Verifying Relational Explanations: A Probabilistic Approach

Abisha Thapa Magar*, Anup Shakya*, Somdeb Sarkhel†, Deepak Venugopal*
*University of Memphis
{thpmagar, ashakya, dvngopal}@memphis.edu
†Adobe Research
{sarkhel}@adobe.com

***Abstract*—Explanations on relational data are hard to verify since the explanation structures are more complex (e.g. graphs). To verify interpretable explanations (e.g. explanations of predictions made in images, text, etc.), typically human subjects are used since it does not necessarily require a lot of expertise. However, to verify the quality of a relational explanation requires expertise and is hard to scale-up. GNNExplainer is arguably one of the most popular explanation methods for Graph Neural Networks. In this paper, we develop an approach where we assess the uncertainty in explanations generated by GNNExplainer. Specifically, we ask the explainer to generate explanations for several counterfactual examples. We generate these examples as symmetric approximations of the relational structure in the original data. From these explanations, we learn a factor graph model to quantify uncertainty in an explanation. Our results on several datasets show that our approach can help verify explanations from GNNExplainer by reliably estimating the uncertainty of a relation specified in the explanation.**

I. Introduction

Relational data is ubiquitous in nature. Healthcare records, social networks, biological data and educational data are all inherently relational in nature. Graphs are the most common representations for relational data and several Deep Neural Network (DNN) based techniques have been developed for reasoning about graph-structured data. Popular Graph Neural Networks (GNNs) include Graph Convolution Networks (GCNs) [1] and Graph Attention Networks (GATs) [2]. At the same time, to improve trust in predictions made by the GNN, we need to be able to explain these predictions. Shakya et al. [3] present a framework for verifying embeddings in GNNs to strengthen the trustworthiness of GNN predictions. While there have been several methods related to explainable AI (XAI) [4], it should be noted that explaining GNNs is perhaps more challenging than explaining non-relational machine learning algorithms that work on i.i.d (independent and identically distributed) data. For instance, techniques such as LIME [5] or SHAP [6] explain a prediction based on interpretable features such as pixel-patches (for images) or words (for language). The quality of explanations produced using such methods are generally verified with human subjects since typically, anyone can understand the explanations that are produced and thus can judge their quality. However, in the case of GNNs, explanations are much more complex and cannot be verified easily through human subjects. Specifically, consider GNNExplainer [7] arguably one of the most widely used explainers for GNNs. Given a relational graph where the task is to classify nodes in the graph, GNNExplainer produces a subgraph as the explanation for a node prediction. Clearly, it is very hard to verify such an explanation using human subjects since the explanation is quite abstract. If the ground truth for the explanation is known in the form of graph structures, then it is easy to verify a relational explanation. However, this does not scale up since considerable domain expertise may be needed in this case to generate correct explanations. Therefore, in this paper, we develop a probabilistic method where we verify explanations based on how the explainer explains counterfactual examples.

The main idea in our approach is to learn a distribution over explanations for variants of the input graph and quantify uncertainty in an explanation based on this distribution. In particular, each variant can be considered as a counterfactual to the true explanation and we represent the distribution over these explanations in the form of a probabilistic graphical model (PGM). In particular, we impose a constraint where the distribution is over explanations for symmetrical counterfactual examples. Intuitively, if the input to the explainer changes, since real-world data has symmetries [8], our distribution will be represented over more likely counterfactual examples.

To learn such a distribution over symmetric counterfactual explanations, we perform a Boolean factorization of the relations specified in the original graph and learn low-rank approximations for them. Specifically, a low-rank approximation represents all the relationships in the data by a smaller number of Boolean patterns. To do this, it introduces symmetries into the approximated relational graph [9]. We explain each of the symmetric approximations using GNNExplainer and represent the distribution over these in the form of a factor graph [10]. We calibrate this using Belief Propagation [11] to compute the distributions over relations specified in an explanation. To quantify uncertainty in an explanation generated by GNNExplainer on the original graph, we measure the reduction in uncertainty in the calibrated factor graph when we inject

This research was supported by NSF awards #2008812 and #1934745. The opinions, findings, and results are solely the authors' and do not reflect those of the funding agencies.

979-8-3503-2445-7/23/$31.00 ©2023 IEEE

knowledge of the explanation into the factor graph.

We perform experiments on several benchmark relational datasets for node classification using GCNs. In each case, we estimate the uncertainty of relations specified in explanations given by GNNExplainer. We use the McNemar's statistical test to evaluate the significance of these estimations on the model learned by the GCN. We compare our approach with the estimates of uncertainty that are directly provided by GNNExplainer. We show that the McNemar's test reveals that using our approach to estimate the uncertainty of an explanation is statistically more reliable than using the estimates produced by GNNExplainer.

II. Related Work

Mittelstadt et al. [12] compare the emerging field of explainable AI (XAI) with what explanations mean in other fields such as social sciences, philosophy, cognitive science or law. It turns out that in these fields, there exists a vast amount of research on different forms of explanation. Typically it has been shown that humans psychologically prefer counterfactual explanations [13]. Schnake et al. [14] show a novel way to naturally explain GNNs by identifying groups of edges contributing to a prediction using higher-order Taylor expansion. GraphLIME [15], an extension of the LIME framework designed for graph data, is another popular explanation method that attributes the prediction result to specific nodes and edges in the local neighborhood. Luo et al. [16] introduce PGExplainer which parameterizes the process of generating explanations to improve the generalizability of explanations. Vu et al. [17] present PGM-Explainer which can generate explanations in the form of a PGM, where the dependencies in the explained features are demonstrated in terms of conditional probabilities. There is also a lot of research work on evaluating the explanations of these explainers. Faber et al. [18] argue that the current explanation methods cannot detect ground truth and they propose three novel benchmarks for evaluating explanations. Sanchez-Lengeling et al. [19] present a systematic way of evaluating these explanation methods by introducing properties like accuracy, consistency, faithfulness, and stability. Although there has been a wide range of research in this field, the uncertainty in the explanation of the GNNs has not been studied. In this paper, we verify the relational explanations in GNN and quantify these uncertainties.

