xAI-Drop: Don't Use What You Cannot Explain

Vincenzo Marco De Luca
University of Trento
vincenzomarco.deluca@unitn.it

Antonio Longa
University of Trento
antonio.longa@unitn.it

Pietro Liò
University of Cambridge
pl219@cam.ac.uk

Andrea Passerini
University of Trento
andrea.passerini@unitn.it

Abstract

Graph Neural Networks (GNNs) have emerged as the predominant paradigm for learning from graph-structured data, offering a wide range of applications from social network analysis to bioinformatics. Despite their versatility, GNNs face challenges such as lack of generalization and poor interpretability, which hinder their wider adoption and reliability in critical applications. Dropping has emerged as an effective paradigm for improving the generalization capabilities of GNNs. However, existing approaches often rely on random or heuristic-based selection criteria, lacking a principled method to identify and exclude nodes that contribute to noise and over-complexity in the model. In this work, we argue that explainability should be a key indicator of a model's quality throughout its training phase. To this end, we introduce xAI-Drop, a novel topological-level dropping regularizer that leverages explainability to pinpoint noisy network elements to be excluded from the GNN propagation mechanism. An empirical evaluation on diverse real-world datasets demonstrates that our method outperforms current state-of-the-art dropping approaches in accuracy, and improves explanation quality.

1 Introduction

The capacity to effectively process networked data has a wide range of potential applications, including recommendation systems [1], drug design [2], and urban intelligence [3]. Graph Neural Networks (GNN) [4–7] have emerged as a powerful and versatile paradigm to address multiple tasks involving networked data, from node and graph classification [8–11] to link prediction [12–14] and graph generation [15–17].

Despite their effectiveness and popularity, GNNs face various challenges that prevent their wider adoption and reliability in critical applications [18–20], such as lack of generalization [21]poor interpretability [22], oversmoothing [23, 24], and oversquashing [25]. A significant challenge for GNNs is their vulnerability to noise, as irrelevant or noisy node features can propagate through the layers and degrade the model's performance. Dropping [26] has emerged as an effective paradigm to reduce noise and improve GNN robustness. Dropping can be performed at different granularities, from dropping single messages [27] to dropping edges [28], or even nodes [1, 29]. However, existing approaches often rely on random or heuristic-based selection criteria, and lack a principled method to identify and exclude nodes that contribute to noise and over-complexity in the model.

In this paper, we argue that explainability should be considered a first-class citizen in determining which elements of the graph should be dropped to increase the robustness of the learned GNN. Consider a GNN being trained for node classification. Our intuition is that the fact that the prediction for a given node has a poor explanation is a symptom of a suboptimal function being learned and that this symptom is more harmful if the prediction has high confidence. Figure 1 provides a graphical illustration of this intuition. Guided by this rationale we present xAI-Drop, a novel topological-level

De Luca et al., xAI-Drop: Don't Use What You Cannot Explain. *Proceedings of the Third Learning on Graphs Conference (LoG 2024)*, PMLR 269, Virtual Event, November 26–29, 2024.

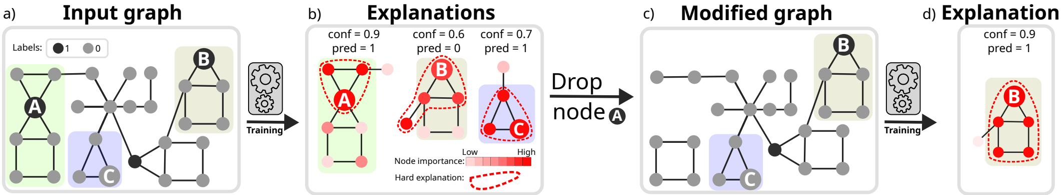


Figure 1: Illustration of the rationale behind XAI-DROP. Panel (a) shows a Barabási-Albert network with house-shaped motifs randomly attached. The task here is to classify nodes as either the top of a house (label 1) or otherwise (label 0). It is easy to see that a triangle is an approximate pattern for the positive class. The figure highlights three prototypical nodes (A, B, C) which are parts of a triangle, where only two of them (A, B) are also the top of a house (triangle and houses highlighted for readability). Panel (b) reports the explanation of a GNN trained on the network, for the three highlighted nodes. Node A has a high confidence because it has both the correct (the house) and spurious (the two triangles) patterns. However, its explanation is mostly based on the (simpler) spurious triangle, which is not sufficient to explain its confidence (as shown by the lower confidence of nodes B and C). Removing node A (Panel (c)) prevents the network from focusing on the spurious pattern so that the correct pattern is eventually learned (Panel (d), with node C omitted as no longer predicted as label 1).

dropping regularizer that leverages explainability and over-confidence to pinpoint noisy network elements to be excluded from the GNN propagation mechanism during each training epoch.

An empirical evaluation on diverse real-world datasets demonstrates that our method outperforms current state-of-the-art dropping approaches in accuracy, and improves explanation quality. Our main contributions can be summarized as follows:

- We identify local explainability *during training* as a driving principle to discard noisy information in the GNN learning process.
- We introduce XAI-DROP, an explainability-guided dropping framework for GNN training.
- We show that XAI-DROP consistently outperforms alternative dropping strategies and xAI-based regularization approaches across various node classification and link prediction benchmarks.
- We demonstrate the effectiveness of XAI-DROP in improving explanation quality.

The rest of the paper is organized as follows. We start by reviewing related work (Section 2) and then introduce the relevant background (Section 3). Our XAI-DROP framework is presented in Section 4 and experimentally evaluated in Section 5. Finally, conclusions are drawn in Section 6.

2 Related Work

Dropping. Dropping strategies are commonly used in neural networks to prevent overfitting [30] by randomly setting a portion of neurons to zero during training, which helps the network learn more robust features. In GNNs, this approach has been extended to the topological level, altering message propagation between nodes, often to reduce oversmoothing [23]. DropEdge [28] was the first to introduce this concept by randomly dropping edges during training based on a Bernoulli distribution. Inspired by DropEdge, subsequent methods include DropNode [31] which drops nodes and their connections, and its variants DropGNN [29], which removes nodes also at test time; DropMessage [27], which drops messages during propagation; and DropAGG [32], which omits the aggregation step for some nodes. While these methods use random sampling, alternative strategies for component dropping have also been explored in the literature. FairDrop [33] combines randomicity and fairness to adjust graph topology for link prediction tasks. Learn2Drop [34] is a learnable graph sparsification procedure deciding which edges to drop to retain maximal similarity to the original network. Beta-Bernoulli Graph Drop Connect (BBGDC) [35] adapts the drop rate of the edges during training based on a Beta-Bernoulli distribution. All these methods rely on random or heuristic-based selection criteria. In this work, we show how a more principled XAI-based method to identify potentially harmful components substantially outperforms existing dropping strategies.

Post-hoc explanability. Several works investigate post-hoc methods to explain the predictions of GNN models. GNN explainers can be categorized into model-level and instance-level explainers.

Model-level explainers [36–38] aim at providing a global understanding of a trained model, e.g., as motifs or rules driving the model to predict a certain class. In contrast, instance-level explainers [39–42] aim at identifying components of a given input that are responsible for the model's prediction for that input, and are thus more appropriate to design an XAI-based dropping strategy. Instance-level explainers can be grouped into five categories [43]: decomposition, surrogate, gradient, perturbation, and generation based. Decomposition-based methods break down the input to identify explanations [44]. Surrogate-based methods rely on an interpretable surrogate to explain the prediction of the original model [40, 45]. Gradient-based methods define explanations in terms of the gradient of the network output with respect to the elements of the input graph [44, 46]. Perturbation-based methods manipulate the input to obtain interpretable subgraphs [39, 47], while generation-based methods generate subgraphs that can explain the model output [48]. In this work, we used a gradient-based method, specifically an approximation of the saliency map [49], due to its computational efficiency (see Appendix C). However, our framework is flexible and can be applied to any explainer that produces node-level explainability scores (see Appendix D).

XAI-based regularization. Several xAI-driven approaches have been proposed to enhance the performance of deep learning methods [50], from addressing interactive data augmentation [51] to enabling automated pruning [52]. A few approaches have been recently proposed to explicitly introduce XAI-based regularization strategies during the training stage of GNNs. MATE [53] applies an optimization procedure via meta-learning to enhance explainability of the resulting model. ExPass [54] works at the message passing level by weighting messages with the importance of nodes as defined by PGExplainer [47], while ENGAGE [55] presents Smoothed Activation Maps [55] to perturb low scores edges and features. These methods however fail to consider the quality of the explanation and are heavily parameterized, resulting in substantial computational overhead, learnability issues, and eventually suboptimal performance. Our experimental evaluation shows how our simple XAI-driven dropping strategies outperform these methods in terms of *both* accuracy and explainability.

3 Preliminaries

In this section, we provide an overview of the fundamental concepts underlying our approach.

Graph. A graph is a tuple $G = (\mathcal{V}, \mathcal{E}, X_{\mathcal{V}}, X_{\mathcal{E}})$, where $\mathcal{V}$ is a set of vertices or nodes, $\mathcal{E}$ is a set of edges between the nodes, $X_{\mathcal{V}}$ and $X_{\mathcal{E}}$ are node features and edge features, respectively. Node and edge features may be empty. The set of edges $\mathcal{E}$ can be represented as an adjacent matrix $A \in R^{|\mathcal{V}|\times|\mathcal{V}|}$, where $A_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}$, 0 otherwise. In this paper, we will focus on undirected graphs, in which edges have no directions, i.e., $A_{ij} = A_{ji}$. Given $v \in \mathcal{V}$, the set $\mathcal{N}_v = \{u \in \mathcal{V} : (u, v) \in \mathcal{E}\}$ denotes the neighborhood of v in G.

Graph Neural Network (GNN). A GNN is a class of neural network architecture specifically designed to process graph data [56–58]. A GNN leverages a message-passing scheme to propagate information across nodes in a graph. GNNs iteratively learn node representations $\mathbf{h}_v$ by aggregating information from neighboring nodes. In most cases, the propagation mechanism for an entire layer can be compactly represented using the adjacency matrix A, the node embedding matrix $H^{(l)}$ and one or more layer-specific weight matrices $W^{(l-1)}$. For instance, layerwise propagation in GCN [58] can be written as:

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

where $\tilde{A} = A + I_{|\mathcal{V}|}$ is the adjacency matrix enriched with self loops, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ and σ is a non-linear activation function such ReLU or sigmoid. Dropping strategies, including our xAI-Drop approach, can be formalized in terms of modifications to the adjacency matrix A.

