The Intelligible and Effective Graph Neural Additive Networks

Maya Bechler-Speicher
Blavatnik School of Computer Science
Tel-Aviv University

Amir Globerson
Blavatnik School of Computer Science
Tel-Aviv University*

Ran Gilad-Bachrach
Department of Bio-Medical Engineering
Edmond J. Safra Center for Bioinformatics
Tel-Aviv University

Abstract

Graph Neural Networks (GNNs) have emerged as the predominant approach for learning over graph-structured data. However, most GNNs operate as black-box models and require post-hoc explanations, which may not suffice in high-stakes scenarios where transparency is crucial. In this paper, we present a GNN that is interpretable by design. Our model, Graph Neural Additive Network (GNAN), is a novel extension of the interpretable class of Generalized Additive Models, and can be visualized and fully understood by humans. GNAN is designed to be fully interpretable, offering both global and local explanations at the feature and graph levels through direct visualization of the model. These visualizations describe exactly how the model uses the relationships between the target variable, the features, and the graph. We demonstrate the intelligibility of GNANs in a series of examples on different tasks and datasets. In addition, we show that the accuracy of GNAN is on par with black-box GNNs, making it suitable for critical applications where transparency is essential, alongside high accuracy.

1 Introduction

In many domains, ranging from biology to fraud detection, Artificial Intelligence (AI) is applied to data with graph structure. Neural Networks, and specifically Graph Neural Networks (GNNs), have emerged as the predominant approach in these applications (see, for example, Zhou et al. [1]). While GNNs demonstrate high accuracy, in terms of the correctness of their predictions, they often function as black-box models; thus, their decision-making processes are opaque. Transparency is vital for assessing potential biases or safety risks, and is particularly critical in high-stakes areas such as criminal justice, healthcare, and finance, where decisions significantly impact individuals' lives. In such contexts, interpretable models, despite sometimes being less accurate, may be preferred over complex black-box models [2]. Furthermore, the transparency of automated decision making processes is increasingly becoming a legal mandate. While there is ongoing debate over whether the European Union's General Data Protection Regulation (GDPR) implies a *"right to explanation"* [3, 4], the proposed European AI Act explicitly addresses this issue, stating that *"To address concerns related to opacity and complexity of certain AI systems and help deployers to fulfill their obligations under this regulation, transparency should be required for high-risk AI systems before they are placed on the market or put into service"* [5].

---

*Now also at Google Research

In this context, interpretability refers to the ease with which a human can understand the reasoning behind model decisions or the general logic of a model's operation. It is important to distinguish between interpretability and explainability [2]. Interpretability relates to models that are inherently comprehensible by design, while explainability pertains to post-hoc methods that elucidate aspects of black-box models [6]. These explanations often come without correctness guarantees [7, 8] and may not provide a complete description of the model and its predictions, potentially failing to expose hidden pitfalls [9, 10, 11].

Methods for model explainability or interpretability can be categorized into local and global types. Local methods, such as SHAP [12] and LIME [13], elucidate individual predictions made by a model, whereas global methods, such as feature-importance [14] and partial dependence plots [15], provide holistic insights about the model, i.e., explain the overarching logic of the model decision-making [16]. However, it has been noted that local explainability methods may not consistently align with their global counterparts [17]. Moreover, local explanations may be inadequate for verifying fairness and other risks [8].

In this work, we introduce the Graph Neural Additive Networks (GNAN), an interpretable-by-design GNN that offers both transparency and accuracy. GNAN is a glass-box model [18] that allows for both local and global interpretability. GNAN extends the family of Generalized Additive Models (GAMs) [19], to accommodate graph data. GAMs are known for their ability to fit complex, nonlinear functions while remaining interpretable and have proven effective across various domains [20, 21, 22, 23, 24]. They operate by learning shape functions for each feature and then linearly combining these functions, making it easy to interpret them, as the influence of each feature on the prediction is independent of other features and can be visualized through their corresponding shape functions. Similarly, GNAN's interpretability is achieved through an architecture that restricts the use of cross-products of features and graphs' topology, thereby reducing its complexity compared to other GNNs. Nonetheless, we demonstrate that GNAN, despite its limited capacity, matches the performance of more expressive GNNs on several real-world datasets. Additionally, GNAN does not rely on iterative local message-passing, avoiding the computational bottlenecks commonly associated with such GNNs [25].