III. Background

A. Graph Convolutional Networks

Given a graph $\mathcal{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X})$ with nodes $\mathbf{V} = \{x_1 \dots x_n\}$, edges $\mathbf{E} = \{e_1 \dots e_k\}$ s.t. $e_k \in (x_i, x_j)$ and $x_i, x_j \in \mathbf{V}$ and features $\mathbf{X} = \{\mathbf{X}_i\}_{i=1}^n$ s.t. $\mathbf{X}_i \in \mathbb{R}^d$. GCN learns representations of nodes from their neighbors by using convolutional layers which is used to classify nodes. The layer-wise propagation rule is as follows:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{1}$$

where, $H^{(l)} \in \mathbb{R}^{n \times d}$ is the feature matrix for layer l, $\tilde{A}$ is the adjacency matrix of graph $\mathcal{G}$, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ is the degree matrix, $W^{(l)}$ is a layer-specific trainable weight matrix and $\sigma(\cdot)$ is the activation function.

B. GNNExplainer

Given a GCN (or any GNN) Φ trained for node classification makes prediction for a single target node Y. GNNExplainer generates a subgraph of the computation graph as an explanation for the prediction. The objective is formulated as a minimization of the conditional entropy for the predicted node conditioned on a subgraph of the computation graph. Specifically,

$$H(Y|\mathcal{G} = \mathcal{G}_s, \mathbf{X} = \mathbf{X}_s) = -\mathbb{E}_{Y|\mathcal{G}_s, \mathbf{X}_s} \log[P_\Phi(y|\mathcal{G}_s, \mathbf{X}_s)] \tag{2}$$

where $\mathcal{G}_s$ is a subgraph of the computation graph and $\mathbf{X}_s$ is a subset of features. The subgraph is obtained by retaining/removing edges/nodes from the graph.

IV. Verification of Explanations

We develop a likelihood-based approach on top of GNNExplainer, arguably one of the most well-known approaches for explaining relational learning in DNNs, to estimate the uncertainty in an explanation. To do this, we learn a probabilistic graphical model (PGM) that encodes relational structure of explanations. We then perform probabilistic inference over the PGM to estimate the likelihood of a specific explanation.

A. Counterfactual Relational Explanations

Definition 1. *A discrete PGM is a pair (*$\mathbf{X}$,$\mathbf{F}$*), where* $\mathbf{X}$ *is a set of discrete random variables and* $\mathbf{F}$ *is a set of functions,* $\phi \in \mathbf{F}$ *is defined over a subset of variables referred to as being in its scope. The joint probability distribution is the normalized product of all factors.*

$$P(\bar{\mathbf{x}}) = \frac{1}{Z}\prod_{\phi}\phi(\mathbf{x})$$

where Z *is the normalization constant and* $\phi(\mathbf{x})$ *is the value of the function when* $\mathbf{x}$ *is projected on its scope.*

The *primal graph* G of a PGM is the structure of the PGM where nodes represent the discrete random variables and cliques in the graph represent the factors. An undirected PGM is also called as a *Markov Network*. A directed PGM is a *Bayesian Network* where edges represent causal links and factors represent conditional distributions, specifically, the conditional distribution of a node in G given all its parents. For the purposes of generalizing notation, we can consider these as factors. However, in a Markov Network, the product of factors is not normalized and to represent a distribution, we need to normalize this with the partition function, while in a Bayesian Network the product of factors is already normalized. A *factor graph* is a discrete PGM represented as a bi-partite graph, where there are two types of nodes, namely, variables and factors. The edges connect variable nodes to factor nodes. The factor represents a function over the variables connected to it (the scope of the factor). Typically, the variables are connected through a logical relationship in the

factor function. Each factor function has an associated *weight* that encodes confidence in the relationship over variables within its scope. Higher confidence in the relationship implies higher weights and vice-versa. A factor graph can be converted to an equivalent Markov network.

A *relational graph* $\mathcal{G}$ is a graph where nodes represent real-world entities and edges represent binary relationships between the entities. For our purposes, we assume that the relationships in $\mathcal{G}$ are not directed. Let Φ be a DNN trained for the node classification task. That is, let $\mathbf{V} = x_1 \dots x_n$ be the nodes where $\mathbf{X} = \{\mathbf{X}_i\}_{i=1}^n$ are their features and Φ learns to classify nodes into one of C classes, $f : \mathbf{V} \rightarrow C$.

Let $E(\mathcal{G}, \Phi, Y)$ denote the GNNExplainer's explanation for Φ classifying node Y in $\mathcal{G}$. $E(\mathcal{G}, \Phi, Y)$ is a subgraph, i.e., a set of relations/edges (we use relations and edges interchangeably since we assume binary relationships) in $\mathcal{G}$ that explains the label assigned to Y by Φ. We estimate the uncertainty in $E(\mathcal{G}, \Phi, Y)$ based on a PGM distribution over *counterfactual relational explanations*.

Definition 2. *Given an explanation $E(\mathcal{G}, \Phi, Y)$, a counterfactual relational explanation (CRE) is $E(\hat{\mathcal{G}}, \Phi, Y)$, where $\mathcal{G}$ and $\hat{\mathcal{G}}$ differ in at least one relation.*

B. Boolean Factorization

Note that computing the full set of CREs is not scalable since the size of the CRE set is exponential in the size of the relational graph. Therefore, we focus on a subset of CREs that best quantify uncertainty in the explanation. Before formalizing our approach, we illustrate this with a simple example. Consider the example shown in Fig. 1. To generate CREs, instead of modifying relations randomly, we add/remove relations that result in *symmetrical structures* as shown in the example. Thus, under the hypothesis that symmetries are ubiquitous in the real-world [8], symmetrical CREs are likely to explain more probable counterfactual examples. Thus, a PGM over symmetrical CREs will better encode uncertainty in explanations.

