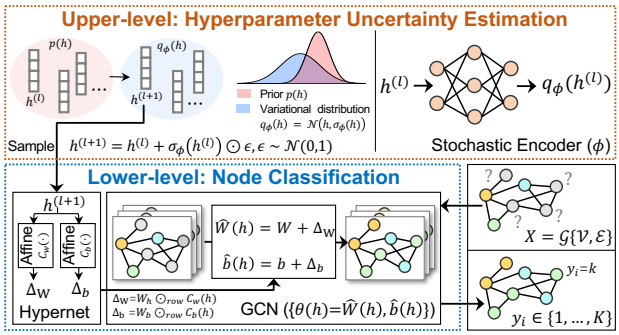


Figure 1: Illustration of the proposed HyperU-GCN model using a bilevel formulation. The upper-level problem reasons hyperparameter uncertainty by developing a probabilistic hypernetwork, while the lower-level learns GCN weights over the optimized variational hyperparameter distribution.

- A new probabilistic hypernetwork is introduced to explicitly model *hyperparameter uncertainty* through a variational Bayesian treatment, enabling a tractable way to squeeze the predictive uncertainty from high-dimensional parameter space to low-dimensional hyperparameter space.
- We propose a novel automated graph learning model, namely HyperU-GCN, by unifying uncertainty estimation and HPO with a bilevel formulation. The HyperU-GCN could lead to robust hypernetworks by learning how GCN weights respond to a hyperparameter distribution, and also provides calibrated predictions via Bayesian model averaging.
- The proposed method has shown an effective solution to provide highly competitive and well-calibrated prediction results, supported by extensive experimental results on six public datasets in terms of accuracy and ECE [12].

2 RELATED WORK

Automated Graph Learning. Graph neural networks achieve great success in complex relationship modeling, but generally require heavy human effort in the algorithm design [38]. Automated graph learning reduces such a dependence by combining the AutoML strategies and GNN models. One of the popular direction is the graph neural architecture search (GNAS) [5, 11, 41, 43], which explores the optimal network configurations in the pre-defined search space, *i.e.*, activation functions, pooling methods, number of layers, and aggregation functions, *etc.* Another popular automated direction is the hyperparameter optimization (HPO) [6, 8, 20]. Specifically, random search and grid search [1] could be directly adopted. Bayesian optimization [4] and hypergradient-based methods [8] also serve as representative tuning strategies. Recently, ST-GCN [44] incorporates hypernetworks [23] into GCNs to jointly optimize hyperparameters and model weights within a population-based training framework. Despite the promising performance, the confidence calibration of the model has not been considered.

Confidence Calibration. There has been growing interest in calibrating the confidence of the automated decision-making systems. Well-calibrated models provide the proper expectation of the success (*i.e.*, classification probability), resulting in trustworthy and more interpretable predictions. A variety of confidence calibration methods are proposed for decades. Histogram binning [39], Isotonic regression [40], and Platt scaling [27] mainly focus on the binary-model calibration. The temperature scaling [12] is one of the most popular methods to calibrate the multiclass methods without sacrificing the predictive performance. Recent work [36] post-processes the logits of the GCN model to obtain the calibrated results. Notably, uncertainty estimation (UE) [18, 24] also benefits the confidence calibration by modeling the probability distribution of the predicted labels. For example, Monte Carlo dropout [33] tackles the problem of over-fitting (overconfident), yielding better-calibrated predictions. Yet, calibrating the graph neural networks from the uncertainty estimation perspective seems to get little research attention so far.

3 METHODOLOGY

3.1 Preliminary

We study the supervised node classification task on the given undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n nodes over K classes, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges between nodes. We denote $\mathbf{A} \in \mathbb{R}^{n \times n}$ as the adjacent matrix for $\mathcal{E}$, $\mathbf{X} \in \mathbb{R}^{n \times d}$ as the node feature matrix, and $\mathbf{Y} \in \mathbb{R}^{n \times K}$ as the label matrix, where $\mathbf{x_i} \in \mathbb{R}^d$ represents the feature of node v_i and $y_i \in \{1, \ldots, K\}$ is the corresponding one-hot label. Let $\mathcal{D} = \{(\mathbf{x}, y)\}$ be the training data, and $f_{\theta,h}$: $\mathbf{x}, \mathbf{A} \rightarrow y$ be the prediction model with GNNs, parameterized by model parameters θ and hyperparameters h. We define the loss function as $\mathcal{L}(h, \theta) = \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}}[-\log p(y|\mathbf{x}; \mathbf{A}))]$ where $p(y|\mathbf{x}; \mathbf{A}) = f_{\theta,h}(\mathbf{x}, \mathbf{A})$. Following the transductive setting [35], we treat the graph structure $\mathbf{A}$ as a fixed quantity and may omit it when no confusion occurs. By calculating $\mathcal{L}(h, \theta)$ on the training/validation datasets, we have the training loss $\mathcal{L}_\mathcal{T}$ and validation loss $\mathcal{L}_\mathcal{V}$, respectively.

