A Deferred details for Section 3

A.1 Technical proofs for theoretical results

Proof of Lemma 2. Hereafter, we condition on the entire unordered graph, all the attribute and label information, and the index sets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{ct}}$. We define the scores evaluated at the original node indices as

$$v_v = V(\mathbf{x}_v, y_v; \{z_i\}_{i\in\mathcal{D}_{\text{train}}\cup\mathcal{D}_{\text{valid}}}, \{\mathbf{x}_v\}_{v\in\mathcal{D}_{\text{calib}}\cup\mathcal{D}_{\text{test}}}, \mathcal{V}, \mathcal{E}), \quad v \in \mathcal{D}_{\text{calib}} \cup \mathcal{D}_{\text{test}} \subseteq \mathcal{V}.$$

By Assumption 1, for any permutation π of $\mathcal{D}_{\text{ct}}$, we always have

$$v_v = V(\mathbf{x}_v, y_v; \{z_i\}_{i\in\mathcal{D}_{\text{train}}\cup\mathcal{D}_{\text{valid}}}, \{\mathbf{x}_{\pi(v)}\}_{v\in\mathcal{D}_{\text{calib}}\cup\mathcal{D}_{\text{test}}}, \mathcal{V}_\pi, \mathcal{E}_\pi).$$

That is, given $\mathcal{D}_{\text{ct}}$, the evaluated score at any $v \in \mathcal{D}_{\text{ct}}$ remains invariant no matter which subset of $\mathcal{D}_{\text{ct}}$ is designated as $\mathcal{D}_{\text{calib}}$. This implies that the scores are fixed after conditioning:

$$[V_1, \ldots, V_{n+m}] = [v_v]_{v\in\mathcal{D}_{\text{ct}}},$$

where we use [] to emphasize unordered sets. Thus, the calibration scores $\{V_i\}_{i=1}^n$ is a subset of size n of $[v_v]_{v\in\mathcal{D}_{\text{ct}}}$. Note that under random splitting in the transductive setting, any permutation π of $\mathcal{D}_{\text{ct}}$ occurs with the same probability, which gives the conditional probability in Lemma 2. □

Proof of Theorem 3. Throughout this proof, we condition on the entire unordered graph, all the attribute and label information, and the index sets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{ct}}$. By Lemma 2, after the conditioning, the unordered set of $\{V_i\}_{i=1}^{n+m}$ is fixed as $[v_v]_{v\in\mathcal{D}_{\text{ct}}}$, and $\{V_i\}_{i=1}^n$ is a simple random sample from $[v_v]_{v\in\mathcal{D}_{\text{ct}}}$. As a result, any test sample $V(X_{n+j}, Y_{n+j})$, $j = 1, \ldots, m$ is exchangeable with $\{V_i\}_{i=1}^n$. By standard theory for conformal prediction [43], this ensures *valid marginal coverage*, i.e., $\mathbb{P}(Y_{n+j} \in \widehat{C}(X_{n+j})) \geq 1 - \alpha$, where the expectation is over all the randomness.

We now proceed to analyze the distribution of $\widehat{\text{Cover}}$. For notational convenience, we write $N = m + n$, and view $\mathcal{D}_{\text{ct}}$ as the 'population'. In this way, $\{V_i\}_{i=1}^n$ is a simple random sample from $[v_v]_{v\in\mathcal{D}_{\text{ct}}}$. For every $\eta \in \mathbb{R}$, we define the 'population' cumulative distribution function (c.d.f.)

$$F(\eta) = \frac{1}{N}\sum_{v\in\mathcal{D}_{\text{ct}}} \mathbb{1}\{v_v \leq \eta\},$$

which is a deterministic function. We also define the calibration c.d.f. as

$$\widehat{F}_n(\eta) = \frac{1}{n}\sum_{v\in\mathcal{D}_{\text{calib}}} \mathbb{1}\{v_v \leq \eta\} = \frac{1}{n}\sum_{i=1}^n \mathbb{1}\{V_i \leq \eta\},$$

which is random, and its randomness comes from which subset of $\mathcal{D}_{\text{ct}}$ is $\mathcal{D}_{\text{calib}}$. By definition,

$$\widehat{\eta} = \inf\{\eta \colon \widehat{F}_n(\eta) \geq (1-q)(1+1/n)\}.$$

Since the scores have no ties, we know

$$\widehat{F}_n(\widehat{\eta}) = \lceil (1-q)(n+1)\rceil/n.$$

The test-time coverage can be written as

$$\begin{aligned}
\widehat{\text{Cover}} &= \frac{1}{m}\sum_{j=1}^m \mathbb{1}\{V_{n+j} \leq \widehat{\eta}\} \\
&= \frac{1}{N-n}\Big(\sum_{v\in\mathcal{D}_{\text{ct}}} \mathbb{1}\{v_v \leq \widehat{\eta}) - \sum_{v\in\mathcal{D}_{\text{calib}}} \mathbb{1}\{v_v \leq \widehat{\eta})\Big) \\
&= \frac{N}{N-n}F(\widehat{\eta}) - \frac{n}{N-n}\widehat{F}_n(\widehat{\eta}) = \frac{N}{N-n}F(\widehat{\eta}) - \frac{\lceil (1-q)(n+1)\rceil}{N-n}.
\end{aligned}$$

Now we characterize the distribution of $\widehat{\eta}$. For any $\eta \in \mathbb{R}$, by the definition of $\widehat{\eta}$,

$$\mathbb{P}(\widehat{\eta} \leq \eta) = \mathbb{P}\big(n\widehat{F}_n(\eta) \geq (n+1)(1-q)\big) = \mathbb{P}\big(n\widehat{F}_n(\eta) \geq \lceil (n+1)(1-q)\rceil\big).$$