Formally, let $\mathbf{P}$ represent all the relations in $\mathcal{G}$. We want to approximate $\mathbf{P}$ which can be represented as a $n \times m$ matrix using at most k Boolean *patterns*. Specifically, the objective is as follows.

$$\arg\min_{\mathbf{QR}} |\mathbf{P} \ominus (\mathbf{Q} \otimes \mathbf{R})| \tag{3}$$

where $\mathbf{Q}$ is a Boolean matrix of size $n \times k$ and $\mathbf{R}$ is Boolean matrix of size $k \times m$ rows. The Boolean operations are defined as follows. $A \oplus B = A \vee B$, $A \ominus B = (A \wedge \neg B) \vee (\neg A \wedge B)$ and $X^{n \times m} = A^{n \times k} B^{k \times m}$, where $X_{ij} = \vee_{l=1}^{k} A_{il} B_{lj}$. The l-th column of $\mathbf{Q}$ and the l-th row of $\mathbf{R}$ is called as the l-th Boolean pattern. To solve the above optimization problem, we use Boolean Matrix Factorization (BMF). The smallest number of patterns for which we can exactly recover $\mathbf{P}$, i.e., the objective value is equal to 0, is known as the Boolean rank of $\mathbf{P}$. It is known that computing the Boolean rank is a NP-hard problem.

Definition 3. *A low-rank approximation for $\mathbf{P}$ with Boolean rank r is a factorization with k patterns such that $k < r$.*

Since in a low-rank approximation, we use fewer patterns than the rank, it results in a symmetric approximation of the original matrix [9]. While there are several approaches for Boolean low-rank approximation [20], we use a widely used approach implemented in NIMFA [21]. Specifically, the problem is formulated as a nonlinear programming problem and solved with a penalty function algorithm. The factorization reduces the original matrix into a binary basis and mixture coefficients. By thresholding the product of the binary basis and mixture coefficient matrices, we obtain the low-rank Boolean approximation of the original matrix. Since it is hard to compute the exact rank, we use an iterative approach to obtain the set of symmetrical CREs. Specifically, for the base explanation, we perform low rank approximation with a starting rank and progressively increase the rank until the objective function in Eq. (3) is below a stopping criteria.

C. Factor Graph Model

We represent a distribution over the set of symmetrical CREs $\mathcal{S}$ using a factor graph. Specifically, each factor function represents a logical relationship between variables in the CRE. One such commonly used relationship in logic is a set of *Horn clauses* of the form $x_i \wedge x_j \Rightarrow Y$, where x_i, x_j represent variables in the explanation and Y is the target of the explanation. However, note that for the factor functions, we have used the logical and ($\wedge$) rather than implication ($\Rightarrow$) since the implication tends to produce uniform distributions. Specifically, whenever the head (left-side) of the horn clause is false, the clause becomes true which is not the ideal logical form for us since we want to quantify the influence of the head on the body (right hand side of the clause which is the target of explanation) when the head is true. Thus, a logical-and works better in practice. For ease of exposition, we assume that the target of an explanation has a binary class. For multi-class targets, we just create p different clauses of the same form each of which corresponds to one of p classes.

Let $\{R_i\}_{i=1}^K$ be the union of all binary relations specified in the explanations in $\mathcal{S}$. The probability distribution over $\mathcal{S}$ is defined as a *log-linear* model as follows.

$$P(\mathcal{S}) = \frac{1}{Z} \exp(\sum_{i=1}^{K} w_i n(R_i, \mathcal{S})) \tag{4}$$

where $n(R_i, \mathcal{S})$ is the number of true clauses of the form $x_{i1} \wedge x_{i2} \wedge T$, where x_{i1}, x_{i2} are entities related by R_i in $\mathcal{S}$, w_i is a real-valued weight for $x_{i1} \wedge x_{i2} \wedge Y$, Z is the normalization constant, i.e., $\sum_{\mathcal{S}'} \exp(\sum_{i=1}^{K} w_i n(R_i, \mathcal{S}'))$.

To learn the weights in Eq. (4), we maximize the likelihood over $\mathbf{S} \in \mathcal{S}$. Specifically,

$$\log \ell(\mathcal{S}) = w_i \sum_{i=1}^{K} n(R_i, \mathcal{S}) - logZ \tag{5}$$

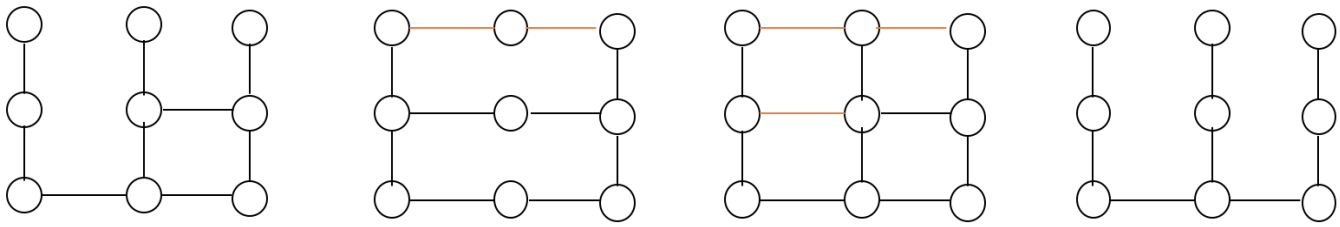


Fig. 1: Given the original graph (the first one), different symmetric approximations of the original graph are shown.