GNN explainability. Intuitively, given a graph G and a trained GNN f, an explanation is a subgraph $G_{exp} \subset G$ that contains the information that is relevant for f to perform inference on G. We use $G_{exp}(v)$ to denote the local explanation for the GNN output for node v. In this study, we employ the saliency map method [49], an instance-based and gradient-based explainer that computes the attribution for each input by performing backpropagation to the input space. The general idea is that the magnitude of the derivative provides insights into the most influential features, which, when perturbed, result in the highest difference in the output space. Formally, it is defined as

$$G_{exp}(v) = \frac{\partial f_v(G)}{\partial X_v} \tag{1}$$

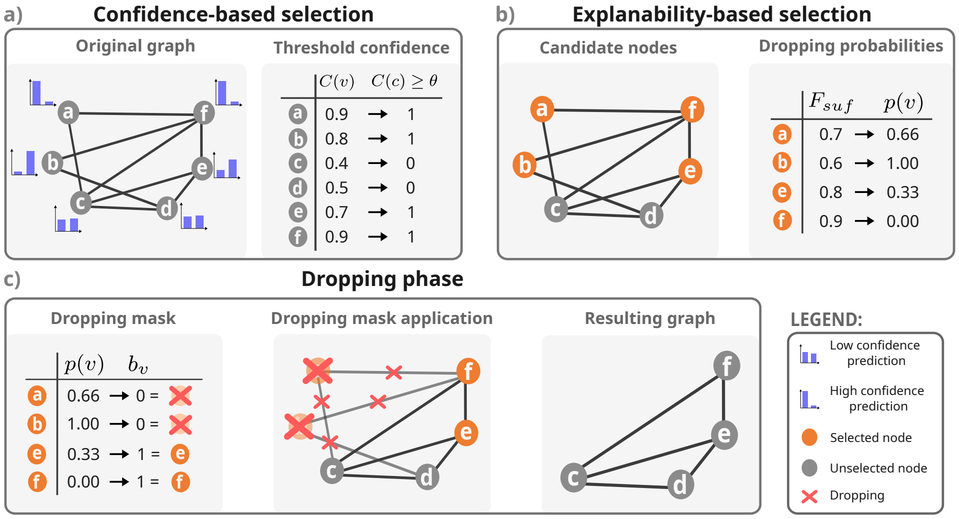
Where $f_v(G)$ is the prediction of the model for node v, and X_v is the feature vector of node v.

Fidelity sufficiency (F_{suf}). Fidelity sufficiency [59] is a popular explainability metric for GNNs. It measures the distance between the probability predicted by f when fed with the entire graph G and the probability when fed with the explanation G_{exp} respectively:

$$F_{suf}(v) = 1 - d(f_v(G), f_v(G_{exp}(v))) \tag{2}$$

with $d(\mathbf{p}, \mathbf{p}')$ being a distance over probability distributions, which in our work we have identified with the Kullback–Leibler divergence.

4 Explainability-Based Dropping



	$C(v)$		$C(c) \geq \theta$
a	0.9	→	1
b	0.8	→	1
c	0.4	→	0
d	0.5	→	0
e	0.7	→	1
f	0.9	→	1

	F_{suf}		$p(v)$
a	0.7	→	0.66
b	0.6	→	1.00
e	0.8	→	0.33
f	0.9	→	0.00

	$p(v)$		b_v
a	0.66	→	0 = ✗
b	1.00	→	0 = ✗
e	0.33	→	1 = e
f	0.00	→	1 = f

Figure 2: A graphical representation of the node dropping strategy (XAI-DROPNODE) employed by the XAI-DROP algorithm for node classification. Panel **a)** shows confidence-based selection, where nodes are selected if the model’s confidence is equal to or greater than a specified threshold θ. Panel **b)** presents the computation of fidelity sufficiency scores and dropping probabilities for the nodes selected in panel **a)**. Lastly, panel **c)** illustrates the computation of the node dropping mask by Bernoulli sampling, and the resulting graph after dropping nodes and their associated edges.

XAI-DROP is based on the combination of two concepts: explainability and (over)confidence. On the one hand, a poor local explanation can be seen as a symptom of an unreliable prediction for the corresponding node, making it a good candidate for being dropped to reduce noise during training. On the other hand, a highly confident prediction for a node indicates that the network is very confident about the features the prediction is based upon, that in principle should correspond to the local explanation. A confident prediction with a poor explanation is thus a combination one would like to avoid as much as possible. Building on these intuitions, XAI-DROP is a general framework that implements a dropping strategy that targets samples with poor explanations and high certainty. In presenting the XAI-DROP framework we focus on the node classification task (and node dropping) with XAI-DROPNODE, but the method can be readily applied to link prediction, as discussed in Section 4.4. In the following, we focus on the transductive setting, which is by far the most common in node classification. The approach can also be applied to inductive settings, like graph classification tasks.

Figure 2 presents a graphical representation of the XAI-DROPNODE approach, which consists of two main phases: node selection and dropping. The node selection phase (further detailed in Section 4.1) consists of four steps. In the first step, the most certain nodes are extracted as candidates for dropping, by setting a threshold θ ($\theta = 0.7$ in Figure 2) over the predicted confidence for the most probable class. In the second step, the fidelity sufficiency score of these nodes is computed using Eq. 2 to assess the local explanation of these predictions. This score is then mapped into a dropping probability $p(v)$ for the node by applying an appropriate transformation (detailed in Section 4.1), such that the nodes having the worst explanations will have the highest probability of being dropped. Finally, a decision on whether to retain or drop each candidate node v is made according to a Bernoulli distribution parameterized by $p(v)$.

4.1 Node selection

First, a forward step is computed on the entire set of nodes $\mathcal{V}$, including training, validation, and test nodes. This first step aims to compute the confidence score for each node v which is computed as:

$$C(v) = \text{max}_y P(y|X_v) \tag{3}$$

where X_v is the feature vector associated with node v. From this large set of nodes, only the most confident nodes are selected as the candidate dropping set, $\mathcal{V}' \subset \mathcal{V}$. For each node $v \in \mathcal{V}'$, its local explanation $G_{exp}(v)$ is computed using an approximation of the saliency map [49] as explainer, which helps to further reduce the computational overhead of generating explanations (see Appendix C and E for details). We opted for a gradient-based explainer because it is not computationally demanding and does not require ground truth explanations, but the method is agnostic with respect to the explanation method being used. Fidelity sufficiency scores are then computed according to Eq. 2. Note that the metric requires a hard explanation, while the produced explanations are soft masks (i.e., a real value associated with each node indicating its importance for the prediction being explained). In this manuscript, we discretize soft explanations by selecting the top 25% of the edges as part of the hard explanation. Nonetheless, the approach can in principle be applied to soft explanations by weighting messages according to the generated explanations.

The next step assigns dropping probabilities to the nodes in $\mathcal{V}'$, such that worse explanations, i.e., low fidelity sufficiency, correspond to higher dropping probabilities. Given a predefined dropping probability p, the idea is to adjust dropping probabilities for individual nodes according to their fidelity sufficiency, without affecting the expected number of nodes selected for dropping. The dropping probability of node $v \in \mathcal{V}'$ is adjusted by mapping the fidelity sufficiency scores into a Gaussian distribution by applying the Box-Cox transformation [60] which is defined as follows:

$$\phi(x;\lambda) = \begin{cases} \frac{(x+1)^{\lambda}-1}{\lambda} & \lambda \neq 0 \\ log(x+1) & \lambda = 0 \end{cases} \tag{4}$$

Where x represents the response variable, which, in our case, is set to $x = 1 - F_{suf}$, indicating that lower sufficiency corresponds to a higher dropping probability. The parameter λ is learnable and selected through log-likelihood maximization to enhance the normality of the transformed data [60]. Finally, the values are normalized and shifted to achieve a mean equal to p. Preliminary experiments showed that this solution achieves better results than using the empirical distribution (Appendix L.)

All nodes $u \in \mathcal{V} \setminus \mathcal{V}'$ that do not belong to the confidence node subset retain the default dropping probability, $p(u) = p$. Overall, this procedure biases the dropping probability to encourage the dropping of potentially noisy nodes, while guaranteeing that the expected number of nodes selected for dropping is equal to the predefined dropping probability p. An empirical evaluation of different strategies to detect candidate noisy nodes is reported in Appendix G.

4.2 Dropping

Once the biased dropping probabilities $p(v)$ have been computed, they can be employed to alter the propagation of information to regularize the learning. In detail, xAI-DropNode removes nodes from the node set $\mathcal{V}$ based on a node dropping mask $\mathbf{b} \in \{0,1\}^{|\mathcal{V}|}$ defined as follows:

$$b_v \sim Bernoulli(1 - p(v)) \tag{5}$$

Once a node is dropped, all its incident edges $\mathcal{I}_v = \{(u,w) \in \mathcal{E} : u = v \text{ or } w = v\}$ are also removed. Following [27], this operation can be compactly represented in terms of a modified adjacency matrix:

$$A' = B\ A\ B \tag{6}$$

where B is a diagonal matrix having the elements of $\mathbf{b}$ on the main diagonal (and zero elsewhere).

4.3 Overall procedure

The overall algorithm for xAI-Drop is outlined in Algorithm 1. The algorithm takes as input a graph G, the GNN architecture to be trained f, and the hyper-parameters θ and p, for further detail about these hyper-parameters refer to Appendix F. In each epoch, the algorithm selects the nodes with a prediction confidence score of at least θ, and computes their explainability in terms of fidelity sufficiency F_suf. The fidelity sufficiency values are then used to determine the node-specific dropping probabilities, guaranteeing that a fraction p of the nodes is dropped in expectation. These probabilities are in turn used to select nodes (and corresponding incident edges) to be dropped and produce an adjusted adjacency matrix A'. Finally, A' is used for another round of training of the GNN f. Note that the dropping procedure is only performed at training time. Indeed, the inference procedure (last line of the algorithm) employs the full adjacency matrix A, consistently with what is done by existing dropping strategies in the literature.

