Appendix

This is the appendix of our work: **'GNNEvaluator: Evaluating GNN Performance On Unseen Graphs Without Labels'**. In this appendix, we present additional information regarding the *GNN model evaluation* problem and corresponding solutions, including in-depth discussions that distinguish our method from related works, dataset statistics used in our experiments, comprehensive details about our compared baseline methods, training details with the performance of well-trained GNN models on both the test set of the observed graph and the unseen test graphs (for ground-truth reference), and more experimental results with detailed analysis.

A Related Work

Predicting Model Generalization Error. Our work is relevant to the line of research on predicting model generalization error, which aims to develop a good estimator of a model's classifier performance on unlabeled data from the unknown distributions in the target domain, when the estimated models' classifier has been trained well in the source domain with labeled data [6, 9, 34, 5, 10, 4]. Typically, Guillory et al. [10] proposed to estimate a classifier's performance of convolutional neural network (CNN) models on image data based on a designed criterion, named difference of confidences (DoC), that estimates and reflects model accuracy. And Garg et al. [9] proposed to learn a score-based Average Thresholded Confidence (ATC) by leveraging the softmax probability of a CNN classifier, whose accuracy is estimated as the fraction of unlabeled images that receive a score above that threshold. In contrast, Deng et al. [6, 4] directly predicted CNN classifier accuracy by deriving distribution distance features between training and test images with a linear regression model.

However, these existing methods mostly focus on evaluating CNN model classifiers on image data in computer vision, and the formulation of evaluating GNNs for graph structural data still remains under-explored in graph machine learning. Concretely, conducting the model evaluation on GNNs for graph structural data has two critical challenges: (1) different from Euclidean image data, graph structural data lies in non-Euclidean space with complex and non-independent node-edge interconnections, so that its node contexts and structural characteristics significantly vary under wide-range graph data distributions, posing severer challenges for GNN model evaluation when serving on unknown graph data distributions. (2) GNNs have entirely different convolutional architectures from those of CNNs, when GNN convolution aggregates neighbor node messages along graph structures. Such that GNNs trained on the observed training graph might well fit its graph structure, and due to the complexity and diversity of graph structures, serving well-trained GNNs on unlabeled test distributions that they have not encountered before would incur more performance uncertainty.

Hence, in this work, we first investigate the GNN model evaluation problem for graph structural data with a clear problem definition and a feasible solution, taking a significant step towards understanding and evaluating GNN models for practical GNN deployment and serving. Our proposed two-stage GNN model evaluation framework directly estimates the node classification accuracy results of specific well-trained GNNs on practical unseen graphs without labels. Our method could not only facilitate model designers to better evaluate well-trained GNNs' performance in practical serving, but also provide users with confidence scores or model selection guidance, when using well-trained GNNs for inferring on their own test graphs.

Unsupervised Graph Domain Adaption. Our work is also relevant to the unsupervised graph domain adaption (UGDA) problem, whose goal is to develop a GNN model with both labeled source graphs and unlabeled target graphs for better generalization ability on target graphs. Typically, existing UGDA methods focus on mitigating the domain gap by aligning the source graph distribution with the target one. For instance, Yang et al. [33] and Shen et al. [27] optimized domain distance loss based on the statistical information of datasets, *e.g.*, maximum mean discrepancy (MMD) metric. Moreover, DANE [39] and UDAGCN [29] introduced domain adversarial methods to learn domain-invariant embeddings across the source domain and the target domain. Therefore, the critical distinction between our work and UGDA is that our work is primarily concerned with evaluating the GNNs' performance on unseen graphs without labels, rather than improving the GNNs' generalization ability when adapting to unlabeled target graphs. Besides, different from UGDA which uses the unlabeled target graphs in their model design and training stage, the GNN model evaluation problem explored by our work is not allowed to access the unlabeled test graphs, *i.e.*, unseen in the whole GNN model evaluation process.

