Residual Reweighted Conformal Prediction for Graph Neural Networks

Zheng Zhang[*1] **Jie Bao**[*2,3] **Zhixin Zhou**[4] **Nicolo Colombo**[5] **Lixin Cheng**[6] **Rui Luo**[†1,2]

[1]Department of System Engineering, City University of Hong Kong, China
[2]Chengdu Research Institute, City University of Hong Kong, China
[3]Faculty of Electronic Information Engineering, Huaiyin Institute of Technology, China
[4]Alpha Benito Research, USA
[5]Centre for Intelligent Systems, Royal Holloway, University of London, UK
[6]Shenzhen People's Hospital, China

Abstract

Graph Neural Networks (GNNs) excel at modeling relational data but face significant challenges in high-stakes domains due to unquantified uncertainty. Conformal prediction (CP) offers statistical coverage guarantees, but existing methods often produce overly conservative prediction intervals that fail to account for graph heteroscedasticity and structural biases. While residual reweighting CP variants address some of these limitations, they neglect graph topology, cluster-specific uncertainties, and risk data leakage by reusing training sets. To address these issues, we propose Residual Reweighted GNN (RR-GNN), a framework designed to generate minimal prediction sets with provable marginal coverage guarantees.

RR-GNN introduces three major innovations to enhance prediction performance. First, it employs Graph-Structured Mondrian CP to partition nodes or edges into communities based on topological features, ensuring cluster-conditional coverage that reflects heterogeneity. Second, it uses Residual-Adaptive Nonconformity Scores by training a secondary GNN on a held-out calibration set to estimate task-specific residuals, dynamically adjusting prediction intervals according to node or edge uncertainty. Third, it adopts a Cross-Training Protocol, which alternates the optimization of the primary GNN and the residual predictor to prevent information leakage while maintaining graph dependencies. We validate RR-GNN on 15 real-world graphs across diverse tasks, including node classification, regression, and edge weight prediction. Compared to CP baselines, RR-GNN achieves improved efficiency over state-of-the-art methods, with no loss of coverage.

[*]These authors contributed equally to this work.
[†]Corresponding author: ruiluo@cityu.edu.hk

1 INTRODUCTION

Graph Neural Networks (GNNs) have achieved state-of-the-art performance on graph-structured data across various applications like recommendation systems, knowledge graphs, and molecular modeling Lam et al. [2023], Li et al. [2022], Wu et al. [2022]. As GNNs are increasingly applied in high-stakes areas such as healthcare and autonomous systems, accurately assessing prediction uncertainty becomes paramount. A common approach to predicting uncertainty is to construct prediction intervals that capture the probability of true outcomes. While several methods of predicting uncertainty have been explored Hsu et al. [2022], Zhang et al. [2020], Lakshminarayanan et al. [2017], they typically lack rigorous theoretical guarantees on interval validity Wang et al. [2021]. Improving uncertainty quantification for GNNs with probabilistic guarantees is critical to ensuring their safe, trusted application in real-world settings.

Previous residual reweight methods Papadopoulos et al. [2008] often relied on fixed weight loss functions and importance weighting techniques, which do not adequately capture heteroscedasticity, leading to inaccurate quantification of prediction errors across diverse samples. Guan Guan [2023] proposed Localized Conformal Prediction (LCP), and Han et al. proposed Split Localized Conformal Prediction (SLCP) Han et al. [2022] which uses kernel-based weights. However, kernel methods are well-known to suffer from the curse of dimensionality, making them less effective in handling complex data, particularly structured data such as graphs. Previous graph-based methods often relied on simplistic partitioning techniques that ignored the complex topological structure between nodes and edges Clarkson [2023], thereby lacking the capability to perform the samplewise normalization. Conformal Prediction (CP) is a machine learning framework for uncertainty quantification that constructs prediction intervals for any underlying point predictor in a theoretically valid manner Vovk et al. [2005].

Proceedings of the 41st Conference on Uncertainty in Artificial Intelligence (UAI 2025), PMLR 244:5025–5042.

Due to its principled formulation, rigorous guarantees, and distribution-free nature, CP has enabled uncertainty estimation across diverse applications, including computer vision Angelopoulos et al. [2020], Bates et al. [2021], causal inference Lei and Candès [2021], Jin et al. [2023], Yin et al. [2024], time series Gibbs and Candes [2021], Zaffran et al. [2022], and drug discovery Jin and Candès [2023]. CP leverages a "calibration" dataset to output prediction sets for new test samples that provably cover the true outcome with at least 1 - α probability, where α is a user-specified error tolerance. CP is based on a nonconformity measure/score that measures the dissimilarity between a data point and others, reflecting disagreements according to the algorithm's feature-relationship assumptions. Crucially, each nonconformity score can represent a single algorithm, by defining a distinct CP predictor Papadopoulos et al. [2008]. Additionally, adding a Residual-Reweighting (RR) factor can refine prediction intervals Papadopoulos et al. [2011], Lei et al. [2018]. By assigning weights to errors covariately, RR helps mitigate heteroscedasticity impacts on accuracy and reliability. Overall, carefully constructing the nonconformity measure and incorporating RR are pivotal to prediction performance.

Previous residual reweight methods often relied on fixed-weight loss functions and importance weighting techniques to address sample imbalance in various applications. However, these approaches have significant limitations: they often overlook relationships between samples, fail to leverage the topological structure of the data, and inadequately analyze prediction residuals when dynamically adjusting sample weights. Existing methods, such as normalized nonconformity functions Johansson et al. [2014], Kath and Ziel [2021] and local reweighted conformal methods Papadopoulos et al. [2008], involve training separate models to predict errors but struggle to capture the heteroscedasticity of the data. Similarly, prior graph-based methods often relied on simplistic partitioning techniques that ignored complex relationships between nodes and edges Aggarwal [2015]. Before the Cross-Training Protocol, machine learning models, including GNNs, faced challenges with information leakage during training Vepakomma et al. [2020]. Simultaneous optimization of multiple models on the same dataset often caused one model's learning to adversely affect the other, leading to biased predictions and reduced performance Kapoor and Narayanan [2023], Hitaj et al. [2017].

We propose a novel residual-reweighted nonconformity measure for graph neural networks (RR-GNN), which optimizes model performance by independently predicting expected accuracy. RR-GNN consists of two GNNs: the Conformal GNN, which generates predictions based on input features and true labels, and the Residual GNN, trained on prediction residuals to calibrate outputs.

We present the key contributions of this work:

- We propose a novel framework that integrates conformal prediction with GNNs to enhance uncertainty quantification in graph-structured data.
- To address the heteroscedasticity of graph data, we design a novel nonconformity score that reweights residuals using a separately trained GNN.
- We introduce a graph-based Mondrian CP, where nodes are clustered based on graph structure, enabling fine-grained, context-aware prediction intervals Aggarwal [2015].
- To prevent information leakage between the primary GNN and residual-prediction GNN, we develop a cross-training strategy where models iteratively update each other. This ensures independence between calibration and training data—a critical requirement for CP validity—while maintaining model performance Vepakomma et al. [2020], Kapoor and Narayanan [2023], Hitaj et al. [2017].

2 RELATED WORK

Conformal prediction (CP) Vovk et al. [2005] is a methodology designed to generate prediction regions for variables of interest, facilitating the estimation of model uncertainty by providing prediction sets rather than point estimates. CP has been successfully applied to both classification Luo and Zhou [2024], Luo and Colombo [2024], Luo and Zhou [2025a] and regression tasks Luo and Zhou [2025d,e]. Its flexibility allows adaptation to various real-world scenarios, including segmentation Luo and Zhou [2025b], games Luo et al. [2024], Bao et al. [2025], time-series forecasting Su et al. [2024], and graph-based applications Luo et al. [2023], Tang et al. [2025], Luo and Zhou [2025c], Wang et al. [2025], Luo and Colombo [2025].

Graph Neural Networks (GNNs) have become foundational models for learning from graph-structured data. Kipf et al. Kipf and Welling [2016a] introduced a seminal unsupervised GNN for learning low-dimensional embeddings. Cai et al. Cai et al. [2021] proposed the Line Graph Neural Network (LGNN), which reformulates link prediction as a node classification problem in the line graph. Kollias et al. Kollias et al. [2022] developed DiGAE, a directed graph auto-encoder with parameterized message passing for node classification and link prediction. GNNs are also adaptable to regression tasks Luo and Krishnamurthy [2023b] by modifying the output layer and loss functions, which has potential applications in network segregation prediction and control Luo et al. [2021], Krishnamurthy et al. [2021], Luo et al. [2022b,a], Luo and Krishnamurthy [2023a].

For heterogeneous graphs, specialized architectures have been proposed. Wang et al. Wang et al. [2019] introduced the Heterogeneous Graph Attention Network (HAN), which employs hierarchical attention to capture both meta-path-

based and semantic-level importance, achieving state-of-the-art results. Iyer et al. Iyer et al. [2021] presented the Bi-Level Attention GNN (BAGNN), utilizing a bi-level attention mechanism to learn complex relationships in heterogeneous data.

While existing GNN approaches focus primarily on achieving high prediction accuracy, they often lack flexibility, adaptability, and mechanisms for uncertainty quantification. Conformal prediction addresses these gaps by offering prediction intervals and enhanced generalization. Notably, Huang et al. Huang et al. [2024] extended CP to GNNs with CF-GNN, improving uncertainty quantification. Robustness in these frameworks has been further enhanced by importance weighting schemes. Guo et al. Guo et al. [2017] introduced a causal inference-driven weighting technique, whereas Volpi et al. Volpi et al. [2018] proposed a distribution matching-based strategy to mitigate distribution shifts.

Despite these advancements, rigorous guarantees on coverage and reliability for uncertainty quantification in GNNs remain an open challenge Bhagat et al. [2011].

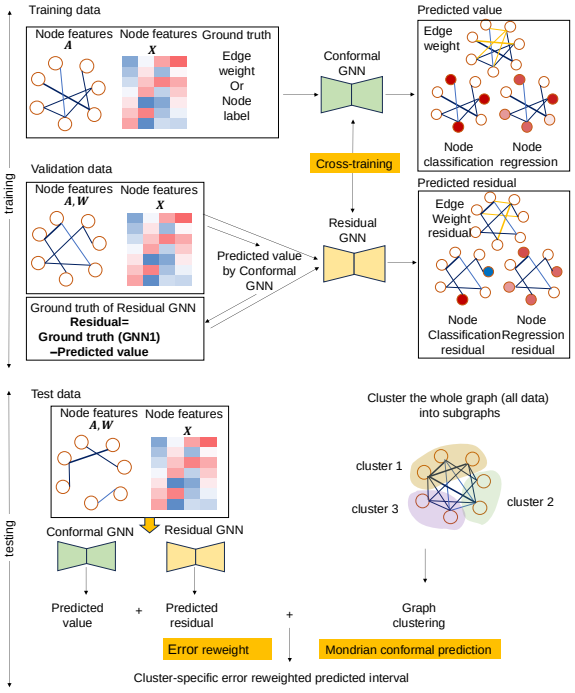

Figure 1: The RR-GNN pipeline consists of three parts: 1) Conformal GNN Training: A GNN is trained for edge weight prediction, node regression, or classification, outputting prediction intervals with uncertainty quantification and marginal coverage guarantees; 2) Residual GNN Training: A separate Residual GNN is trained on validation data to predict residuals between true values and Conformal GNN predictions; 3) Residual Reweighting: Prediction intervals from the Conformal GNN and residual weights from the Residual GNN are integrated, and Mondrian conformal prediction incorporates graph-structured clustering. The Conformal and Residual GNNs are trained alternately, with the Residual GNN correcting the Conformal GNN's final predictions.

3 METHODOLOGY

RR-GNN combines conformal prediction with a GNN-based framework to effectively address graph-structured data tasks. The model accepts a graph as input, either undirected or directed and yields predictions for edge weights or node features as output. RR-GNN provides prediction intervals instead of a single point value, which can better capture the uncertainty in the data. For new test samples, the probability of the predicted interval would cover the true outcome with at least 1 - α with rigorous theoretical guarantees.

RR-GNN performs graph-based Mondrian CP, in which the input graph is clustered to subnetworks to capture the community structure of the graph. It begins by training a predictive model on a training dataset named Conformal GNN, which gives the main prediction according to the tasks. Next, RR-GNN trains a Residual GNN model based on the validation set to predict the residual of Conformal GNN, which is next used to calculate prediction errors used as a reweight factor to establish nonconformity scores, measuring how unusual the data point looks relative to previous examples. After determining a significance level based on the desired confidence, RR-GNN generates a cluster-specific fixed prediction interval based on the distribution of nonconformity scores in the calibration set of each cluster. The predicted interval of test data is given by the combination of predicted point estimation, residual prediction and the cluster-specific fixed interval. Notably, throughout this process, each dataset is divided into four distinct parts: training data, validation data, calibration data, and test data, with only the test data lacking labels. Algorithm 1 and Figure 1 show the implementation details of training the main and residual models crossly simultaneously Cowell et al. [2006], Peste et al. [2021]. The Conformal GNN model is trained using a cross-training process with the Residual GNN model.

We clarify the notation here. The input is a graph, $\mathcal{G} = (\mathbb{V}, \boldsymbol{E})$, with node set $\mathbb{V}$ and edge set $\boldsymbol{E} \subseteq \mathbb{V} \times \mathbb{V}$. Assume the graph has n nodes with m features. Let $\boldsymbol{X} \in \mathbb{R}^{n \times m}$ be the node feature matrix, and $\boldsymbol{X}_{i,:} \in \mathbb{R}^m$ be the feature vector of the ith node. The binary adjacency matrix of $\mathcal{G}$, $\boldsymbol{A} \in \{0,1\}^{n \times n}$, encodes the binary (unweighted) connecting structure of the graph. For the weighted graph, we use $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ to represent the weighted adjacency matrix.

In RR-GNN, taking the weighted graph as an example. We partition the edge set $\boldsymbol{E}$ into three disjoint subsets: $\boldsymbol{E}^{train}$, $\boldsymbol{E}^{val}$, $\boldsymbol{E}^{calib}$ and $\boldsymbol{E}^{test}$, while satisfying $\boldsymbol{E} = \boldsymbol{E}^{train} \cup$

Algorithm 1 Crossly-Training Algorithm for RR-GNN

Input: Model C_0 (Conformal GNN) weights $\boldsymbol{\theta}_1 \in \mathbb{R}^N$, Residual Model C_1 (Residual GNN) weights $\boldsymbol{\theta}_2 \in \mathbb{R}^N$, loop limit n
for $i = 1$ **to** n **with step** 1 **do**
  if i is odd **then**
    Train Model C_0 with gradients and update $\boldsymbol{\theta}_1$ using the training data.
  else
    Get the residual from Model C_0 as the label based on validation data.
    Train Residual Model C_1 with gradients and update $\boldsymbol{\theta}_2$ based on validation data.
  end if
end for

$\boldsymbol{E}^{val} \cup \boldsymbol{E}^{calib} \cup \boldsymbol{E}^{test}$. We define

$$\mathbf{W}^{train} = \begin{cases} W_{ij}, & \text{if } (i,j) \in \boldsymbol{E}^{train}; \\ \delta, & \text{if } (i,j) \in \boldsymbol{E}^{val} \cup \boldsymbol{E}^{calib} \cup \boldsymbol{E}^{test}; \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where $\delta > 0$ is a small positive constant to represent a minimal edge weight. The Conformal GNN is represented by:

$$[\hat{\boldsymbol{y}}, \hat{\boldsymbol{y}}^{\alpha/2}, \hat{\boldsymbol{y}}^{1-\alpha/2}] = g_{\boldsymbol{\theta}_1}\left(\boldsymbol{W}^{train}, \boldsymbol{X}\right), \tag{2}$$

where y is the label for a specific task; $g_{\boldsymbol{\theta}_1}$ represents the GNN-based model mapping the input to the label; $\hat{\boldsymbol{y}}^{\alpha/2}$ and $\hat{\boldsymbol{y}}^{1-\alpha/2}$ is the predicted $\alpha/2$ and $1-\alpha/2$ quantile of the label. The Residual GNN is represented by the following equation:

$$\hat{\boldsymbol{R}} = g_{\boldsymbol{\theta}_2}\left(\boldsymbol{W}^{val}, \boldsymbol{X}\right), \tag{3}$$

where $\hat{\boldsymbol{R}}$ is the predicted residual of Conformal GNN prediction; $g_{\boldsymbol{\theta}_2}$ represents the GNN-based model mapping the input to the residual. Then, we will predict residuals for the calibration set and calculate the nonconformity score

$$\boldsymbol{V} = s\left(g_{\boldsymbol{\theta}_1}\left(\boldsymbol{W}^{calib}, \boldsymbol{X}\right), g_{\boldsymbol{\theta}_2}\left(\boldsymbol{W}^{calib}, \boldsymbol{X}\right)\right), \tag{4}$$

where $s(.)$ is the nonconformity function. We will perform graph-based partition: the nodes in the graph $G = (V, E)$ are clustered into K groups $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K$ using Louvain clustering, where $G_{(m)} = (V_{(m)}, E_{(m)})$ is the subgraph for the cluster m. Next, we will get an interval factor for each cluster, $d_{(m)}$, which is a quantile of nonconformity scores according to the significant level. The predicted interval for a test data point in cluster m is

$$\begin{aligned} \boldsymbol{C}_{(m)} = [&l\left(g_{\boldsymbol{\theta}_1}\left(\boldsymbol{W}^{test}, \boldsymbol{X}\right), g_{\boldsymbol{\theta}_2}\left(\boldsymbol{W}^{test}, \boldsymbol{X}\right), d_{(m)}\right), \\ &u\left(g_{\boldsymbol{\theta}_1}\left(\boldsymbol{W}^{test}, \boldsymbol{X}\right), g_{\boldsymbol{\theta}_2}\left(\boldsymbol{W}^{test}, \boldsymbol{X}\right), d_{(m)}\right)], \end{aligned} \tag{5}$$

where $l(.)$ and $u(.)$ represent the lower and upper bound of the prediction interval.