Algorithm 1 xAI-Drop algorithm for node classification tasks. $G = (\mathcal{V}, \mathcal{E}, \mathbf{X}_{\mathcal{V}}, \mathbf{X}_{\mathcal{E}})$ is a graph, f is the GNN, θ, p are hyper-parameters

```
procedure xAI-Drop($G = (\mathcal{V}, \mathcal{E}, \mathbf{X}_{\mathcal{V}}, \mathbf{X}_{\mathcal{E}})$,$f$, $\theta$, $p$)
    for $e \in$ Epochs do
        $\mathcal{V}' \leftarrow$ HIGHEST-CONFIDENCE$(G, \mathcal{V}, f, \theta)$            ▷ Equation 3
        for $v \in \mathcal{V}'$ do
            $G_{exp}(v) \leftarrow$ SALIENCY-MAP$(G, v)$            ▷ Equation 1
            $F_{suf}(v) \leftarrow$ FIDELITY$(f, G, G_{exp}(v))$            ▷ Equation 2
        end for
        $\mathbf{p} \leftarrow$ DROPPING-PROBABILITIES$(F_{suf}, p)$            ▷ Equation 5
        $A' \rightarrow$ xAI-DropNode$(G, \mathbf{p})$            ▷ Equation 6
        $f \leftarrow$ TRAIN$(f, G, A')$
    end for
    $\mathcal{Y} \leftarrow$ EVALUATE$(f, G, A)$
end procedure
```

4.4 xAI-Drop for Link Prediction

While we focused on node classification in describing the approach for the sake of clarity, the xAI-Drop framework can also be applied to link prediction, where the goal is dropping edges that have highly confident but poorly explained predictions. In this setting, which we name xAI-DropEdge, the confidence score is computed at the edge level, rather than the node level, and the explainer will produce an explanation for each edge. The explanations of the edge predictions are assessed by aggregating the scores such that each edge has a corresponding normalized fidelity sufficiency. xAI-DropEdge produces a dropping probability score for each edge in the input graph and then the procedure drops the edges according to a Bernoulli distribution, i.e., $B_{ij} \sim Bernoulli(1 - p(e_{ij}))$. See Appendix I for further details. In our experimental analysis, we will show the effectiveness of xAI-Drop with its two main variants: xAI-DropNode for node classification tasks and xAI-DropEdge for link prediction tasks.

We presented xAI-Drop for transductive learning settings, which include unsupervised nodes in the message propagation process. The approach can also be applied to inductive settings, like graph classification, where it boils down to dropping entire training instances.

5 Experiments

Our experimental evaluation aims to address the following research questions:

Q1: Does xAI-Drop outperform alternative dropping strategies?

Q2: Does xAI-Drop outperform alternative xAI-driven strategies?

Q3: Does xAI-Drop improve explainability?

Q4: Does xAI-Drop work on beyond node classification?

We start by presenting the experimental setting and then discuss the results answering these questions.

5.1 Experimental setting

Datasets: We employed three widely used datasets for node classification: Cora [61], Citeseer [62], and PubMed [63]. Each dataset is composed of a single graph with thousands of labeled nodes. We utilize the publicly available train, validation, and test node splits [63]. We employed the same datasets for link prediction, where we used 10% of the edges for the validation set, and 20% of the edges for the test set. Detailed dataset statistics are presented in Appendix A.

Competitors: We compare our xAI-Drop approach with state-of-the-art dropping strategies. Random dropping methods include: DropMessage [27], which removes random features from messages; DropEdge [28], which randomly removes edges; DropNode [1, 29], which removes random nodes and all of their incident edges; and DropAgg [32], which discards messages to sampled nodes during aggregation. Non-random methods include Learn2Drop [34], using parameterized networks to prune irrelevant edges, and BBGDC [35], which adapts edge drop rates during training. As our method is XAI-based, we also compare it to XAI-based GNN regularization techniques: ExPass [54] adjusts message weights based on importance scores; MATE [53] uses meta-learning to optimize explainer performance; and ENGAGE [55] introduces a novel explainer for data augmentation to enhance GNN robustness.

GNN Architectures: By operating on the adjacency matrix, xAI-Drop is agnostic about the underlying GNN architecture. To demonstrate its versatility, we implemented it with three widely recognized GNN architectures: Graph Convolutional Networks (GCN) [5], Graph Attention Networks (GAT) [64] and Graph Isomorphism Network (GIN) [65]. For the random strategies, we retained the hyper-parameters that were optimized in the original work evaluating them [27]. The same holds for ExPass [54]. For ENGAGE [55] and MATE [53] we optimized hyper-parameters ourselves over the validation set, as the available configuration was not usable (ENGAGE was trained on different splits rather than the standard ones, while MATE was mostly evaluated on synthetic data). In the case of xAI-Drop, we maintained the same GNN hyper-parameters as those used in random strategies to isolate the role of the explainability component. Concerning xAI-Drop-specific hyper-parameters, we fixed $p = 0.5$ in all settings for simplicity, which implies dropping on average 50% of the edges/nodes (while the dropping probability p is optimized for each single dataset and GNN architecture in the case of the random strategies [27]), and set the confidence threshold θ to 0.9, 0.45, and 0.95 for, respectively, Cora, CiteSeer and PubMed, after a coarse-grained optimization on the validation set. For ablation studies about these hyperparameters refer to Appendix F.

Metrics: Multiclass accuracy is used for addressing research questions **Q1** and **Q2**. To answer research question **Q3**, we employed the standard saliency map explainer as defined in Eq.1, instead of its approximated variant used for training as defined in Section C, and computed the accuracy sufficiency, defined as:

$$A_{suf}(G) = \frac{1}{|\mathcal{V}_{\text{test}}|} \sum_{v \in \mathcal{V}_{\text{test}}} \mathbb{1}\left(\text{argmax}(f_v(G)) = \text{argmax}(f_v(G_{exp}(v)))\right) \tag{7}$$

where $\mathcal{V}_{\text{test}}$ is the set of test nodes in G, $G_{exp}(v)$ is the (thresholded version of the) saliency map defined in Eq. 1, and $\mathbb{1}(\cdot)$ is the indicator function. While in the case of the link prediction task (**Q4**) we used Area Under the Curve (AUC) for assessing the model predictions.

Availability: The code for reproducing experiments in this paper is available as a GitHub repository[1].

5.2 Experimental results

R1: xAI-Drop outperforms alternative dropping strategies. Table 1 shows the test accuracy of the different approaches we tested on node classification. Mean and standard deviation over 5 runs with different initialization seeds are reported. Comparing our xAI-Drop strategies with the blocks of random and learning-based dropping approaches, the advantage of XAI-driven dropping is evident[2]. xAI-DropNode consistently outperforms its random counterpart (DropNode which in turn improves

[1] Source code on Github

[2] Notice that the results in the table are different from those in the original publications of each respective method, as we had to rerun them all (retaining their optimal hyper-parameters or optimizing them on the

over the baseline method with no dropping) on all datasets and for all GNN architectures, despite the fact that the biased dropping probability (Eq. 5) is applied to the most confident nodes. More importantly, it outperforms all alternative dropping strategies, both random-based and learning-based[3]. Indeed, xAI-DropNode consistently scores as the best method in almost all scenarios. These results support our intuition that explainability can be an effective metric to guide the identification and removal of noisy information in GNN training.

R2: xAI-Drop outperforms alternative xAI-driven strategies. xAI-Drop is not the first method to propose the use of explainability to enhance training. The XAI-based block in Table 1 reports test accuracy of existing alternative xAI-based regularization methods. These methods underperform with respect to our xAI-Drop strategies, likely because of the increased complexity of their training process with respect to our dropping schemes. It is important to remind here that these xAI-based competitors have been developed with additional goals in mind with respect to regularization, namely improving explainability (MATE), alleviating oversmoothing (ExPass), or increasing robustness to adversarial attacks (ENGAGE). For large-scale datasets refer to Appendix J.

		GCN			**GAT**			**GIN**		
	Model	Cora	CiteS	PubM	Cora	CiteS	PubM	Cora	CiteS	PubM
	Baseline	79.0±0.3	67.1±0.5	76.9±1.2	78.4±1.2	68.1±0.7	77.3±0.7	78.2±1.0	67.5±1.0	76.7±0.8
Random	DropEdge	80.0±0.5	68.4±0.6	77.5±0.4	79.8±0.3	68.3±0.7	77.3±0.4	79.2±0.7	69.0±0.8	77.3±0.6
	DropMess	80.8±0.5	70.8±0.5	78.1±0.3	80.1±0.6	69.5±0.8	77.5±0.5	78.8±0.5	69.7±0.5	77.9±0.6
	DropNode	80.0±0.5	69.4±0.4	78.0±0.4	78.6±1.3	67.5±0.7	77.4±0.2	79.4±1.0	69.4±0.6	77.2±0.5
	DropAggr	80.0±0.6	68.8±0.6	78.3±0.3	80.8±0.8	67.3±1.2	77.7±0.2	78.6±0.5	68.2±1.6	76.9±0.4
Learn	BBGDC	74.2±0.3	67.4±0.3	74.1±0.6	-	-	-	-	-	-
	Learn2Drop	79.3±1.1	68.6±0.9	77.1±1.4	80.5±0.7	70.5±0.9	77.4±0.6	78.4±1.4	68.1±0.9	77.0±1.1
xAI	MATE	80.3±0.4	68.4±0.3	74.3±0.5	80.0±0.8	69.2±0.6	76.2±0.7	81.1±0.9	71.2±1.3	78.8±1.2
	ExPass	82.2±0.6	72.9±0.4	76.2±0.3	80.3±0.8	70.2±0.3	76.8±0.7	78.5±0.5	69.2±0.3	76.9±0.9
	ENGAGE	81.8±0.4	72.4±0.4	78.6±0.5	81.6±0.2	72.2±0.4	77.6±0.5	81.0±1.1	71.7±1.4	77.2±1.7
xAI-DropNode		**82.8**±0.5	**74.0**±0.4	**81.5**±0.7	**82.6**±0.5	**72.6**±0.4	**80.7**±0.5	**83.0**±0.4	**73.0**±0.6	**79.6**±0.7

Table 1: Node classification test set accuracy (in percentage). Mean and standard deviation over 5 runs with different initialization seeds. The best performing method is boldfaced.

