A Meta-Learning Approach for Training Explainable Graph Neural Networks

Indro Spinelli, *Graduate Student Member, IEEE*, Simone Scardapane, and Aurelio Uncini, *Member, IEEE*

***Abstract*— In this article, we investigate the degree of explainability of graph neural networks (GNNs). The existing explainers work by finding global/local subgraphs to explain a prediction, but they are applied after a GNN has already been trained. Here, we propose a meta-explainer for improving the level of explainability of a GNN directly at training time, by steering the optimization procedure toward minima that allow post hoc explainers to achieve better results, without sacrificing the overall accuracy of GNN. Our framework (called MATE, MetA-Train to Explain) jointly trains a model to solve the original task, e.g., node classification, and to provide easily processable outputs for downstream algorithms that explain the model's decisions in a human-friendly way. In particular, we meta-train the model's parameters to quickly minimize the error of an instance-level GNNExplainer trained on-the-fly on randomly sampled nodes. The final internal representation relies on a set of features that can be "better" understood by an explanation algorithm, e.g., another instance of GNNExplainer. Our model-agnostic approach can improve the explanations produced for different GNN architectures and use any instance-based explainer to drive this process. Experiments on synthetic and real-world datasets for node and graph classification show that we can produce models that are consistently easier to explain by different algorithms. Furthermore, this increase in explainability comes at no cost to the accuracy of the model.**

***Index Terms*— Explainable Artificial Intelligence (AI), graph classification, graph neural network (GNN), meta learning, node classification.**

I. Introduction

GRAPH neural networks (GNNs) are neural network models designed to adapt and perform inference on graph domains, i.e., sets of nodes with sparse connectivity [2], [15], [21], [23]. While a few models were already proposed in between 2005 and 2010 [7], [9], [18], [19], the interest in literature has increased dramatically over the past few years, thanks to the broader availability of data, processing power, and automatic differentiation frameworks. This is part of a larger movement toward applying neural networks to more general types of data, such as manifolds and points clouds, going under the name of "geometric deep learning" [3].

Manuscript received September 17, 2021; revised February 4, 2022; accepted April 26, 2022. This work was supported in part by the CHIST-ERA under Grant CHIST-ERA-19-XAI-009. *(Corresponding author: Indro Spinelli.)*

The authors are with the Department of Information Engineering, Electronics and Telecommunications (DIET), Sapienza University of Rome, 00184 Rome, Italy (e-mail: indro.spinelli@uniroma1.it; simone.scardapane@uniroma1.it; aurelio.uncini@uniroma1.it).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3171398.

Digital Object Identifier 10.1109/TNNLS.2022.3171398

Not surprisingly, GNNs have been applied to a very broad set of scenarios, from medicine [22] to road transportation [29], many of which possess significant risks and challenges and a potentially large impact on the end-users. Because of this, several researchers have investigated techniques to help explain the predictions done by a trained GNN, such as identifying the most critical portions of the graph that contributed to a certain inference [25], to help mitigate those risks and simplify the deployment of the models. Explanation methods can be broadly categorized as model-level explainers [17], [20], [26], which try to extract global explanatory patterns from the trained model, and instance-level algorithms [5], [12], [16], [25], [28], which try to explain individual predictions performed by the model. In this article, we focus on the latter group, although we hypothesize that the ideas underlying the algorithm we propose can also be extended to the former.

More generally, a limitation of most explanatory methods is that they are applied only *after* a model has been trained. However, not every trained GNN is necessarily easy to explain. In critical scenarios, the end-user has no straightforward way to possibly trade off a small amount of accuracy to increase the quality of explanations. We hypothesize that because of the highly nonconvex shape of the optimization landscape of a neural network, multiple models can have similar accuracy but possibly different behaviors when explained. In these scenarios, the literature currently lacks an easy way to steer the optimization of a GNN toward an appropriate level of explainability.

In this article, we investigate **M**et**A**-**T**rain to **E**xplain (MATE), an algorithm to this end that is grounded in the meta-learning literature, most notably, model-agnostic meta-learning (MAML) [6]. The key idea of this article is to find explainable GNNs, by training models that can quickly converge to good explanations when a known instance-level explainer (e.g., GNNExplainer [25]) is applied.

A. Contribution of This Article

We develop a framework to train GNNs such that they can be easily *explained* using any instance-level algorithm. During training, for each iteration we first optimize the model to solve an explanation task, inspired by GNNExplainer, on a random subset of nodes. Then, we meta-update the model starting from the new estimate of its parameters, backpropagating through the explanation's steps.

On a wide range of experiments, we show that MATE consistently finds models whose parameters provide a better

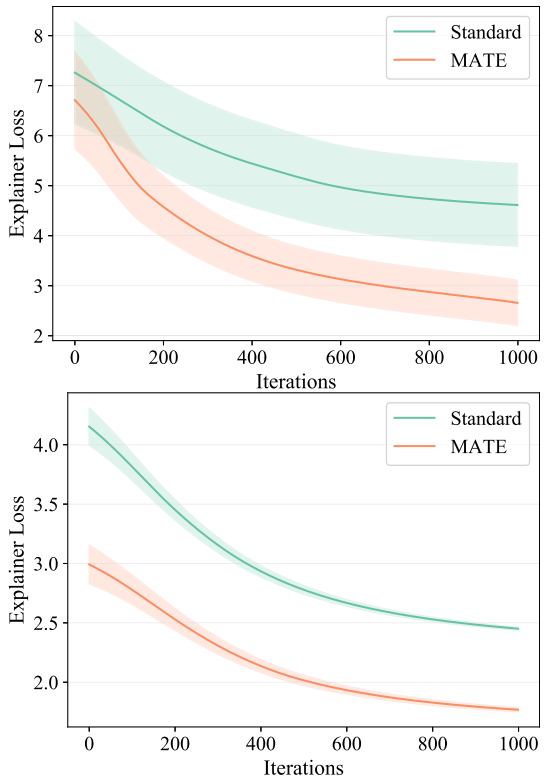


Fig. 1. These plots contain GNNExplainer's losses for every node explained in the evaluation of BA-shapes (Top) and Tree-grids (Bottom) datasets. The local minima found with MATE allows the optimization process to start from a lower value. This often translates into a lower final value and sometimes steeper slopes. We believe that this is the cause of better explainability scores obtained on the MATE-trained models.