Graph Out-of-distribution (OOD) Generalization & Detection. Out-of-distribution (OOD) generalization [18, 48] on graphs aims to develop a GNN model given several different but related source domains, so that the developed model can generalize well to unseen target domains. Li et al. [18] categorized existing graph OOD generalization methodologies into three conceptually different branches, *i.e.*, data, model, and learning strategy, based on their positions in the graph machine learning pipeline. We would like to highlight that, even ODD generalization and our proposed GNN model evaluation both pay attention to the general graph data distribution shift issue, we have different purposes: our proposed GNN model evaluation aims to evaluate well-trained GNNs' performance on unseen test graphs, while ODD generalization aims to develop a new GNN model for improve its performance or generalization capability on unseen test graphs. Moreover, OOD detection [32], aims to detect test samples drawn from a distribution that is different from the training distribution, with the definition of distribution to be well-defined according to the application in the target domain. Hence, the primary difference between OOD detection and our GNN model evaluation is, OOD detection works on detecting OOD test samples at the data level and our GNN model evaluation works on evaluating well-trained GNN's performance on ODD data at the model level.

B Dataset Statistics

We provide the details of dataset statistics used in our experiments in Table. A1, where these datasets meet the covariate shift assumption between any pair of test cases.

Table A1: Statistical details of the experimental real-world datasets.

Dataset	# of Nodes	# of Edges	# of Features	# of Labels
ACMv9	7410	11135	7537	6
Citationv2	4715	6717	7537	6
DBLPv8	5578	7341	7537	6

C Baseline Methods Details

Average Thresholded Confidence (ATC) & Its Variants. This metric [9] learns a threshold on CNN's confidence to estimate the accuracy as the fraction of unlabeled images whose confidence scores exceed the threshold as:

$$\mathrm{ATC} = \frac{1}{M}\sum_{j=1}^{M} \mathbf{1}\left\{ s\left(\mathrm{Softmax}\left(f_{\boldsymbol{\theta}^*}\left(\mathbf{x}'_j\right)\right)\right) < t\right\}, \tag{9}$$

where $f_{\boldsymbol{\theta}^*}(\cdot)$ denotes the well-trained CNN's classifier with the optimal parameter $\boldsymbol{\theta}^*$, and $s(\cdot)$ denotes the score function working on the softmax prediction of $f_{\boldsymbol{\theta}^*}(\cdot)$. When the context is clear, we will use $f(\cdot)$ for simplification. We adopted two different score functions, deriving two variants as: (1) Maximum confidence variant ATC-MC with $s(f(\mathbf{x}'_j)) = \max_{k\in\mathcal{Y}} f_k(\mathbf{x}'_j)$; and (2) Negative Entropy variant ATC-NE with $s(f(\mathbf{x}'_j)) = \sum_k f_k(\mathbf{x}'_j)\log\left(f_k(\mathbf{x}'_j)\right)$, where $\mathcal{Y} = \{1, 2, \ldots, C\}$ is the label space. And for t in Eq. (9), it is calculated based on the validation set of the observed training set $(\mathbf{x}_u^{\text{val}}, \mathbf{y}_u^{\text{val}}) \in \mathcal{S}_{\text{val}}$:

$$\frac{1}{N_{\text{val}}}\sum_{u=1}^{N_{\text{val}}} \mathbf{1}\left\{ s\left(\mathrm{Softmax}\left(f_{\boldsymbol{\theta}^*}\left(\mathbf{x}_u^{\text{val}}\right)\right)\right) < t\right\} = \frac{1}{N_{\text{val}}}\sum_{u=1}^{N_{\text{val}}} \mathbf{1}\left\{ p\left(\mathbf{x}_u^{\text{val}}; \boldsymbol{\theta}^*\right) \neq \mathbf{y}_u^{\text{val}}\right\}, \tag{10}$$

where $p(\cdot)$ denotes the predicted labels of samples. For all the calibration variants with '-c' in our experiments, they conduct standard Temperature Scaling [11] with following equations:

$$p_u = \max_k \mathrm{Softmax}\left(\mathbf{z}_u / T\right)^{(k)}, \tag{11}$$

Table A2: Hyperparameters of learning rate (lr) and weight decay (wd) in training different GNNs and MLP.