R3: xAI-Drop improves explainability. Table 2 reports accuracy sufficiency (Eq. 7) over the entire set of data (training, validation and test), again with mean and standard deviation over the 5 runs. As expected, XAI-based approaches improve explainability with respect to the baseline. It is interesting to highlight that dropping strategies are also quite effective in improving explainability, confirming the beneficial effect of dropping on training robustness. Notably, xAI-DropNode again stands out as the best performing method in all settings. The improvement over the other XAI-based approaches highlights the effectiveness of using XAI as a dropping strategy in isolating the most relevant part of the input graphs. For additional xAI metrics analysis refer to Appendix G.

R4: xAI-Drop outperforms alternative strategies and improves explainability on link prediction tasks. Tables 3 and 4 report test set area under curve (AUC, a standard performance metric used in link prediction) and explainability (as measured by accuracy sufficiency) respectively, when xAI-Drop is applied to link prediction tasks. Results confirm the generality of our XAI-driven dropping strategy, as xAI-DropEdge outperforms all competitors (both dropping or XAI-based strategies) on all datasets, in terms of both prediction quality and explainability.

6 Conclusion

In this work we introduced a simple XAI-based regularization framework for GNN training that selects nodes (for node classification) or edges (for link prediction) with highly confident predictions

validation set, as explained in Section 5.1) in order to compute explainability metrics in addition to accuracy. Accuracy comparisons with the results reported in the original papers are reported in Appendix B, and confirm the advantage of the xAI-Drop strategy.

[3] We omit the results of BBGDC on GAT and GIN, because the method was specifically designed for GCN architectures and it failed to learn usable models when applied to GAT and GIN architectures.

		GCN			GAT			GIN		
	Model	Cora	CiteSeer	PubMed	Cora	CiteSeer	PubMed	Cora	CiteSeer	PubMed
	Baseline	92.8±0.3	88.3±1.2	92.9±0.8	90.6±1.0	89.6±0.8	92.5±0.6	91.3±0.9	88.8±0.7	92.4±0.8
Random	DropEdge	93.2±0.7	89.4±1.7	94.1±1.2	90.2±1.1	90.9±0.7	94.2±0.8	90.2±1.1	90.9±0.7	93.8±1.6
	DropMess	92.9±0.7	89.2±0.5	92.7±0.9	90.1±0.4	91.5±0.6	92.9±0.5	91.4±0.5	91.1±0.9	92.5±1.1
	DropNode	93.7±0.4	90.9±0.5	93.2±0.7	92.9±1.1	92.7±0.8	93.5±0.9	93.1±1.6	92.7±0.7	93.0±1.2
	DropAggr	93.8±1.1	88.9±1.2	93.0±0.8	90.6±0.9	89.9±1.3	92.4±0.9	91.2±1.8	89.4±1.4	92.7±0.9
Learn	BBGDG	89.2±0.3	82.3±0.4	84.2±0.4	-	-	-	-	-	-
	Learn2Drop	90.1±1.5	88.6±1.8	93.1±1.0	88.4±1.9	87.7±1.3	92.8±0.9	91.5±1.0	90.2±1.2	93.0±0.9
xAI	MATE	94.6±0.7	92.1±0.8	92.9±1.2	94.0±1.4	92.5±0.9	93.6±1.0	94.0±1.4	92.5±0.9	93.2±1.1
	ExPass	92.8±0.5	90.6±0.6	93.9±0.4	90.7±0.3	89.3±0.8	92.3±0.4	90.9±0.9	89.9±0.8	92.7±0.6
	ENGAGE	92.9±0.7	90.7±1.0	94.3±0.8	90.2±1.3	91.4±1.1	94.0±0.5	92.9±1.4	92.0±1.5	94.2±0.7
	XAI-DROPNODE	**97.2**±0.6	**95.2**±0.7	**97.3**±0.9	**95.9**±0.8	**94.8**±1.0	**96.7**±0.9	**96.4**±1.0	**95.5**±1.3	**97.0**±0.5

Table 2: Explainability of the different methods for node classification as measured by accuracy sufficiency. The best performing method is boldfaced.

		GCN			**GAT**			**GIN**		
	Model	Cora	CiteS	PubM	Cora	CiteS	PubM	Cora	CiteS	PubM
	Baseline	88.0±1.0	86.7±1.3	94.5±0.2	88.3±1.1	85.6±1.9	89.4±0.3	89.1±1.3	87.0±1.9	90.1±0.5
Random	DropEdge	94.1±0.7	90.5±1.3	94.6±0.3	92.3±0.3	94.6±0.7	93.8±0.7	92.2±1.0	91.9±1.1	93.0±0.9
	DropMess	92.4±0.9	90.8±0.5	92.1±0.8	92.1±0.8	90.4±0.7	91.5±0.7	91.7±0.9	91.2±0.8	91.5±1.6
	DropNode	95.0±0.8	91.4±0.4	94.2±0.8	93.2±1.3	90.7±0.8	91.4±0.5	94.1±1.2	92.8±1.5	93.9±1.3
	DropAggr	90.5±0.6	90.9±0.5	92.3±0.5	90.8±0.4	90.3±0.9	91.5±0.8	90.5±0.9	91.2±1.1	91.4±1.0
L.	Learn2Drop	89.6±0.6	89.5±0.9	90.3±0.5	90.1±0.7	91.0±1.2	92.1±0.9	90.2±0.6	92.1±1.6	91.6±1.1
xAI	FairDrop	90.1±0.7	88.4±1.4	94.8±0.2	87.8±1.0	87.1±1.1	87.1±0.6	90.1±1.2	89.3±0.7	89.9±0.9
	MATE	91.8±0.6	90.4±0.6	93.3±0.9	90.9±0.8	88.2±0.6	92.2±0.7	90.5±1.2	86.9±1.0	91.9±1.2
	ExPass	88.1±1.3	87.2±0.4	92.8±0.9	88.5±1.0	86.2±0.7	92.0±0.8	87.9±1.3	86.4±0.8	90.6±0.5
	XAI-DROPEDGE	**97.5**±0.7	**98.6**±0.9	**96.8**±1.1	**96.8**±1.0	**98.4**±0.8	**95.9**±0.9	**95.2**±1.2	**94.8**±0.5	**95.5**±1.3

Table 3: Test set AUC on link prediction, reported as the mean and standard deviation over 5 runs with different initialization seeds. The best performing method is boldfaced.

		GCN			GAT			GIN		
	Model	Cora	CiteSeer	PubMed	Cora	CiteSeer	PubMed	Cora	CiteSeer	PubMed
	Baseline	93.5±0.8	91.6±1.0	94.0±1.2	92.9±1.3	90.8±1.1	93.8±0.9	93.0±1.3	92.1±0.9	93.0±0.5
Random	DropEdge	92.1±1.1	90.9±1.5	94.7±1.2	92.3±1.1	90.9±1.2	94.5±1.2	92.4±1.2	90.9±0.7	93.9±1.5
	DropMess	93.1±1.1	91.9±0.9	93.6±0.8	91.0±0.5	91.5±0.6	93.9±0.5	92.7±1.0	92.3±0.9	92.9±1.0
	DropNode	93.0±0.9	91.9±1.1	94.9±1.2	93.1±1.4	91.7±1.7	92.8±1.0	92.8±1.3	93.1±1.0	92.9±1.2
	DropAggr	92.8±1.1	89.9±1.3	95.0±0.6	91.3±1.1	90.9±1.2	94.6±1.1	92.2±0.6	91.9±1.6	92.1±0.7
L.	Learn2Drop	91.9±1.6	90.6±1.2	93.2±1.0	92.4±1.3	87.7±1.3	92.1±1.7	92.1±1.3	91.0±1.4	91.0±0.6
xAI	FairDrop	93.3±0.9	91.3±1.0	94.0±0.9	92.2±0.5	91.9±0.5	91.4±1.1	92.5±0.7	92.2±1.2	92.5±0.9
	MATE	94.0±0.9	92.5±1.1	94.3±1.2	94.4±1.2	94.0±0.8	93.2±1.1	94.2±1.5	93.5±0.9	94.2±1.1
	ExPass	94.2±0.9	92.2±0.8	92.6±1.2	94.0±1.1	92.9±1.0	93.9±0.4	91.2±1.2	90.5±1.1	92.1±1.4
	XAI-DROPEDGE	**96.4**±1.0	**93.9**±1.2	**95.8**±0.8	**95.3**±1.2	**94.4**±1.3	**95.2**±0.9	**95.1**±1.3	**93.8**±1.2	**94.8**±1.1

Table 4: Explainability of the different methods for link prediction as measured by accuracy sufficiency. The best performing method is boldfaced.

but poor explanations as candidates for dropping. Our experimental evaluation clearly showed that the proposed framework outperforms alternative dropping strategies as well as other XAI-based regularization techniques in terms of both accuracy and explainability. These promising results highlight the role of explainability-based regularization in improving training dynamics. Future work include the exploration of the connection between explainability-based regularization and out-of-domain generalization, the application of similar XAI-based solutions to design augmentation strategies, and the study of explainability-based dropping for other classes of deep learning architectures.

Acknowledgments

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Health and Digital Executive Agency (HaDEA). Neither the European Union nor the granting authority can be held responsible for them. Grant Agreement no. 101120763 - TANGO. We acknowledge the support of the MUR PNRR project FAIR - Future AI Research (PE00000013) funded by the NextGenerationEU.

References

[1] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019. 1, 7

[2] Dejun Jiang, Zhenxing Wu, Chang-Yu Hsieh, Guangyong Chen, Ben Liao, Zhe Wang, Chao Shen, Dongsheng Cao, Jian Wu, and Tingjun Hou. Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models. *Journal of Cheminformatics*, 13(1):12, Feb 2021. ISSN 1758-2946. doi: 10.1186/s13321-020-00479-8. 1

[3] Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, 207:117921, 2022. 1

[4] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2, 2005. doi: 10.1109/IJCNN.2005.1555942. 1

[5] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 7

[6] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Message passing neural networks. *Machine learning meets quantum physics*, pages 199–214, 2020.

[7] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021. 1

[8] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017. 1

[9] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

[10] Jian Tang and Renjie Liao. Graph neural networks for node classification. *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 41–61, 2022.