starting point for three different instance-level explainers, i.e., GNNExplainer [25], PGExplainer [17], and SubgraphX [28]. We hypothesize that MATE can efficiently steer the optimization process toward better minima (in terms of post hoc explanation). Fig. 1 shows GNNExplainer's loss on two datasets and provides empirical support to our hypothesis. When GNNExplainer interprets the outputs of an MATE-trained model, it starts and ends its optimization process with significantly lower values. To the best of our knowledge, this is the first work proposing an algorithm to improve the degree of explainability of a GNN at training time through a meta-learning framework.

B. Organization of This Article

The rest of this article is structured as follows. In Section I-C, we overview the existing works in the literature. In Section II, we introduce the framework of message-passing GNNs (see Section II-A), instance-level explainers for GNNs (see Section II-B), and we describe in detail GNNExplainer (see Section II-C). Then, in Section III, we introduce MATE, integrating GNNExplainer in a meta-learning bilevel optimization problem to train more interpretable networks. We evaluate our algorithm extensively in Section IV, where we compare the explanations we obtain when running both GNNExplainer and PGExplainer on the trained networks. Finally, we conclude with some final remarks and a series of future improvements in Section VI.

C. Related Works

Providing instance-level explanations in graph domains is more challenging than in other domains (such as computer vision) because of the richness of the underlying data and the general irregularity of the connectivity between nodes. In particular, a single prediction on a node of the graph can depend simultaneously on the features of the node itself, on the features of neighboring nodes (because of the way a GNN diffuses the information over the graph), and even on graph-level properties or specific properties of the community in which the node is residing [26]. Instance-level methods, such as the seminal GNNExplainer [25], work by extracting highly sparse masks from the computational subgraph underlying a single prediction to identify relevant node and edge features. PGExplainer [17] learns a parameterized model trained on the entire dataset to predict edge importance. GraphMask [20] follows a similar approach but predicts which edges can be dropped without changing the model's prediction. Furthermore, it computes the importance of an edge for every layer while PGExplainer only focuses on the input space. SubgraphX [28] explains its predictions by exploring different subgraphs with Monte Carlo tree search. It uses Shapley values to measure the subgraph importance and guide the search. The literature contains different types of algorithms such as global explainers [26], counterfactual explainers [16], and algorithms developed for heterogeneous graphs [24]. For a complete review, we suggest the work of [27].

Our algorithm strongly differs from the literature reviewed up to this point. In fact, it is not an explainer, but rather a training procedure drawing inspiration from MAML [6], whose objective is to facilitate the work of post hoc explainers. MAML is a bilevel optimization method that learns the parameters of a neural network to prepare it for fast adaptation to different tasks, i.e., it finds a set of model parameters that stay relevant for several tasks rather than a single one. Formally, MAML achieves this by adapting the model's parameters θ over a randomly sampled batch of tasks using a few gradient descent updates and a few examples drawn from each one. This process generates different versions of the model's parameters, each one adapted for a specific task. Then, it exploits these modified parameters as the starting point for the meta-update of the global model's parameters. From the trained model, the model can adapt to any additional task sampled from the same distribution with only a small number of gradient descent steps. In our extension of MAML, we identify a task as the explanation of a given node or a graph using a specific instance-level explainer. MATE's overall goal is to modify the training procedure of GNN such that after training, it is easier for any explainer to find an optimum of its corresponding optimization problem. To this end, we introduce an additional set of parameters that work as a surrogate for the explainer during GNN's training. This surrogate is trained on-the-fly and guides the inner loop optimization in which we adapt the GNN to the explanation. We hypothesize that this

optimization pushes the GNN toward a set of parameters that provide better starting point for the explanation process.

II. Preliminaries

A. Graph Neural Networks

The basic idea of GNNs is to combine local (nodewise) updates with suitable message passing across the graph, following the graph topology. In particular, we can represent the input graph $\mathcal{G}$ with the $n \times d$ matrix X collecting all node features (with n being the number of nodes and d the number of features for each node) and by the adjacency matrix $A \in \{0, 1\}^{n \times n}$ encoding its topology. A two-layer graph convolutional network (GCN) [14] working on it is defined as

$$f_\theta(\mathcal{G}) = f_\theta(X, A) = \text{softmax}(D\phi(DX\theta_1)\theta_2) \tag{1}$$

where ϕ is an elementwise nonlinearity [such as the Rectified Linear Unit (ReLU) $\phi(\cdot) = \max(0, \cdot)$], and D is a diffusion operator like the normalized Laplacian or any appropriate shift operator defined on the graph. The model is parameterized by the vector of trainable weights $\theta = [\theta_1, \theta_2]$. The softmax function normalizes the output to a probability distribution over predicted output classes. For tasks of node classification [14], we also know the desired label for a subset of nodes, and we wish to infer the labels for the remaining nodes. Graph classification is easily handled by considering sets of graphs defined as above, with a label associated with every graph (see [8]). In this case, we need a pooling operation before the softmax to compress the node representations into a global representation for the entire graph. In both the scenarios, we optimize the network with a gradient-based optimization, minimizing the cross-entropy loss

$$\mathcal{L} = -\sum_{c=1}^{C} 1[y = c] \log P_\theta(Y = y|\ \mathcal{G}) \tag{2}$$

where C is the total number of classes, 1 is the indicator function, and $P_\theta(Y = y|\ \mathcal{G})$ is the probability assigned by the model f_θ to class y for a single node or graph, depending on the task we wish to solve.