Datasets	ACMv9		Citationv2		DBLPv8	
Models	lr	wd	lr	wd	lr	wd
GCN	0.01	1.00E-05	0.01	1.00E-05	0.01	1.00E-05
SAGE	0.005	1.00E-06	0.005	1.00E-06	0.005	1.00E-06
GAT	0.005	1.00E-06	0.005	1.00E-06	0.005	1.00E-06
GIN	0.01	1.00E-06	0.01	1.00E-06	0.01	1.00E-06
MLP	0.001	1.00E-05	0.001	1.00E-05	0.001	1.00E-05

where $\mathbf{z}_u$ denotes the network output logits before softmax and T is the temperature scaling factor.

Threshold-based Method. This is an intuitive solution introduced by [6], which is not a learning-based method. It follows the basic assumption that a class prediction will likely be correct when it has a high softmax output score. Then, the threshold-based method would provide the estimated accuracy of a model as:

$$\mathrm{ACC} = \frac{\sum_{i=1}^{M} \mathbf{1}\left\{\max\left(f_{\boldsymbol{\theta}^*}(\mathbf{x}_j')\right) > \tau\right\}}{M}, \tag{12}$$

where τ is the pre-defined thresholds as $\tau = \{0.7, 0.8, 0.9\}$ on the output softmax logits of CNNs. This metric calculates the percentage of images in the entire dataset whose maximum entry of logits are greater than the threshold τ, which indicates these images are correctly classified.

AutoEval & Our Adaption AutoEval-G. This is a learning-based method of estimating classifier accuracy by utilizing dataset-level feature statistics [6]. AutoEval synthesizes a meta image dataset $\mathcal{D}$ (a dataset comprised of many datasets) from the observed training dataset $\mathcal{D}_{\text{ori}}$, and conducts the linear regression to learn the CNN classifier's accuracy based on the dataset-level feature statistics between the observed training image data and unseen real-world unlabeled image data. The accuracy linear regression with the dataset statistic feature f_{linear} is written as:

$$\begin{aligned} f_{\text{linear}} &= \mathrm{FD}\left(\mathcal{D}_{\text{ori}}, \mathcal{D}\right) = \|\boldsymbol{\mu}_{\text{ori}} - \boldsymbol{\mu}\|_2^2 + \mathrm{Tr}\left(\boldsymbol{\Sigma}_{\text{ori}} + \boldsymbol{\Sigma} - 2\left(\boldsymbol{\Sigma}_{\text{ori}}\boldsymbol{\Sigma}\right)^{\frac{1}{2}}\right), \\ \mathrm{ACC} &= w_1 f_{\text{linear}} + w_0 \end{aligned} \tag{13}$$

where FD is the Fréchet distance [7] to measure the domain gap with the mean feature vectors $\boldsymbol{\mu}_{\text{ori}}$ and $\boldsymbol{\mu}$, the covariance matrices $\boldsymbol{\Sigma}_{\text{ori}}$ and $\boldsymbol{\Sigma}$ of $\mathcal{D}_{\text{ori}}$ and $\mathcal{D}$, respectively. And $\mathrm{Tr}(\cdot)$ denotes the trace of the matrix, w_1 and w_0 denote the parameters of linear regression.

We make the following necessary adaptions to expand the AutoEval method to graph structural data on GNN model evaluation, deriving **AutoEval-G** by: (1) we synthesize a meta-graph set $\mathcal{G}_{\text{meta}}$ as demonstrated in Sec. 3.2 of the main manuscript; (2) we calculate the Maximum Mean Discrepancy (MMD) distance as AutoEval-G's graph dataset discrepancy representation $f_{\text{linear}}^{'}$, which measures the graph distribution gap between the observed training graph $\mathcal{S}$ and each meta-graph $g_{\text{meta}} \in \mathcal{G}_{\text{meta}}$ with their node embeddings from the well-trained $\mathrm{GNN}_{\mathcal{S}}$ as:

$$f_{\text{linear}}^{'} = \mathrm{MMD}(\mathcal{S}, g_{\text{meta}}) = \mathrm{MMD}(\mathbf{Z}_{\mathcal{S}}^*, \mathbf{Z}_{\text{meta}}^{(i,*)}). \tag{14}$$

where $\mathbf{Z}_{\mathcal{S}}^* = \mathrm{GNN}_{\mathcal{S}}^*(\mathbf{X}, \mathbf{A})$, and $\mathbf{Z}_{\text{meta}}^{(i,*)} = \mathrm{GNN}_{\mathcal{S}}^*(\mathbf{X}_{\text{meta}}^i, \mathbf{A}_{\text{meta}}^i)$.