Note that $n\widehat{F}_n(\eta) = \sum_{v\in\mathcal{D}_{\text{calib}}} \mathbb{1}\{v_v \le \eta\}$ is the count of data in $\mathcal{D}_{\text{calib}}$ such that the score is below η. By the simple random sample (i.e., sampling without replacement), $n\widehat{F}_n(\eta)$ follows a hyper-geometric distribution with parameter $N, n, NF(\eta)$. That is,

$$\mathbb{P}(n\widehat{F}_n(\eta) = k) = \frac{\binom{NF(\eta)}{k}\binom{N-NF(\eta)}{n-k}}{\binom{N}{n}}, \quad 0 \le k \le NF(\eta).$$

Denoting the c.d.f. of hypergeometric distribution as $\Phi_{\text{HG}}(k; N, n, NF(\eta))$, we have

$$\mathbb{P}(\widehat{\eta} \le \eta) = 1 - \Phi_{\text{HG}}\big(\lceil (n+1)(1-q)\rceil - 1; N, n, NF(\eta)\big).$$

Then, for any $t \in [0,1]$,

$$\begin{aligned}\mathbb{P}\big(\widehat{\text{Cover}} \le t\big) &= \mathbb{P}\left(\frac{N}{N-n}F(\widehat{\eta}) - \frac{\lceil (1-q)(n+1)\rceil}{N-n} \le t\right) \\ &= \mathbb{P}\left(F(\widehat{\eta}) \le \frac{\lceil (1-q)(n+1)\rceil + (N-n)t}{N}\right).\end{aligned}$$

Since $F(\cdot)$ is monotonely increasing,

$$\mathbb{P}\big(\widehat{\text{Cover}} \le t\big) = \mathbb{P}\left(\widehat{\eta} \le F^{-1}\left(\frac{\lceil (1-q)(n+1)\rceil + (N-n)t}{N}\right)\right),$$

where $F^{-1}(s) = \inf\{\eta\colon F(\eta) \ge s\}$ for any $s \in [0,1]$. Plugging in the previous results on the distribution of $\widehat{\eta}$, we have

$$\begin{aligned}\mathbb{P}\big(\widehat{\text{Cover}} \le t\big) &= 1 - \Phi_{\text{HG}}\left(\lceil (n+1)(1-q)\rceil - 1; N, n, NF\left(F^{-1}\big(\tfrac{\lceil (1-q)(n+1)\rceil + (N-n)t}{N}\big)\right)\right) \\ &= 1 - \Phi_{\text{HG}}\left(\lceil (n+1)(1-q)\rceil - 1; N, n, N\tfrac{\lceil\lceil (1-q)(n+1)\rceil + (N-n)t\rceil}{N}\right) \\ &= 1 - \Phi_{\text{HG}}\left(\lceil (n+1)(1-q)\rceil - 1; N, n, \lceil\lceil (1-q)(n+1)\rceil + (N-n)t\rceil\right) \\ &= 1 - \Phi_{\text{HG}}\left(\lceil (n+1)(1-q)\rceil - 1; N, n, \lceil (1-q)(n+1)\rceil + \lceil (N-n)t\rceil\right)\end{aligned}$$

where the second equality uses the fact that $F(\eta) \in \{0, 1/N, \ldots, (N-1)/N, 1\}$, hence $F(F^{-1}(s)) = \lceil Ns\rceil/N$ for any $s \in [0,1]$. By tower property, such an equation also holds for the unconditional distribution, marginalized over all the randomness. This completes the proof of Theorem 3. □

A.2 Additional visualization of test-time coverage

In this part, we provide more visualization of the distributions of test time coverage $\widehat{\text{Cover}}$ under various sample size configurations. We note that such results also apply to standard application of split conformal prediction when the non-conformity score function V is independent of calibration and test samples, so that Assumption 1 is satisfied.

Figures 6 and 7 plot the p.d.f. of $\widehat{\text{Cover}}$ for $\alpha = 0.05$ and $\alpha = 0.1$, respectively, when fixing n and varying the test sample size m. The y-axis is obtained by computing $\mathbb{P}(t_{k-1} < \widehat{\text{Cover}} \le t_k)/(t_k - t_{k-1})$ at $x = (t_{k-1} + t_k)/2$ for a sequence of evenly spaced $\{t_k\} \in [0,1]$. All figures in this paper for p.d.f.s are obtained in the same way. We see that $\widehat{\text{Cover}}$ concentrates more tightly around the target value $1-\alpha$ as m and n increases.

Figures 8 and 9 plot the p.d.f. of $\widehat{\text{Cover}}$ for $\alpha = 0.05$ and $\alpha = 0.1$, respectively, where we fix $N = m + n$ but vary the calibration sample size n. This mimics the situation where the total number of nodes on the graph is fixed, while we may have flexibility in collecting data as the calibration set. We observe a tradeoff between the calibration accuracy determined by n and the test-sample concentration determined by n. The distribution of $\widehat{\text{Cover}}$ is more concentrated around $1-\alpha$ when m and n are relatively balanced.