B. Instance-Level Explanations for GNNs

Instance-level methods provide input dependent explanations, by identifying the important input features for the model's predictions. There exist different strategies for extracting this information. Following the taxonomy introduced in [27], we focus our analysis on perturbation-based methods [16], [17], [20], [25]. All these algorithms use a common scheme. They monitor the prediction's change with different input perturbations to study the importance scores associated with each edge or node's features. These methods generate some masks associated with graph features. Then masks, treated as optimization parameters, are applied to the input graph to generate a new graph highlighting the most relevant connections. This new graph is fed to the GNN to evaluate the masks and update them following different rules. The important features for the predictions should be the ones selected by the masks.

Following the notation from Section II-A, let $f_\theta(\mathcal{G})$ be a trained GNN model parameterized by weights θ, making predictions from the input graph $\mathcal{G}$ having node features X and the adjacency matrix A. Given a node v whose prediction we wish to explain, we denote by $\mathcal{G}_v^c = (X_v^c, A_v^c)$ the subgraph participating in the computation of $f_\theta(\mathcal{G})$. For example, for a two-layer GCN, $\mathcal{G}_v^c$ contains all the neighbors up to order two described with the associated adjacency matrix $A_v^c \in \{0, 1\}^{n \times n}$ and their set of features $X_v^c = \{x_j | j \in X_v^c\}$. Instance-level explanation methods generate random masks for the input graph $\mathcal{G}_v^c$ and treat them as training variables $W = [M_v^x, M_v^a]$. These masks are applied to the computational subgraph generating the explanation subgraph $\mathcal{G}_v^e = (X_v^e, A_v^e)$. The explanation is the solution of the optimization problem over $\mathcal{G}_v^e$

$$G_v^{e*} = \arg\min_W \left\{ \mathcal{L}_e\left(\mathcal{G}_v^c, \theta, W\right) \right\} \tag{3}$$

where $\mathcal{L}_e$ is an explanation objective, such that lower values correspond to "better" explanations. In the following, we describe briefly the procedure for GNNExplainer [25], although our proposed approach can be easily extended to any method of the form (3).

C. GNNExplainer

In GNNExplainer [25], the masks are applied to the computational subgraph via pairwise multiplication

$$\mathcal{G}_v^e = \left(X_v^e, A_v^e\right) = \left(X_v^c \odot \sigma\left(M_v^x\right), A_v^c \odot \sigma\left(M_v^a\right)\right) \tag{4}$$

where $W = [M_v^x, M_v^a]$ are the explainer's parameters, $\odot$ denotes the elementwise multiplication, and σ denotes the sigmoid. Then, it optimizes the masks by maximizing the mutual information (MI) between the original prediction and the one obtained with the masked graph. Different regularization terms encourage the masks to be discrete and sparse. Formally, GNNExplainer defines the following optimization framework:

$$\arg\max_{\mathcal{G}_v^e} \ \text{MI}\left(Y, \mathcal{G}_v^e\right) = H\left(Y|\ \mathcal{G}_v^c\right) - H\left(Y|\ \mathcal{G}_v^e\right) \tag{5}$$

where MI quantifies the change in the conditional entropy $H(\cdot)$ (or probability prediction) when v's computational graph is limited to the explanation subgraph. The first term of the equation is constant for a trained GNN. Hence, maximizing the MI corresponds to minimizing the conditional entropy

$$H\left(Y|\mathcal{G}_v^e\right) = -\mathbb{E}_{Y|\mathcal{G}_v^e}\left[\log P_\theta\left(Y|\mathcal{G}_v^e\right)\right] \tag{6}$$

which gives us the subgraph that minimizes the uncertainty of the network's (parameterized by θ) prediction when GNN computation is limited to $\mathcal{G}_v^e$.

When we are interested in the reason behind the prediction of a certain class for a certain node, the conditional entropy is replaced with a cross-entropy objective. Finally, we can use gradient descent-based optimization to find the optimal values for the masks minimizing the following objective:

$$\mathcal{L}_e = -\sum_{c=1}^{C} 1[y = c] \log P_\theta\left(Y = y|\ \mathcal{G}_v^e\right). \tag{7}$$

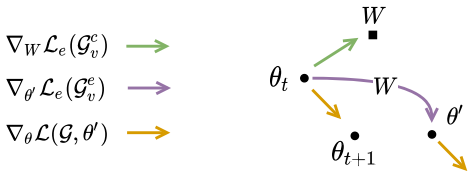


Fig. 2. Diagram of our algorithm, which optimizes for an explainable representation θ.

GNNExplainer deploys some regularization strategies to obtain its explanations. First, an elementwise entropy encourages the masks to be discrete. Second, the sum of all masks' elements (equivalent to their ℓ_1 norm) penalizes large subgraphs. All these strategies contribute to the fact that $\mathcal{G}_v^e$ tends to be a small connected network containing the node to be explained.

III. Proposed Approach

The methods described in Section II-B work by explaining a single instance after the GNN has been trained, decoupling the two steps. In this section, we aim to train models that can be biased toward producing explainable predictions during their training. We first define the problem setup and present the general form of our algorithm. MAML [6] works by optimizing models that can be quickly trained on new tasks. In our framework, we define an "explanation task" on a randomly sampled node and train a model that can quickly converge to a good explanation according to the desired metric (in our case, GNNExplainer). Our goal is a model with more interpretable outputs. By adapting the GNN's parameters for these "explanation tasks," we promote an internal representation based on easily interpretable features (see Fig. 2).