However,clearly Z is intractable to compute. Specifically, we need to sum the probabilities over all possible explanations that can be derived from $\{R_i\}_{i=1}^{K}$ which is exponentially large. Therefore, using gradient descent over the weights, we obtain the following equation for the gradient,

$$\frac{\partial \ell}{\partial w_i} = n(R_i, \mathcal{S}) - \mathbb{E}[n(R_i, \mathcal{S})] \tag{6}$$

It turns out that we can approximate the expectation from the maximum probability (called the MAP) solution to estimate the gradient efficiently (similar to the voted perceptron in [22]). Specifically, using the current weights, we compute the MAP solution which gives us an assignment to each entity in $\mathcal{S}$. From this, we estimate the expected $n(R_i, \mathcal{S})$ as follows. For each explanation $\mathbf{S} \in \mathcal{S}$, we check whether the clause that connects the entities corresponding to R_i in $\mathbf{S}$ is satisfied based on the assignments in the MAP solution. $\mathbb{E}[n(R_i, \mathcal{S})]$ is the total number of satisfied clauses. Finally, we update the weights, $w_i^{(t)} = w_i^{(t-1)} - \epsilon \frac{\partial \ell}{\partial w_i}$, where ϵ is the learning rate. However, it turns out that the initialization of the weights plays an important role in the weights that we eventually converge to [23]. Therefore, we use an initialization based on the explanation scores given by GNNExplainer. That is, we initialize a weight w_i corresponding to relation R_i as the average score assigned by GNNExplainer for R_i over all $\mathbf{S} \in \mathcal{S}$.

D. Uncertainty Estimation

We use Belief Propagation (BP) [24] to estimate the uncertainty in an explanation from the factor graph. Specifically, BP is a message-passing algorithm that uses sum-product computations to estimate probabilities. Specifically, the idea is that a node computes the product of all messages coming into it, and sums out itself before sending its message to its neighbors. In a factor graph, there are two types of messages, i.e., messages from variable nodes to factor nodes and messages from factor nodes to variable nodes. The messages from variable to factor nodes involves only a product operation and the messages from the factor to variable nodes involve both a sum and product operation.

$$v_{var(i)\to fac(s)}(x_i) \propto \prod_{t \in N(i)\setminus s} \mu_{fac(t)\to var(i)}(x_i) \tag{7}$$

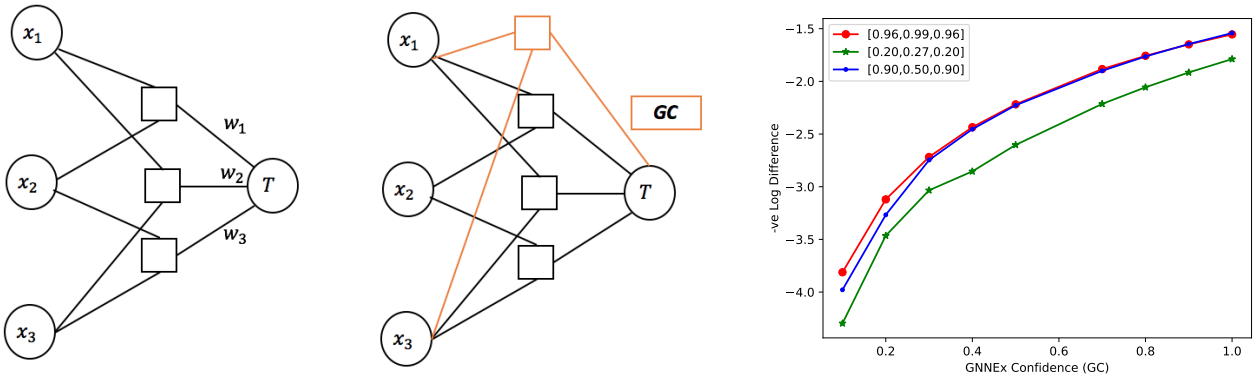
$$\mu_{fac(s)\to var(i)}(x_i) \propto \sum_{x_{N(s)\setminus i}} f_s(x_{N(s)}) \prod_{j \in N(s)\setminus i} v_{var(j)\to fac(s)}(x_j) \tag{8}$$

where $var(i)$ represents a variable node i, $N(i) \setminus s$ represents all the neighbors of i except s, f_s is a factor node. Thus, each variable node multiplies incoming messages from factors and passes this to other factor nodes. The factor nodes multiply the messages with its factor function and sums out all the variables except the one that the message is destined for. In practice, the messages are normalized to prevent numerical errors. The message passing continues until the messages converge. In this case, we say that the factor graph is calibrated and we can now derive marginal probabilities from the calibrated factor graph by multiplying the converged messages coming into a variable node. Specifically,

$$p(x_i) \propto \prod_{t \in N(i)} \mu_{fac(t)\to var(i)}(x_i) \tag{9}$$

We estimate uncertainty of an explanation from the calibrated factor graph. Specifically, let $Q_1 \ldots Q_k$ denote relations in the explanation $E(\mathcal{G}, \Phi, Y)$. We estimate probabilities from the factor graph denoted by $\mathcal{F}$ that we learn from the symmetric CREs $\mathcal{S}$. Intuitively, a relation Q_i is important in $E(\mathcal{G}, \Phi, Y)$ if it is important in $\mathcal{S}$. To quantify this, we compute the change in distributions when we re-calibrate $\mathcal{F}$ by adding new factors obtained from the explanation $Q_1 \ldots Q_k$.

We illustrate our approach with an example in Fig. 2. As shown here, we have a factor graph with three factors, where each factor explains the influence of a pair of nodes on the target. The weights of the factors (w_1, w_2, w_3) encode the uncertainty in the relationship specified by the factors. To quantify the uncertainty of a relation between say x_1, x_3 in explaining T, we obtain the joint distribution over $P(x_1, x_3)$ after calibration using belief propagation. Now, suppose the GNNExplainer gives us an explanation for T that specifies a single relation between x_1, x_3 with a confidence equal to GC. Our goal is to quantify the reduction in uncertainty given this new explanation. To do this, we add a new factor with weight GC and re-calibrate to obtain a modified distribution $\hat{P}(x_1, x_3)$.



(a) (b) (c)

Fig. 2: (a) shows the original factor graph, (b) shows the added factor based on a new explanation x_1, x_3 with confidence GC (c) shows the difference in joint probabilities $p(x_1, x_3)$ before and after the factor is added for different values of GC.