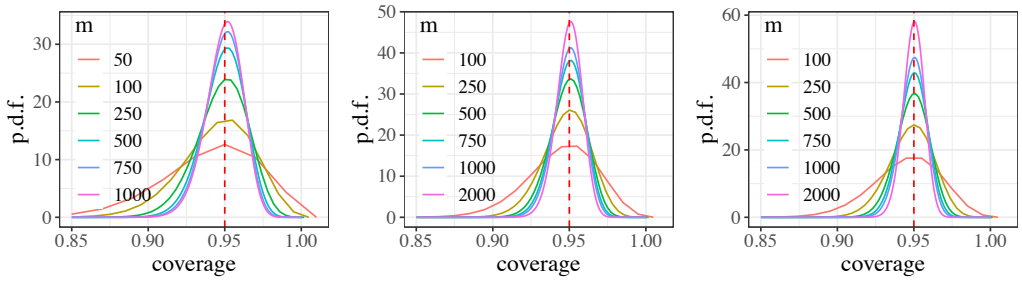

Figure 6: P.d.f. of test-time coverage $\widehat{\text{Cover}}$ for $n = 500$ (left), 1000 (middle), 2000 (right) and $\alpha = 0.05$ with curves representing different values of m, the test sample size.

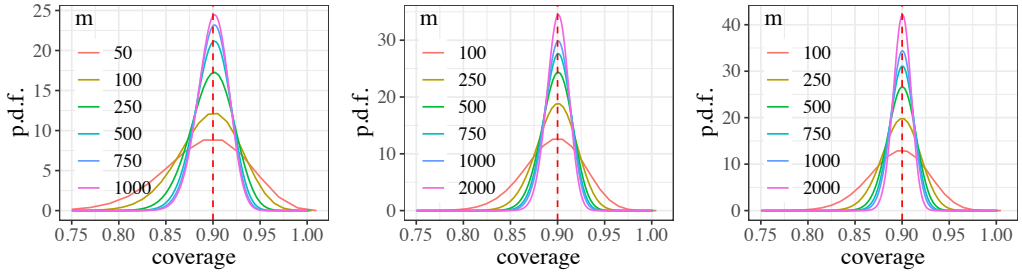

Figure 7: P.d.f. of test-time coverage $\widehat{\text{Cover}}$ for $n = 500$ (left), 1000 (middle), 2000 (right) and $\alpha = 0.1$ with curves representing different values of m, the test sample size.

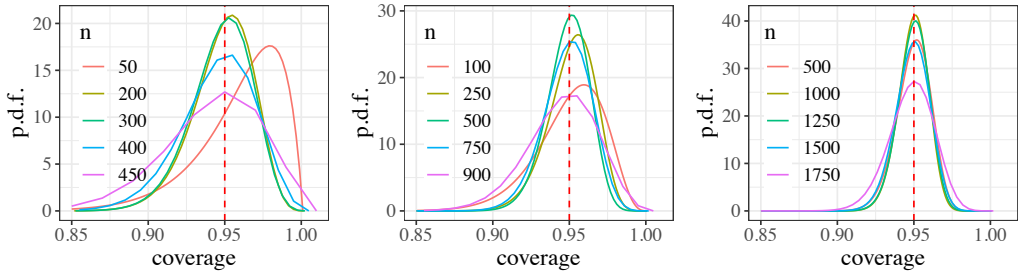

Figure 8: P.d.f. of test-time coverage $\widehat{\text{Cover}}$ for $N = m + n = 500$ (left), 1000 (middle), 2000 (right) and $\alpha = 0.05$ with curves representing different values of n, the calibration sample size.

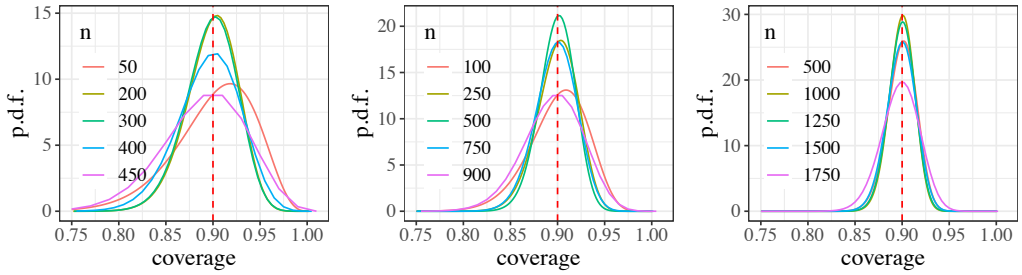

Figure 9: P.d.f. of test-time coverage $\widehat{\text{Cover}}$ for $N = m + n = 500$ (left), 1000 (middle), 2000 (right) and $\alpha = 0.1$ with curves representing different values of n, the calibration sample size.

B Discussion on full conformal prediction, split conformal prediction

In this part, we discuss the relation of our application of conformal prediction to full conformal prediction and split conformal prediction, two prominent conformal prediction methods proposed by Vovk and his coauthors in [43]. Split conformal prediction is mostly widely used due to its computational efficiency, where exchangeability is usually ensured by independence (which is not obvious for graph data) as we discussed briefly in the introduction.

Full conformal prediction (FCP) is arguably the most versatile form of conformal prediction. Given calibration data $Z_i = (X_i, Y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \ldots, n$, and given a test point $X_{n+1} \in \mathcal{X}$ whose outcome $Y_{n+1} \in \mathcal{Y}$ is unknown, at every hypothesized value $y \in \mathcal{Y}$, FCP uses any algorithm S to train the following scores

$$S_i^y = S(X_i, Y_i; Z_1, \ldots, Z_{i-1}, Z_{i+1}, \ldots, Z_n, (X_{n+1}, y)), \quad i = 1, \ldots, n,$$

where S is symmetric in the arguments $Z_1, \ldots, Z_{i-1}, Z_{i+1}, \ldots, Z_n, (X_{n+1}, y)$, as well as