[11] Filippo Maria Bianchi and Veronica Lachi. The expressive power of pooling in graph neural networks. *Advances in neural information processing systems*, 36, 2024. 1

[12] Weimin Li, Lin Ni, Jianjia Wang, and Can Wang. Collaborative representation learning for nodes and relations via heterogeneous graph neural network. *Knowledge-Based Systems*, 255: 109673, 2022. 1

[13] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

[14] Veronica Lachi, Francesco Ferrini, Antonio Longa, Bruno Lepri, and Andrea Passerini. A simple and expressive graph neural network based method for structural link representation. In *ICML 2024 Workshop on Geometry-grounded Representation Learning and Generative Modeling*, 2024. URL `https://openreview.net/forum?id=EGGSCLyVrz`. 1

[15] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR, 2018. 1

[16] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022.

[17] Manfred Jaeger, Antonio Longa, Steve Azzolin, Oliver Schulte, and Andrea Passerini. A simple latent variable model for graph learning and inference. In *Learning on Graphs Conference*, pages 26–1. PMLR, 2024. 1

[18] Lilapati Waikhom and Ripon Patgiri. Graph neural networks: Methods, applications, and opportunities. *arXiv preprint arXiv:2108.10733*, 2021. 1

[19] Wenlong Liao, Birgitte Bak-Jensen, Jayakrishnan Radhakrishna Pillai, Yuelong Wang, and Yusen Wang. A review of graph neural networks and their applications in power systems. *Journal of Modern Power Systems and Clean Energy*, 10(2):345–360, 2021.

[20] Fan Liang, Cheng Qian, Wei Yu, David Griffith, and Nada Golmie. Survey of graph neural networks and applications. *Wireless Communications and Mobile Computing*, 2022(1):9261537, 2022. 1

[21] Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, pages 3419–3430. PMLR, 2020. 1

[22] Antonio Longa, Steve Azzolin, Gabriele Santin, Giulia Cencetti, Pietro Liò, Bruno Lepri, and Andrea Passerini. Explaining the explainers in graph neural networks: a comparative study. *ACM Computing Surveys*. 1

[23] Han Xuanyuan, Tianxiang Zhao, and Dongsheng Luo. Shedding light on random dropping and oversmoothing. In *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*. 1, 2

[24] T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023. 1

[25] Jhony H Giraldo, Konstantinos Skianis, Thierry Bouwmans, and Fragkiskos D Malliaros. On the trade-off between over-smoothing and over-squashing in deep graph neural networks. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pages 566–576, 2023. 1

[26] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 1

[27] Taoran Fang, Zhiqing Xiao, Chunping Wang, Jiarong Xu, Xuan Yang, and Yang Yang. Dropmessage: Unifying random dropping for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4267–4275, 2023. 1, 2, 5, 7, 14

[28] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019. 1, 2, 7

[29] Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. Dropgnn: Random dropouts increase the expressiveness of graph neural networks. *Advances in Neural Information Processing Systems*, 34:21997–22009, 2021. 1, 2, 7

[30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 2

[31] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33:22092–22103, 2020. 2, 14

[32] Bo Jiang, Yong Chen, Beibei Wang, Haiyun Xu, and Bin Luo. Dropagg: Robust graph neural networks via drop aggregation. *Neural Networks*, 163:65–74, 2023. 2, 7, 14

[33] Indro Spinelli, Simone Scardapane, Amir Hussain, and Aurelio Uncini. Fairdrop: Biased edge dropout for enhancing fairness in graph representation learning. *IEEE Transactions on Artificial Intelligence*, 3(3):344–354, 2021. 2

[34] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 779–787, 2021. 2, 7, 14

[35] Arman Hasanzadeh, Ehsan Hajiramezanali, Shahin Boluki, Mingyuan Zhou, Nick Duffield, Krishna Narayanan, and Xiaoning Qian. Bayesian graph neural networks with adaptive connection sampling. In *International conference on machine learning*, pages 4094–4104. PMLR, 2020. 2, 7, 14

[36] Xiaoqi Wang and Han Wei Shen. GNNInterpreter: A probabilistic generative model-level explanation for graph neural networks. In *The Eleventh International Conference on Learning Representations*, 2023. 3

[37] Steve Azzolin, Antonio Longa, Pietro Barbiero, Pietro Lio, and Andrea Passerini. Global explainability of GNNs via logic combination of learned concepts. In *The Eleventh International Conference on Learning Representations*, 2023.

[38] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 430–438, 2020. 3

[39] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019. 3, 15

[40] Minh Vu and My T Thai. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. *Advances in neural information processing systems*, 33:12225–12235, 2020. 3, 15

[41] Siqi Miao, Mia Liu, and Pan Li. Interpretable and generalizable graph learning via stochastic attention mechanism. In *International Conference on Machine Learning*, pages 15524–15543. PMLR, 2022.

[42] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *International conference on machine learning*, pages 12241–12252. PMLR, 2021. 3

[43] Jaykumar Kakkad, Jaspal Jannu, Kartik Sharma, Charu Aggarwal, and Sourav Medya. A survey on explainability of graph neural networks. *arXiv preprint arXiv:2306.01958*, 2023. 3

[44] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. Explainability methods for graph convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10772–10781, 2019. 3

[45] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 2022. 3

[46] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017. 3, 15

[47] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 33:19620–19631, 2020. 3, 15

[48] Wenqian Li, Yinchuan Li, Zhigang Li, HAO Jianye, and Yan Pang. Dag matters! gflownets enhanced explainer for graph neural networks. In *The Eleventh International Conference on Learning Representations*, 2022. 3

[49] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 3, 5, 15

[50] Leander Weber, Sebastian Lapuschkin, Alexander Binder, and Wojciech Samek. Beyond explaining: Opportunities and challenges of xai-based model improvement. *Information Fusion*, 92:154–176, 2023. 3

[51] Patrick Schramowski, Wolfgang Stammer, Stefano Teso, Anna Brugger, Franziska Herbert, Xiaoting Shao, Hans-Georg Luigs, Anne-Katrin Mahlein, and Kristian Kersting. Making deep neural networks right for the right scientific reasons by interacting with their explanations. *Nature Machine Intelligence*, 2(8):476–486, 2020. 3

[52] Seul-Ki Yeom, Philipp Seegerer, Sebastian Lapuschkin, Alexander Binder, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognition*, 115:107899, 2021. 3

[53] Indro Spinelli, Simone Scardapane, and Aurelio Uncini. A meta-learning approach for training explainable graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 3, 7

[54] Valentina Giunchiglia, Chirag Varun Shukla, Guadalupe Gonzalez, and Chirag Agarwal. Towards training gnns using explanation directed message passing. In *Learning on Graphs Conference*, pages 28–1. PMLR, 2022. 3, 7, 14

[55] Yucheng Shi, Kaixiong Zhou, and Ninghao Liu. Engage: Explanation guided data augmentation for graph representation learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 104–121. Springer, 2023. 3, 7, 14

[56] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605. 3

[57] Alessio Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009.

[58] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. 3

[59] Mandeep Rathee, Thorben Funke, Avishek Anand, and Megha Khosla. Bagel: A benchmark for assessing graph neural network explanations. *arXiv preprint arXiv:2206.13983*, 2022. 4

[60] George EP Box and David R Cox. An analysis of transformations. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 26(2):211–243, 1964. 5

[61] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000. 7

[62] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pages 89–98, 1998. 7

[63] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016. 7

[64] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. 7

[65] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018. 7

[66] Federico Baldassarre and Hossein Azizpour. Explainability techniques for graph convolutional networks. In *ICML 2019 Workshop" Learning and Reasoning with Graph-Structured Representations"*, 2019. 15

[67] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413, 2020. 21

A Dataset Statistics

Table 5 outlines the key characteristics of the dataset, including the number of classes, nodes, and unidirectional edges in the networks.

B Additional results

The computation of explainability for different architectures, methods, and datasets requires access to the weights of the model. To achieve this goal, we have retrained the models with a specific regularization method on each, in case the authors do not release the weights of the models. For

	# classes	# nodes	# edges
CiteSeer	6	3,327	9,104
Cora	7	2,708	10,556
PubMed	3	19,717	88,648
OGBN-Arxiv	40	169,343	2,332,486

Table 5: Dataset statistics.

these motivations, we have reported in Table 1 the accuracy that we get by retraining from scratch the model of interest with the hyperparameters written in the paper or source and in case of missing hyperparameters information, by applying grid search during hyperparameter optimization. Table 6 reports for each method the results from the corresponding original paper as reported under the source column, when available. Results confirm the advantage of xAI-DropNode over all alternatives.

			GCN			GAT		
	Model	Source	Cora	CiteSeer	PubMed	Cora	CiteSeer	PubMed
	Baseline	[27]	80.7±0.4	70.8±0.5	75.9±0.7	81.4±0.6	70.1±0.6	77.2±0.5
Random	DropEdge	[27]	81.7±0.9	71.4±0.7	79.1±0.8	81.8±0.8	71.1±1.0	77.7±0.8
	DropMess	[27]	83.3±0.6	71.8±0.6	79.2±0.5	82.2±0.7	71.5±0.7	78.1±0.5
	DropNode	[31]	84.5±0.4	74.2±0.3	80.0±0.3	84.3±0.4	73.2±0.4	79.2±0.6
	DropAggr	[32]	83.1	72.8	-	83.6±0.7	72.9±0.5	-
Learning	Learn2Drop	[34]	82.8±0.3	72.7±0.2	79.8±0.2	84.4±0.2	73.7±0.3	79.3±0.1
	BBGDC	[35]	81.8±1.0	71.5±0.6	-	-	-	-
xAI-Based	ExPass	[54]	-	-	76.2±0.3	-	-	-
	ENGAGE	[55]	84.1±0.2	72.4±0.5	-	83.8±0.5	72.4±0.5	-
xAI-DropNode			**84.7**±0.7	**74.6**±0.9	**82.0**±0.9	**84.5**±0.6	**73.6**±0.6	**81.2**±0.8

Table 6: Test set accuracy on GCN (in percentage). The number of runs for computing standard deviations, when available, can be found in the corresponding paper reported under the "Source" column.