3.1 NODE REGRESSION/CLASSIFICATION

For the node regression/classification tasks, we partition the node set $\boldsymbol{V}$ into three disjoint subsets: $\boldsymbol{V}^{train}$, $\boldsymbol{V}^{val}$, $\boldsymbol{V}^{calib}$ and $\boldsymbol{V}^{test}$, while satisfying $\boldsymbol{V} = \boldsymbol{V}^{train} \cup \boldsymbol{V}^{val} \cup \boldsymbol{V}^{calib} \cup \boldsymbol{V}^{test}$. We define the label of node $\boldsymbol{y}$ as:

$$\boldsymbol{y}^{train} = \begin{cases} y_i, & \text{if } i \in \boldsymbol{V}^{train}; \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

We use the training label $\boldsymbol{y}^{train}$ and validation label $\boldsymbol{y}^{val}$ to train the Conformal GNN and get the parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, respectively. Then, we will predict residuals for the calibration set and calculate the nonconformity score for each node from the calibration set.

$$\boldsymbol{V}^{calib} = s\left(g_{\boldsymbol{\theta}_1}\left(\boldsymbol{W}, \boldsymbol{X}\right), g_{\boldsymbol{\theta}_2}\left(\boldsymbol{W}, \boldsymbol{X}\right), \mathbb{V}^{calib}\right), \tag{7}$$

where the nonconformity score is calculated only on the calibration set. Next, we will get an interval factor, d, which is a quantile of nonconformity scores according to the significant level. The predicted interval for a test data point is

$$\begin{aligned} \boldsymbol{C} = [&l(g_{\boldsymbol{\theta}_1}\left(\boldsymbol{W}, \boldsymbol{X}\right), g_{\boldsymbol{\theta}_2}\left(\boldsymbol{W}, \boldsymbol{X}\right), d, \mathbb{V}^{test}), \\ &u(g_{\boldsymbol{\theta}_1}\left(\boldsymbol{W}, \boldsymbol{X}\right), g_{\boldsymbol{\theta}_2}\left(\boldsymbol{W}, \boldsymbol{X}\right), d, \mathbb{V}^{test})], \end{aligned} \tag{8}$$

where $l(.)$ and $u(.)$ represent the lower and upper bound of the prediction interval, which are calculated based on the predicted node label of the test set.

3.2 RR-GNN ON EDGE WEIGHT PREDICTION

3.2.1 Conformal GAE for Edge Weight Prediction

The GAE Kipf and Welling [2016b], Ahn and Kim [2021] is used for edge weight prediction tasks by learning node embeddings across various types of graphs, including directed graphs Kollias et al. [2022], weighted graphs Zulaika et al. [2022], and graphs with different edge types Samanta et al. [2020]. The edge weight is then given by the similarity of node embeddings. There are two kinds of problem settings in link prediction shown in Figure 1 in appendices material including transductive setting and inductive setting. We focus on the first one. We can see the details in the first part in Figure 5 appendices material. We integrate CP in GAE's framework by making the encoder produce a triple output. We use $\boldsymbol{Z}, \boldsymbol{Z}^{\alpha/2}$, and $\boldsymbol{Z}^{1-\alpha/2} \in \mathbb{R}^{n \times d}$ to represent the mean, $\alpha/2$ quantile, and $(1-\alpha/2)$ quantile of node embedding matrix obtained from a Conformal GAE model. This differs from having three single-output GAE encoders because most network parameters are shared across the three embeddings. The resulting embedding is

$$[\boldsymbol{Z}, \boldsymbol{Z}^{\alpha/2}, \boldsymbol{Z}^{1-\alpha/2}] = f_{\boldsymbol{\theta}}(\boldsymbol{X}, \boldsymbol{A}), \tag{9}$$

where $f_{\boldsymbol{\theta}}$ is the structure of the encoder, and $\boldsymbol{\theta}$ is a learnable parameter. Note that the traditional GNN model is applicable because it could generate d-dimentional output for each node, which represents node embeddings. Directed GAE designed for the directed graph Kollias et al. [2022] is more flexible, using separate source and target embeddings, $\boldsymbol{Z} = [\boldsymbol{Z}^S, \boldsymbol{Z}^T]$. As for undirected GAE, $\boldsymbol{Z}^S = \boldsymbol{Z}^T$. It is similar for $\boldsymbol{Z}^{\alpha/2}$ and $\boldsymbol{Z}^{1-\alpha/2}$.

We take the directed graph as an example for the following description. We next reconstruct the weighted adjacency matrix from the inner product between node embeddings, which is the Conformal GNN-based model.

$$g_{\boldsymbol{\theta}_1}(\boldsymbol{X}, \boldsymbol{A}) = [\hat{\boldsymbol{W}}, \hat{\boldsymbol{W}}^{\alpha/2}, \hat{\boldsymbol{W}}^{1-\alpha/2}] = [\boldsymbol{Z}^S (\boldsymbol{Z}^T)^\top, \boldsymbol{Z}^{S,\alpha/2} (\boldsymbol{Z}^{T,\alpha/2})^\top, \boldsymbol{Z}^{S,1-\alpha/2} (\boldsymbol{Z}^{T,1-\alpha/2})^\top]. \quad (10)$$

where $\hat{\boldsymbol{W}}$, $\hat{\boldsymbol{W}}^{\alpha/2}$, and $\hat{\boldsymbol{W}}^{1-\alpha/2}$ be the mean, $\alpha/2$, and (1-α/2) quantiles of the edge weights.

The loss function $\mathcal{L}_{\mathrm{Conformal-GNN}}$ is given by:

$$\mathcal{L}_{\mathrm{GAE}} + \sum_{(i,j)\in E^{\text{train}}} \rho_{\alpha/2}\left(W_{ij}^{\text{train}}, \hat{W}_{ij}^{\alpha/2}\right) + \rho_{1-\alpha/2}\left(W_{ij}^{\text{train}}, \hat{W}_{ij}^{1-\alpha/2}\right). \quad (11)$$

where $\mathcal{L}_{\mathrm{GAE}}$ is the squared error loss defined in (9) The second term is the pinball loss referenced to Romano et al. [2019], Steinwart and Christmann [2011], defined as

$$\rho_\alpha(y, \hat{y}) := \begin{cases} \alpha(y - \hat{y}) & \text{if } y > \hat{y} \\ (1-\alpha)(y - \hat{y}) & \text{otherwise} \end{cases}$$

The first term is added to train the mean estimator, $\hat{\boldsymbol{W}}$.

$$\mathcal{L}_{\mathrm{GAE}} = \|\boldsymbol{A}^{train} \odot \hat{\boldsymbol{W}} - \boldsymbol{W}^{train}\|. \quad (12)$$

3.2.2 RR-GNN on Edge Weight Prediction

We train a separate Residual GAE model to predict the error of the edge weight prediction of the $g_{\boldsymbol{\theta}_1}$,

$$\hat{R}_{ij}^{val} = g_{\boldsymbol{\theta}_2}\left((i,j); \boldsymbol{A}, \boldsymbol{X}\right), \quad (13)$$

where the label is $R_{ij}^{val} = W_{ij}^{\text{val}} - \hat{W}_{ij}^{val}$, $\hat{W}_{ij}^{val} = g_{\boldsymbol{\theta}_1}\left((i,j); \boldsymbol{A}^{val}, \boldsymbol{X}\right)$ is the output of conformal GAE, and the RR-GAE is trained by minimizing

$$\mathcal{L}_{\text{Residual GNN}} = \|\boldsymbol{A}^{val} \odot \hat{\boldsymbol{R}}^{val} - \boldsymbol{R}^{val}\|_F. \quad (14)$$

We use the standard deviation of these predictions as a proxy of the residual. More concretely, we propose a new nonconformity score function, which is the interval of predicted edge weight reweighted according to the absolute value of the residual as predicted by the RR-GNN model (13).

$$V_{ij}^{\text{RR}} = \max\left\{ \frac{\hat{W}_{ij}^{\alpha/2} - W_{ij}^{\text{calib}}}{\left|\hat{R}_{ij}\right|}, \frac{W_{ij}^{\text{calib}} - \hat{W}_{ij}^{1-\alpha/2}}{\left|\hat{R}_{ij}\right|} \right\}, \quad (15)$$

$$(i,j) \in \boldsymbol{E}^{calib}, \quad (16)$$

where $\hat{\boldsymbol{W}}^{\alpha/2}$ and $\hat{\boldsymbol{W}}^{1-\alpha/2}$ is the predicted edge weight quantile of the Conformal GAE based on the calibration set. Let $d_{(m)}^{\text{RR}}$ be the k-th smallest value in $\{V_{ij}^{\text{RR}} | (i,j) \in \boldsymbol{E}_{(m)}^{calib}\}$ for the cluster m, where $k = \lceil (n/2+1)(1-\alpha) \rceil$ where n is size of $\boldsymbol{E}_{(m)}^{calib}$. The RR prediction intervals for cluster m are:

$$C_{ab}^{(m)} = \left[\hat{W}_{ab}^{\alpha/2} - d_{(m)}^{\text{RR}}\left|\hat{R}_{ab}\right|,\ \hat{W}_{ab}^{1-\alpha/2} + d_{(m)}^{\text{RR}}\left|\hat{R}_{ab}\right|\right], \quad (17)$$

$$(a,b) \in \boldsymbol{E}_{(m)}^{test}, \quad (18)$$

The theoretical guarantees on interval validity can be referenced to Luo and Colombo [2025].

Algorithm 2 Residual Reweighted Conformalized Graph Neural Network for Edge Weight Prediction

1: **Input:** The binary adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n\times n}$, edge weight matrix $\boldsymbol{W} \in \mathbb{R}^{n\times n}$, node features $\boldsymbol{X} \in \mathbb{R}^{n\times m}$, training edges and weights $\boldsymbol{E}^{train}$ and $\boldsymbol{W}^{train}$, validation edges and weights $\boldsymbol{E}^{val}$ and $\boldsymbol{W}^{val}$ (used for training Residual GNN), calibration edges and weights $\boldsymbol{E}^{calib}$ and $\boldsymbol{W}^{calib}$, and test edges $\boldsymbol{E}^{test}$, user-specified error rate $\alpha \in (0,1)$, two GNN models $g_{\boldsymbol{\theta}_1}$ and $g_{\boldsymbol{\theta}_2}$ with trainable parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$.
2: Cluster the whole graph $\boldsymbol{E} = \boldsymbol{E}^{train} \cup \boldsymbol{E}^{val} \cup \boldsymbol{E}^{calib} \cup \boldsymbol{E}^{test}$ into K clusters using Louvain clustering.
3: Train the models $g_{\boldsymbol{\theta}_1}$ and $g_{\boldsymbol{\theta}_2}$ with $\boldsymbol{A}$, $\boldsymbol{X}$, $\boldsymbol{W}^{train}$ and $\boldsymbol{W}^{val}$ according to Algorithm 1.
4: Predict the interval $[\hat{W}_{ij}^{\alpha/2}, \hat{W}_{ij}^{1-\alpha/2}]$ as the output of $g_{\boldsymbol{\theta}_1}$ and the residual $\hat{R}_{ij}$ as the output of $g_{\boldsymbol{\theta}_2}$ using the calibration data as input.
5: Compute the nonconformity score V_{ij}^{RR} for the calibration data according to (15).
6: Compute $d_{(m)}$ = the k-th smallest value in $\{V_{ij}^{\text{RR}} | (i,j) \in \boldsymbol{E}_{(m)}^{calib}\}$, where $k = \lceil (|\boldsymbol{E}_{(m)}^{calib}| + 1)(1-\alpha) \rceil$.
7: Construct a prediction interval for test edges according to (17).
8: **Output:** Prediction of confidence intervals for the test edges $(a,b) \in \boldsymbol{E}^{test}$ with the coverage guarantee according to (17).

3.3 RR-GNN ON NODE REGRESSION

We apply RR-GNN for the node regression task to predict a continuous target variable y_i associated with each node i in a

Table 1: Performance comparison of the proposed models

GNN Model On Chicago Data	GraphConv		SAGEConv		GCNConv		GATConv	
Score Method-CP	cover^x	ineff	cover^x	ineff	cover^x	ineff	cover^x	ineff
GAE	$0.7984^{\pm 0.1181}$	$3.6659^{\pm 0.3313}$	$0.8297^{\pm 0.1264}$	$3.6350^{\pm 0.2231}$	$0.8234^{\pm 0.1213}$	$3.6918^{\pm 0.2454}$	$0.9524^{\pm 0.0333}$	$3.3493^{\pm 0.5910}$
DiGAE	$0.8081^{\pm 0.1257}$	$3.5721^{\pm 0.1951}$	$0.8196^{\pm 0.1215}$	$3.5978^{\pm 0.1884}$	$0.8135^{\pm 0.1361}$	$3.5846^{\pm 0.2050}$	$0.8135^{\pm 0.1319}$	$3.6346^{\pm 0.2432}$
LGNN	$0.9174^{\pm 0.0238}$	$6.7157^{\pm 0.1325}$	$0.9152^{\pm 0.0256}$	$6.5865^{\pm 0.1577}$	$0.9151^{\pm 0.0246}$	$6.5265^{\pm 0.1426}$	$0.9075^{\pm 0.0618}$	$6.0679^{\pm 0.1862}$
Average	0.8477	4.6512	0.8548	4.5998	0.8507	4.6010	0.8912	4.3506
Score Method-CQR	cover^x	ineff	cover^x	ineff	cover^x	ineff	cover^x	ineff
GAE	$0.9514^{\pm 0.0144}$	$3.3652^{\pm 0.1312}$	$0.9517^{\pm 0.0141}$	$3.5878^{\pm 0.2107}$	$0.9578^{\pm 0.0420}$	$4.0504^{\pm 1.2916}$	$0.9524^{\pm 0.0333}$	$3.3292^{\pm 0.5866}$
DiGAE	$0.9205^{\pm 0.0498}$	$3.3135^{\pm 0.1172}$	$0.9223^{\pm 0.0469}$	$3.3872^{\pm 0.1260}$	$0.9250^{\pm 0.0479}$	$3.4241^{\pm 0.1271}$	$0.9089^{\pm 0.0611}$	$3.6158^{\pm 0.2348}$
LGNN	$0.9284^{\pm 0.0296}$	$3.4362^{\pm 0.1029}$	$0.9305^{\pm 0.0258}$	$3.4844^{\pm 0.1233}$	$0.9290^{\pm 0.0284}$	$3.6514^{\pm 0.1050}$	$0.9379^{\pm 0.0261}$	$4.0805^{\pm 0.5445}$
Average	0.9334	3.3716	0.9348	3.4865	0.9373	3.7086	0.9331	3.6752
Score Method-CQR-cluster	cover^x	ineff	cover^x	ineff	cover^x	ineff	cover^x	ineff
GAE	$0.9519^{\pm 0.0318}$	$3.3721^{\pm 0.021}$	$0.9532^{\pm 0.028}$	$3.4862^{\pm 0.035}$	$0.9557^{\pm 0.024}$	$3.7083^{\pm 0.041}$	$0.9541^{\pm 0.032}$	$3.6749^{\pm 0.019}$
DiGAE	$0.9412^{\pm 0.025}$	$3.3645^{\pm 0.018}$	$0.9428^{\pm 0.031}$	$3.4821^{\pm 0.027}$	$0.9443^{\pm 0.029}$	$3.7058^{\pm 0.033}$	$0.9437^{\pm 0.026}$	$3.6724^{\pm 0.022}$
LGNN	$0.9315^{\pm 0.037}$	$3.3582^{\pm 0.015}$	$0.9332^{\pm 0.034}$	$3.4789^{\pm 0.029}$	$0.9351^{\pm 0.031}$	$3.7023^{\pm 0.036}$	$0.9345^{\pm 0.028}$	$3.6698^{\pm 0.024}$
Average	0.9415	3.3649	0.9424	3.4824	0.9450	3.7055	0.9438	3.6720
Score Method-CQR-RR	cover^x	ineff	cover^x	ineff	cover^x	ineff	cover^x	ineff
GAE	$0.9482^{\pm 0.019}$	$3.3018^{\pm 0.017}$	$0.9497^{\pm 0.021}$	$3.3976^{\pm 0.023}$	$0.9513^{\pm 0.016}$	$3.4241^{\pm 0.025}$	$0.9508^{\pm 0.018}$	$3.5372^{\pm 0.020}$
DiGAE	$0.9395^{\pm 0.026}$	$3.2954^{\pm 0.019}$	$0.9411^{\pm 0.028}$	$3.3921^{\pm 0.024}$	$0.9428^{\pm 0.022}$	$3.4207^{\pm 0.027}$	$0.9432^{\pm 0.025}$	$3.5346^{\pm 0.021}$
LGNN	$0.9316^{\pm 0.035}$	$3.2893^{\pm 0.014}$	$0.9335^{\pm 0.032}$	$3.3875^{\pm 0.026}$	$0.9357^{\pm 0.029}$	$3.4174^{\pm 0.028}$	$0.9364^{\pm 0.027}$	$3.5319^{\pm 0.023}$
Average	0.9442	3.2945	0.9414	3.3920	0.9433	3.4207	0.9435	3.5346
Score Method-CQR-RR-Cluster	cover^x	ineff	cover^x	ineff	cover^x	ineff	cover^x	ineff
GAE	$0.9578^{\pm 0.0134}$	$3.1297^{\pm 0.1401}$	$0.9578^{\pm 0.0189}$	$3.0985^{\pm 0.1478}$	$0.9527^{\pm 0.0123}$	$3.1614^{\pm 0.1622}$	$0.9520^{\pm 0.0145}$	$\mathbf{2.8927}^{\pm 0.1223}$
DiGAE	$0.9513^{\pm 0.0415}$	$\mathbf{3.0262}^{\pm 0.1412}$	$0.9501^{\pm 0.0312}$	$\mathbf{2.8976}^{\pm 0.1393}$	$0.9507^{\pm 0.0456}$	$2.9347^{\pm 0.1139}$	$0.9442^{\pm 0.0735}$	$3.0321^{\pm 0.2134}$
LGNN	$0.9438^{\pm 0.0396}$	$3.3562^{\pm 0.0355}$	$0.9473^{\pm 0.0423}$	$3.1422^{\pm 0.0423}$	$0.9497^{\pm 0.0323}$	$\mathbf{2.9913}^{\pm 0.0732}$	$0.9507^{\pm 0.0324}$	$3.5195^{\pm 0.1231}$
Average	**0.9510**	**3.1707**	**0.9517**	**3.0461**	**0.9510**	**3.0291**	**0.9490**	**3.1481**