$$S_{n+1}^y = S(X_{n+1}, y; Z_1, \ldots, Z_n).$$

Here, for $1 \le i \le n$, S_i^y intuitively measures how well the observation (X_i, Y_i) conforms to the observations $Z_1, \ldots, Z_{i-1}, Z_{i+1}, \ldots, Z_n, (X_{n+1}, y)$ with the hypothesized value of y. For instance, when using a linear prediction model, it can be chosen as the prediction residual

$$S_i^y = |Y_i - X_i^\top \widehat{\theta}^y|,$$

where $\widehat{\theta}^y$ is the ordinary least squares coefficient by a linear regression of $Y_1, \ldots, Y_{i-1}, Y_{i+1}, \ldots, Y_n, y$ over $X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n, X_{n+1}$. More generally, one may train a prediction model $\widehat{\mu}^y \colon \mathcal{X} \to \mathcal{Y}$ using $Z_1, \ldots, Z_{i-1}, Z_{i+1}, \ldots, Z_n, (X_{n+1}, y)$, and set $S_i^y = |Y_i - \widehat{\mu}^y(X_i)|$. For a confidence level $\alpha \in (0, 1)$, the FCP prediction set is then

$$\widehat{C}(X_{n+1}) := \Big\{ y \colon \tfrac{1 + \mathbb{1}\{S_i^y > S_{n+1}^y\}}{n+1} \le \alpha \Big\}.$$

Since the original form of FCP involves training $n + 1$ models at each hypothesized value y, its computation can be very intense. It is thus impractical to directly apply FCP to GNN models (i.e., imagining S as the GNN training process on the entire graph with a hypothesized outcome y).

Split conformal prediction (SCP) is a computationally-efficient special case of FCP that is most widely used for i.i.d. data. The idea is to set aside an independent fold of data to output a single trained model. To be specific, we assume access to a given non-conformity score $V \colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, i.i.d. calibration data $Z_i = (X_i, Y_i)_{i=1}^n$, and an independent test sample (X_{n+1}, Y_{n+1}) from the same distribution with Y_{n+1} unobserved. Here by a "given" score, we mean that it is obtained without knowing the calibration and test sample; usually, it is trained on an independent set of data $\{(X_j, Y_j)\}_{j \in \mathcal{D}_{\text{train}}}$ before seeing the calibration and test sample. Then define $V_i = V(X_i, Y_i)$ for $i = 1, \ldots, n$. The SCP prediction set is

$$\widehat{C}(X_{n+1}) = \Big\{ y \colon \tfrac{1 + \mathbb{1}\{V_i > V(X_{n+1}, y)\}}{n+1} \le \alpha \Big\}.$$

The above set is usually convenient to compute, because we only need one single model to obtain V. The validity of SCP usually relies on the independence of V to calibration and test data as we mentioned in the introduction. However, the application of SCP to GNN model is also not straightforward: as we discussed in the main text, the model training step already uses the calibration and test samples, and the nodes are correlated.

Indeed, our method can be seen as a middle ground between FCP and SCP: it only requires one single prediction model as SCP does, but allows to use calibration and test data in the training step as FCP does. In our method introduced in the main text, there exists a fixed function $V \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ (provided by APS and CQR) such that

$$S_i^y = V(\widehat{\mu}(X_i), Y_i), \quad S_{n+1}^y = V(\widehat{\mu}(X_{n+1}, y),$$

where $\widehat{\mu}$ is the final output from the second GNN model whose training process does not utilize the outcomes $Y_1, \ldots, Y_n$ and y, but uses the features $X_1, \ldots, X_n$ and X_{n+1}.

Algorithm 1: Pseudo-code for CF-GNN algorithm.

Input: Graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$; a trained base GNN model GNN_θ; non-conformity score function $V(X, Y)$; pre-specified mis-coverage rate α, Randomly initialized ϑ for the conformal correction model GNN_ϑ.

while *not done* **do**

  for *i in {1, ..., $|\mathcal{V}_{\text{cor-calib}} \cup \mathcal{V}_{\text{cor-test}}|$}* **do**

    $\widehat{\mu}(X_i) = \text{GNN}_\theta(\mathbf{X_i}, G)$ `// Base GNN output scores`

    $\tilde{\mu}(X_i) = \text{GNN}_\vartheta(\widehat{\mu}(X_i), G)$ `// Correction model output scores`

  end

  $n, m = |\mathcal{V}_{\text{cor-calib}}|, |\mathcal{V}_{\text{cor-test}}|$ `// Size of correction calib/test set`

  $\widehat{\alpha} = \frac{1}{n+1} * \alpha$ `// Finite-sample correction`

  $\widehat{\eta} = \text{DiffQuantile}(\{V(X_i, Y_i) | i \in \mathcal{V}_{\text{cor-calib}}\})$ `// Compute non-conformity scores`

  if Classification **then**

    $\mathcal{L}_{\text{Ineff}} = \frac{1}{m} \sum_{i \in \mathcal{V}_{\text{cor-test}}} \frac{1}{|\mathcal{Y}|} \sum_{k \in \mathcal{Y}} \sigma(\frac{V(X_i, k) - \widehat{\eta}}{\tau})$

    `// Inefficiency proxy for classification tasks`

  end

  if Regression **then**

    $\mathcal{L}_{\text{Ineff}} = \frac{1}{m} \sum_{i \in \mathcal{V}_{\text{cor-test}}} (\tilde{\mu}_{1-\alpha/2}(X)_i + \widehat{\eta}) - (\tilde{\mu}_{\alpha/2}(X)_i - \widehat{\eta})$

    `// Inefficiency proxy for regression tasks`

    $\mathcal{L}_{\text{Ineff}} += \gamma \frac{1}{m} \sum_{i \in \mathcal{V}_{\text{cor-test}}} (\tilde{\mu}_{1-\alpha/2}(X)_i - \widehat{\mu}_{1-\alpha/2}(X)_i)^2 + (\tilde{\mu}_{\alpha/2}(X)_i - \widehat{\mu}_{\alpha/2}(X)_i)^2$

    `// Consistency regularization term`

  end

  $\vartheta \leftarrow \vartheta - \nabla_\vartheta \mathcal{L}_{\text{Ineff}}$ `// Optimizing` ϑ `to reduce inefficiency`

end

C Algorithm overview

We describe the pseudo-code of CF-GNN in Algorithm 1.