The graph in Fig, 2 (c) shows the -ve log difference between the two distributions (we show it for the case $x_1 = x_3 = 1$). A larger value indicates that the reduction in uncertainty is larger. As seen here, if the prior uncertainty is high (the green plot), then, it requires GC to be very large (over 90%) to achieve the same level of reduction in uncertainty as is the case for a much smaller GC (around 60%) when the prior uncertainty is low (the red/blue plots). Further, even though the weight vector for the red plot has larger values than those in the blue plot, we see that the reduction in uncertainty is comparable over all values of GC. This is because for the red plot, all the weights are large and therefore, the relationship between x_1, x_3 is not significantly more important than the other relationships. Thus, additional knowledge that x_1, x_3 is an explanation does not result in a large reduction in uncertainty. On the other hand, as seen by the weights in the blue plot, the factor encoding the relationship between x_1, x_3 has a much larger weight compared to that for x_2, x_3. Thus, there is a significant difference when the target is explained with a relation between x_1, x_3 as compared to something else, say x_2, x_3. Therefore, knowledge that a relation between x_1, x_3 is the explanation will greatly reduce uncertainty.

Formally, let us assume that at most k new factors $\hat{t}_1 \ldots \hat{t}_k$ are introduced by the GNNExplainer's explanation whose scope has the variable x_i. Each of these factors has a weight equal to the confidence (a value between 0 and 1) assigned by GNNExplainer which quantifies its confidence that the corresponding relation is an explanation for the target variable. Thus, the new messages from variable i to factors will be of the form,

$$v_{var(i)\to fac(s)}(x_i) \propto \prod_{j=1}^{k} \mu_{fac(\hat{t}_j)\to var(i)}(x_i) \prod_{t\in N(i)\setminus s} \mu_{fac(t)\to var(i)}(x_i) \quad (10)$$

The marginal after re-calibration is given by,

$$p(x_i) \propto \prod_{j=1}^{k} \mu_{fac(\hat{t}_j)\to var(i)}(x_i) \prod_{t\in N(i)} \mu_{fac(t)\to var(i)}(x_i) \quad (11)$$

Thus, as the confidence values grow larger, then the messages get amplified in each iteration of BP and the uncertainty reduces since the marginal probability becomes larger. In our case, we store the joint probabilities for all related entities in the symmetric CRE set. Let $p(x_i, x_j)$ be the joint distribution computed for related entities x_i, x_j after calibration. Let $\hat{p}(x_i, x_j)$ denote the joint distribution after re-calibration upon adding factors based on the GNNExplanation. We compute the difference between $p(x_i, x_j)$ and $\hat{p}(x_i, x_j)$ based on the logical structure of the factors. Specifically, since we assume that the structure is a conjunction over $x_i \wedge x_j \wedge Y$, where Y is the explanation target. We compute the average difference over all cases where the formula is satisfied. In the binary case, this corresponds to $x_i = x_j = T = 1$. This difference is a measure of reduction in uncertainty when x_i, x_j is related in the explanation.

Algorithm 1 summarizes our full approach. Our input is the GNNExplainer's explanation for a relational graph $\mathcal{G}$, using DNN Φ for target Y. Our output is a measure for the reduction

Algorithm 1: Uncertainty Quantification

Input: GNNExplanation $E(\mathcal{G}, \Phi, Y)$ with confidences $\mathbf{GC}$
Output: δ_{x_i,x_j}, the uncertainty for the relation connecting x_i, x_j in $E(\mathcal{G}, \Phi, Y)$

```
// Computing the symmetric CREs
```
Initialize rank r
Initialize err as ∞
while $err < t$ **do**
```
    // Low rank approximation for G with
       rank r
```
    $\mathbf{P}$ = Adjacency matrix for $\mathcal{G}$
    Factorize $\mathbf{P}$ into $\mathbf{Q}, \mathbf{R}$ with r Boolean patterns
    $\hat{\mathbf{P}} = \mathbf{Q}^\top \mathbf{R}$
    $\hat{\mathcal{G}}$ = Relational graph with adjacency matrix $\hat{\mathbf{P}}$
    $\mathcal{S} = \mathcal{S} \cup E(\hat{\mathcal{G}}, \Phi, Y)$
    Increment r
    $err = |\mathbf{P} - \hat{\mathbf{P}}|$
```
// Factor Graph
```
FG = Factor Graph of $\mathcal{S}$
Initialize weights of FG using GNN confidences in explanations for $\mathcal{S}$
Learn weights of FG using gradient descent
```
// Calibration
```
$BP(FG)$ = Calibrated FG
$p(x_i, x_j)$ = Joint distribution using $BP(FG)$
for *each* $(x_i, x_j) \in E(\mathcal{G}, \Phi, Y)$ **do**
    Add factor over (x_i, x_j, T) to FG
$\hat{BP}(FG)$ = Re-calibrate FG
$\hat{p}(x_i, x_j)$ = Joint distribution using $\hat{BP}(FG)$
return $p(x_i, x_j) - \hat{p}(x_i, x_j)$

in uncertainty when relation $x_i, x_j \in E(\mathcal{G}, \Phi, Y)$ is part of an explanation for Y. We start by computing the low rank approximations from the input relational graph $\mathcal{G}$. We then explain each of the CREs with GNNExplainer to obtain the set of symmetric CREs $\mathcal{S}$. Next, we construct a factor graph FG from $\mathcal{S}$ and learn its weights. We calibrate FG using belief propagation and obtain the joint distribution $p(x_i, x_j)$. We then add factors to FG from relations in $E(\mathcal{G}, \Phi, Y)$ with weights equal to the explanation confidence assigned to the relations. We then re-calibrate the changed FG and compute $\hat{p}(x_i, x_j)$. We return the difference between $p(x_i, x_j)$ and $\hat{p}(x_i, x_j)$.