A. Problem Setup

We define the main task, being node or graph classification, as optimizing the original objective function $\mathcal{L}$ with gradient-based techniques (e.g., cross-entropy over the labeled nodes of the graph). We will take into consideration node classification tasks being the extension to graph classification trivial. Then we define a single "explanation task" $\mathcal{T}_v^e = \{f_\theta, \mathcal{G}_v^e, \mathcal{L}_e\}$ for any randomly sampled node v from the graph $\mathcal{G}$. The explanation task requires an explanatory subgraph $\mathcal{G}_v^e$ produced by an explainer trained on-the-fly during the optimization of the model f_θ using its current state θ. The loss $\mathcal{L}_e$ provides task-specific feedback. It is the same loss used during the explainer's optimization with $\mathcal{G}_v^e$ fixed instead of θ. By optimizing the model's parameters θ for a few steps of gradient descent, we will adapt the model's parameters to the explanation, producing a new set of parameters θ'. In Section III-B, we will explain step by step how we use these adapted parameters to perform a meta-optimization for the model's principal objective.

B. Meta-Explanation

We want to steer the optimization process to find a set of parameters that help post hoc explanation algorithms to provide relevant interpretation of the model prediction. We present a graphical representation of our framework in Fig. 3.

Differently from MAML, MATE has two different optimization objectives. The first is the explanation objective $\mathcal{L}_e$ as described in (7) and optimized in the inner loop. The second is the main objective $\mathcal{L}$ (2), a standard cross-entropy loss for the main classification task, optimized in the outer loop. The first one uses the computational subgraph of a randomly sampled node. The second one instead exploits the entire graph.

To perform a single update, we start the inner loop by sampling at random a node v from the graph and extracting its computational subgraph $\mathcal{G}_v^c$. Explaining v's prediction will be our target or our "explanation task." We continue by initializing and training a GNNExplainer minimizing (7) for K gradient steps (with K being a hyperparameter) to obtain the explanation subgraph for v based on the current GNN parameters. A single update is in the form

$$W = W - \delta \nabla_W \mathcal{L}_e\left(\mathcal{G}_v^c, W\right) \tag{8}$$

where we take the gradient of the explanation loss with respect to the explainer's parameters regarding the model's ones fixed. The step size δ, like all the next step sizes, may be fixed or meta-learned.

At this point, we can define our "explanation task" $\mathcal{T}_v^e = \{f_\theta, \mathcal{G}_v^e, \mathcal{L}_e\}$. We adapt the model parameters to $\mathcal{T}_v^e$ using T gradient descent updates. Again, a single update takes the form

$$\theta' = \theta' - \alpha \nabla_{\theta'} \mathcal{L}_e\left(\mathcal{G}_v^e, \theta'\right) \tag{9}$$

where θ' is the vector of the adapted parameters and $\mathcal{G}_v^e$ is the explanation subgraph. We compute the gradient with respect to the model's parameters leaving the explainer's masks fixed. Like the previous update, we have another hyperparameter paired with T, α, representing the step size for adaptation.

The model's parameters are trained by optimizing for the performance of $f_{\theta'}$ with respect to θ for the main classification task, exploiting the entire graph structure. The meta-update is defined as

$$\theta = \theta - \beta \nabla_\theta \mathcal{L}\left(\mathcal{G}, \theta'\right) \tag{10}$$

where β is the meta step size. The meta-optimization updates θ using the objective computed with the adapted model's parameters θ'. We outline the framework in Algorithm 1.

The meta-gradient update involves a gradient through a gradient. We use the Higher library [10] to handle the additional backward passes and to deploy the Adam optimizer [13] to perform the actual updates. The extension to the graph classification task is trivial. Instead of sampling a random node for the explanation, we select an entire graph from the current batch used for the model's update.

IV. Experimental Evaluation

In this section, we evaluate MATE with several experiments over synthetic and real-world datasets. We first describe the datasets and experimental setup. Then, we present the results on both node and graph classification. With qualitative and quantitative evaluations, we demonstrate that GNNExplainer, PGExplainer, and SubgraphX provide better explanation results when used on models trained with

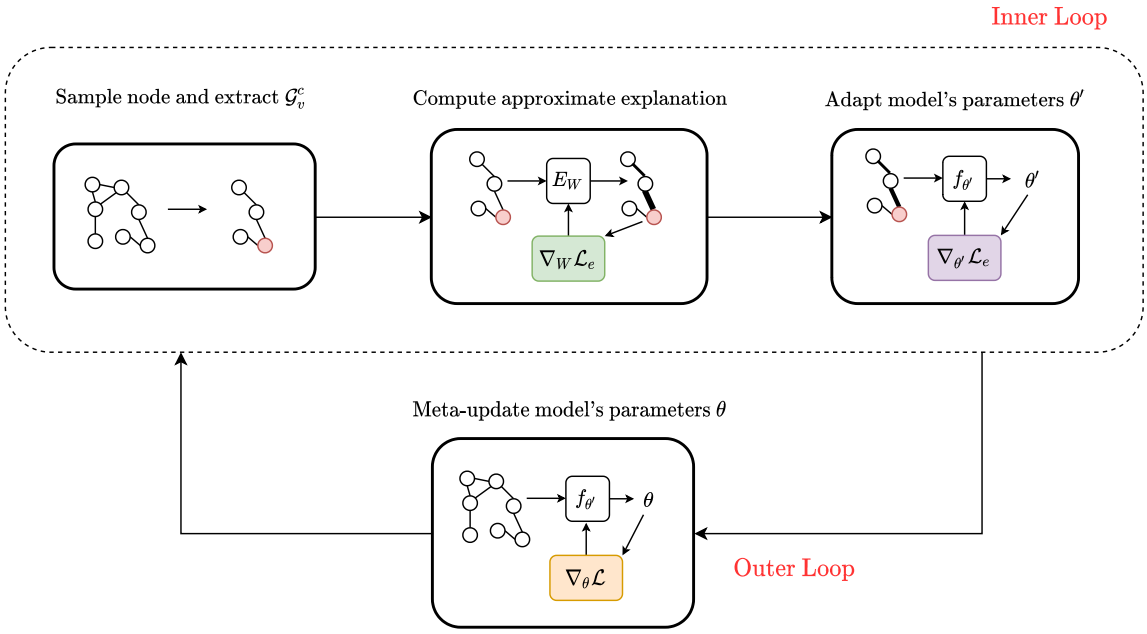


Fig. 3. Schematics of our meta-learning framework for improving GNN's explainability at training time. MATE steers the optimization procedure toward more interpretable minima in the inner loop, meanwhile optimizing for the original task in the outer one. The inner loop adapts the model's parameters to a single "explanation task." It starts with the sampling of a random node and its computational subgraph. Then, we train GNNExplainer to explain the current model's prediction. Afterward, we can adapt the model's parameters to the "explanation task" ending in a new model's state. Finally, we meta-update the original parameters minimizing the cross-entropy loss computed with the adapted parameters.