D Deferred details for experiments

D.1 Hyperparameters

Table 5 reports our set of hyperparameter ranges. We conduct 100 iterations of Bayesian Optimization for CF-GNN with the validation set inefficiency proxy as the optimization metric. To avoid overfitting, each iteration only uses the first GNN run. The optimized hyperparameters are then used for all 10 GNN runs and we then reported the average and standard deviation across runs. Each experiment is done with a single NVIDIA 2080 Ti RTX 11GB GPU.

Table 5: Hyperparameter range for CF-GNN.

Task	Param.	Range
Classification	GNN_ϑ Hidden dimension	[16,32,64,128,256]
	Learning rate	[1e-1, 1e-2, 1e-3, 1e-4]
	GNN_ϑ Number of GNN Layers	[1,2,3,4]
	GNN_ϑ Base Model	[GCN, GAT, GraphSAGE, SGC]
	τ	[10, 1, 1e-1, 1e-2, 1e-3]
Regression	GNN_ϑ Hidden dimension	[16,32,64,128,256]
	Learning rate	[1e-1, 1e-2, 1e-3, 1e-4]
	GNN_ϑ Number of GNN Layers	[1, 2, 3, 4]
	GNN_ϑ Base Model	[GCN, GAT, GraphSAGE, SGC]
	Reg. loss coeff. γ	[1, 1e-1]

D.2 Baseline Details

We report the details about baselines below and the hyperparameter range in Table 6.

1. Temperature Scaling [13] divides the logits with a learnable scalar. It is optimized over NLL loss in the validation set.
2. Vector Scaling [13] has a scalar to scale the logits for each class dimension and adds an additional classwide bias. It is optimized over NLL loss in the validation set.
3. Ensemble Temperature Scaling [49] learns an ensemble of uncalibrated, temperature-scaled calibrated calibrators.
4. CaGCN [44] uses an additional GCN model that learns a temperature scalar for each node based on its neighborhood information.
5. GATS [17] identifies five factors that affect GNN calibration and designs a model that accounts for these five factors by using per-node temperature scaling and attentive aggregation from the local neighborhood.
6. QR [25] uses a pinball loss to produce quantile scores. It is CQR without the conformal prediction adjustment.
7. MC dropout [9] turns on dropout during evaluation and produces K predictions. We then take the 95% quantile of the predicted distribution. We also experimented with taking a 95% confidence interval but 95% quantile has better coverage, thus we adopt the quantile approach.
8. BayesianNN [23] model the label with normal distribution and the model produces two heads, where one corresponds to the mean and the second log variance. We then calculate the standard deviation as the square root of the exponent of log variance. Then we take the [mean-1.96*standard deviation, mean+1.96*standard deviation] for the 95% interval.

Table 6: Hyperparameter range for baselines.

Baseline	Param.	Range
Temperature Scaling	No hyperparameter	Not Applicable
Vector Scaling	No hyperparameter	Not Applicable
Ensemble Temp Scaling	No hyperparameter	Not Applicable
CaGCN	Dropout Hidden dimension Number of GNN Layers Weight Decay	[0.3, 0.5, 0.7] [16, 32, 64, 128, 256] [1,2,3,4] [0, 1e-3, 1e-2, 1e-1]
GATS	Dropout Hidden dimension Number of GNN Layers Weight Decay	[0.3, 0.5, 0.7] [16, 32, 64, 128, 256] [1,2,3,4] [0, 1e-3, 1e-2, 1e-1]
MC Dropout	Number of Predictions	[100, 500, 1,000]
BayesianNN	No hyperparameter	Not Applicable

D.3 Dataset

For node classification, we use the common node classification datasets in Pytorch Geometric package. For node regression, we use datasets in [20]. We report the dataset statistics at Table 7.

D.4 Marginal coverage and inefficiency across GNN architectures

We additionally conduct marginal coverage and inefficiency comparisons of CF-GNN over the vanilla CP across 4 different GNN architectures: GCN, GAT, GraphSAGE, and SGC. The result for marginal

Table 7: Dataset statistics.

Domain	Dataset	Task	# Nodes	# Edges	# Features	# Labels
Citation	Cora	Classification	2,995	16,346	2,879	7
	DBLP	Classification	17,716	105,734	1,639	4
	CiteSeer	Classification	4,230	10,674	602	6
	PubMed	Classification	19,717	88,648	500	3
Co-purchase	Computers	Classification	13,752	491,722	767	10
	Photos	Classification	7,650	238,162	745	8
Co-author	CS	Classification	18,333	163,788	6,805	15
	Physics	Classification	34,493	495,924	8,415	5
Transportation	Anaheim	Regression	914	3,881	4	–
	Chicago	Regression	2,176	15,104	4	–
Geography	Education	Regression	3,234	12,717	6	–
	Election	Regression	3,234	12,717	6	–
	Income	Regression	3,234	12,717	6	–
	Unemployment	Regression	3,234	12,717	6	–
Social	Twitch	Regression	1,912	31,299	3,170	–