In Section 4, we showcase through a series of examples how users can interpret GNAN and gain precise insights into the connections between the target and the graph, the target and the features, and the interplay between features and graph information. In some cases, an exact description of the model can be visualized through only a few figures. We also demonstrate how the interpretability of GNAN allows users to debug their model, a process that can be used for ensuring consistency with prior knowledge and avoiding biases and safety risks. In Section 5, we compare the performance of GNAN with other GNN architectures. This comparison underscores that sacrificing performance for intelligibility is not necessary, as the performance of GNAN is comparable to that of commonly used black-box GNNs.

The main contributions of this work are:

1. The extension of Generalized Additive Models (GAMs) to graph data.

2. The introduction of a fully interpretable-by-design model for graph prediction tasks, demonstrating that its explanations provide both global and local insights, through visualizations of the model itself, and include debugging capabilities.

3. The demonstration that GNAN achieves good performance on common real-world graph datasets, despite its limited capacity. This observation supports previous findings that some real-world graph problems are simple and do not require the capacity of other GNNs.

Thus we argue that GNAN is suitable for high-stakes applications due to its interpretability and performance.

2 Related work

Generalized Additive Models   Generalized Additive Models (GAMs) are a class of statistical models that build upon generalized linear models by incorporating non-linear functions for each variable while maintaining additivity [19, 20, 21]. Essentially, GAMs model the expected value of the target variable as a sum of univariate functions of the features. Formally, in GAMs, a prediction

for an input $\mathbf{x}$ is computed by $\sigma\left(\sum f_k\left(\mathbf{x}_k\right)\right)$ where σ is a predefined activation function, such as the sigmoid[2], and the f_k's are shape functions learned during the training process. This approach extends generalized linear models, in which predictions are computed by $\sigma\left(\sum \mathbf{w}_k\mathbf{x}_k\right)$ where w is a learned weight vector.

GAMs are more expressive than generalized linear models while remaining interpretable, as the effect of each predictor is modeled separately. For example, they can capture non-monotone effects of features, which generalized linear models cannot achieve without feature engineering. Traditionally, GAMs utilize splines or other smooth shape functions to model the non-linear relationships between each feature and the target variable. However, other methods, such as trees, have been proposed to fit the shape functions [24]. Recently, Agarwal et al. [26] suggested using neural networks to learn the shape functions. This approach combines the representational power of deep learning with the interpretability of additive models.

Graph Neural Networks Graph Neural Networks (GNNs) [27, 28, 29, 30] have emerged as the leading approach for learning over graph data. The fundamental idea behind GNNs is to use neural-networks that combine node features with graph-structure. A commonly used family of GNNs is message-passing GNNs, where the representations of nodes are updated in iterations through neighborhood aggregations. This aggregation is done, for example, through a convolution-like operation or an attention mechanism. [31, 32, 33]

Various non-message-passing approaches have been explored to disentangle the node features from the graph structure. Such approaches were shown to enhance performance across diverse applications [34, 35, 36]. Disentanglement can also reduce overfitting, as popular GNNs which do entangle features and graph-structure, were shown to have the tendency to overfit non-informative graph information [37] GNAN uses these concepts in order to achieve a model that is both high-performing and fully interpretable.

There are different prediction tasks on graphs [38]. In *graph tasks*, the goal is to predict properties of entire graphs. For example, a graph could represent a molecule, and the goal would be to predict its toxicity level. In *node tasks*, the goal is to predict a property of a node (vertex) within a graph. An example of a node task is predicting whether a user in a social network is a human or a bot. In *link prediction tasks*, the goal is to determine whether there is an edge between two nodes of a graph. In this work, we focus on graph tasks and node tasks. Although link prediction tasks are not within the scope of this work, it is possible to view these problems as node tasks on the dual line graph [39].