V. Experiments

A. Evaluation Procedure

We evaluate our approach by measuring the significance of the uncertainty measures that we obtain for an explanation. Specifically, we learn a GCN for node prediction from the input graph and explain target nodes using GNNExplainer on the learned GCN. For each explanation, we apply our approach to compute the uncertainty scores for the relations in the explanation. We then modify the original graph based on these scores and observe changes in predictions made by the GCN. Specifically, we use an approximate statistical test known as the McNemar's test [25] for quantifying differences in prediction. We do this to avoid Type I errors. That is errors made by an approximate statistical test where a difference is detected even though no difference exists. In the well-known work by Dietterich [26] on comparing predictions made by difference classifiers, it is shown that the McNemar's test has a low Type I error. McNemar's test is also widely used to evaluate paired binomial data in medicine [27]. We run this test as follows. Let $\mathcal{G}$ be the original relational graph and Φ be the GCN learned from $\mathcal{G}$. We run our approach to obtain uncertainty measures for explanations in predictions made by Φ on $\mathcal{G}$ and we remove the relation with the i-th highest score (higher score means lower uncertainty that the relation is part of the explanation) from $\mathcal{G}$ for each of the target nodes. Thus, we get a reduced graph denoted by $\mathcal{G}^{(i)}$. We then learn a new GCN $\Phi^{(i)}$ on $\mathcal{G}^{(i)}$ and compare the predicted values in Φ with the predicted values in $\Phi^{(i)}$ through the McNemar's test. If the removed relations are significant, then we would observe a higher score in the McNemar's test along with a small p-value that rules out the null hypothesis that there is no significant difference in predictions made by Φ and $\Phi^{(i)}$.

Datasets	# nodes	# edges	# classes
BAShapes	700 *# Base nodes*: 300 *Motif shape*: house *Motif size*: 5 *Motif number*: 80	2055	4
BACommunity	1400 *union of two* *BA-Shapes graphs*	3872	7
TreeCycle	871 *Base shape*: balanced tree of height 8 *Motif shape*: cycle *Motif size*: 6 *Motif number*: 80	962	2
TreeGrid	1231 *Base shape*: balanced tree of height 8 *Motif shape*: grid *Motif size*: 3	1705	2
Cora	2708	10556	7
Citeseer	3327	9228	6
Cornell	183	298	5
Texas	183	325	5
Wisconsin	251	515	5

TABLE I: Benchmarks used for evaluation.

B. Setup

We implement our approach using the Deep Geometric Learning (DGL) library in Pytorch. We run all our experiments on a single Tesla GPU machine with 64GB RAM. For the factor graph learning and to perform inference using belief propagation, we used the implementations in the pgmpy library which is an open-source python implementation for PGMs. For the GCN, we varied the hidden dimensions size between 16 and 512, and chose the one with the optimal validation accuracy for a given dataset. We used a maximum of 10K epochs for training the GCN. We used the GNNExplainer from DGL to explain node predictions made by the GCN. We set the number of hops to 2 in the explanations, i.e., any relation that is an explanation for a target node is at most two hops away from the target. For the low rank approximation, we used

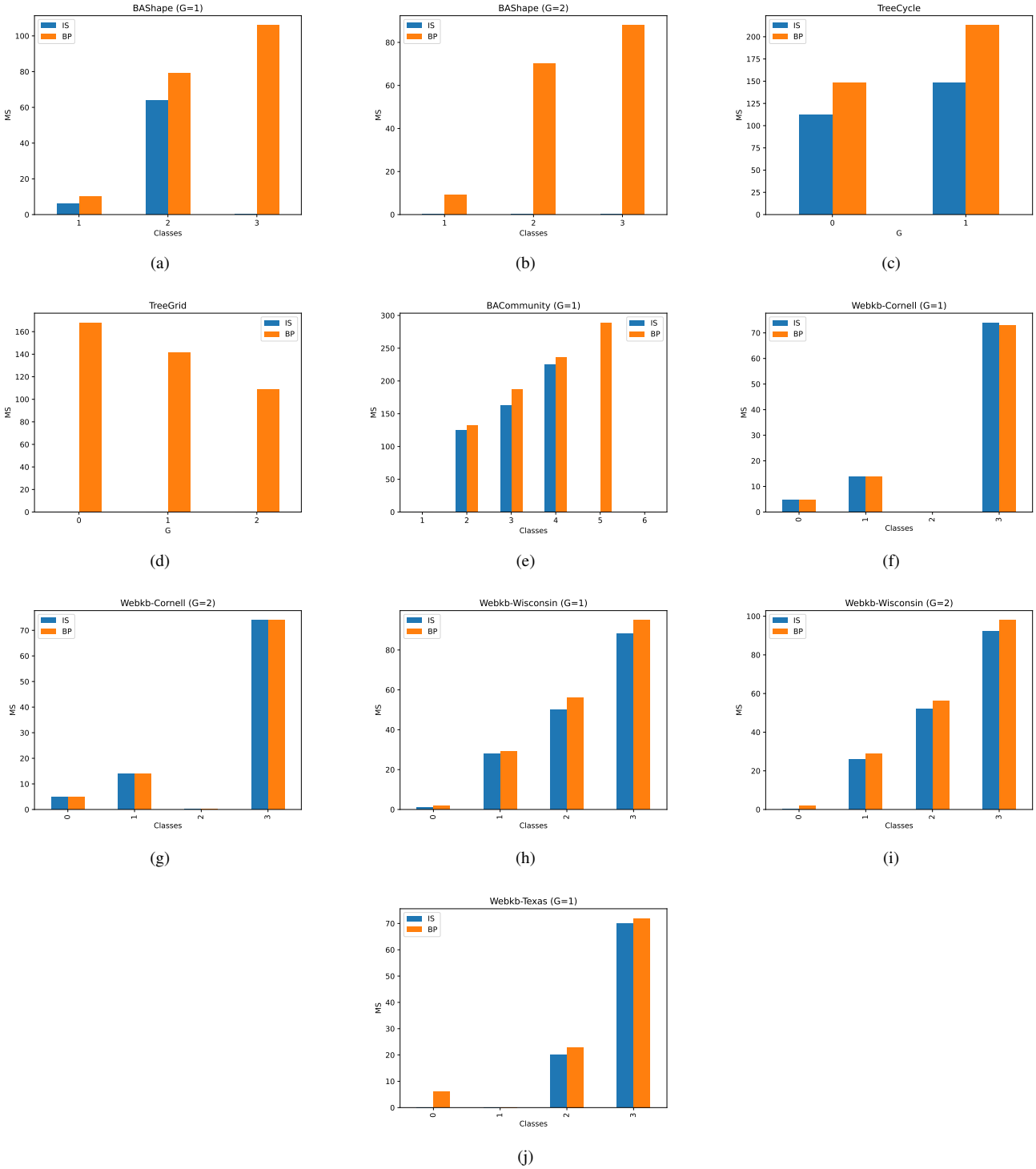


Fig. 3: Results from McNemar's test to verify uncertainty quantification in benchmarks.