Algorithm 1 : MATE

Data: Input Graph $\mathcal{G} = (X, A)$
Require: α, β, γ step size hyperparameters, (K,T) number of gradient-based optimization steps.

Initialize model's parameters θ.
while not done **do**
  Random sample v from $\mathcal{G}$ and extract computational graph $\mathcal{G}_v^c$
  Initialize explainer's parameters W
  for K steps **do**
    Compute explainer's parameters (θ fixed) $W = W - \delta \nabla_W \mathcal{L}_e(\mathcal{G}_v^c, W)$
  end for
  for T steps **do**
    Adapt model's parameters (W fixed) $\theta' = \theta' - \alpha \nabla_{\theta'} \mathcal{L}_e(\mathcal{G}_v^e, \theta')$
  end for
  Meta-update $\theta = \theta - \beta \nabla_\theta \mathcal{L}(\mathcal{G}, \theta')$
end while

MATE, in some cases improving the state-of-the-art in explaining node/graph classification predictions. At the same time, we show that our framework does not impact the classification accuracy of the model. We based our implementation[1] upon the code develop in [11].

[1]https://github.com/ispamm/MATE

A. Datasets

Synthetic datasets are very common in the evaluation of explanation techniques. These datasets contain graph motifs determining the node or graph class. The relationships between the nodes or graphs and their labels are easily understandable by humans. The motifs represent the approximation of the explanation's ground truth. In our evaluation, we consider four synthetic datasets for node classification and one for graph classification. Barabási-Albert (BA)-shapes generates a base graph with BA [1] and attaches randomly a house-like, five-node motif. It has four labels, one for the base graph, one for the top node of the house motifs, and one for the two upper nodes, followed by the last label for the bottom ones of the house. BA-Community has eight classes and contains two BA-shapes graphs with randomly attached edges. The memberships of the BA-shapes graphs and the structural location determine the labels. In Tree-cycle, the base graph is a balanced tree graph with a depth equal to 8. The motifs are a six-node cycle. In this case, we have just two labels, motifs and nonmotifs. Tree-grids substitute the previous motif with a grid of nine nodes. Concerning graph classification, BA-2motifs has 800 graphs and two labels. Each network has a base component generated with the BA model. Then one between the cycle and house-like motif, injected in the graph, determines the resulting label. All the node features are vectors containing all 1 s. The other dataset for graph classification is MUTAG [4], a molecular dataset. The dataset contains several molecules represented as graphs where nodes represent atoms and edges chemical bonds. The molecules are labeled based on their mutagenic effect on a specific bacterium.

TABLE I
EXPLANATION ACCURACY OBTAINED ON THE MODEL TRAINED WITH AND WITHOUT OUR META-TRAINING FRAMEWORK

	BA-shapes	BA-community	Tree-cycles	Tree-grids	BA-motifs	MUTAG
GNNExp	0.763 ± 0.006	0.640 ± 0.004	0.479 ± 0.019	0.668 ± 0.002	0.491 ± 0.004	0.637 ± 0.002
MATE+GNNExp	0.851 ± 0.003	0.688 ± 0.004	0.523 ± 0.012	0.628 ± 0.001	0.500 ± 0.001	0.680 ± 0.002
PGExp	0.997 ± 0.001	0.868 ± 0.012	0.793 ± 0.035	0.423 ± 0.012	0.101 ± 0.073	0.811 ± 0.076
MATE+PGExp	**1.000** ± 0.000	**0.910** ± 0.008	**0.870** ± 0.011	**0.853** ± 0.010	**0.963** ± 0.020	**0.873** ± 0.100
SubgraphX	0.548 ± 0.002	0.473 ± 0.002	0.617 ± 0.007	0.516 ± 0.006	0.610 ± 0.006	0.529 ± 0.002
MATE+SubgraphX	0.564 ± 0.008	0.525 ± 0.001	0.642 ± 0.008	0.613 ± 0.015	0.555 ± 0.011	0.576 ± 0.001

TABLE II
MODEL'S ACCURACY WITH AND WITHOUT OUR META-TRAINING FRAMEWORK OBTAINED WITH EARLY STOPPING. THE SCORES ARE IN THE FORMAT TRAIN/VALIDATION/TEST

	BA-shapes	BA-community	Tree-cycles	Tree-grids	BA-2motifs	MUTAG
Standard	0.96/1.0/1.0	0.85/0.74/0.73	0.95/0.98/0.97	0.97/0.99/0.98	0.99/1.0/0.99	0.83/0.84/0.79
MATE	0.97/1.0/1.0	0.84/0.70/0.74	0.96/0.98/0.97	0.97/0.98/0.98	0.99/0.99/0.99	0.82/0.82/0.78

As discussed in [4], carbon rings with chemical groups NH_2 or NO_2 are present in mutagenic molecules. A good explainer should identify such patterns for the corresponding class. However, Luo *et al.* [17] observed that carbon rings exist in both mutagen and nonmutagenic graphs.