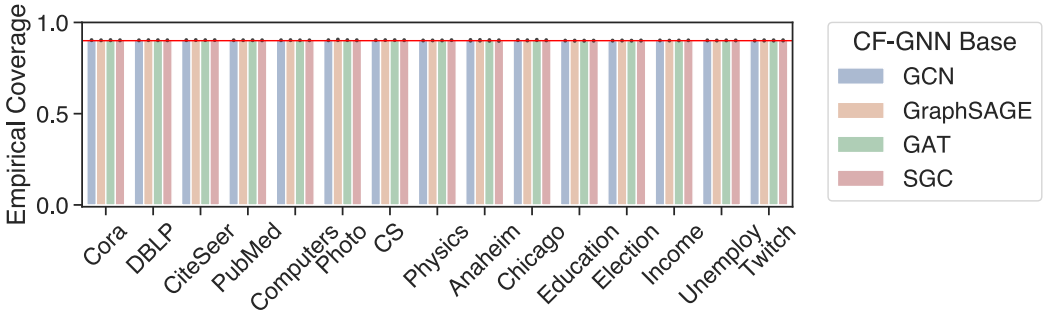

Figure 10: Empirical coverage across 15 datasets with 10 independent runs of GNN, using CF-GNN.

coverage is in Figure 10. The result for inefficiency is in Table 8. We observe consistent improvement in inefficiency reduction across these architectures, suggesting CF-GNN is a GNN-agnostic efficiency improvement approach.

D.5 CF-GNN with Regularized Adaptive Prediction Sets

To further showcase that CF-GNN is a versatile framework that adapts to any advancement in non-conformity scores, we experiment on RAPS [2], which regularizes APS to produce a smaller prediction set size. We report the performance using the GCN backbond in Table 9. We observe that CF-GNN still obtains impressive inefficiency reduction compared to the vanilla application of RAPS to GNN.

D.6 Conditional coverage on full set of network features

We report the full set of network features and calculate the worse-slice coverage in Table 10 for target coverage of 0.9 and Table 11 for a target coverage of 0.95. We observe that CF-GNN achieves satisfactory conditional coverage across a wide range of diverse network features.

D.7 Prediction accuracy versus uncertainty calibration

As we discussed in the main text, the original GNN trained towards optimal prediction accuracy does not necessarily yield the most efficient prediction model; this is corrected by the second GNN in CF-GNN which improves the efficiency of conformal prediction sets/intervals. With our approach, one can use the output of the original GNN for point prediction while that of the second GNN for efficient uncertainty quantification, without necessarily overwriting the first accurate prediction model. However, a natural question here still remains, which is that whether applying the second

Table 8: Empirical inefficiency measured by the size/length of the prediction set/interval for node classification/regression. The result uses APS for classification and CQR for regression. We report the average and standard deviation calculated from 10 GNN runs with each run of 100 conformal splits.