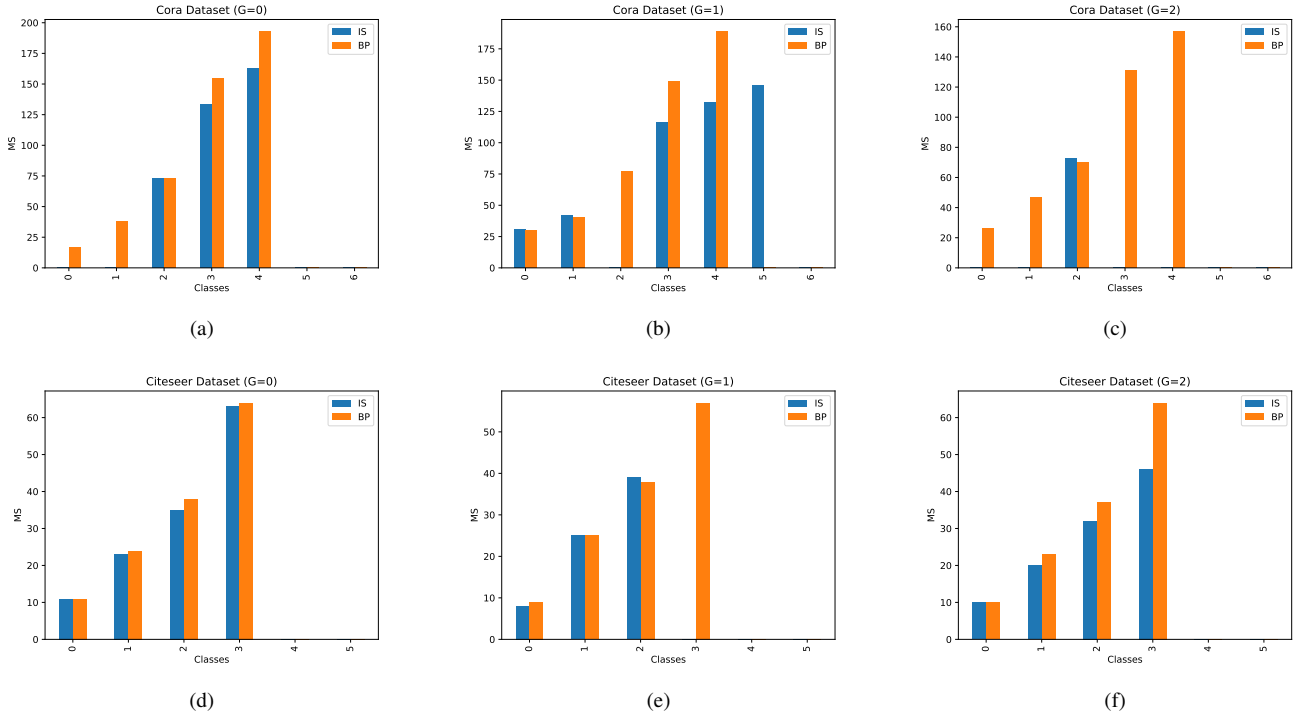


Fig. 4: Results from McNemar's test to verify uncertainty quantification in benchmarks.

Nimfa [21] which is a python library for Nonnegative Matrix Factorization. We used the default parameters for BMF but set the number of iterations between 10K and 100K depending on the dataset. We initialized the rank to one where the approximation error was less than 25% of the total number of edges in the input graph and stopped increasing the rank if we observed that the rank plateaued or if the approximation error was within 5% of the total number of edges.

C. Datasets

We use the benchmark datasets shown in Table I to evaluate our approach. These benchmarks are widely used in explanations for GNNs and also in relational learning. Specifically, we compare our results with uncertainty estimates from GNNExplainer. That is, for each relation in the explanation, GNNExplainer assigns a score based on the conditional entropy equation Eq. (2). A higher score for a relation indicates its higher importance in the explanation. As described in our evaluation procedure, we create a reduced graph based on these scores to compare them with our approach. We denote our approach as BP and the GNNExplainer based scores as IS in the results.

In some of the cases, the explanation contains a single node or a single edge, we filter out such cases from our results. We run the McNemar's test for each class separately and report the statistic values. For the BAShapes, BACommunity, TreeGrid and TreeCycle datasets, we show results for all classes except when the node class is equal to 0 which corresponds to the base nodes. In this case, we explain the nodes from the motifs attached to the base nodes which have non-zero class values. In case the statistic is not significant, i.e., the null hypothesis is true that there is no change in predictions made by $\Phi^{(i)}$ and Φ, we report the statistic as 0. When a class contained less than 10 nodes, we do not report that class in the results since the p-values were not significant. We show the results for $\Phi^{(1)}$, $\Phi^{(2)}$), ... (labeled as $G = 1, G = 2, \ldots$ in the graphs) as long as roughly, the same number of edges are removed for both IS and BP. Our data and implementation are available here[1].