GNNs explanations The inherent complexity of graph-structured data poses unique challenges for explainability. Most approaches for explaining black-box GNNs focus on providing a sub-graph or a similar structure that can explain a certain example. This is done either as a post-hoc explanation for GNNs [40, 41, 42] or by adjusting the data a priory [43, 44, 45]. For example, the method suggested in Ying et al. [41] identifies both important subgraph structures and node features influencing the GNN's predictions by maximizing the mutual information between the prediction and the distribution of possible subgraph structures and node features. Yin et al. [43] suggested a structural pattern learning module that is learning through pre-training. GNAN, on the contrary to these methods, does not aim to provide an explanation through a proxy object like a subgraph, nor does it require modification to the data, or the training process. Instead, GNAN is a interpretable by design, and its exact description can be visualized through its learned shape functions. In particular, the exact relation between the target, the features, and the graph can be visualized and conveyed to users.

3   Graph Neural Additive Networks

In this section, we introduce the Graph Neural Additive Networks (GNAN). We begin by defining some essential notation. A graph G has a set of N vertices, where each vertex is associated with a d-dimensional feature vector. Specifically, $\mathbf{x}_i \in \mathbb{R}^d$ represents the feature vector of the i'th node in G. We define the distance $\text{dist}(j, i)$ between node j and node i within the graph G as the number of edges in the shortest path from j to i. This definition implies that the distance from a node to itself is zero. In cases where no path exists from j to i, we set $\text{dist}(j, i) = \infty$. For enhanced readability, we denote vectors in boldface, and an entry k of a vector $\mathbf{x}$ is denoted by $[\mathbf{x}]_k$. We begin

---

[2]In the context of Generalized Linear Models (GLMs) σ can be though of as the inverse of the link-function.

by describing GNAN for applications such as binary classification and regression where the model output is one-dimensional. At the end of this section, we discuss extensions to scenarios such as multi-class classification, where the model output is multi-dimensional.

GNAN generates a representation $\mathbf{h}_i \in \mathbb{R}^d$ for each node i by learning a distance function $\rho(x; \theta) : \mathbb{R} \to \mathbb{R}$ and a set of feature shape functions $\{f_k\}_{k=1}^d, f_k(x; \theta_k) : \mathbb{R} \to \mathbb{R}$. Each of these functions is a neural network, and is optimized through back-propagation. For brevity, we omit the parameterization θ and θ_k for the remainder of this section. In GNAN, the k'th entry of the representation $\mathbf{h}_i$ for the i'th node is defined as follows:

$$[\mathbf{h}_i]_k = \sum_{j=1}^N \frac{1}{\#\text{dist}_i(j,i)} \cdot \rho\left(\frac{1}{1+\text{dist}(j,i)}\right) \cdot f_k\left([\mathbf{x}_j]_k\right)$$

where $\#\text{dist}_i(j,i)$ represents the number of nodes at distance $\text{dist}(j,i)$ from node i. The underlying rationale for this definition is as follows: each node's k'th feature is transformed by a shape function f_k, independently from other features. The effect the k'th feature value of node j has on node i's representation is influenced by their distance. Specifically, if $\text{dist}(j,i) = l$, then the cumulative influence of all nodes at distance l from node i is captured by $\rho(1/(1+l))$. This is achieved by the normalization term $1/\#\text{dist}_i(j,i)$. Here, ρ's argument $1/(1+l)$ scales the distance such that a distance of 0 (the self-distance of a node) is mapped to 1, and an infinite distance, which implies no path exists, is scaled to 0. Thus, ρ spans the interval $[0,1]$.

The representation of each node is dependent on the entire graph, yet the function ρ enables weighting the influence from nodes, based on their distance. This enables, for example, diminishing the impact of distant nodes, or close neighbors. For each node i, the weighted sum of the transformed feature vectors of all other nodes is computed, with weights assigned according to their distance from i. This weighted sum is computed after the shape functions are applied to the distances and the features of each node.