Task	GNN Model	GCN	GraphSAGE	GAT	SGC
	Dataset	CP $\longrightarrow$ CF-GNN	CP $\longrightarrow$ CF-GNN	CP $\longrightarrow$ CF-GNN	CP $\longrightarrow$ CF-GNN
Node classif.	Cora	$3.80_{\pm.28} \xrightarrow{-58.44\%} 1.58_{\pm.22}$	$6.73_{\pm.19} \xrightarrow{-76.50\%} 1.58_{\pm.15}$	$4.14_{\pm.16} \xrightarrow{-62.53\%} 1.55_{\pm.10}$	$3.88_{\pm.19} \xrightarrow{-62.23\%} 1.47_{\pm.10}$
	DBLP	$2.43_{\pm.03} \xrightarrow{-49.20\%} 1.23_{\pm.01}$	$3.91_{\pm.01} \xrightarrow{-68.27\%} 1.24_{\pm.01}$	$2.02_{\pm.06} \xrightarrow{-37.51\%} 1.26_{\pm.01}$	$2.44_{\pm.04} \xrightarrow{-48.93\%} 1.24_{\pm.02}$
	CiteSeer	$3.86_{\pm.11} \xrightarrow{-74.50\%} 0.99_{\pm.01}$	$5.88_{\pm.02} \xrightarrow{-83.07\%} 1.00_{\pm.01}$	$3.18_{\pm.25} \xrightarrow{-68.56\%} 1.00_{\pm.01}$	$3.79_{\pm.14} \xrightarrow{-73.43\%} 1.01_{\pm.02}$
	PubMed	$1.60_{\pm.02} \xrightarrow{-19.44\%} 1.29_{\pm.04}$	$1.93_{\pm.28} \xrightarrow{-36.95\%} 1.22_{\pm.03}$	$1.37_{\pm.02} \xrightarrow{-10.54\%} 1.23_{\pm.02}$	$1.60_{\pm.02} \xrightarrow{-24.27\%} 1.21_{\pm.02}$
	Computers	$3.56_{\pm.13} \xrightarrow{-50.22\%} 1.77_{\pm.11}$	$6.00_{\pm.10} \xrightarrow{-45.74\%} 3.26_{\pm.48}$	$2.33_{\pm.11} \xrightarrow{-8.18\%} 2.14_{\pm.12}$	$3.44_{\pm.14} \xrightarrow{-45.69\%} 1.87_{\pm.15}$
	Photo	$3.79_{\pm.13} \xrightarrow{-57.03\%} 1.63_{\pm.17}$	$4.52_{\pm.47} \xrightarrow{-37.22\%} 2.84_{\pm.67}$	$2.24_{\pm.21} \xrightarrow{-19.38\%} 1.81_{\pm.16}$	$3.81_{\pm.14} \xrightarrow{-62.86\%} 1.41_{\pm.05}$
	CS	$7.79_{\pm.29} \xrightarrow{-55.83\%} 3.44_{\pm.33}$	$14.68_{\pm.02} \xrightarrow{-88.63\%} 1.67_{\pm.14}$	$6.87_{\pm.48} \xrightarrow{-73.08\%} 1.85_{\pm.10}$	$7.76_{\pm.25} \xrightarrow{-73.92\%} 2.02_{\pm.22}$
	Physics	$3.11_{\pm.07} \xrightarrow{-65.36\%} 1.08_{\pm.10}$	$4.91_{\pm.01} \xrightarrow{-72.97\%} 1.33_{\pm.08}$	$2.00_{\pm.19} \xrightarrow{-45.23\%} 1.09_{\pm.06}$	$3.10_{\pm.08} \xrightarrow{-57.22\%} 1.32_{\pm.12}$
Average Improvement		-53.75%	-63.75%	-40.63%	-56.07%
Node regress.	Anaheim	$2.89_{\pm.39} \xrightarrow{-25.00\%} 2.17_{\pm.11}$	$2.37_{\pm.05} \xrightarrow{-23.12\%} 1.82_{\pm.07}$	$3.12_{\pm.38} \xrightarrow{-31.27\%} 2.14_{\pm.11}$	$2.94_{\pm.24} \xrightarrow{-24.90\%} 2.21_{\pm.16}$
	Chicago	$2.05_{\pm.07} \xrightarrow{-0.48\%} 2.04_{\pm.17}$	$2.08_{\pm.05} \xrightarrow{-7.90\%} 1.92_{\pm.09}$	$1.95_{\pm.04} \xrightarrow{-68.15\%} 0.62_{\pm.93}$	$2.02_{\pm.03} \xrightarrow{-1.37\%} 1.99_{\pm.07}$
	Education	$2.56_{\pm.02} \xrightarrow{-5.07\%} 2.43_{\pm.05}$	$2.20_{\pm.04} \xrightarrow{+8.44\%} 2.38_{\pm.08}$	$2.48_{\pm.05} \xrightarrow{-2.76\%} 2.41_{\pm.04}$	$2.55_{\pm.02} \xrightarrow{-2.80\%} 2.48_{\pm.04}$
	Election	$0.90_{\pm.01} \xrightarrow{+0.21\%} 0.90_{\pm.02}$	$0.87_{\pm.01} \xrightarrow{-0.80\%} 0.86_{\pm.02}$	$0.89_{\pm.00} \xrightarrow{-1.23\%} 0.88_{\pm.02}$	$0.90_{\pm.00} \xrightarrow{-0.42\%} 0.90_{\pm.02}$
	Income	$2.51_{\pm.12} \xrightarrow{-4.58\%} 2.40_{\pm.05}$	$2.08_{\pm.04} \xrightarrow{+32.23\%} 2.75_{\pm.23}$	$2.35_{\pm.03} \xrightarrow{-0.23\%} 2.34_{\pm.07}$	$2.42_{\pm.01} \xrightarrow{+3.04\%} 2.49_{\pm.04}$
	Unemploy.	$2.72_{\pm.03} \xrightarrow{-10.83\%} 2.43_{\pm.04}$	$2.75_{\pm.06} \xrightarrow{-12.90\%} 2.39_{\pm.05}$	$2.80_{\pm.08} \xrightarrow{-14.56\%} 2.40_{\pm.04}$	$2.72_{\pm.02} \xrightarrow{-11.05\%} 2.42_{\pm.04}$
	Twitch	$2.43_{\pm.10} \xrightarrow{-1.36\%} 2.39_{\pm.07}$	$2.48_{\pm.09} \xrightarrow{-3.06\%} 2.40_{\pm.07}$	$2.50_{\pm.14} \xrightarrow{-5.53\%} 2.36_{\pm.07}$	$2.42_{\pm.08} \xrightarrow{-1.43\%} 2.38_{\pm.06}$
Average Improvement		-6.73%	-1.02%	-17.68%	-5.56%

Table 9: Comparison with other non-conformity scores that reduce inefficiency.

Size	CP $\longrightarrow$ CF-GNN
Cora	$1.67_{\pm.11} \xrightarrow{-15.35\%} 1.42_{\pm.05}$
DBLP	$1.39_{\pm.02} \xrightarrow{-5.00\%} 1.32_{\pm.01}$
CiteSeer	$1.30_{\pm.07} \xrightarrow{-19.85\%} 1.04_{\pm.04}$
PubMed	$1.23_{\pm.01} \xrightarrow{+2.40\%} 1.26_{\pm.02}$
Computers	$1.58_{\pm.02} \xrightarrow{-4.59\%} 1.51_{\pm.05}$
Photo	$1.34_{\pm.01} \xrightarrow{-10.47\%} 1.20_{\pm.01}$
CS	$1.29_{\pm.04} \xrightarrow{-6.13\%} 1.21_{\pm.02}$

GNN drastically changes the prediction accuracy. This question is more relevant to the classification problem since for regression our method only adjusts the confidence band. For classification, we consider top-1 class prediction as the "point prediction". We present its accuracy "Before" and "After" the correction in Table 12, which shows that this correction typically does not result in a visible change in accuracy. In addition, in a new experiment on Cora, we find that 100% of the top-1 class from the base GNN are in CF-GNN's prediction sets. The potential to develop steps that explicitly consider point prediction accuracy is an exciting avenue for future research.