C Approximated saliency map

Saliency map is an explainer that produces an importance score for each node feature given a single model prediction. In general, saliency map is applied to a single node v to get the local explanation in the k-hop neighborhood of the node of interest v. For computational efficiency, rather than applying one forward step for each candidate node, we compute a single forward step for the entire set of candidate nodes. The node feature importance will then be the gradient of a single forward step on the entire set of candidate nodes, rather than the gradient of a forward step on just one candidate node. Once we have obtained the node feature importance, the aggregation of them has been considered as node importance score. These node scores are then used to generate the explanation subgraph where the top-$K\%$ most important neighboring nodes are retained. This means that a batch of candidate nodes is associated with a single explanation subgraph rather than associating one different explanation subgraph with each single candidate node in the batch.

This approximated variant of the Saliency Map method dramatically reduces the time complexity of the approach. Let's consider the worst case scenario in the transductive node classification task: all the nodes' predictions have a confidence score (Equation 3) higher than the threshold confidence $C(v) > \theta, \forall v \in \mathcal{V}$, so that all the nodes are candidate noisy nodes $v \in V', \forall v \in \mathcal{G}$. Then n different explanations have to be computed, with $n = |\mathcal{V}|$ being the number of nodes in the graph. In this setting, the standard Saliency Map procedure would require n different forward steps, one for each node: $f_v(G); \forall v \in \mathcal{V}$ to compute n different local explanations. On the contrary, Approximated Saliency Map applies a single forward step on the entire input graph $f(G)$, and the prediction is backpropagated to leverage the gradient of each feature in the input graph as the explanation importance score. Finally, these feature-level importance scores are averaged at the node level to get the importance of each node. Once we get these importance scores, for each node, the connections with the $\theta\%$-most relevant neighboring nodes are retained, while the incident edges from the not-relevant nodes are dropped to prevent the propagation of direct incoming messages during the forward step. This simple approximation reduces also the computational costs required to create n different explanation subgraphs. Once the explanation graph, which retains only the most important connections according to the Approximated Saliency Map explainer, has been isolated, a single forward step is used to compute the prediction for all the nodes in the graph having the original topology $f(G)$, and another forward step computes the prediction for all the nodes in the graph having the explanation topology $f(G_{exp})_{V'}$. Finally, the Kullback-Lieber (KL) divergence sufficiency score $KL_{suf}(v)$(Equation 8) is computed, for each node in the set of candidate noisy nodes $v \in V'$, between the two probability distributions obtained by feeding the model with the original graph and the explanation subgraph respectively.

Wrapping up, the overhead introduced by XAI-DROP in the case of node classification consists of:

- performing (only) one forward step to compute the explanations, regardless of the number of nodes in the set of candidate noisy nodes;
- performing (only) two forward steps to compute the predictions for the original graph and the explanation subgraph;
- computing the KL-divergence.

D Explainer comparison

In this section, we explore the flexibility of XAI-DROP to the usage of alternative explainers with respect to the Approximated Saliency Map used in the main paper. In Figure 3, we present the performance and training time of our method using different explainers in terms of accuracy (left axis) and training time (right axis). For this ablation study, we have explored different types of explainers:

- Gradient-based (Integrated Gradients [46], Saliency-Map [49])
- Perturbation-based (GNNExplainer [39], PGExplainer [47])
- Decomposition-based (CAM [66])
- Surrogate-based (PGM-Explainer [40])

Quite remarkably, XAI-DROP manages to improve performance with respect to the random node-dropping strategy *regardless of the explainer being used*. On the other hand, this plot highlights the substantial computational advantage of Approximated Saliency Map (and to a lesser extent Saliency Map and CAM) over more complex alternatives, without incurring in a reduction of generalization capabilities.

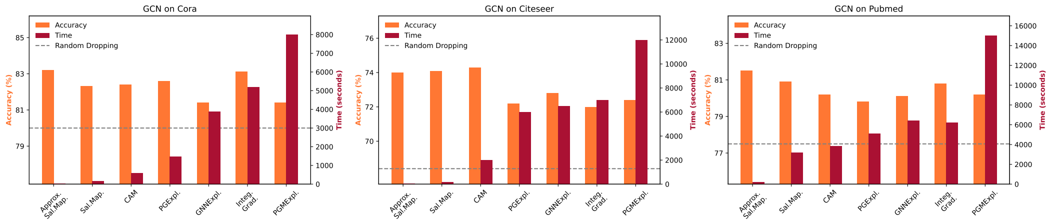


Figure 3: Test accuracy (left axis) and training time (right axis) when using different explainers for XAI-DROPNODE applied on Cora (left), Citeseer (center), and Pubmed (right) for node classification with GCN architecture. The dotted line represents the accuracy achieved when using the baseline DropNode random strategy.

E Computational complexity

Training time is a crucial challenge in designing topological regularizers for GNNs. In Figure 4, we report the training time required for each method by iterating training for the same number of epochs when using Cora as dataset and GCN as architecture. While baseline and random methods (i.e. DropEdge, DropMessage, DropNode, DropAgg) are extremely fast, and almost have the same training time, the additional operations computed in other dropping strategies, introduce additional overhead. In Figure 4 it is clear that there are strategies whose computational overhead is crucial in analyzing their performance. ExPass introduces a relevant overhead due to the usage of GNNExplainer for producing explanations. GNNExplainer is a computationally demanding explainer because it requires a separate iterative learning procedure and applying it to the input graph dramatically slows down the training procedure. Also MATE, Learn2Drop, and ENGAGE require a double training procedure. MATE introduces a Meta-learning approach and is a model that requires more parameters than the traditional dropping procedure to stabilize its training dynamics. Learn2Drop, apart from training the model for node classification, needs to train the model to learn a denoised topology and this objective requires a lot of parameters. ENGAGE incorporates an unsupervised step for learning robust embeddings by optimizing a contrastive objective and a supervised step for doing prediction on top of these embeddings. The unsupervised step requires deep, highly non-linear functions and many parameters to learn to be effective. Our approach XAI-DROPNODE, despite the introduced overhead, thanks to the approximated variant of Saliency Map, defined in Appendix C, and the Node selection based on confidence, still guarantees a manageable training time.

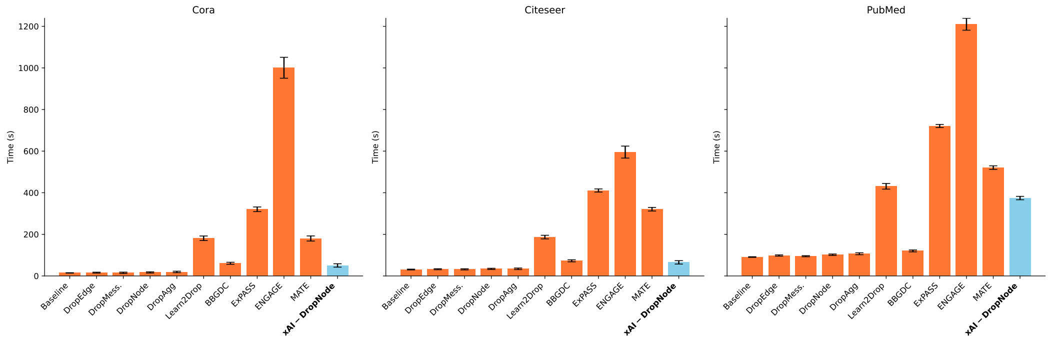


Figure 4: The histogram of the time in seconds required for training GCN on Cora, Citeseer, and PubMed with each regularization method used for node classification.

In the analysis of the computational complexity, the number of parameters required by each method also plays a crucial role. As in Figure 4 the analysis is conducted on the Cora dataset trained with GCN. In Figure 5, the number of parameters is plotted with a log-scale histogram rather than a linear-scale histogram due to the huge amount of parameters required by ENGAGE for the unsupervised training procedure, before the finetuning stage. Other methods which requires much more parameters than the other strategies are Learn2Drop and MATE, because of the need for parameters to, respectively, optimize the robustness of the topology and the Meta-Learning inner stage. On the contrary, the released version of BBGDC simply uses a wider hidden layer. Finally, the baseline, random drop strategies (DropEdge, DropMess, DropNode, DropAgg), and xAI-guided methods (ExPass and xAI-DropNode) use the same hidden size and the same network depth. The small number of additional parameters introduced by ExPass is due to the choice of a parametric explainer. Figure 5 confirms, as Figure 4, that XAI-DROPNODE does not introduce a meaningful computational overhead with respect to other random dropping strategies; and at the meantime, XAI-DROPNODE exhibits an evident computational advantage with respect to learnable and alternative xAI-based dropping strategies.

F Hyperparameter sensitivity

F.1 Confidence

One of the most important hyperparameters in our method is the confident threshold $\theta \in [0, 1]$. This hyperparameter is necessary to decide whether a node is a candidate noisy node or not. To fully comprehend its rule, we can start by analyzing the two extreme cases:

- if $\theta = 0$: all nodes in the graph are candidate noisy nodes, regardless of their confidence. The consequence is that the dropping probability of each node will be biased exclusively based on its explanation quality.

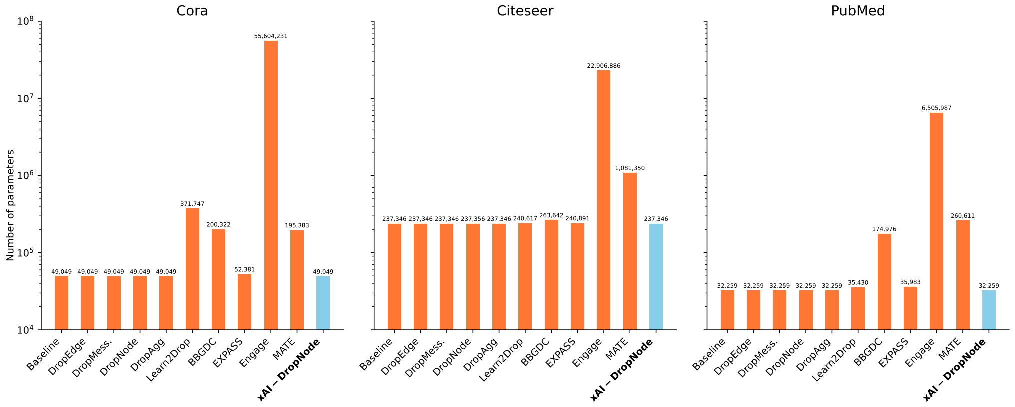


Figure 5: The log-scale histogram of the parameters used for training GCN for node classification task with Cora, Citeseer, and Pubmed datasets.