graph. Firstly, we train a traditional GNN model (Conformal GNN) for the node regression task using the training set. The GNN model learns a function $f : \mathcal{G} \to \mathbb{R}^n$, where $\mathcal{G}$ is the input graph and $f(\mathcal{G})_i$ represents the predicted target variable for node i. Node regression minimizes the distance between the direct output and labels.

$$\hat{\boldsymbol{y}} = f_{\text{GNN}}(\mathbf{X}, \mathbf{A}) \tag{19}$$

$$\mathcal{L} = |\boldsymbol{y} - \hat{\boldsymbol{y}}|^2 \tag{20}$$

Here, we use the GNN-based model, GAE, to deal with several cases, which generates the embedding using an encoder function and generate node regression using the decoder function. For the following steps, a Residual GNN predicts the residual of the node labels, generating the weight for the non-conformity measure when computing prediction sets. The details of the algorithm are shown in appendices Algorithm 2.

3.4 RR-GNN ON NODE CLASSIFICATION

The node classification problem is a fundamental task in graph-based machine learning, where the goal is to predict a discrete label or class for each node in a given graph.

We first trained the Conformal GNN model, a function $f : \mathcal{G} \to \{0, 1, ..., K\}^n$, that maps the node features to the corresponding labels with K classes. Compared with regression, node classification uses binary cross-entropy loss as the loss function:

$$\hat{\boldsymbol{y}} = f_{\text{GNN}}(\boldsymbol{X}, \boldsymbol{A}) \tag{21}$$

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} y_{i,k} \log(\hat{y}_{i,k}) \tag{22}$$

We then generate the residual of the predicted value and the actual label as the label for the Residual GNN. We do a softmax operation to get a vector, representing the probability of the node belonging to each class:

$$\hat{\boldsymbol{l}} = \text{softmax}(\hat{\boldsymbol{y}}) \tag{23}$$

The residual $\hat{\boldsymbol{r}}$ is obtained by:

$$\hat{\boldsymbol{r}} = \hat{\boldsymbol{l}} - \boldsymbol{y} \tag{24}$$

Where $\boldsymbol{y}$ is the one-hot label of which class the node belongs to. In Algorithm 4 appendices material, we set a small positive real number ϵ (1e-9) to avoid the denominator equal to 0. In addition, we need the differentiable quantile method in equation (21). Since the non-conformity score is usually differentiable, it only requires differentiable quantile calculation where there are well-established methods available Chernozhukov et al. [2010], Blondel et al. [2020].

4 RESULTS

4.1 EMPIRICAL ANALYSIS

In this section, we showcase the application of the proposed RR-GNN on 15 datasets for edge weight prediction, node regression, and node classification problems. We conduct a comparative analysis of the performance of RR-GNN and four competitors based on two metrics. For the data split, 30% of the data was designated for training, 30% for validation, 20% for testing, and the remaining 20% for calibration.

Datasets: To evaluate the effectiveness of RR-GNN algorithm, we conduct experiments on four categories of benchmark graph datasets: 1) traffic datasets, 2) citation connection datasets, 3) social network datasets, and 4) additional datasets.

Table 2: Results of RR-GNN on Node Regression Datasets

Dataset	GraphSAGE		SGC		GCN		GATS	
Metrics	cover$^{\mathcal{F}}$	ineff	cover$^{\mathcal{F}}$	ineff	cover$^{\mathcal{F}}$	ineff	cover$^{\mathcal{F}}$	ineff
Anaheim: CF-GNN	$0.9520^{\pm 0.0669}$	$\mathbf{1.9231}^{\pm 0.0483}$	$0.9559^{\pm 0.0617}$	$2.2031^{\pm 0.0241}$	$0.9519^{\pm 0.0531}$	$2.3782^{\pm 0.0533}$	$0.9523^{\pm 0.0302}$	$2.1499^{\pm 0.0463}$
Anaheim: Cluster-GNN	$0.9532^{\pm 0.042}$	$1.8954^{\pm 0.037}$	$0.9561^{\pm 0.035}$	$2.1423^{\pm 0.031}$	$0.9528^{\pm 0.041}$	$2.2451^{\pm 0.029}$	$0.9541^{\pm 0.028}$	$2.0321^{\pm 0.025}$
Anaheim: RR-GAE	$0.9539^{\pm 0.038}$	$1.8732^{\pm 0.032}$	$\mathbf{0.9567}^{\pm 0.031}$	$2.0987^{\pm 0.028}$	$0.9532^{\pm 0.036}$	$2.1934^{\pm 0.026}$	$0.9563^{\pm 0.024}$	$1.9623^{\pm 0.022}$
Anaheim: Clsuter-RR-GAE	$\mathbf{0.9543}^{\pm 0.0320}$	$1.9647^{\pm 0.0197}$	$0.9577^{\pm 0.0657}$	$\mathbf{2.0188}^{\pm 0.0246}$	$\mathbf{0.9585}^{\pm 0.0413}$	$\mathbf{2.2179}^{\pm 0.0254}$	$\mathbf{0.9638}^{\pm 0.0302}$	$\underline{\mathbf{1.8996}}^{\pm 0.0249}$
Chicago: CF-GNN	$0.9448^{\pm 0.0519}$	$2.3426^{\pm 0.0384}$	$0.9486^{\pm 0.0247}$	$1.0423^{\pm 0.0372}$	$0.9505^{\pm 0.0447}$	$2.0456^{\pm 0.0443}$	$0.9508^{\pm 0.0569}$	$1.1396^{\pm 0.0686}$
Chicago: Cluster-GNN	$0.9461^{\pm 0.039}$	$2.2894^{\pm 0.034}$	$0.9492^{\pm 0.031}$	$1.1895^{\pm 0.029}$	$0.9513^{\pm 0.037}$	$1.8742^{\pm 0.031}$	$0.9516^{\pm 0.042}$	$1.1254^{\pm 0.045}$
Chicago: RR-GAE	$0.9472^{\pm 0.035}$	$2.2673^{\pm 0.029}$	$\mathbf{0.9498}^{\pm 0.028}$	$1.2567^{\pm 0.026}$	$0.9519^{\pm 0.033}$	$1.6923^{\pm 0.027}$	$0.9519^{\pm 0.038}$	$1.1489^{\pm 0.039}$
Chicago: Cluster-RR-GAE	$\mathbf{0.9476}^{\pm 0.0426}$	$\mathbf{2.2291}^{\pm 0.0325}$	$0.9546^{\pm 0.0328}$	$\mathbf{1.2012}^{\pm 0.0250}$	$\mathbf{0.9538}^{\pm 0.0356}$	$\underline{\mathbf{1.5769}}^{\pm 0.0252}$	$\mathbf{0.9540}^{\pm 0.0362}$	$1.1283^{\pm 0.0256}$
Education: CF-GNN	$0.9501^{\pm 0.0242}$	$2.3808^{\pm 0.0427}$	$0.9500^{\pm 0.0285}$	$2.4892^{\pm 0.0351}$	$0.9483^{\pm 0.0408}$	$2.4380^{\pm 0.0452}$	$0.9502^{\pm 0.0392}$	$2.4209^{\pm 0.0376}$
Education: Cluster-GNN	$0.9513^{\pm 0.031}$	$2.3145^{\pm 0.038}$	$0.9517^{\pm 0.033}$	$2.3721^{\pm 0.032}$	$0.9496^{\pm 0.035}$	$2.2894^{\pm 0.034}$	$0.9518^{\pm 0.036}$	$2.3256^{\pm 0.033}$
Education: RR-GAE	$0.9529^{\pm 0.029}$	$2.1932^{\pm 0.027}$	$0.9534^{\pm 0.030}$	$2.1478^{\pm 0.028}$	$0.9508^{\pm 0.032}$	$2.0321^{\pm 0.029}$	$0.9532^{\pm 0.031}$	$2.1423^{\pm 0.030}$
Education: Cluster-RR-GAE	$\mathbf{0.9599}^{\pm 0.0417}$	$\mathbf{2.0573}^{\pm 0.0280}$	$\mathbf{0.9586}^{\pm 0.0225}$	$\mathbf{2.0445}^{\pm 0.0239}$	$\mathbf{0.9580}^{\pm 0.0333}$	$\mathbf{1.8731}^{\pm 0.0260}$	$\mathbf{0.9594}^{\pm 0.0386}$	$\underline{\mathbf{1.9075}}^{\pm 0.0221}$
Election: CF-GNN	$0.9498^{\pm 0.0211}$	$0.9268^{\pm 0.0429}$	$0.9495^{\pm 0.0215}$	$0.9279^{\pm 0.0302}$	$0.9506^{\pm 0.0473}$	$0.9009^{\pm 0.0282}$	$0.9488^{\pm 0.0363}$	$0.9136^{\pm 0.0681}$
Election: Cluster-GNN	$0.9503^{\pm 0.028}$	$0.9152^{\pm 0.038}$	$0.9501^{\pm 0.027}$	$0.9124^{\pm 0.035}$	$0.9512^{\pm 0.041}$	$0.8723^{\pm 0.031}$	$0.9496^{\pm 0.033}$	$0.8945^{\pm 0.042}$
Election: RR-GAE	$0.9509^{\pm 0.025}$	$0.9037^{\pm 0.029}$	$0.9523^{\pm 0.024}$	$0.8956^{\pm 0.028}$	$0.9518^{\pm 0.036}$	$0.8234^{\pm 0.026}$	$0.9514^{\pm 0.030}$	$0.8562^{\pm 0.035}$
Election: Cluster-RR-GAE	$\mathbf{0.9558}^{\pm 0.0215}$	$\mathbf{0.9213}^{\pm 0.0279}$	$\mathbf{0.9567}^{\pm 0.0242}$	$\mathbf{0.9487}^{\pm 0.0259}$	$\mathbf{0.9510}^{\pm 0.0432}$	$\mathbf{0.9343}^{\pm 0.0341}$	$\mathbf{0.9567}^{\pm 0.0317}$	$\underline{\mathbf{0.6698}}^{\pm 0.0201}$

We apply RR-GNN on the traffic network and traffic flow data from Chicago and Anaheim to predict each node's edge weight and traffic volume Stabler et al. [2018]. Chicago dataset consists of 541 nodes representing road junctions and 2150 edges representing road segments with directions, while the Anaheim dataset consists of 413 nodes and 858 edges. In this context, each node is characterized by a two-dimensional feature $\boldsymbol{X}_{i,:} \in \mathbb{R}^2$ representing its coordinates, and each edge is associated with a weight that signifies the traffic volume passing through the corresponding road segment. We collect three widely used citation network datasets for the citation datasets: Cora, PubMed, and CiteSeer. We apply RR-GNN to paper classification and citation prediction. Social network datasets like Twitch, CS, and Physics have become increasingly important resources for graph machine-learning research.

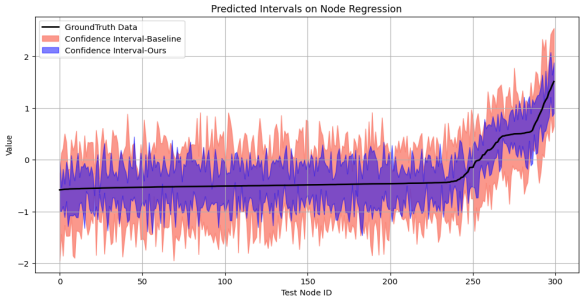

Figure 2: The prediction interval of node regression generated by CF-GNN and RR-GAE. The x-axis represents the node, which is sorted by the label. The y-axis represents the prediction intervals of nodes. The error rate α is 0.05. Blue and red represent the results of RR-GAE and CFNN-GAE, respectively.

Metrics: We use inefficiency (ineff) and weighted symmetric calibration (WSC) as evaluation metrics (details of ineff and WSC can be accessed in appendices material). Lower ineff and higher WSC indicate better performance. To generate prediction intervals, we independently sample 1000 vectors v from the unit sphere in $\mathbb{R}^{2m}$ space. The parameters a, b, δ are fine-tuned via grid search. Additionally, 25% of

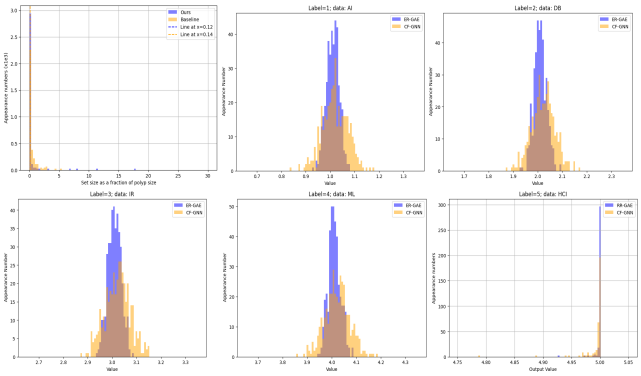

Figure 3: The histogram of predicted values of the node classification task on dataset MedPub. The error rate α is 0.05. Each sub-figure represents one classification from 0 to 5. The x-axis is the predicted value from the model. The y-axis is the frequency corresponding to the predicted value. Blue and yellow represent the results of RR-GAE and CF-GNN, respectively.

test data is utilized to estimate optimal v, a, b, δ values. The conditional coverage is then calculated on the remaining 75% of test data.

Models and baselines: There are three basic GNN models for the model: 1) GAE (Section 3.2.1), 2) line graph neural network (LGNN Cai et al. [2021]) and 3) a directed variant of GAE, called DiGAE. We use two nonconformity score, including 1) CP Huang et al. [2024] and 2) conformal quantile regression (CQR). For the encoder in the basic GNN models, we choose 4 different structures: 1) GraphSAGE Hamilton et al. [2017], 2) SAGEConv Morris et al. [2019], 3) GCN Kipf and Welling [2016a], and 4) GAT Veličković et al. [2017]. We name the model that combines conformal prediction (CP, as described in work Huang et al. [2024]) with graph autoencoder (GAE, Section 3.2.1) as CP-GAE. Similarly, we name the model with the nonconformity score and the GNN models. For example, the models use CQR, and GAE is referred to as CQR-GAE. As for node classification task, three models are added: HAN, Ensemble TS

and CaGCN which are latest models baseline on this task. Additionally, if the residual reweighting approach is performed based on the CQR approach, we also add RR to the name of the models. For example, the residual reweighted CQR-GAE is referred to as CQR-RR-GAE. The baseline models are the above models without residual reweight.