E Extended Related Works

Uncertainty quantification for graph neural networks. Uncertainty quantification (UQ) is a well-studied subject in general machine learning and also recently in GNNs. For multi-class classification, the raw prediction scores are often under/over-confident and thus various calibration methods are proposed for valid uncertainty estimation such as temperate scaling [13], vector scaling [13], ensemble temperate scaling [49], and so on [14, 27, 39, 1]. Recently, specialized calibration methods that leverage network principles such as homophily have been developed: examples include CaGCN [44] and GATS [17]. In regression, various methods have been proposed to construct prediction intervals, such as quantile regression [25, 41, 38], bootstrapping with subsampling, model ensembles, and dropout initialization [9, 28, 26, 35], and bayesian approaches with strong modeling assumptions on parameter and data distributions [23, 19]. However, these UQ methods can fail to provide statistically

Table 10: CF-GNN achieves conditional coverage. We use Cora/Twitch as an example classification/regression dataset.

Target: 0.9	Classification		Regression	
Model	CP	CF-GNN	CP	CF-GNN
Marginal Cov.	0.90±.02	0.90±.01	0.91±.02	0.91±.03
Cond. Cov. (Input Feat.)	0.89±.04	0.90±.03	0.90±.07	0.86±.08
Cond. Cov. (Cluster)	0.82±.07	0.89±.03	0.90±.06	0.88±.07
Cond. Cov. (Between)	0.82±.06	0.89±.03	0.86±.08	0.88±.07
Cond. Cov. (PageRank)	0.71±.08	0.87±.05	0.87±.09	0.89±.07
Cond. Cov. (Load)	0.83±.05	0.90±.03	0.86±.08	0.88±.07
Cond. Cov. (Harmonic)	0.89±.04	0.87±.05	0.88±.08	0.91±.06
Cond. Cov. (Degree)	0.79±.05	0.89±.04	0.86±.08	0.89±.06

Table 11: CF-GNN achieves conditional coverage. We use Cora/Twitch as an example classification/regression dataset.

Target: 0.95	Classification		Regression	
Model	CP	CF-GNN	CP	CF-GNN
Marginal Cov.	0.95±.01	0.95±.01	0.96±.02	0.96±.02
Cond. Cov. (Input Feat.)	0.94±.02	0.94±.03	0.95±.04	0.94±.05
Cond. Cov. (Cluster)	0.89±.06	0.93±.04	0.96±.03	0.96±.03
Cond. Cov. (Between)	0.81±.06	0.95±.03	0.94±.05	0.94±.05
Cond. Cov. (PageRank)	0.78±.06	0.94±.03	0.94±.05	0.94±.05
Cond. Cov. (Load)	0.81±.06	0.94±.03	0.94±.05	0.95±.05
Cond. Cov. (Harmonic)	0.88±.04	0.95±.03	0.96±.04	0.95±.04
Cond. Cov. (Degree)	0.83±.05	0.88±.06	0.94±.04	0.94±.04

rigorous and empirically valid coverage guarantee (see Table 1). In contrast, CF-GNN achieves valid marginal coverage in both theory and practice. Uncertainty quantification has also been leveraged to deal with out-of-distribution detection and imbalanced data in graph neural networks [50, 10]. While it is not the focus here, we remark that conformal prediction can also be extended to tackle such issues [18], and it would be interesting to explore such applications for graph data.

Conformal prediction for graph neural networks. As we discussed, the application of conformal prediction to graph-structured data remains largely unexplored. At the time of submission, the only work we awared of is [7], who claims that nodes in the graph are not exchangeable in the inductive setting and employs the framework of [3] to construct conformal prediction sets using neighborhood nodes as the calibration data. In contrast, we study the transductive setting where certain exchangeablility property holds and allows for flexibility in the training step. We also study the efficiency aspect that is absent in [7]. In addition, there have been concurrent works [15, 33] that observe similar exchangeability and validity of conformal prediction in either transductive setting or other network models. In particular, [15] proposes a diffusion-based method that aggregates non-conformity scores of neighbor nodes to improve efficiency, while our approach learns the aggregation of neighbor scores, which is more general than their approach. [32] studies the exchangeability for node regression under certain network models instead of our transductive setting with GNNs, and without considering the efficiency aspect. With a growing recent interest in conformal prediction for graphs, there are even more recent works that focus on validity [32] and link prediction [34].

Efficiency of conformal prediction. While conformal prediction enjoys distribution-free coverage for any non-conformity score based on any prediction model, its efficiency (i.e., size of prediction sets or length of prediction intervals) varies with specific choice of the scores and models. How to achieve desirable properties such as efficiency is a topic under intense research in conformal prediction. To this end, one major thread designs good non-conformity scores such as APS [37] and CQR [36]. More recent works take another approach, by modifying the training process of the prediction model to

Table 12: CF-GNN does not change the top-1 class prediction accuracy for classification tasks.

Dataset	Before	After
Cora	$0.844_{\pm 0.004}$	$0.843_{\pm 0.016}$
DBLP	$0.835_{\pm 0.001}$	$0.832_{\pm 0.002}$
CiteSeer	$0.913_{\pm 0.002}$	$0.911_{\pm 0.002}$

further improve efficiency. This work falls into the latter case. Our idea applies to any non-conformity scores, as demonstrated with APS and CQR, two prominent examples of the former case. Related to our work, ConfTr [40] also simulates conformal prediction so as to train a prediction model that eventually leads to more efficient conformal prediction sets. However, our approach differs from theirs in significant ways. First, ConfTr modifies model training, while CF-GNN conducts post-hoc correction without changing the original prediction. Second, ConfTr uses the training set to simultaneously optimize model prediction and efficiency of conformal prediction, while we withhold a fraction of calibration data to optimize the efficiency. Third, our approach specifically leverages the rich topological information in graph-structured data to achieve more improvement in efficiency. Finally, we also propose a novel loss for efficiency in regression tasks.