B. Baselines

We use the same GNN architectures described in [11]. The model has three graph convolutional layers and an additional fully connected classification layer. For node classification, the last layer takes as input the concatenation of the three intermediate outputs. For graph classification, instead, it receives the concatenation of max and mean pooling of the final output. Concerning the explainers, we use GNNExplainer [25], PGExplainer [17], and SubgraphX [28]. Our baselines will be the explanations provided by the three explainers over the outputs of the GNN architecture described previously, trained in a standard fashion. We train GNN with Adam [13] and an early stopping strategy on a validation split. GNNExplainer and PGExplainer with the GNNs use the hyperparameters fine-tuned by Holdijk *et al.* [11]. For SubgraphX, we used the hyperparameters of the original implementation [28]. We repeat the explanation steps ten times with different seeds and report in Tables I, V, and VI the mean AUC score with the standard deviation.

C. Metrics

Like in recent works, we divide the evaluation into quantitative and qualitative experiments. For the quantitative part, following [11], [17], and [25], we compute the AUC score between the edges inside motifs, considered as positive edges, and the importance weights provided by the explanation methods. Every connection outside the motif has a negative label. High scores for the edges in the ground-truth explanation corresponds to higher explanation accuracy. The qualitative evaluation, instead, provides a visualization of the chosen subgraph. Given the mask, we select all the edges that have the weights satisfying a predefined threshold. Then, we choose only the nodes that are in a direct subgraph together with the node-to-be-explained. Finally, we select only the top-k edges where k is the number of connections in the motifs. Darker edges have higher weights in the mask than the lighter ones. Nodes are color-coded by their ground-truth label.

D. Hyperparameters

MATE requires two sets of hyperparameters. The first regards the optimization process with the tuples (K, δ) and (T, α) and meta step size β. The tuples drive the explainer training and the adaptation procedure. We set $(K = 30, \delta = 0.03), \alpha = 0.0001$ for all the datasets. Then we set $\beta = 0.003$ and $\beta = 0.001$ for the node and graph classification dataset, respectively. We fine-tuned the number of adaptation steps T for each dataset selecting from the values $[1, 3, 5, 10]$. The second parameter set is the one of GNNExplainer. In this case, we used the same hyperparameters of [11].

V. RESULTS

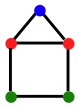
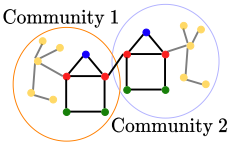
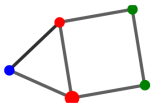
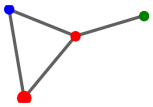
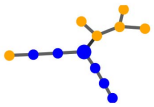
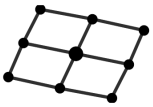
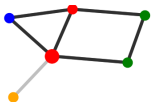
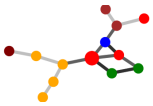
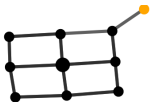
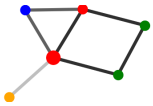
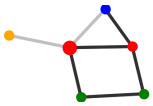
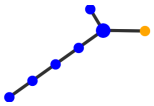
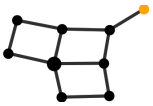
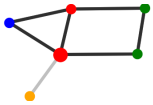
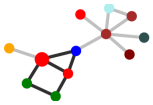
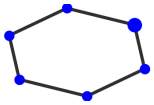
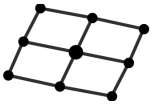
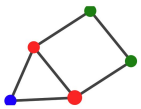
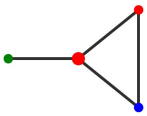
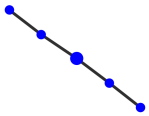
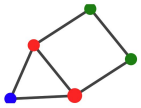
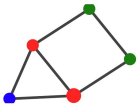
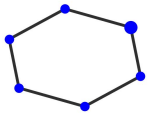
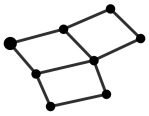
We investigate the question: Does a model trained to be explainable improves the performances of post hoc explanation algorithms?

A. Quantitative

In Table I, we report the results in terms of the explanation accuracy obtained using three different explainers on models trained with and without MATE. We report that in 89% of the cases, the MATE-trained models helped the explainers to outscore their counterparts who interpreted standard models. The average increments are 4.6% points on GNNExplainer, 24.6% on PGExplainer, and 4.7% on SubgraphX. Part of the success of the combination META+PGExplainer is because Holdijk *et al.* [11] were not able to replicate the original results on Tree-grids. Yet, the results of PGExplainer over the

TABLE III

VISUALIZATION OF THE EXPLANATION SUBGRAPHS FOR THE NODE CLASSIFICATION TASK. NODE COLORS REPRESENT NODE LABELS. DARKNESS OF THE EDGES SIGNALS IMPORTANCE FOR CLASSIFICATION. THE GROUND-TRUTH MOTIF IS PRESENTED IN THE FIRST ROW

	BA-shapes	BA-community	Tree-cycles	Tree-grids
Motif				
GNNExp				
MATE+GNNExp				
PGExp				
MATE+PGExp				
SubgraphX				
MATE+SubgraphX				

model trained with MATE are comparable with the ones presented in [17]. We used the same hyperparameters regardless of the explainers used for imputation. Therefore, we believe there is still some margin for improvement with a fine-tuning targeting the explainer's accuracy.

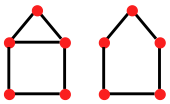
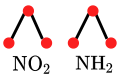
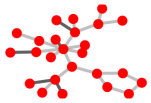
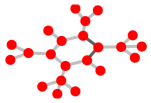
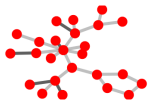
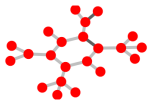
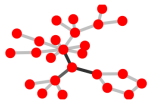
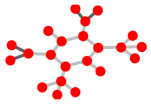
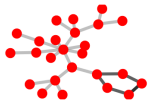
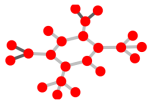
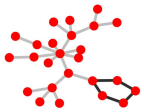
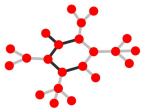
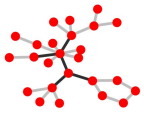
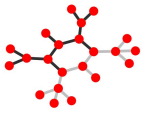
In Table II, we report the accuracies obtained by the GNN model obtained via standard optimization and with our framework. We show that there are no relevant changes in the utility of the model when optimized to be explainable.