We use four popular graph neural network (GNN) model structures for encoder and decoder - GCN Kipf and Welling [2016a], GraphConv Morris et al. [2019], GAT Veličković et al. [2017], and GraphSAGE Hamilton et al. [2017] - as the base graph convolution layers for both the CP and CQR based models.

Experiment result: As for the task of edge weight prediction, we ran the experiment 10 times and split the data into training, validation, and the combined calibration and test sets for each dataset and model. We conduct 100 random splits of calibration and testing edges to perform the baseline model and RR-GNN and evaluate the empirical coverage. Our method achieved 6.15% to 28.87% reduce on inefficiency interval length shown in table 1 and 2 on the task of edge weight prediction. In addition, we conducted a paired t-testing on the mean inefficiency values for our method (RR) compared to CP and CQR, taking the meaning value of each (12 values in total in Table 1) after repeating the experiments 10 times. The p-values are 0.00035 and 0.00024. Coverage is defined as the probability that the ground truth value lies within the predicted confidence interval. Our method allows control of the coverage through the parameter α, where the expected coverage is $1 - \alpha$. In our manuscript, we set $\alpha = 0.05$, corresponding to a target coverage of 0.95. As shown in Tables 1-3 of the manuscript, the empirical coverages achieved by our method are close to 0.95, indicating that the coverage is well controlled and aligns with the expected theoretical value.

We conducted a conditional coverage Equation 37 in appendices material) of RR-GNN and baseline methods on the Chicago and Anaheim traffic dataset for the edge weight prediction task. The results presented in Table 1 show that the overall RR-GNN models outperform others in terms of inefficiency (as defined in Equation 36 in appendices material) and conditional coverage (Equation 37 in appendices material). This indicates that the RR variants can strike a better balance between capturing the uncertainty in the predictions and maintaining a high level of accuracy. We also find that GAE and LGNN outperform DiGAE, highlighting the efficacy of the autoencoder approach in weight prediction from Table 1. We showcase the prediction interval produced by model of LGNN with RR(CQR) in Figure 4. Furthermore, Figure 4 also illustrates the adaptability of the RR models by generating the smallest prediction intervals of varying sizes, which aligns with the data characteristics.

For the node regression task, we apply the models to 7 different datasets: 1) Anaheim traffic dataset, 2) Chicago traffic dataset, 3) Education dataset, 4) Income dataset, 5) Unemploy dataset, and 7) Twitch dataset. Table 4 in appendices material shows that our method outperforms the baseline model, CF-GNN Huang et al. [2024], both on WSC score (coverage) and inefficiency on 7 datasets. Besides, the visualization result in Figure 2 shows that we have a smaller interval size than that from CF-GNN. Similarly, for the task-node classification, we compared RR-GNN and CF-GNN on 8 datasets: 1) Cora, 2) DBLP, 3) CiteSeer, 4) PubMed, 5) Computer, 6) Photo, 7) CS and 8) Physics. Number and visual results can be seen in Table 4 and Figure 3. Our model achieves better results in both accuracy and inefficiency. It is worth to mention that we employ categorical cross-entropy as the loss function for multi-class classification. The loss function is given by:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{i,k}\log(\hat{y}_{i,k}), \tag{25}$$

where $\hat{y}_{i,k}$ represents the predicted probability of node i belonging to class k, and $\sum_{k=1}^{K}\hat{y}_{i,k} = 1$.

In summary, leveraging the RR-based models can generate prediction intervals that are both efficient and well-calibrated, making them a more suitable choice for different network-based tasks, especially real-world transportation applications where accurate and reliable predictions are crucial for informed decision-making.

Furthermore, we compared the predicted residual between CF-GNN and RR-GNN. Figure 7 in the appendices material shows the predicted residual of road's traffic volume for Chicago and Anathm. The residual value of CF-GNN is higher than that of RR-GAE. Figure 6 in appendices material also shows the residual of the U.S.A election result, where we can see that the global residual/difference between the model output and ground truth from RR-GAE is much lower than these baselines.

4.2 ABLATION STUDY

To address the computational cost concerns, we tested the time and memory usage on random graphs and analyzed the time complexity of the algorithm.

Specifically, we tested three types of random graphs: Erdős-Rényi Erdos [1961], Barabási-Albert Lei et al. [2018], and Watts-Strogatz small-world model Romano et al. [2019]. For each type, we generated graphs with node counts of 100, 10000, 50000, and 100000, respectively. Each node had 128 features, and the models were trained for 100 epochs. We present results using GAE (Graph Autoencoder) and DiGAE, combined with three different GNN (Graph Neural Network) encoders: GraphSAGE, GCN (Graph Convolutional Network), and GAT (Graph Attention Network) shown in Table8. The used GPU space is increasing linearly along with the node size, while processing time increases at

Table 3: Results of Ours (RR-GNN) on Node Classification Datasets

Dataset	HAN		SGC		CaGCN		GATS	
Dataset	cover^x	ineff	cover^x	ineff	cover^x	ineff	cover^x	ineff
Cora: CF-GNN	$0.9456^{\pm 0.0569}$	$1.6284^{\pm 0.0483}$	$0.9461^{\pm 0.0603}$	$1.6633^{\pm 0.0441}$	$0.9473^{\pm 0.0556}$	$1.6344^{\pm 0.0418}$	$0.9464^{\pm 0.0702}$	$1.6278^{\pm 0.0334}$
Cora: Cluster-GAE	$0.9458^{\pm 0.0532}$	$1.61201^{\pm 0.0431}$	$0.9459^{\pm 0.0612}$	$1.6537^{\pm 0.0432}$	$0.9385^{\pm 0.0529}$	$1.6188^{\pm 0.0328}$	$0.9482^{\pm 0.0453}$	$1.6013^{\pm 0.0313}$
Cora: RR-GAE	$0.9460^{\pm 0.0542}$	$1.6100^{\pm 0.0415}$	$0.9462^{\pm 0.0581}$	$1.6297^{\pm 0.0428}$	$0.9432^{\pm 0.0573}$	$1.6251^{\pm 0.0367}$	$0.9475^{\pm 0.0624}$	$1.6146^{\pm 0.0351}$
Cora: Cluster-RR-GAE	$\mathbf{0.9478}^{\pm 0.0523}$	$\mathbf{1.5896}^{\pm 0.0354}$	$\mathbf{0.9490}^{\pm 0.0643}$	$\mathbf{1.5907}^{\pm 0.0432}$	$\mathbf{0.9465}^{\pm 0.0759}$	$\mathbf{1.6175}^{\pm 0.0354}$	$\mathbf{0.9508}^{\pm 0.0554}$	$\mathbf{1.6114}^{\pm 0.0287}$
DBLP: CF-GNN	$0.9501^{\pm 0.0523}$	$1.5723^{\pm 0.0683}$	$\mathbf{0.9451}^{\pm 0.0617}$	$1.5274^{\pm 0.0416}$	$0.9473^{\pm 0.0596}$	$1.5644^{\pm 0.0733}$	$0.9467^{\pm 0.0717}$	$1.5729^{\pm 0.0463}$
DBLP: Cluster-GAE	$0.9497^{\pm 0.0512}$	$1.5489^{\pm 0.0492}$	$0.9457^{\pm 0.0583}$	$1.4873^{\pm 0.0449}$	$0.9452^{\pm 0.0684}$	$1.5569^{\pm 0.0317}$	$0.9479^{\pm 0.0673}$	$1.5814^{\pm 0.0376}$
DBLP: RR-GAE	$0.9499^{\pm 0.0531}$	$1.5351^{\pm 0.0473}$	$0.9462^{\pm 0.0528}$	$1.4286^{\pm 0.0541}$	$0.9458^{\pm 0.0702}$	$1.5512^{\pm 0.0295}$	$0.9485^{\pm 0.0589}$	$1.5725^{\pm 0.0349}$
DBLP: Cluster-RR-GAE	$\mathbf{0.9518}^{\pm 0.0509}$	$\mathbf{1.5467}^{\pm 0.0427}$	$0.9503^{\pm 0.0428}$	$\mathbf{1.3563}^{\pm 0.0626}$	$\mathbf{0.9484}^{\pm 0.0624}$	$\mathbf{1.5371}^{\pm 0.0248}$	$\mathbf{0.9505}^{\pm 0.0469}$	$\mathbf{1.5570}^{\pm 0.0356}$
CiteSeer: CF-GNN	$0.9528^{\pm 0.0203}$	$1.1680^{\pm 0.0439}$	$0.9525^{\pm 0.0257}$	$\mathbf{1.1827}^{\pm 0.0552}$	$0.9496^{\pm 0.0392}$	$1.2310^{\pm 0.0332}$	$0.9508^{\pm 0.0309}$	$1.2396^{\pm 0.0416}$
CiteSeer: Cluster-GAE	$0.9532^{\pm 0.0218}$	$1.1653^{\pm 0.0427}$	$0.9561^{\pm 0.0274}$	$1.1854^{\pm 0.0483}$	$0.9507^{\pm 0.0365}$	$1.2237^{\pm 0.0311}$	$0.9523^{\pm 0.0332}$	$1.2298^{\pm 0.0384}$
CiteSeer: RR-GAE	$0.9538^{\pm 0.0853}$	$1.1621^{\pm 0.0552}$	$0.9579^{\pm 0.0536}$	$1.1782^{\pm 0.0415}$	$0.9512^{\pm 0.0358}$	$1.2189^{\pm 0.0276}$	$0.9535^{\pm 0.0447}$	$1.2085^{\pm 0.0361}$
CiteSeer: Cluster-RR-GAE	$\mathbf{0.9556}^{\pm 0.0918}$	$\mathbf{1.1539}^{\pm 0.0615}$	$\mathbf{0.9598}^{\pm 0.0561}$	$1.1678^{\pm 0.0372}$	$\mathbf{0.9526}^{\pm 0.0363}$	$\mathbf{1.2016}^{\pm 0.0289}$	$\mathbf{0.9562}^{\pm 0.0428}$	$\mathbf{1.1408}^{\pm 0.0361}$
PubMed: CF-GNN	$0.9502^{\pm 0.0207}$	$1.4680^{\pm 0.0361}$	$0.9508^{\pm 0.0276}$	$1.4272^{\pm 0.0325}$	$0.9516^{\pm 0.0458}$	$1.5310^{\pm 0.0514}$	$0.9512^{\pm 0.0434}$	$1.4396^{\pm 0.0485}$
PubMed: Cluster-GAE	$0.9507^{\pm 0.0352}$	$1.3985^{\pm 0.0374}$	$0.9513^{\pm 0.0419}$	$1.4083^{\pm 0.0341}$	$0.9519^{\pm 0.0462}$	$1.4521^{\pm 0.0483}$	$0.9514^{\pm 0.0427}$	$1.4198^{\pm 0.0491}$
PubMed: RR-GAE	$0.9510^{\pm 0.0386}$	$1.3528^{\pm 0.0357}$	$0.9516^{\pm 0.0453}$	$1.3992^{\pm 0.0328}$	$0.9520^{\pm 0.0469}$	$1.3815^{\pm 0.0301}$	$0.9515^{\pm 0.0432}$	$1.4085^{\pm 0.0503}$
PubMed: Cluster-RR-GAE	$\mathbf{0.9526}^{\pm 0.0483}$	$\mathbf{1.3275}^{\pm 0.0392}$	$\mathbf{0.9520}^{\pm 0.0482}$	$\mathbf{1.3897}^{\pm 0.0339}$	$\mathbf{0.9521}^{\pm 0.0473}$	$\mathbf{1.3732}^{\pm 0.0296}$	$\mathbf{0.9515}^{\pm 0.0419}$	$\mathbf{1.3989}^{\pm 0.0522}$
Computers: CF-GNN	$0.9471^{\pm 0.0276}$	$3.3680^{\pm 0.3499}$	$0.9492^{\pm 0.0235}$	$3.8272^{\pm 0.0292}$	$0.9457^{\pm 0.0435}$	$3.2310^{\pm 0.0652}$	$0.9478^{\pm 0.0325}$	$3.1396^{\pm 0.0586}$
Computers: Cluster-GAE	$0.9476^{\pm 0.0321}$	$3.1523^{\pm 0.3287}$	$0.9490^{\pm 0.0273}$	$3.4821^{\pm 0.0315}$	$0.9461^{\pm 0.0418}$	$2.8945^{\pm 0.0583}$	$0.9479^{\pm 0.0382}$	$2.9634^{\pm 0.0541}$
Computers: RR-GAE	$0.9481^{\pm 0.0473}$	$2.8937^{\pm 0.0328}$	$0.9493^{\pm 0.0298}$	$2.7324^{\pm 0.0394}$	$0.9464^{\pm 0.0436}$	$2.6745^{\pm 0.0352}$	$0.9479^{\pm 0.0623}$	$2.8033^{\pm 0.0259}$
Computers: Cluster-RR-GAE	$\mathbf{0.9503}^{\pm 0.0553}$	$\mathbf{2.7423}^{\pm 0.0258}$	$\mathbf{0.9509}^{\pm 0.0315}$	$\underline{\mathbf{2.6343}}^{\pm 0.0413}$	$\mathbf{0.9418}^{\pm 0.0436}$	$\mathbf{2.5471}^{\pm 0.0365}$	$\mathbf{0.9354}^{\pm 0.0584}$	$\underline{\mathbf{2.7739}}^{\pm 0.0272}$
Photo: CF-GNN	$0.9511^{\pm 0.0275}$	$3.2680^{\pm 0.0395}$	$0.9515^{\pm 0.0263}$	$2.2276^{\pm 0.0354}$	$0.9486^{\pm 0.0419}$	$2.2010^{\pm 0.0387}$	$0.9509^{\pm 0.0391}$	$2.1986^{\pm 0.0286}$
Photo: Cluster-GAE	$0.9523^{\pm 0.0289}$	$3.0125^{\pm 0.0362}$	$0.9517^{\pm 0.0291}$	$2.1224^{\pm 0.0338}$	$0.9491^{\pm 0.0396}$	$2.1076^{\pm 0.0352}$	$0.9510^{\pm 0.0374}$	$2.0059^{\pm 0.0263}$
Photo: RR-GAE	$0.9527^{\pm 0.0852}$	$2.7843^{\pm 0.0415}$	$0.9518^{\pm 0.0894}$	$2.0451^{\pm 0.0331}$	$0.9495^{\pm 0.0821}$	$2.0128^{\pm 0.0513}$	$0.9511^{\pm 0.0439}$	$1.9015^{\pm 0.0254}$
Photo: Cluster-RR-GAE	$\mathbf{0.9554}^{\pm 0.0723}$	$\mathbf{2.5474}^{\pm 0.0456}$	$\mathbf{0.9534}^{\pm 0.0913}$	$\mathbf{2.0026}^{\pm 0.0316}$	$\mathbf{0.9504}^{\pm 0.0342}$	$\mathbf{2.0003}^{\pm 0.0370}$	$\mathbf{0.9498}^{\pm 0.0512}$	$\mathbf{1.7093}^{\pm 0.0234}$
CS: CF-GNN	$0.9438^{\pm 0.0224}$	$1.8669^{\pm 0.0347}$	$0.9435^{\pm 0.0284}$	$1.6272^{\pm 0.0452}$	$0.9476^{\pm 0.0416}$	$3.6310^{\pm 0.0325}$	$0.9478^{\pm 0.0317}$	$2.7396^{\pm 0.0286}$
CS: Cluster-GAE	$0.9451^{\pm 0.0253}$	$1.8324^{\pm 0.0332}$	$0.9448^{\pm 0.0316}$	$1.6229^{\pm 0.0428}$	$0.9483^{\pm 0.0387}$	$3.1957^{\pm 0.0301}$	$0.9481^{\pm 0.0293}$	$2.5641^{\pm 0.0269}$
CS: RR-GAE	$0.9472^{\pm 0.0573}$	$1.8453^{\pm 0.0365}$	$0.9461^{\pm 0.0528}$	$1.6205^{\pm 0.0384}$	$0.9435^{\pm 0.0546}$	$2.8932^{\pm 0.0275}$	$0.9483^{\pm 0.0362}$	$2.4785^{\pm 0.0241}$
CS: Cluster-RR-GAE	$\mathbf{0.9502}^{\pm 0.0601}$	$\mathbf{1.8430}^{\pm 0.0361}$	$\mathbf{0.9501}^{\pm 0.0528}$	$\mathbf{1.6183}^{\pm 0.0361}$	$\mathbf{0.9516}^{\pm 0.0525}$	$\underline{\mathbf{2.5469}}^{\pm 0.0227}$	$\mathbf{0.9485}^{\pm 0.0329}$	$\mathbf{2.3889}^{\pm 0.0238}$
Physics: CF-GNN	$0.9495^{\pm 0.0243}$	$1.2218^{\pm 0.0463}$	$0.9507^{\pm 0.0292}$	$1.2430^{\pm 0.0324}$	$0.9489^{\pm 0.0257}$	$1.2005^{\pm 0.0604}$	$0.9505^{\pm 0.0275}$	$1.2243^{\pm 0.0246}$
Physics: Cluster-GAE	$0.9498^{\pm 0.0267}$	$1.2205^{\pm 0.0428}$	$0.9510^{\pm 0.0319}$	$1.2418^{\pm 0.0346}$	$0.9491^{\pm 0.0283}$	$1.2069^{\pm 0.0551}$	$0.9506^{\pm 0.0298}$	$1.2231^{\pm 0.0239}$
Physics: RR-GAE	$0.9501^{\pm 0.0573}$	$1.2198^{\pm 0.0283}$	$0.9512^{\pm 0.0501}$	$1.2412^{\pm 0.0385}$	$0.9493^{\pm 0.0326}$	$1.2145^{\pm 0.0423}$	$0.9507^{\pm 0.0442}$	$1.2298^{\pm 0.0249}$
Physics: Cluster-RR-GAE	$\mathbf{0.9518}^{\pm 0.0511}$	$\mathbf{1.2050}^{\pm 0.0223}$	$\mathbf{0.9528}^{\pm 0.0542}$	$\mathbf{1.2279}^{\pm 0.0419}$	$\mathbf{0.9508}^{\pm 0.0334}$	$\mathbf{1.1998}^{\pm 0.0438}$	$\mathbf{0.9522}^{\pm 0.0493}$	$\mathbf{1.2187}^{\pm 0.0238}$

a rate lower than linear with respect to the size. Our findings indicate that the time and space requirements are within acceptable limits for these configurations.