Particularly, this study focuses on automated graph learning based on hyperparameter optimization (HPO), which can be formulated as a bilevel optimization problem [23, 34] as the following:

$$h^* = \arg\min_h \mathcal{L}_\mathcal{V}(h, \theta^*) \text{ s.t. } \theta^* = \arg\min_\theta \mathcal{L}_\mathcal{T}(h, \theta), \quad (1)$$

where the *lower-level* problem finds optimal model weights by minimizing $\mathcal{L}_\mathcal{T}$, while the *upper-level* problem automatically tunes the hyperparameter h to minimize $\mathcal{L}_\mathcal{V}$ upon the minimizers θ^*. One possible way for solving Eq. (1) is to transform the bilevel optimization into a single-level problem by substituting a best-response function $\theta^*(h)$ to replace θ^*. To this end, we leverage hypernetworks [13, 21, 29] $\theta(h)$, which acts as a mapping function to approximate $\theta^*(h)$, to further solve the HPO problem as

$$h^* = \arg\min_h \mathcal{L}_\mathcal{V}(h, \theta(h)). \quad (2)$$

We adopt the self-tuning GCN (ST-GCN) [44] to implement $\theta(h)$, enabling to solve Eq. (2) with hypergradients – $\partial\mathcal{L}_\mathcal{V}/\partial h$. Given the weight $W \in \mathbb{R}^{D_{in} \times D_{out}}$ and bias $b \in \mathbb{R}^{D_{out}}$ of one GCN layer, the hypernetwork $\theta(h)$ maps hyperparameters h to GCN weights by

$$\hat{W}(h) = W + W_h \odot_{row} C_w(h) \text{ and } \hat{b}(h) = b + b_h \odot_{row} C_b(h), \quad (3)$$

where W_h/b_h is the counterpart parameter for W/b, $C_w(h) \in \mathbb{R}^{D_{out}}$ and $C_b(h) \in \mathbb{R}^{D_{out}}$ are scaled embeddings of h by linear transformations, $\odot$ represents an element-wise multiplication, and $\odot_{row}$ denotes a row-wise rescaling. By developing hypernetworks, the model parameters are given by $\theta(h) = \{\hat{W}(h), \hat{b}(h)\}$.

3.2 Hyperparameter Uncertainty

Bayesian nerual networks (BNN) [22, 25] are widely used for predictive uncertainty estimation. By applying Bayesian model averaging [7], which marginalizes model parameters by $p(y|\mathbf{x},\mathcal{D}) = \int p(y|\mathbf{x},\theta)p(\theta|\mathcal{D})d\theta$, BNNs could yield well-calibrated predictions by capturing the model (epistemic) uncertainty. However, it is generally intractable to directly compute the parameter posterior due to the high-dimensional parameter space and computational complexity. In light of this, we propose a new perspective to investigate predictive uncertainty over parameters – hyperparameters uncertainty – by squeezing model uncertainty into a relatively low-dimensional hyperparameter space. We first formulate the predictive uncertainty upon both model parameters θ and hyperparameters h as

$$p(y|\mathbf{x},\mathcal{D}) = \iint p(y|\mathbf{x},\theta)p(\theta|h)p(h|\mathcal{D})d\theta dh. \tag{4}$$

Notably, a large hyperparameter uncertainty will lead to a high-variance of posterior distribution over model parameters, which further impacts the predictive uncertainty estimation. By marginalizing out θ, we can obtain a clear formulation w.r.t h as

$$p(y|\mathbf{x},\mathcal{D}) = \int p(y|\mathbf{x},h)p(h|\mathcal{D})dh. \tag{5}$$