B. Qualitative

In this section, we analyze the qualitative aspect of the explanation subgraphs computed by the post hoc explainers over a GNN trained with and without our meta-training framework. Since GNNExplainer and PGExplainer output a soft explanation mask, the intensity of the edges in the subgraph reflects the associated confidence. SubgraphX assigns the same importance to each edge being part of the explanation. We present the results for the node classification task in Table III. GNNExplainer and PGExplainer provide explanations with darker edges inside the motifs on the MATE-trained models, highlighting greater confidence. Most notably, all the explainers find the cycle motif in Tree-cycles when taking the meta-trained model as input. We report a slight improvement or comparable results for all the other datasets. In Table IV, we show the interpretations over the graph classification task. In this scenario, PGExplainer is the best performing model among the baselines, but only the variant trained with our meta-training approach is capable of perfectly highlighting both the five-node cycle motif and the NH_2 and NO_2 motifs. The combination MATE-SubgraphX on the BA-2motif could improve neither the quantitative nor the qualitative evaluation. However, the same combination correctly includes the ground-truth motifs on MUTAG instead of focusing on the carbon ring alone.

TABLE IV

VISUALIZATION OF THE EXPLANATION SUBGRAPHS FOR THE GRAPH CLASSIFICATION. DARKNESS OF THE EDGES SIGNALS IMPORTANCE FOR CLASSIFICATION. THE GROUND-TRUTH MOTIF IS IN THE FIRST ROW

	BA-2motifs	MUTAG
Motif		
GNNExp		
MATE+GNNExp		
PGExp		
MATE+PGExp		
SubgraphX		
MATE+SubgraphX		

C. Ablation

We performed an ablation study on the Tree-cycles dataset. In particular, we observed the change in GNN accuracy and explainability score when varying the number of optimization steps in MATE's inner loop. Table V shows the change when acting on the number of GNNExplainer's training steps. We can observe that this value does not influence the GNN accuracy performances. However, we have found a sweet spot in the range $K = [20, 50]$ for the explanation scores of both the explainers. Table VI shows what happens when we perform a different number of adaptation steps T on the "explanation task." This hyperparameter has a greater impact on both model's accuracy and explainability score. Increasing T worsens the accuracy performance especially for the maximum tested value of $T = 10$. Surprisingly, PGExplainer shares this behavior; meanwhile, GNNExplainer performances increase with higher values of T. We have found a similar behavior for all the datasets taken into consideration.

TABLE V

RESULTS OF AN ABLATION STUDY ON THE EFFECT OF THE NUMBER OF GNNEXPLAINER OPTIMIZATION STEPS ON THE MODEL'S ACCURACY AND AUC SCORE. THE ABLATION STUDY IS PERFORMED USING THE TREE-CYCLES DATASET AVERAGING TEN RUNS

T=3	Tr/Val/Te	PGExplainer	GNNExplainer
K=10	0.95/0.98/0.94	0.847 ± 0.002	0.473 ± 0.016
K=20	0.95/0.98/0.97	0.850 ± 0.007	0.494 ± 0.020
K=30	0.94/0.98/0.97	0.900 ± 0.006	0.542 ± 0.016
K=40	0.96/0.98/0.96	**0.918** ± 0.004	**0.547** ± 0.017
K=50	0.96/0.98/0.97	0.902 ± 0.006	0.534 ± 0.016
K=60	0.95/0.98/0.94	0.838 ± 0.003	0.500 ± 0.019
K=70	0.96/0.98/0.97	0.821 ± 0.004	0.546 ± 0.016
K=80	0.95/0.98/0.97	0.901 ± 0.004	0.546 ± 0.016
K=90	0.96/0.98/0.97	0.838 ± 0.006	0.503 ± 0.020

TABLE VI

RESULTS OF AN ABLATION STUDY ON THE EFFECT OF THE NUMBER OF ADAPTATION STEPS ON THE MODEL'S ACCURACY AND AUC SCORE. THE ABLATION STUDY IS PERFORMED USING THE TREE-CYCLES DATASET AVERAGING TEN RUNS

K=30	Tr/Val/Te	PGExplainer	GNNExplainer
T=1	0.95/0.96/0.95	**0.893** ± 0.009	0.495 ± 0.020
T=3	0.95/0.98/0.97	0.870 ± 0.011	0.523 ± 0.012
T=5	0.94/0.98/0.94	0.832 ± 0.004	**0.554** ± 0.017
T=10	0.90/0.92/0.92	0.850 ± 0.003	0.545 ± 0.018

VI. CONCLUSION AND FUTURE WORKS

In this work, we presented MATE, a meta-learning framework for improving the level of explainability of a GNN at training time. Our approach steers the optimization procedure toward more interpretable minima meanwhile optimizing for the original task. We produce easily processable outputs for downstream algorithms that explain the model's decisions in a human-friendly way. In particular, we optimized the model's parameters to minimize the error of GNNExplainer trained on-the-fly on randomly sampled nodes. Our model-agnostic approach can improve the explanation produced for different GNN architectures by different post hoc explanation algorithms. Experiments on synthetic and real-world datasets showed that the meta-trained model is consistently easier to explain by GNNExplainer, PGExplainer, and SubgraphX. A small ablation demonstrated how MATE balances the model's accuracy with the explainability of its outputs. Furthermore, this increase in explainability does not impact the model's prediction performances. Future works may study the feasibility of this approach for other domains like images, audio, and video.

REFERENCES

[1] R. Albert and A. Barabási, "Statistical mechanics of complex networks," *Rev. Mod. Phys.*, vol. 74, no. 1, pp. 47–97, Jan. 2002.