- if $\theta = 1$: no node has a confidence larger than the threshold, and the strategy boils down to random dropping.

In Figure 6, we have reported how the tuning of the confidence hyperparameter θ affects test accuracy on Cora, Citeseer, and Pubmed trained with GCN for Node Classification task. Results show that both completely explainability-guided ($\theta = 0$) and completely random ($\theta = 1$) strategies are suboptimal, and that a threshold around 0.8 is reasonable for all datasets.

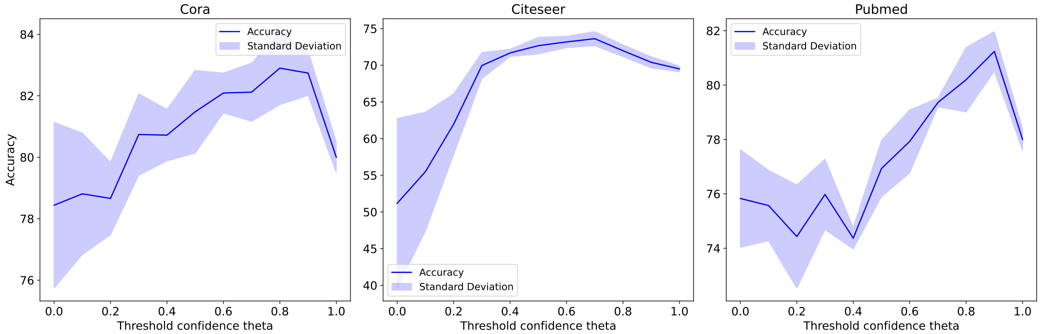


Figure 6: Test Accuracy and standard deviations on some node classification datasets (Cora, Citeseer, and Pubmed) trained with GCN varying confidence threshold θ.

F.2 Dropping probability

The dropping probability p is a crucial hyperparameter for properly applying xAI-Drop. As with any dropping strategies, the tuning of this hyperparameter strongly depends on the input graph, and is also related to the design of the GNN. In Figure 7 we show how the test accuracy varies when changing the dropping probability. From this empirical evidence, we note that larger datasets such as Pubmed have better results for larger values of p, i.e., with a more aggressive dropping. Furthermore, it is interesting to notice that removing the 90% of the input nodes leads to results similar to the baseline, which does not apply any dropping.

G Evolution of node confidence and explainability, ablation studies

The xAI-Drop method relies on a crucial intuition: the combination of confidence and explanation quality can be used as a proxy for pinpointing harmful nodes in a graph, the removal of which stabilizes training. To better show how training evolves using xAI-Drop, Figure 8 reports a series of confusion matrices (at different stages of training) reporting nodes with high confidence and

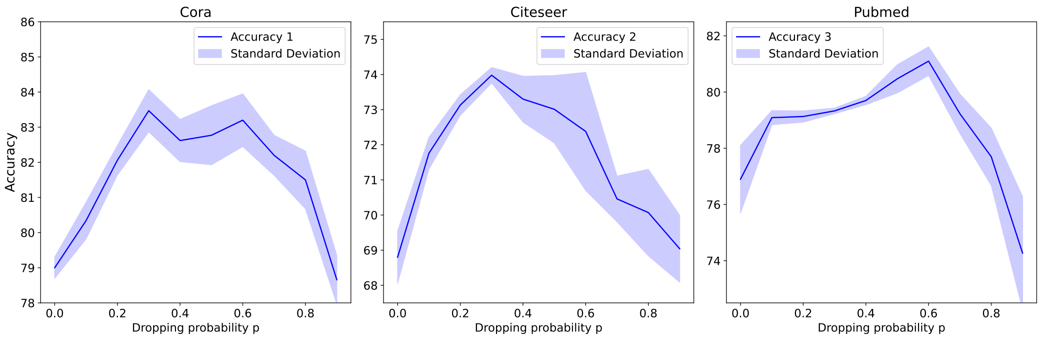


Figure 7: Test accuracy on Node classification with GCN on multiple datasets (Cora, Citeseer, Pubmed) varying dropping probability θ.

good explanations (HC-GE), high confidence and poor explanations (HC-PE), low confidence and good explanations (LC-GE) and low confidence and poor explanations (LC-PE). Results show how most nodes have initially low confidence. Thanks to training, the confidence of nodes increases, but high-confidence nodes are equally distributed among good-explanation and poor-explanation ones. While training progresses, increasingly more nodes have high confidence and good explanations, as expected.

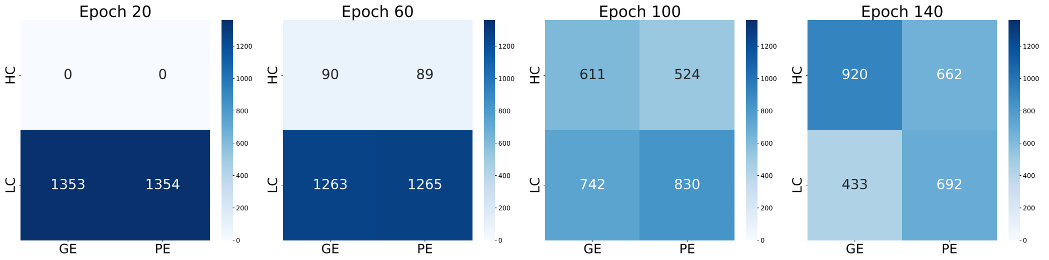


Figure 8: Confusion matrices for an increasing number of training epochs, showing nodes with high confidence and good explanations (HC-GE), high confidence and poor explanations (HC-PE), low confidence and good explanations (LC-GE) and low confidence and poor explanations (LC-PE).

Figure 9 shows the histogram of the dropping probability of a node averaged over the set of training epochs. Clearly, the histogram converges to a Delta Dirac (on $p = 0.5$) for the random strategy (DropNode, right plot), which corresponds to a uniform dropping probability for all nodes. On the contrary, xAI-Drop (left histogram) significantly biases the behavior of nodes over training, so that part of the nodes is consistently identified as harmful (often dropped during training) or beneficial (mostly retained during training), stabilizing training.

Finally, Table 7 presents the results of an ablation study where we altered the dropping strategy. Alternatives explored include confidence-only (high or low), explanation-only (good or poor) and their combinations. Results clearly indicate the advantage of the xAI-Drop strategy focusing on high-confidence, poorly explained cases. It is important to highlight that dropping low confidence nodes is especially detrimental, most likely because it destabilizes training removing instances that still need to be properly learned.

H Post-Hoc Explanation evaluation across explanations metrics

In this section we evaluate the quality of the explanations obtained using xAI-Drop and its competitors (with a GCN) in terms of alternative explanation quality metrics, namely KL-Necessity and KL-Sufficiency.

KL-Sufficiency follows the definition of sufficiency described in Equation 2, where the distance criterion is the Kullback-Lieber divergence as reported in 8 between the two probability

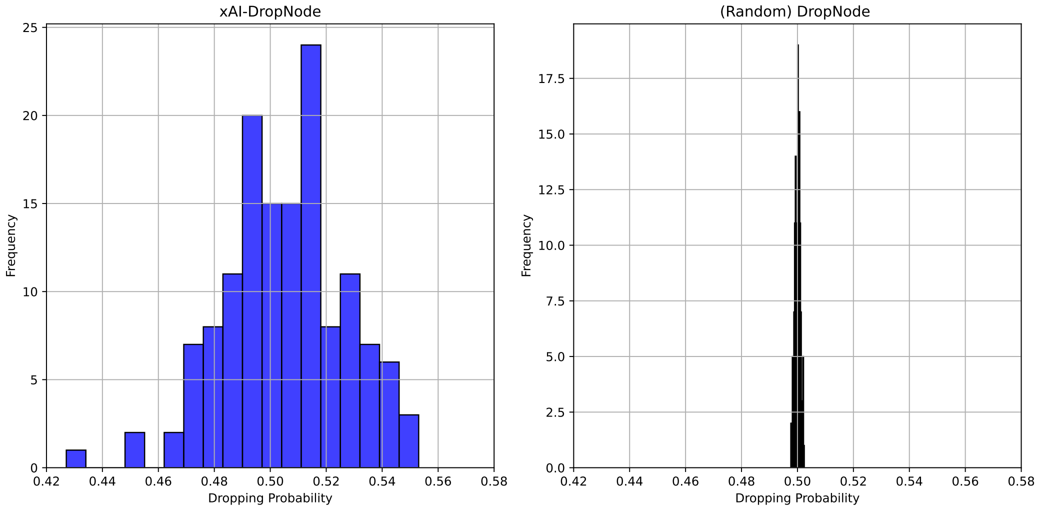


Figure 9: Histograms showing the average dropping probability of each node in a graph computed over all the training epochs for XAI-DROPNODE (left) and DROPNODE (right) respectively.

Noisy node selection Criterion	Cora
Random	80.0±0.5
HighConfidence	80.6±0.4
LowConfidence	74.5±1.5
LowConfidence+PoorXAI	78.4±0.9
HighConfidence+GoodXAI	79.8±0.5
LowConfidence+GoodXAI	77.4±1.6
LowConfidence+Random	76.9±1.4
HighConfidence+Random	81.2±0.9
PoorXAI	80.4±0.9
GoodXAI	79.5±0.5
xAI-Drop	**82.8±0.5**

Table 7: Test set accuracy (in percentage) on Cora dataset for node classification trained with GCN by comparing different metrics for identifying noisy nodes. The standard deviation is computed over three runs.

distributions($(f_v(G)_i)$, $f_v(G_{exp}(v))$) produced by feeding the GNN, respectively, the original graph G and the explanation subgraph $G_{exp}(v)$ for the node $v \in G$.

$$KL_{suf}(v) = \sum_{i=1}^{c} (f_v(G))_i \; log \left(\frac{(f_v(G))_i}{f_v(G_{exp}(v))_i} \right) \tag{8}$$

KL-Necessity, on the other hand, removes the explanation from the neighborhood of the node of interest, to define whether the explanation is necessary for producing the same prediction. It is computed as the KL-distance between the probability distributions produced by feeding the entire graph $f_v(G)_i$ and the non-relevant subgraph $f_v(G \setminus G_{exp}(v))$:

$$KL_{nec}(v) = \sum_{i=1}^{c} (f_v(G))_i \; log \left(\frac{(f_v(G))_i}{f_v(G \setminus G_{exp}(v))_i} \right) \tag{9}$$

Figure 10 reports results of the different methods in terms of KL-Sufficiency *and* KL-Necessity, as only a reasonable trade-off between the two is an indicator of a good quality explanation. Results clearly indicate that XAI-DROP scores the best trade-off between the two metrics, thus achieving the best explanations for all datasets.