Our method, RR-GNN, involves training two GNN models with a cross-training protocol. Subsequently, RR-GNN uses Louvain clustering to generate interval predictions. The time complexity of the GNN component is $O(m)$, where m represents the number of edges Wu et al. [2020]. The Louvain clustering algorithm has a time complexity of $O(m \log m)$ Kumar et al. [2016]. For the cross-training protocol, we set a fixed number of training iterations. Therefore, the overall time complexity of RR-GNN can be approximated as $O(m \log m)$. Besides, in order to evaluate our method with the best current methods, we list the results in the appendices material. We test Graphormer and GT in tables 6 and 7 in the appendices material. We can see that our RR operation can help improve the performance of Graphormer and GT. In addition, we compare SAN, which is a transformer-based method in Table 7 in the appendices section. The result shows that our RR-GCN is better than SAN overall.

5 CONCLUSION

This paper introduces the Residual Reweighted Conformal Prediction Graph Neural Network (RR-GNN), which enhances graph neural networks (GNNs) by integrating conformal prediction (CP). While traditional GNNs yield point predictions, RR-GNN provides predictive regions reflecting varying confidence levels. Existing nonconformity measures often produce uniform-width regions, neglecting the differing prediction difficulties. RR-GNN overcomes this by employing a novel residual reweighting nonconformity measure that adjusts predictive region widths based on expected accuracy for each example. We validate RR-GNN's effectiveness on 15 datasets, including real-world datasets, like transportation and social networks, across tasks like edge weight prediction and node classification. RR-GNN consistently delivers tighter predictive regions, higher accuracy, and improved efficiency compared to standard GNN methods, advancing uncertainty-aware predictions in graph machine learning.

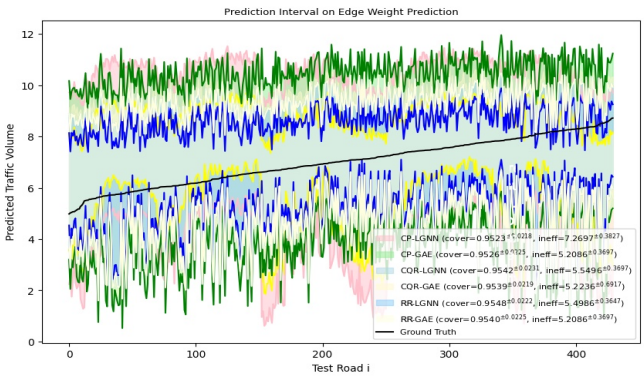

Figure 4: The graph shows the traffic volume prediction intervals generated on the Chicago traffic dataset. All methods set their error rate α at 0.05. The x-axis represents individual roads sorted by their actual/ground truth traffic volumes. The y-axis represents the predicted intervals. Different colors distinguish the results from different prediction methods.

ACKNOWLEDGMENT

This work was partially supported by Hong Kong RGC, City University of Hong Kong grants (Project No. 9610639 and 6000864), and Chengdu Municipal Office of Philosophy and Social Science grant 2024BS013.

REFERENCES

Charu C Aggarwal. Data mining (pdfdrive. com). 2015.

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

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

Jie Bao, Chuangyin Dang, Rui Luo, Hanwei Zhang, and Zhixin Zhou. Enhancing adversarial robustness with conformal prediction: A framework for guaranteed model reliability. In *Proceedings of the Forty-second International Conference on Machine Learning (ICML)*, 2025. to appear.

Stephen Bates, Anastasios Angelopoulos, Lihua Lei, Jitendra Malik, and Michael Jordan. Distribution-free, risk-controlling prediction sets. *Journal of the ACM (JACM)*, 68(6):1–34, 2021.

Smriti Bhagat, Graham Cormode, and S Muthukrishnan. Node classification in social networks. *Social network data analytics*, pages 115–148, 2011.

Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. Fast differentiable sorting and ranking. In *International Conference on Machine Learning*, pages 950–959. PMLR, 2020.

Lei Cai, Jundong Li, Jie Wang, and Shuiwang Ji. Line graph neural networks for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5103–5113, 2021.

Maxime Cauchois, Suyash Gupta, and John Duchi. Knowing what you know: valid and validated confidence sets in multiclass and multilabel prediction. *arXiv preprint arXiv:2004.10181*, 2020.

Victor Chernozhukov, Iván Fernández-Val, and Alfred Galichon. Quantile and probability curves without crossing. *Econometrica*, 78(3):1093–1125, 2010.

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

Charles Cowell, Pamela Clinton Hopkins, Rochell McWhorter, and Debra L Jorden. Alternative training models. *Advances in Developing Human Resources*, 8(4): 460–475, 2006.

Paul Erdos. On the evolution of random graphs. *Bulletin of the Institute of International Statistics*, 38:343–347, 1961.

Isaac Gibbs and Emmanuel Candes. Adaptive conformal inference under distribution shift. *Advances in Neural Information Processing Systems*, 34:1660–1672, 2021.

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

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

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Xing Han, Ziyang Tang, Joydeep Ghosh, and Qiang Liu. Split localized conformal prediction. *arXiv preprint arXiv:2206.13092*, 2022.

Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 603–618, 2017.

Hans Hao-Hsun Hsu, Yuesong Shen, Christian Tomani, and Daniel Cremers. What makes graph neural networks miscalibrated? *Advances in Neural Information Processing Systems*, 35:13775–13786, 2022.

Kexin Huang, Ying Jin, Emmanuel Candes, and Jure Leskovec. Uncertainty quantification over graph with conformalized graph neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.

Roshni G Iyer, Wei Wang, and Yizhou Sun. Bi-level attention graph neural networks. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 1126–1131. IEEE, 2021.

Junteng Jia and Austion R Benson. Residual correlation in graph neural network regression. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 588–598, 2020.

Ying Jin and Emmanuel J Candès. Selection by prediction with conformal p-values. *Journal of Machine Learning Research*, 24(244):1–41, 2023.

Ying Jin, Zhimei Ren, and Emmanuel J Candès. Sensitivity analysis of individual treatment effects: A robust conformal inference approach. *Proceedings of the National Academy of Sciences*, 120(6):e2214889120, 2023.

Ulf Johansson, Henrik Boström, Tuve Löfström, and Henrik Linusson. Regression conformal prediction with random forests. *Machine learning*, 97:155–176, 2014.

Sayash Kapoor and Arvind Narayanan. Leakage and the reproducibility crisis in machine-learning-based science. *Patterns*, 4(9), 2023.

Christopher Kath and Florian Ziel. Conformal prediction interval estimation and applications to day-ahead and intraday power markets. *International Journal of Forecasting*, 37(2):777–799, 2021.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016a.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016b.

Georgios Kollias, Vasileios Kalantzis, Tsuyoshi Idé, Aurélie Lozano, and Naoki Abe. Directed graph auto-encoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7211–7219, 2022.

Vikram Krishnamurthy, Rui Luo, and Buddhika Nettasinghe. Segregation in social networks: Markov bridge models and estimation. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5484–5488. IEEE, 2021.

Vikas Kumar, Anubhav Sisodia, Umesh Maini, and Abhineet Anand. Comparing algorithms of community structure in networks. *Indian Journal of Science and Technology*, 9(44):1–5, 2016.

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

Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523):1094–1111, 2018.

Lihua Lei and Emmanuel J Candès. Conformal inference of counterfactuals and individual treatment effects. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 83(5):911–938, 2021.

Michelle M Li, Kexin Huang, and Marinka Zitnik. Graph representation learning in biomedicine and healthcare. *Nature Biomedical Engineering*, 6(12):1353–1369, 2022.

Rui Luo and Nicolo Colombo. Entropy reweighted conformal classification. In *The 13th Symposium on Conformal and Probabilistic Prediction with Applications*, pages 264–276. PMLR, 2024.

Rui Luo and Nicolo Colombo. Conformal load prediction with transductive graph autoencoders. *Machine Learning*, 114(3):1–22, 2025.

Rui Luo and Vikram Krishnamurthy. Fréchet-statistics-based change point detection in dynamic social networks. *IEEE Transactions on Computational Social Systems*, 11 (2):2863–2871, 2023a.

Rui Luo and Vikram Krishnamurthy. Who you play affects how you play: Predicting sports performance using graph attention networks with temporal convolution. *arXiv preprint arXiv:2303.16741*, 2023b.

Rui Luo and Zhixin Zhou. Trustworthy classification through rank-based conformal prediction sets. *arXiv preprint arXiv:2407.04407*, 2024.

Rui Luo and Zhixin Zhou. Conformity score averaging for classification. In *Proceedings of the Forty-second International Conference on Machine Learning (ICML)*, 2025a. to appear.

Rui Luo and Zhixin Zhou. Conditional conformal risk adaptation. *arXiv preprint arXiv:2504.07611*, 2025b.

Rui Luo and Zhixin Zhou. Conformalized interval arithmetic with symmetric calibration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(18):19207–19215, 2025c.

Rui Luo and Zhixin Zhou. Conformal thresholded intervals for efficient regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(18):19216–19223, 2025d.

Rui Luo and Zhixin Zhou. Volume-sorted prediction set: Efficient conformal prediction for multi-target regression. *arXiv preprint arXiv:2503.02205*, 2025e.

Rui Luo, Buddhika Nettasinghe, and Vikram Krishnamurthy. Echo chambers and segregation in social networks: Markov bridge models and estimation. *IEEE transactions on computational social systems*, 9(3):891–901, 2021.

Rui Luo, Vikram Krishnamurthy, and Erik Blasch. Mitigating misinformation spread on blockchain enabled social media networks. *arXiv preprint arXiv:2201.07076*, 2022a.

Rui Luo, Buddhika Nettasinghe, and Vikram Krishnamurthy. Controlling segregation in social network dynamics as an edge formation game. *IEEE transactions on network science and engineering*, 9(4):2317–2329, 2022b.

Rui Luo, Buddhika Nettasinghe, and Vikram Krishnamurthy. Anomalous edge detection in edge exchangeable social network models. In *Conformal and probabilistic prediction with applications*, pages 287–310. PMLR, 2023.

Rui Luo, Jie Bao, Zhixin Zhou, and Chuangyin Dang. Game-theoretic defenses for robust conformal prediction against adversarial attacks in medical imaging. *arXiv preprint arXiv:2411.04376*, 2024.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019.

Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alex Gammerman. Inductive confidence machines for regression. In *Machine Learning: ECML 2002: 13th European Conference on Machine Learning Helsinki, Finland, August 19–23, 2002 Proceedings 13*, pages 345–356. Springer, 2002.

Harris Papadopoulos, Alex Gammerman, and Volodya Vovk. Normalized nonconformity measures for regression conformal prediction. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2008)*, pages 64–69, 2008.

Harris Papadopoulos, Vladimir Vovk, and Alex Gammerman. Regression conformal prediction with nearest neighbours. *Journal of Artificial Intelligence Research*, 40: 815–840, 2011.

Alexandra Peste, Eugenia Iofinova, Adrian Vladu, and Dan Alistarh. Ac/dc: Alternating compressed/decompressed training of deep neural networks. *Advances in neural information processing systems*, 34:8557–8570, 2021.

Yaniv Romano, Evan Patterson, and Emmanuel Candes. Conformalized quantile regression. *Advances in neural information processing systems*, 32, 2019.

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

Bidisha Samanta, Abir De, Gourhari Jana, Vicenç Gómez, Pratim Kumar Chattaraj, Niloy Ganguly, and Manuel Gomez-Rodriguez. Nevae: A deep generative model for molecular graphs. *The Journal of Machine Learning Research*, 21(1):4556–4588, 2020.

Ben Stabler, Hillel Bar-Gera, and Elizabeth Sall. Transportation networks for research core team. *Transportation Networks for Research. Accessed Month, Day, Year.[Electronic resource]: https://github.com/bstabler/TransportationNetworks (accessed 16.02. 2021)*, 2018.

Ingo Steinwart and Andreas Christmann. Estimating conditional quantiles with the help of the pinball loss. *Bernoulli*, 17(1):211 – 225, 2011. doi: 10.3150/10-BEJ267. URL https://doi.org/10.3150/10-BEJ267.

Xiaoyi Su, Zhixin Zhou, and Rui Luo. Adaptive conformal inference by particle filtering under hidden markov models. *arXiv preprint arXiv:2411.01558*, 2024.

Lingxuan Tang, Rui Luo, Zhixin Zhou, and Nicolo Colombo. Enhanced route planning with calibrated uncertainty set. *Machine Learning*, 114(5):1–16, 2025.

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

Praneeth Vepakomma, Abhishek Singh, Otkrist Gupta, and Ramesh Raskar. Nopeek: Information leakage reduction to share activations in distributed deep learning. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pages 933–942. IEEE, 2020.

Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. *Advances in neural information processing systems*, 31, 2018.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*, volume 29. Springer, 2005.

Ting Wang, Zhixin Zhou, and Rui Luo. Enhancing trustworthiness of graph neural networks with rank-based conformal training. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(20):21261–21268, 2025.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The world wide web conference*, pages 2022–2032, 2019.

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

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

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Mingzhang Yin, Claudia Shi, Yixin Wang, and David M Blei. Conformal sensitivity analysis for individual treatment effects. *Journal of the American Statistical Association*, 119(545):122–135, 2024.

Margaux Zaffran, Olivier Féron, Yannig Goude, Julie Josse, and Aymeric Dieuleveut. Adaptive conformal predictions for time series. In *International Conference on Machine Learning*, pages 25834–25866. PMLR, 2022.

Jize Zhang, Bhavya Kailkhura, and T Yong-Jin Han. Mix-n-match: Ensemble and compositional methods for uncertainty calibration in deep learning. In *International conference on machine learning*, pages 11117–11128. PMLR, 2020.

Unai Zulaika, Ruben Sanchez-Corcuera, Aitor Almeida, and Diego Lopez-de Ipina. Lwp-wl: Link weight prediction based on cnns and the weisfeiler–lehman algorithm. *Applied Soft Computing*, 120:108657, 2022.

6 APPENDICES

6.1 OTHER EXPERIMENT RESULTS

In order to test our method's effcience on more datasets, we did 4 experiments. First is the extended test on node regression on other four datasets shown as 4. The second result shows node classification results comparing with conformal baselines can be found in Table 5. Thirdly, in order to prove our method's generlization, we incorporated additional GNN models into our experiments using the Anaheim Data and Chicago Data datasets. It needs to be clarified that our method can be combined with all other GNN models (Graphormer [1], GT [2]). Details are shown Table 6 7.

6.2 BACKGROUND OF GRAPH AUTOENCODER

6.2.1 Guaranteed Edge Weight Prediction Using GNNs

Let $\mathcal{G} = (\mathbb{V}, \boldsymbol{E})$ be a graph with node set V and edge set $E \subseteq \mathbb{V} \times \mathbb{V}$. Assume the graph has n nodes with f features. Let $\boldsymbol{X} \in R^{n \times m}$ be the node feature matrix, and $X_{i,:} \in R^f$ be the feature vector of the i^{th} node. The binary adjacency matrix of $\mathcal{G}$, $\boldsymbol{A} \in \{0,1\}^{n \times n}$, encodes the binary (unweighted) connecting structure of the graph. It is defined by:

$$A_{ij} = \begin{cases} 1, & \text{if } (i,j) \in \boldsymbol{E}; \\ 0, & \text{otherwise.} \end{cases} \tag{26}$$

Then, we define the weight matrix as $\boldsymbol{W} \in R^{n \times n} \geq 0$, where W_{ij} denotes the weight rather than the binary of the edge connecting node i to node j. In the context of a road system, for example, we can interpret W_{ij} as the volume of traffic transitioning from junction i to junction j.