[2] D. Bacciu, F. Errica, A. Micheli, and M. Podda, "A gentle introduction to deep learning for graphs," *Neural Netw.*, vol. 129, pp. 203–221, Sep. 2020.

[3] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," 2021, *arXiv:2104.13478*.

[4] A. Debnath, R. L. D. Compadre, G. Debnath, A. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity," *J. Med. Chem.*, vol. 34, no. 2, pp. 786–797, Feb. 1991.
[5] L. Faber, A. K. Moghaddam, and R. Wattenhofer, "Contrastive graph neural network explanation," 2020, *arXiv:2010.13663*.
[6] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn. (CoRR)*, Jul. 2017, pp. 1126–1135.
[7] C. Gallicchio and A. Micheli, "Graph echo state networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2010, pp. 1–8.
[8] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, Jul. 2017, pp. 1263–1272.
[9] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, vol. 2, Jul. 2005, pp. 729–734.
[10] E. Grefenstette *et al.*, "Generalized inner loop meta-learning," 2019, *arXiv:1910.01727*.
[11] L. Holdijk, M. Boon, S. Henckens, and L. D. Jong, "[Re] parameterized explainer for graph neural network," ML Reproducibility Challenge, Tech. Rep., 2021.[Online]. Available: https://paperswithcode.com/rc2021
[12] Q. Huang, M. Yamada, Y. Tian, D. Singh, D. Yin, and Y. Chang, "GraphLIME: Local interpretable model explanations for graph neural networks," 2020, *arXiv:2001.06216*.
[13] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2014, pp. 1–5.
[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–4.
[15] M. Li, Z. Ma, Y. G. Wang, and X. Zhuang, "Fast Haar transforms for graph neural networks," *Neural Netw.*, vol. 128, pp. 188–198, Aug. 2020.
[16] A. Lucic, M. T. Hoeve, G. Tolomei, M. de Rijke, and F. Silvestri, "CF-GNNExplainer: Counterfactual explanations for graph neural networks," 2021, *arXiv:2102.03322*.
[17] D. Luo *et al.*, "Parameterized explainer for graph neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 19620–19631.
[18] A. Micheli, "Neural network for graphs: A contextual constructive approach," *IEEE Trans. Neural Netw.*, vol. 20, no. 3, pp. 498–511, Mar. 2009.
[19] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.
[20] M. S. Schlichtkrull, N. D. Cao, and I. Titov, "Interpreting graph neural networks for NLP with differentiable edge masking," 2020, *arXiv:2010.00577*.
[21] I. Spinelli, S. Scardapane, and A. Uncini, "Adaptive propagation graph convolutional network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 10, pp. 1–6, Sep. 2020.
[22] A. Vretinaris, C. Lei, V. Efthymiou, X. Qin, and F. Özcan, "Medical entity disambiguation using graph neural networks," in *Proc. Int. Conf. Manage. Data*, Jun. 2021, pp. 2310–2318.
[23] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.
[24] Y. Yang, Z. Guan, J. Li, W. Zhao, J. Cui, and Q. Wang, "Interpretable and efficient heterogeneous graph convolutional network," *IEEE Trans. Knowl. Data Eng.*, early access, Aug. 6, 2021, doi: 10.1109/TKDE.2021.3101356.
[25] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "GNNExplainer: Generating explanations for graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, p. 9240.
[26] H. Yuan, J. Tang, X. Hu, and S. Ji, "XGNN: Towards model-level explanations of graph neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 430–438.
[27] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," 2020, *arXiv:2012.15445*.
[28] H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji, "On explainability of graph neural networks via subgraph explorations," 2021, *arXiv:2102.05152*.
[29] F. Zhou, Q. Yang, T. Zhong, D. Chen, and N. Zhang, "Variational graph neural networks for road traffic prediction in intelligent transportation systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2802–2812, Apr. 2020.

Indro Spinelli (Graduate Student Member, IEEE) received the master's degree in artificial intelligence (AI) and robotics from the Sapienza University of Rome, Rome, Italy, in 2019, where he is currently pursuing the Ph.D. degree with the Department of Information Engineering, Electronics, and Telecommunications.

He is also a member of the "Intelligent Signal Processing and Multimedia" (ISPAMM) Group, Sapienza University of Rome. His main research interests include graph deep learning and trustworthy machine learning for graph-structured data.

Simone Scardapane is currently an Assistant Professor with the Sapienza University of Rome, Rome, Italy, where he works on deep learning applied to audio, video, and graphs and their application in distributed and decentralized environments. He has authored more than 70 articles in the fields of machine and deep learning.

Dr. Scardapane is a member of the IEEE CIS Social Media Sub-Committee, the IEEE Task Force on Reservoir Computing, and the "Machine Learning in Geodesy" joint-Study Group of the International Association of Geodesy. He is the Chair for the Statistical Pattern Recognition Techniques TC of the International Association for Pattern Recognition and the Chair for the Italian Association for Machine Learning.

Aurelio Uncini (Member, IEEE) received the Laurea degree in electronic engineering from the University of Ancona, Ancona, Italy, in 1983, and the Ph.D. degree in electrical engineering from the University of Bologna, Bologna, Italy, in 1994.

He is currently a Full Professor with the Department of Information Engineering, Electronics and Telecommunications, Sapienza University of Rome, Rome, Italy, where he is teaching neural networks, adaptive algorithm for signal processing, and digital audio processing and he is also the Founder and Director of the "Intelligent Signal Processing and Multimedia" (ISPAMM) Group. His current research interests include adaptive filters, adaptive audio and array processing, machine learning for signal processing, blind signal processing, and multisensors' data fusion.

Dr. Uncini is a member of the Associazione Elettrotecnica ed Elettronica Italiana (AEI), the International Neural Networks Society (INNS), and the Società Italiana Reti Neuroniche (SIREN).