D Well-trained GNN Model Details

In this section, we provide the details of well-trained GNN models in terms of their architecture parameters and training details.

For all GNN and MLP models, the default settings are : (a) the number of layers is 2; (b) the hidden feature dimension is 128; (c) the output feature dimension before the softmax operation is 16. The hyperparameters of training these GNNs and MLP are shown in Table A2. The five seeds for training each type of models (GCN, GraphSAGE, GAT, GIN, MLP) are $\{0, 1, 2, 3, 4\}$, and the node classification performance (accuracy) on the test set of each observed training graph and the ground-truth accuracy on the unseen test graphs without labels are shown in Table A3, Table A4, Table A5, Table A6, and Table A7, respectively.

Table A3: Node classification results in accuracy(%) of GCN with different seeds in terms of the test set of the observed graph and unseen test graphs.

GCN	Observed ACMv9			Observed Citationv2			Observed DBLPv8		
Seeds	A_{test}	GT A→C	GT A→D	C_{test}	GT C→A	GT C→D	D_{test}	GT D→A	GT D→C
0	83.13	45.51	63.88	88.98	46.86	60.18	97.58	70.81	57.65
1	83.74	49.01	68.09	88.45	40.76	61.60	98.30	68.43	57.90
2	82.73	55.08	69.29	88.45	45.51	60.95	98.57	70.04	53.36
3	83.40	48.89	69.85	89.09	46.09	60.58	98.57	68.11	52.85
4	83.87	55.63	68.75	88.24	37.25	58.62	98.66	70.61	58.37

Table A4: Node classification results in accuracy (%) of GraphSAGE with different seeds in terms of the test set of the observed graph and unseen test graphs.

SAGE	Observed ACMv9			Observed Citationv2			Observed DBLPv8		
Seeds	A_{test}	GT A→C	GT A→D	C_{test}	GT C→A	GT C→D	D_{test}	GT D→A	GT D→C
0	82.32	45.49	63.54	88.98	50.59	59.90	98.66	42.24	42.27
1	83.54	48.97	69.67	88.14	48.87	58.03	98.84	39.41	35.21
2	83.67	52.56	71.15	88.45	41.17	54.61	98.39	45.82	55.99
3	82.79	48.19	66.22	88.03	42.20	53.96	98.48	31.30	27.25
4	82.05	47.04	66.67	88.35	37.46	55.70	98.84	36.99	35.14

Table A5: Node classification results in accuracy (%) of GAT with different seeds in terms of the test set of the observed graph and unseen test graphs.

GAT	Observed ACMv9			Observed Citationv2			Observed DBLPv8		
Seeds	A_{test}	GT A→C	GT A→D	C_{test}	GT C→A	GT C→D	D_{test}	GT D→A	GT D→C
0	83.00	44.84	72.53	87.82	41.55	57.94	97.85	64.10	51.77
1	82.73	44.18	57.08	88.67	48.80	60.40	98.84	57.03	42.18
2	81.65	45.07	69.59	88.77	45.74	59.04	98.48	39.65	63.99
3	82.79	43.82	60.18	88.98	41.48	61.03	98.93	62.33	58.43
4	81.71	42.04	57.62	87.50	37.04	60.06	98.30	63.64	52.64

Table A6: Node classification results in accuracy (%) of GIN with different seeds in terms of the test set of the observed graph and unseen test graphs.