D. Results

The results are shown in Fig. 3 and Fig. 4. For BAShapes, BP scores higher McNemar's statistic (MS) values over all classes. Further, using IS, we could not obtain statistical significance for $G = 2$ which indicates poorer uncertainty quantification. For TreeCycle for both $G = 1$ and $G = 2$, the MS scores for BP were larger than IS once again illustrating that our approach yields better quantification of uncertainty. For TreeGrid, we could not obtain statistical significance for any of the values of G. This also indicates that as the structure gets more complex (BAShapes is simpler than TreeCycle and TreeGrid), uncertainty quantification becomes more reliable using our approach. For BACommunity, our results were slightly better than IS and also more significant over some classes. For WebKB, we observed that IS and BP had very similar performance. One of the reasons for this is related to the accuracy of the GCN model. The accuracy here was significantly lower (between 50-60$\%$) for all three datasets in

[1]https://anonymous.4open.science/r/explain

WebKB. Thus, it indicates that when the underlying GCN has poor performance, explanations may be harder to verify. We plan to study this connection further in the future work. For the Cora dataset, BP achieves better MS scores and significance compared to IS for most values of G. As G increases, the significance of IS reduces, for instance at $G = 3$, most of the values produced by IS were statistically insignificant. Citeseer shows similar results, where for $G = 0$, BP and IS gave us similar results, but for larger values of G, the MS values given by BP was much more significant than those for IS. Thus to summarize, over most of the tested benchmarks, using BP, we were able to quantify uncertainty in the relations of the explanation in a more statistically significant manner compared to IS.

VI. Conclusion

Explanations for relational data are harder to interpret since they involve complex structures (e.g. graphs). In this work, we developed an approach to verify such explanations using a probabilistic model. Specifically, we learn a distribution from multiple counterfactual explanations. In particular, we chose counterfactuals to represent symmetrical approximations of the original graph and learned such an approximation using low-rank Boolean factorization. From the counterfactual explanations, we learn a factor graph to estimate uncertainty in relations specified by a new explanation. Our results on several benchmarks show that these estimates are statistically more reliable compared to estimates from GNNExplainer.

In future, we will extend our approach to interactive verification, where we can help debug a model by interacting with a user. Further, we will also explore specific applications in domains such as education where we can verify context-specific explanations.

References

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *CoRR*, vol. abs/1609.02907, 2016.

[2] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2018.

[3] A. Shakya, A. T. Magar, S. Sarkhel, and D. Venugopal, "On the verification of embeddings with hybrid markov logic," in *Proceedings of 23rd IEEE ICDM*, 2023.

[4] D. Gunning, "Darpa's explainable artificial intelligence (XAI) program," in *ACM Conference on Intelligent User Interfaces*, 2019.

[5] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD*, 2016, pp. 1135–1144.

[6] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *NeurIPS*, vol. 30, 2017.

[7] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," in *NeurIPS*, vol. 32, 2019.

[8] F. Ball and A. Geyer-Schulz, "How symmetric are real-world graphs? a large-scale study," *Symmetry*, vol. 10, 2018.

[9] G. Van den Broeck and A. Darwiche, "On the complexity and approximation of binary evidence in lifted inference," in *NeurIPS*, vol. 26, 2013.

[10] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, vol. 21, pp. 28–41, 2004.

[11] J. S. Yedidia, W. Freeman, and Y. Weiss, "Generalized belief propagation," in *NeurIPS*, vol. 13, 2000.

[12] B. Mittelstadt, C. Russell, and S. Wachter, "Explaining explanations in ai," in *Proceedings of FAT'19*, 2019, p. 279–288.

[13] "Explanation in artificial intelligence: Insights from the social sciences," *Artificial Intelligence*, vol. 267, pp. 1–38, 2019.

[14] T. Schnake, O. Eberle, J. Lederer, S. Nakajima, K. T. Schütt, K.-R. Müller, and G. Montavon, "Higher-order explanations of graph neural networks via relevant walks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, pp. 7581–7596, 2022.

[15] Q. Huang, M. Yamada, Y. Tian, D. Singh, and Y. Chang, "Graphlime: Local interpretable model explanations for graph neural networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, pp. 6968–6972, 2023.

[16] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, "Parameterized explainer for graph neural network," in *NeurIPS*, vol. 33, 2020.

[17] M. Vu and M. T. Thai, "Pgm-explainer: Probabilistic graphical model explanations for graph neural networks," in *NeurIPS*, vol. 33, 2020, pp. 12 225–12 235.

[18] L. Faber, A. K. Moghaddam, and R. Wattenhofer, "When comparing to ground truth is wrong: On evaluating gnn explanation methods," in *Proceedings of the 27th ACM SIGKDD*, 2021, p. 332–341.

[19] B. Sanchez-Lengeling, J. Wei, B. Lee, E. Reif, P. Wang, W. Qian, K. McCloskey, L. Colwell, and A. Wiltschko, "Evaluating attribution for graph neural networks," in *NeurIPS*, vol. 33, 2020, pp. 5898–5910.

[20] C. Wan, W. Chang, T. Zhao, M. Li, S. Cao, and C. Zhang, "Fast and efficient boolean matrix factorization by geometric segmentation," *AAAI*, vol. 34, pp. 6086–6093, 2020.

[21] M. Žitnik and B. Zupan, "Nimfa : A python library for nonnegative matrix factorization," *JMLR*, vol. 13, no. 30, pp. 849–853, 2012.

[22] P. Singla and P. Domingos, "Discriminative Training of Markov Logic Networks," in *Proceedings of the Twentieth National Conference on Artificial Intelligence*. AAAI Press, 2005, pp. 868–873.

[23] I. Sutskever and T. Tieleman, "On the convergence properties of contrastive divergence," in *AISTATS*, vol. 9, 2010, pp. 789–795.

[24] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Generalized Belief Propagation," in *NeurIPS*, 2001, pp. 689–695.

[25] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, 1947.

[26] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural computation*, vol. 10, no. 7, pp. 1895–1923, 1998.

[27] M. W. Fagerland, S. Lydersen, and P. Laake, "The mcnemar test for binary matched-pairs data: mid-p and asymptotic are better than exact conditional," *BMC Medical Research Methodology*, vol. 13, no. 1, p. 91, 2013.