Based on Eq. (5), we define hyperparameter uncertainty as the mutual information [3, 9, 24] between the model prediction y and hyperparameter h as the following:

$$\underbrace{\mathbb{I}[y,h|\mathbf{x},\mathcal{D}]}_{\text{hyper uncertainty}} = \underbrace{\mathbb{H}[\mathbb{E}_{p(h|\mathcal{D})}[p(y|\mathbf{x},h)]]}_{\text{total uncertainty}} - \underbrace{\mathbb{E}_{p(h|\mathcal{D})}[\mathbb{H}[p(y|\mathbf{x},h)]]}_{\text{data uncertainty}}. \tag{6}$$

Following [2, 9], we employ variational inference to approximate the hyperparameter posterior by minimizing $KL(q_\phi(h)||p(h|\mathcal{D}))$, where ϕ parameterizes the variational distribution $q_\phi(h)$ and could be implemented as neural networks. Given the data log-evidence as $\log P(\mathbf{Y}|\mathbf{X}) = \sum_{(\mathbf{x},y)\in\mathcal{D}} logp(y|\mathbf{x})$, we derive the evidence lower bound (ELBO) [14, 16] *w.r.t.* $q_\phi(h)$ as

$$logP(\mathbf{Y}|\mathbf{X}) \geq \mathbb{E}_{q_\phi(h)}[\log p(\mathbf{Y}|h,\mathbf{X})] - KL(q_\phi(h)||p(h)), \tag{7}$$

where $p(h)$ is the given hyperparameter prior (*e.g.*, normal distribution) and $p(\mathbf{Y}|h,\mathbf{X}) = \prod_{(\mathbf{x},y)\in\mathcal{D}} p(y|\mathbf{x},h)$. To simplify the integral inside $p(y|\mathbf{x},h)$, we leverage point estimation by formulating $p(\theta|h)$ as a Dirac delta distribution with the deterministic hypernetwork $\theta(h)$, where we have $p(y|\mathbf{x},h) = \int p(y|\mathbf{x},\theta)\delta(\theta-\theta(h))d\theta = p(y|\mathbf{x},\theta(h))$. Eventually, the variation distribution $q_\phi(h)$ is obtained by maximizing the ELBO *w.r.t.* ϕ as

$$\max_\phi \mathbb{E}_{q_\phi(h)}[\mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}} logp(y|\mathbf{x},\theta(h))] - KL(q_\phi(h)||p(h)), \tag{8}$$

where $q_\phi(h)$ and $\theta(h)$ work together as a probabilistic hypernetwork to reason hyperparameter uncertainties.

3.3 HyperU-GCN: Probabilistic Hypernetworks with Bilevel Optimization

In this study, we propose a hyperparameter uncertainty-induced GCN (HyperU-GCN) model by developing the probabilistic hypernetwork ($q_\phi(h)$, $\theta(h)$) introduced in Eq. (8). Specifically, the proposed HyperU-GCN instantiates $\theta(h)$ by using ST-GCN layers [28] in Eq. (3) and formulates the $q_\phi(h)$ as a Gaussian distribution by

$$q_\phi(h) = \mathcal{N}(h, \sigma_\phi(h)), \tag{9}$$

which treats the hyperparameter h as its mean value and employs ϕ as a stochastic hyperparameter encoder to learn the variance centering on each h. By explicitly modeling $q_\phi(h)$, we could obtain new training/validation loss in Eq. (1-2) by rewriting $\mathcal{L}(\phi(h),h)$ as

$$\mathcal{L}(\theta(h), q_\phi(h)) = \mathbb{E}_{q_\phi(h)}[\mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}} - \log p(y|\mathbf{x},\theta(h))]$$

To compute stochastic gradients for ϕ, we utilize the reparameterization trick [15, 16, 26]. Let $h' \sim q_\phi(h)$ be a random variable, then we can obtain it through $h' = h + \sigma_\phi \odot \epsilon$ where $\epsilon \sim \mathcal{N}(0,1)$.

Finally, to incorporate uncertainty into HPO, we formulate the proposed HyperU-GCN as a bilevel optimization problem:

$$\begin{aligned} &\min_{\phi,h} \mathcal{L}_\mathcal{V}(q_\phi(h),\theta(h)) + KL(q_\phi(h)||p(h)) \\ &\text{s.t. } \theta(h) = \arg\min_\theta \mathcal{L}_\mathcal{T}(\theta(h), q_\phi(h)), \end{aligned} \tag{10}$$

where the *upper-level* problem is equivalent to maximize the ELBO w.r.t $q_\phi(h)$ on the validation set, and the *lower-level* problem minimizes the hypernetwork $\theta(h)$ conditioning on $q_\phi(h)$ on the training set. Thus, the bilevel formulation in Eq. (10) unifies two problems – 1) hyperparameter uncertainty estimation and 2) hyperparameter optimization – within an automated graph learning framework.