We partition the edge set $\boldsymbol{E}$ into three disjoint subsets: $\boldsymbol{E}^{train}$, $\boldsymbol{E}^{val}$,$\boldsymbol{E}^{calib}$ and $\boldsymbol{E}^{test}$, while satisfy that $\boldsymbol{E} = \boldsymbol{E}^{train} \cup \boldsymbol{E}^{val} \cup \boldsymbol{E}^{calib} \cup \boldsymbol{E}^{test}$. We assume that the weights of the edges in $\boldsymbol{E}^{train}$ and $\boldsymbol{E}^{val}$ are known. The objective is to estimate the unknown weights of the edges in $\boldsymbol{E}^{test}$. Additionally, we assume that the entire graph structure, represented by the adjacency matrix A, is known.

To mask the validation and test sets, we define

$$\boldsymbol{A}^{train} \in \{0,1\}^{n \times n}, \quad A_{ij}^{train} = \begin{cases} 1, & \text{if } (i,j) \in \boldsymbol{E}^{train}; \\ 0, & \text{otherwise.} \end{cases} \tag{27}$$

$\boldsymbol{A}^{val}$, $\boldsymbol{A}^{calib}$ and $\boldsymbol{A}^{test}$ are defined in the same way based on $\boldsymbol{E}^{val}$, $\boldsymbol{E}^{calib}$ and $\boldsymbol{E}^{test}$, respectively.

If $(i,j) \notin \boldsymbol{E}^{train}$, it is possible to assign a small positive number to the corresponding element $Wtrain_{ij}$, such as an assigned value or the minimum of the existing edge weights. This can represent prior knowledge or assumptions about the unknown edge weight. In the following part, we use a positive constant $\delta > 0$ to represent this minimal or assigned value. Incorporating this unknown edge weight information effectively leverages the underlying graph structure. The resulting weighted adjacency matrix is:

$$\boldsymbol{W}^{train} = \begin{cases} W_{ij}, & \text{if } (i,j) \in \boldsymbol{E}^{train}; \\ \delta, & \text{if } (i,j) \in \boldsymbol{E}^{val} \cup \boldsymbol{E}^{calib} \sup \boldsymbol{E}^{test}; \\ 0, & \text{otherwise,} \end{cases} \tag{28}$$

In the transductive setting (details can been seen in appendix Figure 5(a)), the structure of the entire graph, represented by the adjacency matrix $\boldsymbol{A}$, is known during the training, validation, and testing phases. To calibrate the prediction, we extract a subset from $\boldsymbol{E}^{test}$ as a calibration edge set. This ensures that the calibration and test samples are exchangeable.

Consider edge weight prediction in traffic networks. The road system, $\boldsymbol{A}$, and partial traffic volumes, $\boldsymbol{W}^{train} + \boldsymbol{W}^{val}$, are known. The task is to predict volumes, $\boldsymbol{W}^{test}$, for the remaining roads.

During training, models observe node features and graph structure to learn functions for node classification/regression and embedding. At inference, models deduce edge connections between nodes (Figure 5).

Three GNN approaches are evaluated. The first is a Graph Autoencoder (GAE) Kipf and Welling [2016b] that trains and infers on the full graph. The second is DiGAE Kollias et al. [2022], a directed GAE variant. The third is the

Table 4: Results of RR-GNN on Node Regression Datasets

Dataset	GraphSAGE		SGC		GCN		GATS	
Metrics	cover^x	ineff	cover^x	ineff	cover^x	ineff	cover^x	ineff
Income: CF-GNN	$0.9512^{\pm 0.0264}$	$2.7580^{\pm 0.0342}$	$0.9504^{\pm 0.0405}$	$2.4892^{\pm 0.0302}$	$0.9511^{\pm 0.0250}$	$2.5272^{\pm 0.0318}$	$0.9508^{\pm 0.0329}$	$2.4396^{\pm 0.0328}$
Income: Cluster-GNN	$0.9521^{\pm 0.035}$	$2.6723^{\pm 0.041}$	$0.9513^{\pm 0.038}$	$2.3721^{\pm 0.037}$	$0.9526^{\pm 0.033}$	$2.4189^{\pm 0.036}$	$0.9519^{\pm 0.034}$	$2.3254^{\pm 0.035}$
Income: RR-GAE	$0.9538^{\pm 0.032}$	$2.5342^{\pm 0.038}$	$\mathbf{0.9524}^{\pm 0.036}$	$2.1423^{\pm 0.034}$	$0.9539^{\pm 0.031}$	$2.1932^{\pm 0.033}$	$0.9527^{\pm 0.033}$	$2.1567^{\pm 0.032}$
Income: Cluster-RR-GAE	$\mathbf{0.9552}^{\pm 0.0618}$	$\mathbf{2.1003}^{\pm 0.0492}$	$0.9519^{\pm 0.0513}$	$\mathbf{1.9616}^{\pm 0.0358}$	$\mathbf{0.9566}^{\pm 0.0501}$	$\mathbf{1.9203}^{\pm 0.0354}$	$\mathbf{0.9545}^{\pm 0.0347}$	$\underline{\mathbf{1.8555}}^{\pm 0.0423}$
Unemploy: CF-GNN	$0.9526^{\pm 0.0415}$	$2.2298^{\pm 0.0523}$	$0.9510^{\pm 0.0320}$	$2.4587^{\pm 0.0491}$	$0.9506^{\pm 0.0294}$	$2.5013^{\pm 0.0326}$	$0.9502^{\pm 0.0354}$	$2.4332^{\pm 0.0376}$
Unemploy: Cluster-GNN	$0.9531^{\pm 0.038}$	$2.1932^{\pm 0.045}$	$0.9519^{\pm 0.036}$	$2.3256^{\pm 0.042}$	$0.9513^{\pm 0.034}$	$2.3721^{\pm 0.038}$	$0.9516^{\pm 0.033}$	$2.2894^{\pm 0.039}$
Unemploy: RR-GAE	$0.9542^{\pm 0.035}$	$2.1423^{\pm 0.039}$	$0.9524^{\pm 0.033}$	$2.1932^{\pm 0.036}$	$0.9528^{\pm 0.032}$	$2.2567^{\pm 0.035}$	$0.9523^{\pm 0.031}$	$2.1567^{\pm 0.034}$
Unemploy: Cluster-RR-GAE	$\mathbf{0.9569}^{\pm 0.0419}$	$\mathbf{2.0816}^{\pm 0.0218}$	$\mathbf{0.9517}^{\pm 0.0313}$	$\mathbf{2.0534}^{\pm 0.0367}$	$\mathbf{0.9523}^{\pm 0.0369}$	$\mathbf{2.0480}^{\pm 0.0190}$	$\mathbf{0.9523}^{\pm 0.0448}$	$\underline{\mathbf{1.9503}}^{\pm 0.0312}$
Twitch: CF-GNN	$\mathbf{0.9524}^{\pm 0.0443}$	$2.6634^{\pm 0.0365}$	$0.9523^{\pm 0.0392}$	$2.6835^{\pm 0.0394}$	$0.9529^{\pm 0.0257}$	$2.5409^{\pm 0.0404}$	$0.9515^{\pm 0.0275}$	$2.6243^{\pm 0.0460}$
Twitch: Cluster-GNN	$0.9531^{\pm 0.039}$	$2.5894^{\pm 0.042}$	$0.9528^{\pm 0.037}$	$2.5321^{\pm 0.040}$	$0.9534^{\pm 0.034}$	$2.4892^{\pm 0.038}$	$0.9523^{\pm 0.033}$	$2.4723^{\pm 0.041}$
Twitch: RR-GAE	$0.9539^{\pm 0.036}$	$\mathbf{2.4987}^{\pm 0.039}$	$\mathbf{0.9532}^{\pm 0.035}$	$2.4567^{\pm 0.037}$	$0.9541^{\pm 0.032}$	$2.3721^{\pm 0.036}$	$0.9529^{\pm 0.031}$	$2.3256^{\pm 0.038}$
Twitch: Cluster-RR-GAE	$0.9515^{\pm 0.0367}$	$5.0491^{\pm 0.0513}$	$0.9541^{\pm 0.0284}$	$\underline{\mathbf{2.1005}}^{\pm 0.0189}$	$\mathbf{0.9571}^{\pm 0.0219}$	$\mathbf{2.2398}^{\pm 0.0225}$	$\mathbf{0.9535}^{\pm 0.0280}$	$\mathbf{2.1353}^{\pm 0.0262}$

Table 5: Node Classification Results with Conformal Baselines (Coverage ↑ / Inefficiency ↓)

Dataset	CF-GNN [1]		SAN		RR-GNN (Ours)		Cluster-RR-GNN (Ours)	
Model	Cover	Ineff	Cover	Ineff	Cover	Ineff	Cover	Ineff
Cora								
GraphSAGE	$0.9456^{\pm 0.0569}$	$1.6284^{\pm 0.0483}$	$0.9476^{\pm 0.0532}$	$1.6825^{\pm 0.0541}$	$0.9460^{\pm 0.0542}$	$1.6100^{\pm 0.0415}$	$\mathbf{0.9463}^{\pm 0.0509}$	$\mathbf{1.6076}^{\pm 0.0397}$
SGC	$0.9461^{\pm 0.0603}$	$1.6633^{\pm 0.04415}$	$0.9482^{\pm 0.05348}$	$1.6956^{\pm 0.0236}$	$0.9462^{\pm 0.0581}$	$1.6297^{\pm 0.0428}$	$\mathbf{0.9468}^{\pm 0.0662}$	$\mathbf{1.6017}^{\pm 0.0465}$
GCN	$0.9473^{\pm 0.0556}$	$1.6344^{\pm 0.0418}$	$0.9426^{\pm 0.0453}$	$1.6520^{\pm 0.0344}$	$0.9432^{\pm 0.0573}$	$1.6251^{\pm 0.0367}$	$\mathbf{0.9476}^{\pm 0.0732}$	$\mathbf{1.6315}^{\pm 0.0303}$
GAT	$0.9464^{\pm 0.0702}$	$1.6278^{\pm 0.0334}$	$0.9473^{\pm 0.065}$	$1.6752^{\pm 0.0364}$	$0.9475^{\pm 0.0624}$	$1.6146^{\pm 0.0351}$	$\mathbf{0.9491}^{\pm 0.0539}$	$\mathbf{1.6254}^{\pm 0.0396}$
DBLP								
GraphSAGE	$0.9501^{\pm 0.0523}$	$1.5723^{\pm 0.0683}$	$0.9501^{\pm 0.0420}$	$\mathbf{1.5524}^{\pm 0.0637}$	$0.9499^{\pm 0.0531}$	$1.5351^{\pm 0.0473}$	$\mathbf{0.9503}^{\pm 0.0510}$	$1.5607^{\pm 0.0487}$
SGC	$\mathbf{0.9451}^{\pm 0.0617}$	$1.5274^{\pm 0.0416}$	$0.9235^{\pm 0.0526}$	$1.4520^{\pm 0.0345}$	$0.9462^{\pm 0.0528}$	$1.4286^{\pm 0.0541}$	$0.9443^{\pm 0.0462}$	$\mathbf{1.3921}^{\pm 0.0624}$
GCN	$\mathbf{0.9473}^{\pm 0.0596}$	$1.5644^{\pm 0.0733}$	$0.9445^{\pm 0.0565}$	$1.5984^{\pm 0.0743}$	$0.9458^{\pm 0.0702}$	$1.5512^{\pm 0.0295}$	$0.9430^{\pm 0.0713}$	$\mathbf{1.5491}^{\pm 0.0278}$
GAT	$0.9467^{\pm 0.0717}$	$1.5729^{\pm 0.0463}$	$0.9456^{\pm 0.0624}$	$1.5943^{\pm 0.0425}$	$0.9485^{\pm 0.0589}$	$1.5725^{\pm 0.0349}$	$\mathbf{0.9491}^{\pm 0.0539}$	$\mathbf{1.5720}^{\pm 0.0322}$
CiteSeer								
GraphSAGE	$0.9528^{\pm 0.0203}$	$1.1680^{\pm 0.0439}$	$0.9502^{\pm 0.0164}$	$1.2523^{\pm 0.0416}$	$0.9538^{\pm 0.0853}$	$1.1621^{\pm 0.0552}$	$\mathbf{0.9540}^{\pm 0.0926}$	$\mathbf{1.1679}^{\pm 0.0605}$
SGC	$0.9525^{\pm 0.0257}$	$1.1827^{\pm 0.0552}$	$0.9505^{\pm 0.0645}$	$1.2678^{\pm 0.0532}$	$0.9579^{\pm 0.0536}$	$1.1782^{\pm 0.0415}$	$\mathbf{0.9594}^{\pm 0.0582}$	$\mathbf{1.1898}^{\pm 0.0399}$
GCN	$0.9496^{\pm 0.0392}$	$1.2310^{\pm 0.0332}$	$0.9502^{\pm 0.0536}$	$1.3024^{\pm 0.0324}$	$0.9512^{\pm 0.0358}$	$1.2189^{\pm 0.0276}$	$0.9518^{\pm 0.0373}$	$\mathbf{1.2153}^{\pm 0.0290}$
GAT	$0.9508^{\pm 0.0309}$	$1.2396^{\pm 0.0416}$	$0.9515^{\pm 0.0251}$	$1.3112^{\pm 0.0123}$	$0.9535^{\pm 0.0447}$	$1.2085^{\pm 0.0361}$	$\mathbf{0.9548}^{\pm 0.0491}$	$\mathbf{1.2020}^{\pm 0.0392}$

line graph neural network (LGNN) Cai et al. [2021] that transforms edges to nodes in line graphs.

6.3 APPENDICES FIGURES

6.3.1 Schematic figure for transductive and inductive settings for link prediction.

There are two major braches in link prediction problem settting: transductive and inductive settings. Schematic figure for transductive and inductive settings for edge weight prediction are shown in Figure 5

6.3.2 Learning the error structure

In order to show our method's outperformance in learning the error structure in real-world applications, we tested our method in two datasets: 2016 U.S. county-level presidential election and Traffic volums prediction in Anaheim and Chicago transportation networks. Results are shown in Figure 6 and Figure 7.

6.4 NONCONFORMITY SCORE AND EVALUATION METRICS

Nonconformity Score For the nonconformity score in main paper is for conformal prediction-based RR (CP-RR). On the other hand, for conformal quantile regression-based RR (CQR-RR), The nonconformity score is

$$C_{ab} = \Big[\hat{W}_{ab}^{\alpha/2} - d^{\text{RR}}\big|\hat{W}_{ab}^{1-\alpha/2} - \hat{W}_{ab}^{\alpha/2}\big|, \\ \hat{W}_{ab}^{1-\alpha/2} + d^{\text{RR}}\big|\hat{W}_{ab}^{1-\alpha/2} - \hat{W}_{ab}^{\alpha/2}\big|\Big], \tag{32}$$

$$(a,b) \in \boldsymbol{E}^{test}, \tag{33}$$

Evaluation Metrics: For evaluation, we use the marginal coverage, defined as

$$\text{cover} = \frac{1}{|E^{test}|} \sum_{(i,j)\in E^{test}} \mathbb{1}\big(Wtest_{ij} \in C_{ij}\big), \tag{34}$$

where C_{ij} is prediction interval for edge (i,j). Another one is inefficiency which is defined as

$$\text{ineff} = \frac{1}{|E^{test}|} \sum_{(i,j)\in E^{test}} |C_{ij}|, \tag{35}$$

Table 6: Performance(AUC) Comparison of Graph Transformer Models with RR Enhancement on dataset MolHIV

Model	Method	Cora	CiteSeer	PubMed	OGB-Arxiv
Graphormer	Original	0.763 ± 0.012	0.691 ± 0.015	0.792 ± 0.008	0.718 ± 0.005
	+ RR	**0.781** ± 0.011	**0.705** ± 0.013	**0.803** ± 0.007	**0.729** ± 0.004
Graphormer with Spatial Encoding	Original	0.772 ± 0.010	0.702 ± 0.014	0.801 ± 0.007	0.725 ± 0.005
	+ RR	**0.789** ± 0.009	**0.715** ± 0.012	**0.812** ± 0.006	**0.736** ± 0.004
Graphormer with Graph Structure	Original	0.781 ± 0.011	0.712 ± 0.013	0.808 ± 0.007	0.732 ± 0.004
	+ RR	**0.796** ± 0.010	**0.724** ± 0.011	**0.819** ± 0.006	**0.742** ± 0.003

Table 7: Performance of Graph Transformer Networks (GT) with RR Enhancement

Node Classification	F1 Score	ACM	DBLP	IMDB
Base GT	Original	0.912 ± 0.014	0.938 ± 0.034	0.609 ± 0.023
	+ RR	**0.923** ± 0.012	**0.942** ± 0.025	**0.724** ± 0.036

Algorithm 3 Residual Reweighted Conformalized Graph Neural Network for Node Regression

Input: The binary adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n\times n}$, training node features $\boldsymbol{X} \in R^{n\times m}$, training node set $\mathbb{V}^{train}$ and label , $\boldsymbol{y}^{train}$, validation node set $\mathbb{V}^{val}$ and label- $\boldsymbol{y}^{val}$ (Used for training Residual GNN), calibration nodes $\mathbb{V}^{calib}$ and label $\boldsymbol{y}^{calib}$, and test nodes $\mathbb{V}^{test}$, user-specified error rate $\alpha \in (0,1)$, two GNN model $\mathcal{G}_{\boldsymbol{\theta}_1}$ and $\mathcal{G}_{\boldsymbol{\theta}_2}$ with trainable parameter $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$.