Given the node representations, both node prediction tasks and graph prediction tasks can be implemented. To predict for the i'th node, the computation is as follows:

$$\sigma\left(\sum_{k=1}^d [\mathbf{h}_i]_k\right) \quad ,$$

where the entry-wise sum of the representation vector $\mathbf{h}_i$ is computed and subsequently processed using an activation function such as the sigmoid for classification and the identity for regression. For a prediction over the entire graph, the collective node representation is computed via sum-pooling:

$$\mathbf{h} = \sum_{i=1}^N \mathbf{h}_i \quad .$$

Following this, the entry-wise sum of the graph representation $\mathbf{h}$ is computed and also processed using the activation function:

$$\sigma\left(\sum_{k=1}^d [\mathbf{h}]_k\right) \quad . \tag{1}$$

Once the model is trained, it can be fully described using its univariate functions ρ and $\{f_k\}_{k=1}^d$.

From the definitions provided above, it follows that the entire model can be represented with just a few figures, thus providing global interpretability. For local explanations, it is feasible to examine the contribution of each feature and each node to the predictions. The following representation convey the influence of each node to the k'ts feature:

$$[\mathbf{h}]_k = \sum_{i=1}^N [\mathbf{h}_i]_k = \sum_{j=1}^N f_k\left([\mathbf{x}_j]_k\right) \sum_{i=1}^N \frac{1}{\#\text{dist}_i(j,i)} \cdot \rho\left(\frac{1}{1+\text{dist}(j,i)}\right) \quad .$$
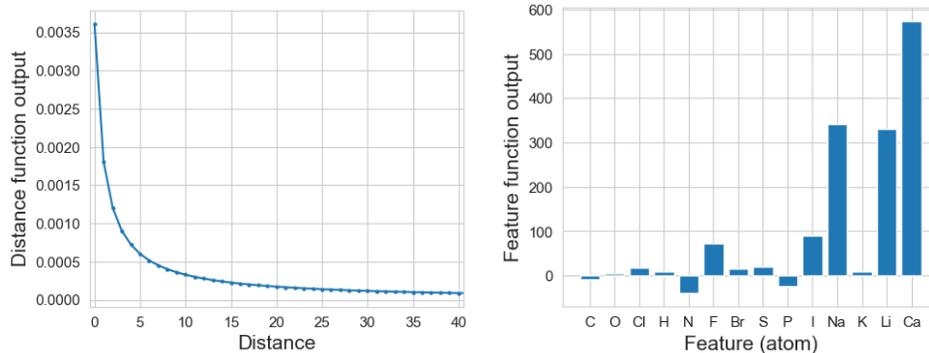
Figure 1: Visualization of the distance and feature functions, learned on Mutagenicity. As the features are binary, the feature functions are evaluated only on the value 1. These plots provide an exact description of the functions' signal processing and a global explanation of how the model uses the distances and features.

Here $[\mathbf{h}_i]_k$ contains the influence of the k'th feature in node i on the prediction. However, from the definition of $[\mathbf{h}_i]_k$ we see that it serves as a mediator for influences of all other nodes. Therefore, the influence of node j on feature k in the final graph representation can be obtained by:

$$f_k\left([\mathbf{x}_j]_k\right) \sum_{i=1}^{N} \frac{1}{\#\mathrm{dist}_i(j,i)} \cdot \rho\left(\frac{1}{1+\mathrm{dist}(j,i)}\right) \quad . \tag{2}$$

We can also extract the total contribution of each node i to the prediction, by summing the contribution of the nodes over the features, as done in the input to σ in Equation 1:

$$\sum_{k=1}^{d}[\mathbf{h}]_k = \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{1}{\#\mathrm{dist}_i(j,i)} \cdot \rho\left(\frac{1}{1+\mathrm{dist}(j,i)}\right) \sum_{k=1}^{d} f_k\left([\mathbf{x}_j]_k\right) \quad . $$

Therefore, the total contribution of node i to the prediction is

$$\sum_{j=1}^{N} \frac{1}{\#\mathrm{dist}_i(j,i)} \cdot \rho\left(\frac{1}{1+\mathrm{dist}(j,i)}\right) \sum_{k=1}^{d} f_k\left([\mathbf{x}_j]_k\right) \quad . \tag{3}$$

Overall, the model facilitates a detailed understanding of local behavior from multiple perspectives.