Remark. The benefits of introducing the HyperU-GCN model lie in two folds. First, modeling hyperparameter uncertainty enables a similar way to apply Bayesian model averaging on GCN weights, which could calibrate model predictions and thus enhance the reliability. Second, training hypernetworks on a hyperparameter distribution usually leads to a more robust response function than training on a single h (like ST-GCN [44]), since $\theta(h)$ can better learn how to respond to different h spreading over $q_\phi(h)$.

We adopt an alternating optimization strategy to solve Eq. (10). During the training, the deterministic hypernetwork $\theta(h)$ will be first updated by calculating $\frac{\partial}{\partial\theta}\mathcal{L}_\mathcal{T}$ with h sampled from the fixed hyperparameter distribution $q_\phi(h)$. Then, the parameter of the stochastic encoder ϕ and hyperparameter h will be optimized with the most recent $\theta(h)$ by fixing all its weights, on the validation set. Specifically, we optimize h with hypergradient $\frac{\partial}{\partial h}\mathcal{L}_\mathcal{V}$, and update ϕ with the following chain rule by

$$\frac{\partial}{\partial\phi}\mathcal{L}_\mathcal{V}(q_\phi(h),\theta(h)) = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)}[\frac{\partial\mathcal{L}_\mathcal{V}}{\partial\theta(h)}\frac{\partial\theta(h)}{\partial h}\frac{\partial h}{\partial\phi} + \frac{\partial\mathcal{L}_\mathcal{V}}{\partial h}\frac{\partial h}{\partial\phi}].$$

To sum up, we train the proposed HyperU-GCN by updating $\theta(h)$ and $q_\phi(h)$ on training and validation sets, alternatively.

4 EXPERIMENTS

Dataset. The experiment is conducted on six benchmark datasets, including three citation network datasets [30], such as Cora, Citeseer, and Pubmed, and three public datasets provided in [32], namely Coauthor Physics, Amazon Computer, and Amazon Photo. For the citation datasets, we use 500 nodes for validation, 1000 nodes for testing, and the remaining nodes for training, as following [28, 44]. For the other three datasets, we randomly select validation nodes with the number as 30× of the # of classes. We use 1000 nodes for testing and the remaining nodes for training, as following [42].

Baselines. We compare HyperU-GCN with six GCN-based methods, including GCN [17], ST-GCN [44], Drop-GCN [10], CaGCN-st [36], GKDE-GCN [42]. We tuned all these methods with the recommended setting provided by their works. Two validation metrics,

Table 1: Comparison of different methods by test accuracy (Acc%) and the expected calibration error (ECE%)

Methods	Cora		Citeseer		Pubmed		Physics		Computers		Photo	
	Acc	ECE	Acc	ECE	Acc	ECE	Acc	ECE	Acc	ECE	Acc	ECE
GCN [17]	86.50	6.08	77.60	6.36	87.00	8.04	96.00	0.55	87.50	1.74	93.70	1.72
ST-GCN [44]	86.00	1.43	79.10	1.68	89.90	2.51	**96.70**	1.06	93.10	0.61	96.60	0.58
Drop-GCN [10]	86.40	4.97	78.30	5.49	87.00	7.65	95.90	0.63	87.30	1.78	93.90	1.80
CaGCN-st [36]	**87.60**	4.22	78.20	2.87	87.90	2.46	94.90	1.86	65.60	4.32	85.50	2.51
GKDE-GCN [42]	87.20	2.26	78.10	1.89	87.30	2.24	96.10	0.90	65.90	3.31	86.00	6.04
HyperU-GCN (Ours)	86.70	**0.53**	**79.40**	**1.22**	**90.20**	**0.78**	96.20	**0.46**	**93.50**	**0.30**	**97.10**	**0.41**