Train the model $g_{\boldsymbol{\theta}_1}$ and $g_{\boldsymbol{\theta}_2}$ with $\boldsymbol{y}^{train}$ and $\boldsymbol{y}^{val}$ according to Algorithm 2 in main paper.
Compute the nonconformity score, which quantifies the interval the predicted calibration node labels:

$$V_i^{\mathrm{RR}} = \max\left\{\frac{\hat{y}_i^{\alpha/2} - y_i^{calib}}{\left|\hat{R}_i\right|}, \frac{y_i^{calib} - \hat{y}_i^{1-\alpha/2}}{\left|\hat{R}_i\right|}\right\}, \tag{29}$$

$$i \in \mathbb{V}^{calib}, \tag{30}$$

Compute $d =$ the kth smallest value in $\{V_i^{\mathrm{RR}}\}$, where $k = \lceil (|\mathbb{V}^{calib}|+1)(1-\alpha)\rceil$;
Construct a prediction interval for test nodes:

$$C_a = \left[\hat{y}_a^{\alpha/2} - d\left|\hat{R}_a\right|, \hat{y}_a^{1-\alpha/2} + d\left|\hat{R}_a\right|\right],\ a \in \mathbb{V}^{test}.$$

Output: Prediction of confidence intervals for the test nodes $a \in \mathbb{V}^{test}$ with the coverage guarantee:

$$p\big(y_a^{test} \in C_a\big) \geq 1-\alpha. \tag{31}$$

which measures the average length of the prediction interval.

In addition to the marginal coverage, we also consider the conditional coverage. Specifically, we measure the coverage over a slab of the feature space $S_{v,a,b} = \big\{[\boldsymbol{X}_{i,:} \parallel \boldsymbol{X}_{j,:}] \in R^{2f} : a \leq v^\top \boldsymbol{X} \leq b\big\}$ Romano et al. [2020], Cauchois et al. [2020], where $[\boldsymbol{X}_{i,:} \parallel \boldsymbol{X}_{j,:}]$ denotes the node feature of two connected nodes of an edge (i,j) and $v \in R^{2f}$ and $a < b \in R$ are chosen adversarially and independently from the data. For any prediction interval f_θ^* and $\delta \in (0,1)$, the *worst slice coverage* is defined as

$$\mathrm{WSC}(f_\theta^*, \delta) = \inf_{\substack{v \in R^{2f},\\ a<b\in R}} \big\{P\big(Wtest_{ij} \in C_{ij} \mid [\boldsymbol{X}_{i,:}\parallel \boldsymbol{X}_{j,:}] \in S_{v,a,b}\big) \text{ s.t. } P([\boldsymbol{X}_{i,:} \parallel \boldsymbol{X}_{j,:}] \in S_{v,a,b}) \geq \delta\big\}. \tag{36}$$

6.5 ALGORITHM FOR NODE CLASSIFICATION

6.6 COMPLETE TABLE OF EXPERIMENT

6.7 THEORETICAL GUARANTEE OF OUR METHOD

7.1. Conformal Prediction

We assume we have access to the graph structure, A, the node features, X, and the weighted adjacency matrix W^{train} (3). Let (a,b) be the endpoints of a test edge. We aim to generate a prediction interval, $C_{ab} = \big(f_\theta\left((a,b), A, X, W^{\text{train}}\right)\big) \subset \mathbb{R}$, for the weight of the such a test edge. The prediction interval should be marginally valid, i.e. it should obey

$$P\left(W_{ab} \in C_{ab}\right) \geq 1-\alpha, \tag{41}$$

where $\alpha \in (0,1)$ is a user-defined error rate. The probability is over the datagenerating distribution. For efficiency, we

GNN Model on Watts-Strogatz	Data Load (s)	Load GPU (MB)	GraphConv			SAGEConv			GATS		
			train	val	GPU space	train	val	GPU space	train	val	GPU space
GAE (100)	0.30	0.73	0.85s	0.31s	0.21MB	0.70s	0.23s	0.38MB	0.97s	0.38s	0.22MB
DiGAE (100)	0.30	0.73	1.08s	0.45s	0.41MB	1.24s	0.40s	0.74MB	1.16s	0.45s	0.43MB
GAE (1000)	1.40	7.31	1.91s	0.74s	0.58MB	4.71s	0.27s	0.75MB	2.05s	0.72s	0.59MB
DiGAE (1000)	1.40	7.31	3.06s	0.56s	1.11MB	2.24s	0.77s	1.44MB	2.05s	0.60s	1.13MB
GAE (10000)	18.62	73.13	2.21s	0.83s	4.29MB	1.88s	0.72s	4.45MB	3.15s	1.02s	4.30MB
DiGAE (10000)	18.62	73.13	2.90s	1.06s	8.12MB	3.45s	0.97s	8.45MB	2.75s	1.13s	8.13MB
GAE (50000)	98.39	366.07	2.74s	0.75s	20.93MB	2.19s	0.58s	20.77MB	2.75s	0.89s	20.77MB
DiGAE (50000)	98.39	366.07	3.34s	0.91s	39.57MB	3.38s	1.27s	39.25MB	5.25s	1.74s	39.26MB
GAE (100000)	196.39	731.48	2.48s	0.91s	42.25MB	2.00s	0.80s	41.82MB	4.37s	1.12s	42.76MB
DiGAE (100000)	196.39	731.48	3.90s	1.18s	78.60MB	2.93s	1.05s	78.51MB	6.22s	2.24s	80.07MB

Table 8: Performance comparison of different GNN models on Watts-Strogatz graphs including data loading overhead.

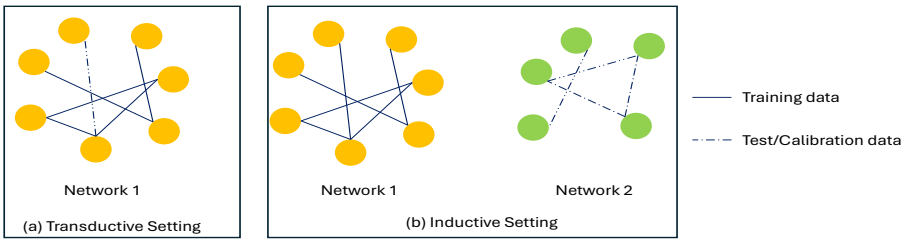

Figure 5: Schematic figure for transductive and inductive settings for edge weight prediction. Different colors indicate the availability of the nodes during the training or testing phases. Solid and dashed lines represent edges used for training and the predicted edge in the testing phases, respectively. (a) Transductive edge weight prediction performs both training and inference on the same graph. (b) Inductive edge weight prediction inference is performed on a new, unseen graph.

focus on the split CP approachPapadopoulos et al. [2002], using the training edge set E^{train} for training and the calibration edge set E^{calib} for calibration. E^{train} is used to fit the prediction model, f_θ, and a conformity score is calculated for each sample in E^{calib}. The conformity score evaluates how well the predictions match the observed labels.

Proposition 1. The prediction intervals generated by split CP (Algorithm 1), CQR (Algorithm 2), and RR are marginally valid, i.e. obey equation 41 in appendices material.

Proof. First, we show that the calibration and test conformity scores defined in equation 29 and 37 in appendices material and (12) in main paper are exchangeable. Given the entire graph structure, A, all the node features, X, and the edge weights of the training edges, W^{train}, the node embeddings are trained based on W^{train}, and the edge weights in the remaining E^{ct} are set randomly, the division of E^{ct} into E^{calib} and E^{test} have no impact on the training process. Consequently, the conformity scores for E^{calib} and E^{test} are exchangeable. In practice, we split E^{ct} into E^{calib} and E^{test} randomly (as detailed in Section 6.2 in appendices material) by converting the graph into its line graph and then selecting nodes uniformly at random.

We also explore an alternative proof which is equivalent to the proof in Huang et al.Huang et al. [2024] but applied within a line graph setting. Consider the original graph $G = (V, E)$ and its corresponding line graph $G' = (V', E')$, where $V' = E$ and E' denotes adjacency between edges in G. After randomly dividing E into $\mathbb{E}^{\text{train}}$ and E^{ct}, and further splitting E^{ct} into E^{calib} and E^{test}, the edges of G transforms into nodes in G'. This setup mirrors the node division in the line graph. We train node embeddings on E^{train} using a graph autoencoder, which aligns with fixing the training node set in G'. Given this fixed training set, any permutation and division of E^{ct} (which corresponds to nodes in G') doesn't affect the training, and thus the conformity scores computed for E^{calib} and E^{test} are exchangeable.

Given this exchangeability of nonconformity scores, the validity of the prediction interval produced by CP and CQR follows from Theorem 2.2 of Lei et al.Lei et al. [2018] and Theorem 1 of Romano, Patterson, and CandesRomano et al. [2019]. Let V be the conformity score of CQR. The RR approach performs a monotone transformation of V, defined as

$$\Phi_{ij}(V) = \frac{V}{\left|\hat{W}_{ij}^{1-\alpha/2} - \hat{W}_{ij}^{\alpha/2}\right|},$$

where i and j are two nodes in the graph[2]. For all (i, j) and all $V, \Phi'_{ij}(V) = \frac{\partial \Phi_{ij}(V)}{\partial V} > 0$, i.e. the transformation is strictly monotonic in V. This implies Φ_{ab} is invertible for any test edge, (a, b). Let Φ_{ab}^{-1} be the inverse of Φ_{ab}. The Inverse Function Theorem implies Φ_{ab}^{-1} is also strictly increasing. Now suppose d^{RR} is the k th smallest value in $\left\{V_{ij}^{\text{RR}}\right\} = \left\{\Phi_{ij}\left(V_{ij}\right)\right\}, k = \left\lceil \left(\left|E^{\text{calib}}\right| + 1\right)(1-\alpha)\right\rceil$. Then for a test edge (a, b),

$$P\left(\Phi_{ab}\left(V_{ab}\right) \leq d^{\text{RR}}\right) = \frac{\left\lceil \left(\left|E^{\text{calib}}\right| + 1\right)(1-\alpha)\right\rceil}{\left|E^{\text{calib}}\right| + 1} \geq 1-\alpha.$$

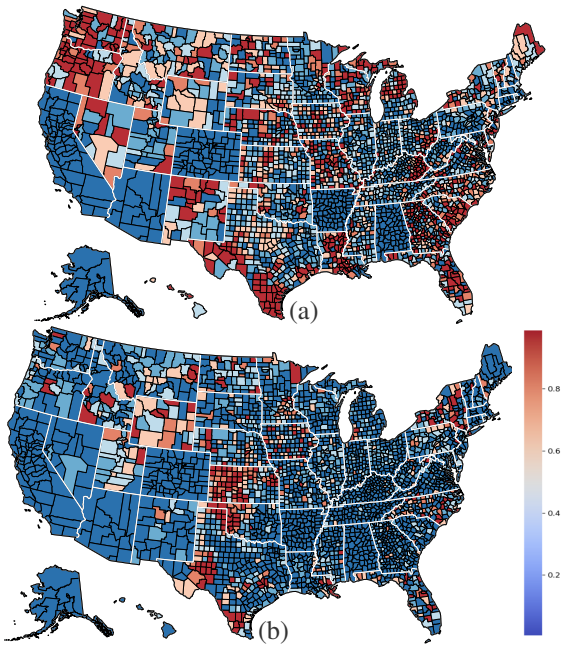

Figure 6: Residual for predicted 2016 U.S. county-level presidential election. (a) The baseline model's residual Jia and Benson [2020], which is the normalized absolute difference between the predicted and ground truth vote count. (b) RR-GNN's residual result. The results indicate that the proposed RR-GNN achieves smaller and more uniform error/residuals.

Using the monotonicity of Φ_{ab}^{-1},

$$\begin{aligned} 1-\alpha &\leq P\left(\Phi_{ab}\left(V_{ab}\right) \leq V_k^{\mathrm{RR}}\right) \\ &= P\left(V_{ab} \leq \Phi_{ab}^{-1}\left(d^{\mathrm{RR}}\right)\right) \\ &= P\left(W_{ab} \in C_{ab}\right) \end{aligned}$$

The final equation is derived from the construction of the prediction interval in C_a in Algorithm 1 in main paper and Algorithm 2 and 3 in appendices material and the validity of CQR. This shows that the prediction intervals based on the reweighted conformity scores are valid. Proof done.

7 OTHER ABLATION STUDIES

2. Stability Validation

We systematically evaluate clustering stability and noise robustness. Results can be seen in table 11. In this table titles, Ori means original setting. NE means noise edge. FN means feature noise.

Algorithm 4 Residual Reweighted Conformalized Graph Neural Network for Node Classification

Input: The binary adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n\times n}$, training node features $\boldsymbol{X} \in \mathbf{R}^{n\times m}$, training node and class label variable $\mathbb{V}^{train}$, $\boldsymbol{l}^{train}$ ($\boldsymbol{l}^{train}$ is the one-hot of $\boldsymbol{l}^{train}$. The situation is also met for $\hat{l}$ and $\hat{l}$ which are class output of model), validation nodes and label $\mathbb{V}^{val}$, $\boldsymbol{l}^{val}$ (Used for training Residual GNN), calibration nodes and label $\mathbb{V}^{calib}$, $\boldsymbol{l}^{calib}$, and test nodes $\mathbb{V}^{test}$, user-specified error rate $\alpha \in (0,1)$, two GNN model $g_{\boldsymbol{\theta}_1}$ and $g_{\boldsymbol{\theta}_2}$ with trainable parameter $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$.

Train the model $g_{\boldsymbol{\theta}_1}$ and $g_{\boldsymbol{\theta}_2}$ with $\boldsymbol{l}^{train}$ and $\boldsymbol{l}^{val}$ according to Algorithm 2 in main paper
Compute the score which quantifies the residual of the calibration node classes $\boldsymbol{l}^{calib}$ projected onto the nearest quantile produced by $g_{\boldsymbol{\theta}_1}$ and $g_{\boldsymbol{\theta}_2}$:

$$V_i^{\mathrm{RR}} = \max\left\{\frac{\left|\hat{l}_i^{\alpha/2} - l_i^{calib}\right|}{\left|\hat{\mathbf{R}}_i\right| + \epsilon}, \frac{\left|l_i^{calib} - \hat{l}_i^{1-\alpha/2}\right|}{\left|\hat{\mathbf{R}}_i\right| + \epsilon}\right\}, \tag{37}$$

$$i \in \mathbb{V}^{calib}, \tag{38}$$

Compute $d =$ the kth smallest value in $\{V_i^{\mathrm{RR}}\}$, where

$$k = DiffQuantile(\lceil(|\mathbb{V}^{calib}|+1)(1-\alpha)\rceil); \tag{39}$$

Construct a prediction interval for test nodes:

$$C_a = \left[\hat{l}_a^{\alpha/2} - d_{(m)}\left|\hat{R}_a\right|, \hat{l}_a^{1-\alpha/2} + d_{(m)}\left|\hat{R}_a\right|\right],\ a \in \mathbb{V}^{test}.$$

Output: Prediction of confidence intervals for the test nodes $(a,b) \in \mathbb{V}^{test}$ with the coverage guarantee:

$$p\left(l_a^{test} \in C_a\right) \geq 1-\alpha. \tag{40}$$

Table 11: Clustering Stability Metrics (Cora Dataset)

Metric	Ori	+20% NE	+15% FN
Adjusted Rand Index	0.85	0.82	0.79
Coverage Variance	0.012	0.015	0.018
Cluster Purity	0.92	0.89	0.85
Calibration Time (s)	42.3	45.1	47.8

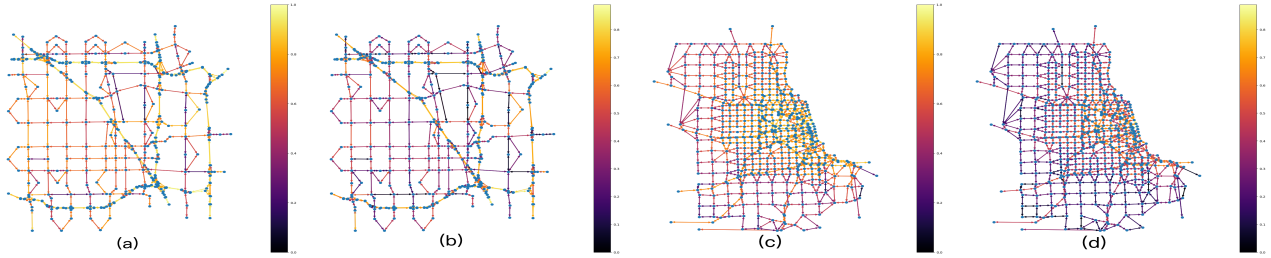

Figure 7: Residual between predicted and actual traffic volumes across roads in two cities under different models. We took the average volume at the start and end points as the road traffic. (a) Residual of predicted roads' traffic volume in Anaheim of baseline model Huang et al. [2024], which is the absolute value of prediction and ground truth. (b) Residual of predicted roads' traffic volume in Anaheim of RR-GAE. (c) Residual of predicted roads' traffic volume in Chicago of baseline model. (d) Residual of predicted roads' traffic volume in Chicago of RR-GAE. For each city, the residuals from the two models were independently normalized to a 0-1 range for comparison purposes.