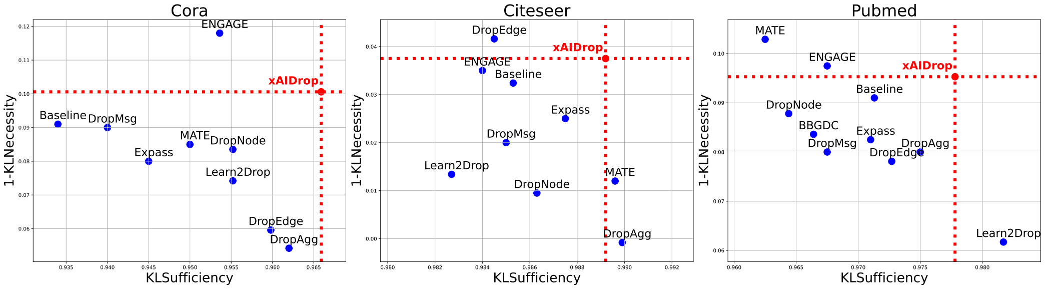


Figure 10: Scatter plot representing the quality of the explanations produced through Saliency Map on GCN across multiple datasets (Cora, Citeseer, Pubmed), measured in terms of KL-Sufficiency (x-axis) and 1- KL-Necessity (y-axis).

I xAI-DropEdge

The overall algorithm for xAI-DropEdge is outlined in Algorithm 2. As in Algorithm 1, the algorithm takes as input a graph $\mathcal{G}$, the GNN architecture to be trained f, and the hyperparameters θ and p. In each epoch, the algorithm from the entire set of edges $\mathcal{E}$ selects the edges $\mathcal{E}'$ with a prediction confidence score higher than the confidence threshold θ. Explanations qualities for all edges in $\mathcal{E}'$ are assessed via the Fidelity sufficiency score F_{suf} (Equation 2). These explanation scores are mapped in probabilities through the Yao-Johnson mapping as described in Equation 4, as happens for nodes in the xAI-DropNode variant. Once the biased dropping probabilities $p(v)$ have been computed, xAI-DropEdge removes edges $e \in \mathcal{E}$ from the edge set $\mathcal{E}$ based on a edge dropping mask $B^{\mathcal{E}} \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{V}|}$ defined as follows:

$$B^{\mathcal{E}}_{i,j} \sim Bernoulli(1 - p((i,j))) \tag{10}$$

where $p((i,j)) = 1$ if $(i,j) \notin \mathcal{E}$. The edge-dropping operation can be compactly represented in terms of Hadamard product between the binary edge dropping mask $B^{\mathcal{E}}$ and the adjacency matrix of the input graph A:

$$A' = A \otimes B^{\mathcal{E}} \tag{11}$$

Algorithm 2 xAI-Drop algorithm for link prediction. $G = (\mathcal{V}, \mathcal{E}, \mathbf{X}_{\mathcal{V}}, \mathbf{X}_{\mathcal{E}})$ is a graph, f is the GNN, θ, p are hyper-parameters

```
procedure xAI-Drop($G = (\mathcal{V}, \mathcal{E}, \mathbf{X}_{\mathcal{V}}, \mathbf{X}_{\mathcal{E}})$, $f$, $\theta$, $p$)
    for $e \in$ Epochs do
        $\mathcal{E}' \leftarrow$ HIGHEST-CONFIDENCE$(G, \mathcal{E}, f, \theta)$          ▷ Equation 3
        for $e \in \mathcal{E}'$ do
            $G_{exp}(e) \leftarrow$ SALIENCY-MAP$(G, e)$          ▷ Equation 1
            $F_{suf}(e) \leftarrow$ FIDELITY$(f, G, G_{exp}(e))$          ▷ Equation 2
        end for
        $\mathbf{p} \leftarrow$ DROPPING-PROBABILITIES$(F_{suf}, p)$          ▷ Equation 10
        $A' \rightarrow$ xAI-DropEdge$(G, \mathbf{p})$          ▷ Equation 11
        $f \leftarrow$ TRAIN$(f, G, A')$
    end for
    $\mathcal{Y} \leftarrow$ EVALUATE$(f, G, A)$
end procedure
```

J Scaling xAI-Drop

Dropping strategies, apart from the advantages analysed in Section 5, are well-known in the literature for their capabilities to enhance learning with deeper architectures. In this section we report results on one large-scale graph (i.e. OGBN-Arxiv [67]) trained on a deeper GNN (i.e. 4 layers), to verify whether xAI-Drop scales also to large input graphs. Test accuracy on this dataset has been tested on GCN for all the competitors (apart from Learn2Drop for computational reasons). Results are shown in Table 8, and confirm the advantage of xAI-Drop over its competitors[4]

	Model	OGBN-Arxiv
-	Baseline	67.1±0.8
Random	DropEdge	70.5±1.0
	DropMess	71.0±0.6
	DropNode	70.7±0.9
	DropAggr	69.8±1.2
L.	BBGDC	68.0±1.1
xAI	MATE	68.8±1.6
	ExPass	70.9±0.8
	ENGAGE	71.5±0.6
xAI-DropNode		**71.7**±1.2

Table 8: Test set accuracy (in percentage) computed on OGBN-Arxiv trained with GCN across different dropping strategies. The standard deviation is computed on three runs.

K Extracted explanations

In Figure 11, we present explanations generated using saliency maps on a standard GCN compared to a GCN with our proposed method. The explanations produced after applying our dropping strategies are notably sparser, resulting in clearer visualizations that enable more reliable insights.

L Probability Mapping

The mapping of the explanation assessments (i.e. Fidelity F_{suf}) is a crucial point in xAI-Drop. It enables to convert raw explanation metrics that are usually compressed in a range far from the default dropping probabilities into a range of probabilities p defined in the range $[0, 1]$. We have tested multiple mapping approaches, but two distributions better fill our needs: empirical cumulative distribution and Gaussian distribution (described in Equation 4). In Table 9 we report an empirical comparison across datasets, architectures, and distributions. For the sake of completeness, we report also Uniform distribution which is the one used by random dropping strategies.

	GCN			GAT			GIN		
Distribution	Cora	CiteSeer	PubMed	Cora	CiteSeer	PubMed	Cora	CiteSeer	PubMed
Uniform	79.0±0.3	67.1±0.5	76.9±1.2	78.4±1.2	68.1±0.7	77.3±0.7	78.2±1.0	67.5±1.0	76.7±0.8
Cumulative	82.6±0.4	72.6±0.6	80.9±0.6	82.5±0.7	71.7±0.8	80.4±0.7	82.0±0.7	72.1±0.6	79.6±0.8
Gaussian	**82.8**±0.5	**74.0**±0.4	**81.5**±0.7	**82.6**±0.5	**72.6**±0.4	**80.7**±0.5	**83.0**±0.4	**73.0**±0.6	**79.6**±0.7

Table 9: Test accuracy across multiple datasets and architectures tested for node classification task when using different methods for mapping explanation metrics into probability distributions. Note that Uniform refers to random dropping strategies.

[4] It is important to remind here that our goal is not that of achieving state-of-the-art results using the most recent, complex architectures, for which running competitors would be prohibitively expensive, but showing consistent advantages over alternative solutions when evaluated under the same experimental conditions.

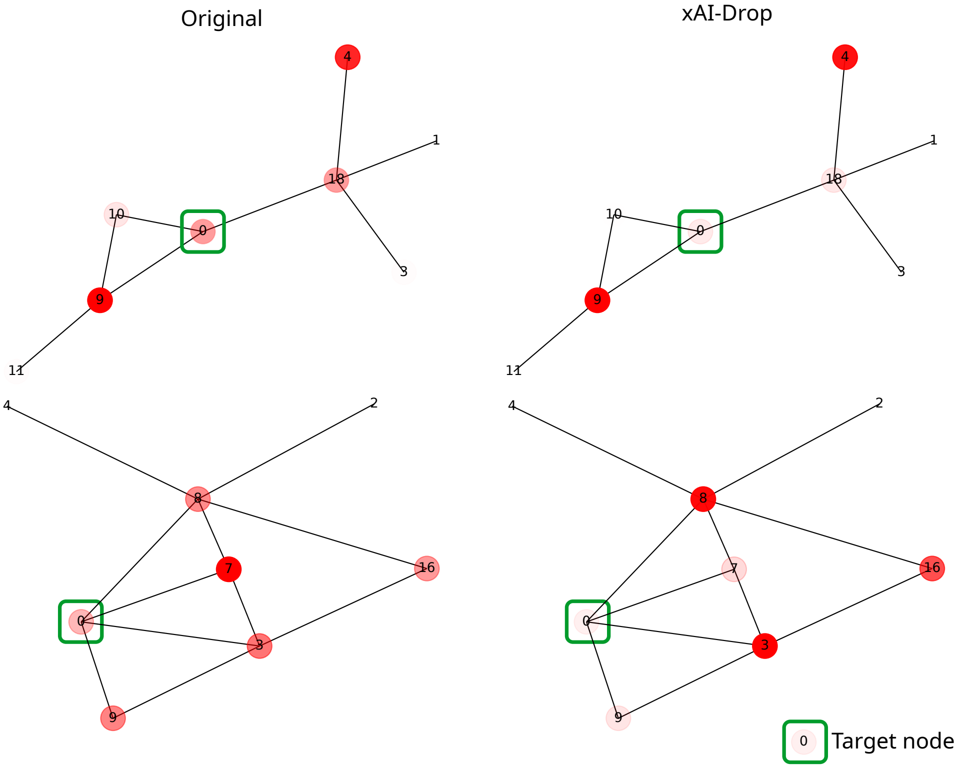


Figure 11: Examples of explanations generated using a saliency map on a GCN trained on the Cora network.