GIN	Observed ACMv9			Observed Citationv2			Observed DBLPv8		
Seeds	A_{test}	GT A→C	GT A→D	C_{test}	GT C→A	GT C→D	D_{test}	GT D→A	GT D→C
0	82.19	50.37	74.94	88.24	33.35	59.47	98.48	35.13	40.78
1	82.52	43.12	51.58	86.97	35.40	64.83	66.70	35.37	28.14
2	82.86	42.93	58.21	81.57	39.28	65.04	98.93	53.91	30.56
3	82.46	47.93	69.79	87.92	44.59	69.72	63.47	30.61	21.53
4	82.39	47.04	78.13	88.56	40.23	65.72	98.93	59.39	45.56

Table A7: Node classification results in accuracy (%) of MLP with different seeds in terms of the test set of the observed graph and unseen test graphs.

MLP	Observed ACMv9			Observed Citationv2			Observed DBLPv8		
Seeds	A_{test}	GT A→C	GT A→D	C_{test}	GT C→A	GT C→D	D_{test}	GT D→A	GT D→C
0	71.59	33.28	38.17	74.89	34.48	37.25	62.67	59.18	51.41
1	70.18	33.47	39.05	74.58	35.30	38.62	63.12	54.44	43.92
2	70.11	42.65	43.67	74.36	34.74	39.12	63.12	53.74	42.59
3	68.62	35.12	39.71	72.56	33.91	37.41	62.94	54.28	44.31
4	69.23	35.95	42.33	74.58	34.41	39.82	64.37	54.08	43.88

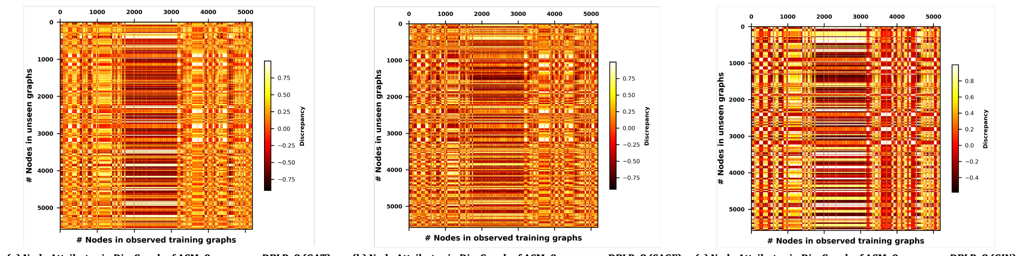
E More Experimental Analysis and Results.

E.1 DiscGraph Statistics

As a vital component of our proposed two-stage GNN model evaluation framework, DiscGraph set captures wide-range and diverse graph data distribution discrepancies. By fully exploiting the latent node embeddings and node class predictions from well-trained GNNs, the DiscGraph set involves representative node attributes incorporating diverse discrepancies through a discrepancy measurement function, leading to effective instructions for the training of `GNNEvaluator`. Hence, in the following, we show the statistical information of our derived DiscGraph set in Table A8. Note that, each generated DiscGraph set is related to a specific dataset and a specific well-trained GNN, as it considers the outputs of node embeddings and predictions on a well-trained GNN trained by a specific graph dataset.

Table A8: Statistics of the proposed DiscGraph in terms of GNNs on ACMv9, Citationv2, DBLPv8, where the accuracy label distributions calculate average minimum and maximum node classification accuracy on GNNs with five random seeds with standard deviations (std).