Table 9: Results of RR-GNN on Node Regression Datasets

Dataset	GraphSAGE		SGC		GCN		GATS	
Metrics	cover^x	ineff	cover^x	ineff	cover^x	ineff	cover^x	ineff
Anaheim: CF-GNN	$0.9520^{\pm 0.0669}$	**1.9231**$^{\pm 0.0483}$	$0.9559^{\pm 0.0617}$	$2.2031^{\pm 0.0241}$	$0.9519^{\pm 0.0531}$	$2.3782^{\pm 0.0533}$	$0.9523^{\pm 0.0302}$	$2.1499^{\pm 0.0463}$
Anaheim: Cluster-GNN	$0.9532^{\pm 0.042}$	$1.8954^{\pm 0.037}$	$0.9561^{\pm 0.035}$	$2.1423^{\pm 0.031}$	$0.9528^{\pm 0.041}$	$2.2451^{\pm 0.029}$	$0.9541^{\pm 0.028}$	$2.0321^{\pm 0.025}$
Anaheim: RR-GAE	$0.9539^{\pm 0.038}$	$1.8732^{\pm 0.032}$	**0.9567**$^{\pm 0.031}$	$2.0987^{\pm 0.028}$	$0.9532^{\pm 0.036}$	$2.1934^{\pm 0.026}$	$0.9563^{\pm 0.024}$	$1.9623^{\pm 0.022}$
Anaheim: Clsuter-RR-GAE	**0.9543**$^{\pm 0.0320}$	$1.9647^{\pm 0.0197}$	$0.9577^{\pm 0.0657}$	**2.0188**$^{\pm 0.0246}$	**0.9585**$^{\pm 0.0413}$	**2.2179**$^{\pm 0.0254}$	**0.9638**$^{\pm 0.0302}$	**1.8996**$^{\pm 0.0249}$
Chicago: CF-GNN	$0.9448^{\pm 0.0519}$	$2.3426^{\pm 0.0384}$	$0.9486^{\pm 0.0247}$	$1.0423^{\pm 0.0372}$	$0.9505^{\pm 0.0447}$	$2.0456^{\pm 0.0443}$	$0.9508^{\pm 0.0569}$	$1.1396^{\pm 0.0686}$
Chicago: Cluster-GNN	$0.9461^{\pm 0.039}$	$2.2894^{\pm 0.034}$	$0.9492^{\pm 0.031}$	$1.1895^{\pm 0.029}$	$0.9513^{\pm 0.037}$	$1.8742^{\pm 0.031}$	$0.9516^{\pm 0.042}$	$1.1254^{\pm 0.045}$
Chicago: RR-GAE	$0.9472^{\pm 0.035}$	$2.2673^{\pm 0.029}$	**0.9498**$^{\pm 0.028}$	$1.2567^{\pm 0.026}$	$0.9519^{\pm 0.033}$	$1.6923^{\pm 0.027}$	$0.9519^{\pm 0.038}$	$1.1489^{\pm 0.039}$
Chicago: Cluster-RR-GAE	**0.9476**$^{\pm 0.0426}$	**2.2291**$^{\pm 0.0325}$	$0.9546^{\pm 0.0328}$	**1.2012**$^{\pm 0.0250}$	**0.9538**$^{\pm 0.0356}$	**1.5769**$^{\pm 0.0252}$	**0.9540**$^{\pm 0.0362}$	$1.1283^{\pm 0.0256}$
Education: CF-GNN	$0.9501^{\pm 0.0242}$	$2.3808^{\pm 0.0427}$	$0.9500^{\pm 0.0285}$	$2.4892^{\pm 0.0351}$	$0.9483^{\pm 0.0408}$	$2.4380^{\pm 0.0452}$	$0.9502^{\pm 0.0392}$	$2.4209^{\pm 0.0376}$
Education: Cluster-GNN	$0.9513^{\pm 0.031}$	$2.3145^{\pm 0.038}$	$0.9517^{\pm 0.033}$	$2.3721^{\pm 0.032}$	$0.9496^{\pm 0.035}$	$2.2894^{\pm 0.034}$	$0.9518^{\pm 0.036}$	$2.3256^{\pm 0.033}$
Education: RR-GAE	$0.9529^{\pm 0.029}$	$2.1932^{\pm 0.027}$	$0.9534^{\pm 0.030}$	$2.1478^{\pm 0.028}$	$0.9508^{\pm 0.032}$	$2.0321^{\pm 0.029}$	$0.9532^{\pm 0.031}$	$2.1423^{\pm 0.030}$
Education: Cluster-RR-GAE	**0.9599**$^{\pm 0.0417}$	**2.0573**$^{\pm 0.0280}$	**0.9586**$^{\pm 0.0225}$	**2.0445**$^{\pm 0.0239}$	**0.9580**$^{\pm 0.0333}$	**1.8731**$^{\pm 0.0260}$	**0.9594**$^{\pm 0.0386}$	**1.9075**$^{\pm 0.0221}$
Election: CF-GNN	$0.9498^{\pm 0.0211}$	$0.9268^{\pm 0.0429}$	$0.9495^{\pm 0.0215}$	$0.9279^{\pm 0.0302}$	$0.9506^{\pm 0.0473}$	$0.9009^{\pm 0.0282}$	$0.9488^{\pm 0.0363}$	$0.9136^{\pm 0.0681}$
Election: Cluster-GNN	$0.9503^{\pm 0.028}$	$0.9152^{\pm 0.038}$	$0.9501^{\pm 0.027}$	$0.9124^{\pm 0.035}$	$0.9512^{\pm 0.041}$	$0.8723^{\pm 0.031}$	$0.9496^{\pm 0.033}$	$0.8945^{\pm 0.042}$
Election: RR-GAE	$0.9509^{\pm 0.025}$	$0.9037^{\pm 0.029}$	$0.9523^{\pm 0.024}$	$0.8956^{\pm 0.028}$	$0.9518^{\pm 0.036}$	$0.8234^{\pm 0.026}$	$0.9514^{\pm 0.030}$	$0.8562^{\pm 0.035}$
Election: Cluster-RR-GAE	**0.9558**$^{\pm 0.0215}$	**0.9213**$^{\pm 0.0279}$	**0.9567**$^{\pm 0.0242}$	**0.9487**$^{\pm 0.0259}$	**0.9510**$^{\pm 0.0432}$	**0.9343**$^{\pm 0.0341}$	**0.9567**$^{\pm 0.0317}$	**0.6698**$^{\pm 0.0201}$
Income: CF-GNN	$0.9512^{\pm 0.0264}$	$2.7580^{\pm 0.0342}$	$0.9504^{\pm 0.0405}$	$2.4892^{\pm 0.0302}$	$0.9511^{\pm 0.0250}$	$2.5272^{\pm 0.0318}$	$0.9508^{\pm 0.0329}$	$2.4396^{\pm 0.0328}$
Income: Cluster-GNN	$0.9521^{\pm 0.035}$	$2.6723^{\pm 0.041}$	$0.9513^{\pm 0.038}$	$2.3721^{\pm 0.037}$	$0.9526^{\pm 0.033}$	$2.4189^{\pm 0.036}$	$0.9519^{\pm 0.034}$	$2.3254^{\pm 0.035}$
Income: RR-GAE	$0.9538^{\pm 0.032}$	$2.5342^{\pm 0.038}$	**0.9524**$^{\pm 0.036}$	$2.1423^{\pm 0.034}$	$0.9539^{\pm 0.031}$	$2.1932^{\pm 0.033}$	$0.9527^{\pm 0.033}$	$2.1567^{\pm 0.032}$
Income: Cluster-RR-GAE	**0.9552**$^{\pm 0.0618}$	**2.1003**$^{\pm 0.0492}$	$0.9519^{\pm 0.0513}$	**1.9616**$^{\pm 0.0358}$	**0.9566**$^{\pm 0.0501}$	**1.9203**$^{\pm 0.0354}$	**0.9545**$^{\pm 0.0347}$	**1.8555**$^{\pm 0.0423}$
Unemploy: CF-GNN	$0.9526^{\pm 0.0415}$	$2.2298^{\pm 0.0523}$	$0.9510^{\pm 0.0320}$	$2.4587^{\pm 0.0491}$	$0.9506^{\pm 0.0294}$	$2.5013^{\pm 0.0326}$	$0.9502^{\pm 0.0354}$	$2.4332^{\pm 0.0376}$
Unemploy: Cluster-GNN	$0.9531^{\pm 0.038}$	$2.1932^{\pm 0.045}$	$0.9519^{\pm 0.036}$	$2.3256^{\pm 0.042}$	$0.9513^{\pm 0.034}$	$2.3721^{\pm 0.038}$	$0.9516^{\pm 0.033}$	$2.2894^{\pm 0.039}$
Unemploy: RR-GAE	$0.9542^{\pm 0.035}$	$2.1423^{\pm 0.039}$	$0.9524^{\pm 0.033}$	$2.1932^{\pm 0.036}$	$0.9528^{\pm 0.032}$	$2.2567^{\pm 0.035}$	$0.9523^{\pm 0.031}$	$2.1567^{\pm 0.034}$
Unemploy: Cluster-RR-GAE	**0.9569**$^{\pm 0.0419}$	**2.0816**$^{\pm 0.0218}$	**0.9517**$^{\pm 0.0313}$	**2.0534**$^{\pm 0.0367}$	**0.9523**$^{\pm 0.0369}$	**2.0480**$^{\pm 0.0190}$	**0.9523**$^{\pm 0.0448}$	**1.9503**$^{\pm 0.0312}$
Twitch: CF-GNN	**0.9524**$^{\pm 0.0443}$	$2.6634^{\pm 0.0365}$	$0.9523^{\pm 0.0392}$	$2.6835^{\pm 0.0394}$	$0.9529^{\pm 0.0257}$	$2.5409^{\pm 0.0404}$	$0.9515^{\pm 0.0275}$	$2.6243^{\pm 0.0460}$
Twitch: Cluster-GNN	$0.9531^{\pm 0.039}$	$2.5894^{\pm 0.042}$	$0.9528^{\pm 0.037}$	$2.5321^{\pm 0.040}$	$0.9534^{\pm 0.034}$	$2.4892^{\pm 0.038}$	$0.9523^{\pm 0.033}$	$2.4723^{\pm 0.041}$
Twitch: RR-GAE	$0.9539^{\pm 0.036}$	**2.4987**$^{\pm 0.039}$	**0.9532**$^{\pm 0.035}$	$2.4567^{\pm 0.037}$	$0.9541^{\pm 0.032}$	$2.3721^{\pm 0.036}$	$0.9529^{\pm 0.031}$	$2.3256^{\pm 0.038}$
Twitch: Cluster-RR-GAE	$0.9515^{\pm 0.0367}$	$5.0491^{\pm 0.0513}$	$0.9541^{\pm 0.0284}$	**2.1005**$^{\pm 0.0189}$	**0.9571**$^{\pm 0.0219}$	**2.2398**$^{\pm 0.0225}$	**0.9535**$^{\pm 0.0280}$	**2.1353**$^{\pm 0.0262}$

Table 10: Node Classification Results with Conformal Baselines (Coverage ↑ / Inefficiency ↓)

Dataset	CF-GNN [1]		DAPS [2]		RR-GNN (Ours)		Cluster-RR-GNN (Ours)	
Model	Cover	Ineff	Cover	Ineff	Cover	Ineff	Cover	Ineff
Cora								
GraphSAGE	$0.9456^{\pm 0.0569}$	$1.6284^{\pm 0.0483}$	$0.9453^{\pm 0.0535}$	$1.8025^{\pm 0.0421}$	$0.9460^{\pm 0.0542}$	$1.6100^{\pm 0.0415}$	**0.9463**$^{\pm 0.0509}$	**1.6076**$^{\pm 0.0397}$
SGC	$0.9461^{\pm 0.0603}$	$1.6633^{\pm 0.04415}$	$0.9452^{\pm 0.0538}$	$1.7856^{\pm 0.0426}$	$0.9462^{\pm 0.0581}$	$1.6297^{\pm 0.0428}$	**0.9490**$^{\pm 0.0662}$	**1.5907**$^{\pm 0.0465}$
GCN	$0.9473^{\pm 0.0556}$	$1.6344^{\pm 0.0418}$	$0.9435^{\pm 0.053}$	$1.7120^{\pm 0.0354}$	$0.9432^{\pm 0.0573}$	$1.6251^{\pm 0.0367}$	**0.9476**$^{\pm 0.0732}$	**1.6315**$^{\pm 0.0303}$
GAT	$0.9464^{\pm 0.0702}$	$1.6278^{\pm 0.0334}$	$0.9480^{\pm 0.065}$	$1.7052^{\pm 0.0384}$	$0.9475^{\pm 0.0624}$	$1.6146^{\pm 0.0351}$	**0.9508**$^{\pm 0.0539}$	**1.6114**$^{\pm 0.0396}$
DBLP								
GraphSAGE	$0.9501^{\pm 0.0523}$	$1.5723^{\pm 0.0683}$	$0.9500^{\pm 0.0420}$	$1.6436^{\pm 0.0627}$	$0.9499^{\pm 0.0531}$	$1.5351^{\pm 0.0473}$	**0.9503**$^{\pm 0.0510}$	**1.5607**$^{\pm 0.0487}$
SGC	**0.9451**$^{\pm 0.0617}$	$1.4274^{\pm 0.0416}$	$0.9427^{\pm 0.0526}$	$1.6020^{\pm 0.0317}$	$0.9462^{\pm 0.0528}$	$1.4286^{\pm 0.0541}$	$0.9443^{\pm 0.0462}$	**1.3921**$^{\pm 0.0624}$
GCN	**0.9473**$^{\pm 0.0596}$	$1.5644^{\pm 0.0733}$	$0.9458^{\pm 0.0565}$	$1.6384^{\pm 0.0703}$	$0.9458^{\pm 0.0702}$	$1.5512^{\pm 0.0295}$	$0.9430^{\pm 0.0713}$	**1.5491**$^{\pm 0.0278}$
GAT	$0.9467^{\pm 0.0717}$	$1.5729^{\pm 0.0463}$	$0.9455^{\pm 0.0685}$	$1.6493^{\pm 0.0455}$	$0.9485^{\pm 0.0589}$	$1.5725^{\pm 0.0349}$	**0.9505**$^{\pm 0.0539}$	**1.5570**$^{\pm 0.0322}$
CiteSeer								
GraphSAGE	$0.9528^{\pm 0.0203}$	$1.1680^{\pm 0.0439}$	$0.9501^{\pm 0.0195}$	$1.3425^{\pm 0.0412}$	$0.9538^{\pm 0.0853}$	$1.1621^{\pm 0.0552}$	**0.9540**$^{\pm 0.0926}$	**1.1679**$^{\pm 0.0605}$
SGC	$0.9525^{\pm 0.0257}$	$1.1827^{\pm 0.0552}$	$0.9513^{\pm 0.0245}$	$1.3578^{\pm 0.0525}$	$0.9579^{\pm 0.0536}$	$1.1782^{\pm 0.0415}$	**0.9594**$^{\pm 0.0582}$	**1.1898**$^{\pm 0.0399}$
GCN	$0.9496^{\pm 0.0392}$	$1.2310^{\pm 0.0332}$	**0.9520**$^{\pm 0.036}$	$1.4026^{\pm 0.0327}$	$0.9512^{\pm 0.0358}$	$1.2189^{\pm 0.0276}$	$0.9518^{\pm 0.0373}$	**1.2153**$^{\pm 0.0290}$
GAT	$0.9508^{\pm 0.0309}$	$1.2396^{\pm 0.0416}$	$0.9513^{\pm 0.0291}$	$1.4152^{\pm 0.0393}$	$0.9535^{\pm 0.0447}$	$1.2085^{\pm 0.0361}$	**0.9562**$^{\pm 0.0491}$	**1.1418**$^{\pm 0.0392}$