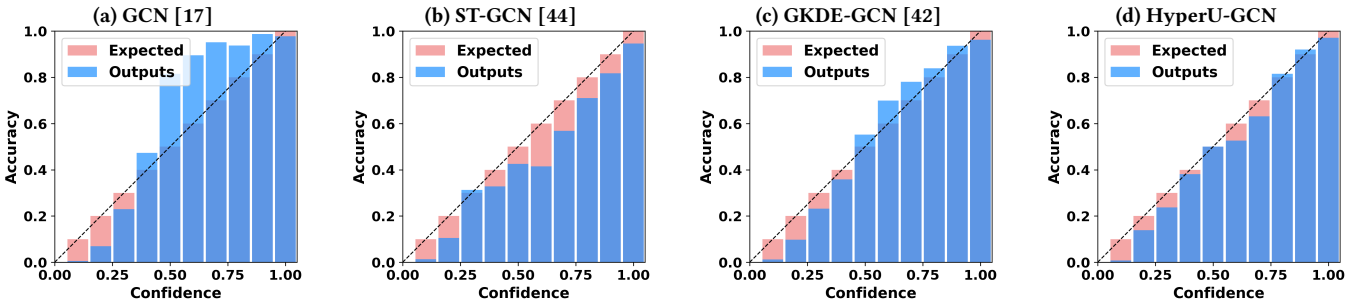


Figure 2: Reliability diagrams of different methods on the Cora dataset. A well-calibration tends to perfectly match model outputs and expected results along the diagonal line.

classification accuracy and expected calibration error (ECE) [12], were used for performance comparison in the experiment.

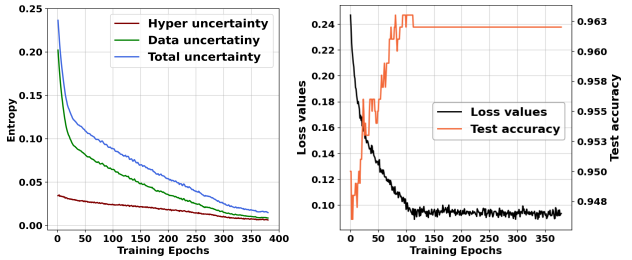
Implementation details. Following [28], we optimize five hyperparameters including three dropout rates in $[0, 0.9]$, one edge dropout rate [28] in $[0, 0.9]$, and one weight decay in $[10^{-6}, 10^{-2}]$. We implemented $\theta(h)$ as the ST-GCN [44] model with 4 layers, each of which has 128 hidden units, and developed $q_\phi(h)$ as a two-layer MLP network with the hidden dimension as five.

4.1 Calibration Performance

Table 1 summarizes the comparison results between different methods in node classification and model calibration by testing accuracy and ECE, respectively. As can be seen, the proposed HyperU-GCN reports better test accuracy among four out of six datasets and achieves the lowest ECE compared with the representative methods such as CaGCN-st [36]. The reliability diagrams in Fig. 2 also show that our method obtains a clear calibration improvement. This is mainly due to the benefits of training GCN weights over a dynamic hyperparameter distribution rather than a fixed configuration. On some datasets, *e.g.*, Cora and Physics, the classification accuracy of HyperU-GCN is slightly lower than the best performance. One possible reason might be the choice of using a single normal distribution cannot capture all the hyperparameter properties on different datasets, which could be improved through modeling the distribution as a mixture of models, *e.g.*, Gaussian mixture models.

4.2 Model Discussion

The proposed method approximates the hyperparameter posterior by learning a variational distribution $q_\phi(h)$, which could be used to estimate hyperparameter uncertainties based on Eq. (6). Fig. 3(a) shows estimation results during the training process. As can be seen, the data uncertainty dramatically drops at first few epochs.



(a) Uncertainty (b) Convergence

Figure 3: Uncertainty estimation and convergence analysis on the Physcis dataset.

The hyperparameter uncertainty curve tends to be stable when the training steps towards the end. This might be due to the nature of HPO in finding an optimal local region. On the other hand, Fig. 3(b) discusses the convergence of training probabilistic hypernetworks. Empirically, our method exhibits a good convergence behavior by training loss and testing accuracy at each epoch.

5 CONCLUSIONS

In this work, we studied the confidence calibration strategy of the automated graph learning model. The proposed solution is to trace the predictive uncertainty induced by hyperparameters for a predictive reliability enhancement. Using a bilevel optimization framework, the proposed HyperU-GCN not only focuses on node classification but also estimates hyperparameter uncertainties with a probabilistic hypernetwork. Specifically, we introduced a stochastic encoder to model the hyperparameter distribution and developed a deterministic hypernetwork to predict model weights over the learned distribution. Experimental results on six public datasets demonstrated that our approach achieves competitive node classification accuracy and state-of-the-art calibration performance.