Statistics		DiscGraph-ACMv9	DiscGraph-Citationv2	DiscGraph-DBLPv8
#Num of Graphs (K)		400	400	400
#Avg. Num of Nodes		2105	1340	1585
#Avg. Num of Edges		1595	927	1205
Accuracy Label Distributions (Min. | Max.$_{\pm\text{std}}$)	GCN	39.95$_{\pm0.00}$ | 80.45$_{\pm1.40}$	13.89$_{\pm7.82}$ | 85.36$_{\pm1.00}$	17.53$_{\pm0.57}$ | 91.95$_{\pm0.56}$
	SAGE	39.95$_{\pm0.00}$ | 77.76$_{\pm2.22}$	21.29$_{\pm7.28}$ | 82.43$_{\pm0.54}$	19.07$_{\pm3.52}$ | 77.27$_{\pm2.03}$
	GAT	39.50$_{\pm1.01}$ | 79.67$_{\pm0.63}$	19.53$_{\pm12.27}$ | 86.13$_{\pm0.47}$	19.58$_{\pm1.34}$ | 87.26$_{\pm3.99}$
	GIN	28.50$_{\pm10.15}$ | 76.06$_{\pm2.27}$	13.48$_{\pm8.24}$ | 81.50$_{\pm2.91}$	19.06$_{\pm5.18}$ | 63.63$_{\pm16.83}$
	MLP	39.92$_{\pm0.04}$ | 71.91$_{\pm1.19}$	33.65$_{\pm7.08}$ | 77.41$_{\pm0.75}$	17.71$_{\pm0.39}$ | 64.68$_{\pm0.41}$



(a) Node Attributes in DiscGraph of ACMv9⟶ unseen DBLPv8 (GAT) (b) Node Attributes in DiscGraph of ACMv9⟶ unseen DBLPv8 (SAGE) (c) Node Attributes in DiscGraph of ACMv9⟶ unseen DBLPv8 (GIN)

Figure A1: Visualizations on discrepancy node attributes in the proposed DiscGraph set for (a) GAT, (b) GraphSAGE, and (c) GIN model evaluation. Darker color denotes a larger discrepancy.

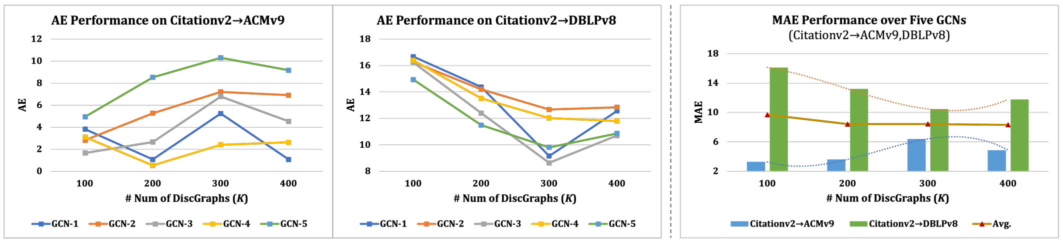


Figure A2: Effects of different numbers of DiscGraphs (K) for `GNNEvaluator` training with Absolute Error (AE) for each GCN model under C→A and C→D cases.

E.2 More Visualization Results of Discrepancy Node Attributes

In Fig. A1, we present more visualization results on discrepancy node attributes in the proposed DiscGraph set for different GNN models, *i.e.*, (a) GAT, (b) GraphSAGE, and (c) GIN, under ACMv9$\rightarrow$DBLPv8 case.

As can be observed, for different GNNs, the node attributes in the proposed DiscGraph set show significant differences, denoting the effectiveness of the proposed discrepancy measurement function, which could capture model-specific discrepancy representations effectively. And such discrepancy representations could instruct our proposed `GNNEvaluator` to learn graph data distribution discrepancies with different well-trained GNNs for accuracy regression.

E.3 More Results on The Number of DiscGraphs

We provide more results for illustrating the effects of different numbers of discGraphs for training the proposed `GNNEvaluator` in Fig. A2. Along with the MAE results reported in Fig. 5 in the main manuscript, we report the Absolute Error (AE) results for each GCN model under C$\rightarrow$A and C$\rightarrow$D cases. Lower AE denotes better performance. It can be observed that different GCN models have different appropriate K values for `GNNEvaluator` training, but they show similar trends for each unseen test graph in the left and the middle figures. For instance, the proposed `GNNEvaluator` has highest AE when it is trained with $K = 100$ DiscGraphs, and comparable AEs with $K = 300$ and $K = 400$ for evaluating all five GCNs under the C$\rightarrow$D case. Nevertheless, in general, as shown in the red 'Avg.' line of the right figure, the performance of the proposed `GNNEvaluator` would not significantly vary along the number of DiscGraphs when evaluating all GCNs that are well-trained on Citationv2 dataset and served on unseen DBLPv8 dataset.