REFERENCES

[1] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research* 13, 2 (2012).

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural network. In *International conference on machine learning*. PMLR, 1613–1622.

[3] 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.

[4] Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. BOHB: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*. PMLR, 1437–1446.

[5] Guosheng Feng, Chunnan Wang, and Hongzhi Wang. 2021. Search For Deep Graph Neural Networks. *arXiv preprint arXiv:2109.10047* (2021).

[6] Matthias Feurer and Frank Hutter. 2019. Hyperparameter optimization. In *Automated machine learning*. Springer, Cham, 3–33.

[7] Tiago M Fragoso, Wesley Bertoli, and Francisco Louzada. 2018. Bayesian model averaging: A systematic review and conceptual classification. *International Statistical Review* 86, 1 (2018), 1–28.

[8] Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. 2017. Forward and reverse gradient-based hyperparameter optimization. In *International Conference on Machine Learning*. PMLR, 1165–1173.

[9] Yarin Gal et al. 2016. Uncertainty in deep learning. (2016).

[10] 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.

[11] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. 2020. Graph Neural Architecture Search.. In *IJCAI*, Vol. 20. 1403–1409.

[12] 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.

[13] David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106* (2016).

[14] Matthew D Hoffman and Matthew J Johnson. 2016. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, Vol. 1.

[15] Durk P Kingma, Tim Salimans, and Max Welling. 2015. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems* 28 (2015).

[16] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

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

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

[19] Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. 2019. DeepGCNs: Can GCNs Go As Deep As CNNs?. In *2019 IEEE/CVF International Conference on Computer Vision*. IEEE, 9266–9275.

[20] Lisha Li, Kevin G Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2017. Hyperband: Bandit-based configuration evaluation for hyperparameter optimization. In *ICLR (Poster)*.

[21] Jonathan Lorraine and David Duvenaud. 2018. Stochastic hyperparameter optimization through hypernetworks. *arXiv preprint arXiv:1802.09419* (2018).

[22] David JC MacKay. 1992. A practical Bayesian framework for backpropagation networks. *Neural computation* 4, 3 (1992), 448–472.

[23] 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).

[24] Andrey Malinin and Mark Gales. 2018. Predictive uncertainty estimation via prior networks. *Advances in neural information processing systems* 31 (2018).

[25] Radford M Neal. 2012. *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media.

[26] Manfred Opper and Cédric Archambeau. 2009. The variational Gaussian approximation revisited. *Neural computation* 21, 3 (2009), 786–792.

[27] 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.

[28] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2019. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903* (2019).

[29] Jürgen Schmidhuber. 1992. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation* 4, 1 (1992), 131–139.

[30] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[31] Murat Sensoy, Lance Kaplan, and Melih Kandemir. 2018. Evidential deep learning to quantify classification uncertainty. *Advances in Neural Information Processing Systems* 31 (2018).

[32] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[33] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[34] Zhiqiang Tao, Yaliang Li, Bolin Ding, Ce Zhang, Jingren Zhou, and Yun Fu. 2020. Learning to Mutate with Hypergradient Guided Population. In *NeurIPS*.

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

[36] 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).

[37] Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2022. Automated Graph Machine Learning: Approaches, Libraries and Directions. *CoRR* abs/2201.01288 (2022). arXiv:2201.01288 https://arxiv.org/abs/2201.01288

[38] Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2022. Automated Graph Machine Learning: Approaches, Libraries and Directions. *arXiv preprint arXiv:2201.01288* (2022).

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

[40] 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.

[41] Huan Zhao, Lanning Wei, and Quanming Yao. 2020. Simplifying architecture search for graph neural network. *arXiv preprint arXiv:2008.11652* (2020).

[42] 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.

[43] Kaixiong Zhou, Qingquan Song, Xiao Huang, and Xia Hu. 2019. Auto-gnn: Neural architecture search of graph neural networks. *arXiv preprint arXiv:1909.03184* (2019).

[44] Ronghang Zhu, Zhiqiang Tao, Yaliang Li, and Sheng Li. 2021. Automated graph learning via population based self-tuning GCN. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2096–2100.