The functions ρ and $\{f_k\}_{k=1}^{d}$ may be implemented using a variety of neural network architectures. In our experiments, we employed multi-layer perceptrons (MLPs) with ReLU activations to implement these functions. Nonetheless, other alternatives are viable, such as employing learning splines for activations to achieve smoother shape functions [46]. Additionally, it is feasible to develop a separate distance network for each feature to enhance the model's capacity. Specifically, rather than utilizing a single function ρ, one can train a distinct function ρ_k for each feature k, which weights the contribution of each feature based on its node's distance. For graph-level tasks, additional feature networks may be integrated prior to aggregating the graph's representation vector, akin to a readout layer in GNNs. These extensions, along with a discussion on a tensor representation of the computation that facilitates efficient GPU utilization, are further elaborated in the Appendix.

For multiclass classification involving C classes, we configure the final layers of the feature shape functions $f_k(x;\theta_k): \mathbb{R} \to \mathbb{R}^{C\times 1}$ and the distance function $\rho(x;\theta): \mathbb{R} \to \mathbb{R}^{C\times 1}$ to accommodate the required dimensionality. The transformed feature vectors and the distance metrics are combined using an element-wise multiplication denoted by $\odot$, as follows:

$$[\mathbf{h}_i]_k = \sum_{j=1}^{N} \frac{1}{\#\mathrm{dist}_i(j,i)} \cdot \rho\left(\frac{1}{1+\mathrm{dist}(j,i)}\right) \odot f_k([\mathbf{x}_j]_k) \quad . $$

For prediction purposes, the sum operator is applied independently across the dimensions corresponding to each class, and a softmax is employed as the activation function.

4 Inteligibility

In this section, we demonstrate the intelligibility of GNAN through visualizations. Each GNAN model is characterized by the univariate learned shape functions ρ and $\{f_k\}_{k=1}^d$, and can thus be depicted as a set of illustrative figures. Below, we present examples of such figures and explain their utility in generating insights. Our focus in this section is on global interpretability, as local interpretability can utilize analogous ways. We showcase GNAN's application on two datasets, with additional examples detailed in the Appendix.

Our initial examples focus on the task of detecting mutation-causing molecules using the Mutagenicity dataset [47]. In this task, molecules are modeled as graphs where nodes correspond to atoms and edges to connections between these atoms. Each atom type is represented by a 14-dimensional one-hot encoding. A GNAN model trained on this dataset is illustrated in Figure 1. On the left, the function ρ is presented, demonstrating how distance impacts prediction, with a clear diminishing influence of more distant atoms. On the right, the shape functions for the features are displayed. Given that the features are binary, each shape function manifests only two values: one when the feature is 0 (indicating that the atom is not of the specified type), and another when the feature is 1 (indicating that the atom is of the specified type). Defining $b = \sum_k f_k(0)$ as the bias term allows us to set $f_k(0) = 0$ for each k, thereby enabling the plotting of only $f_k(1)$. The graphical representation reveals that atoms such as Ca, Na, and Li are predicted to correlate with an increased mutagenicity effect, whereas N and P atoms are predicted to be associated with a slight protective effect.

It is essential to emphasize that Figure 1 displays the entire model comprehensively. This means that combined with the value of the bias term, which is -5.6672 in this case, every crucial detail needed to understand and utilize this model for predictions is contained within this single figure. This stands in stark contrast to methods like feature importance, which offer a limited perspective on models. While the figure provides complete information about the model, presenting additional views can sometimes be helpful.

Figure 2 showcases the cross product of the shape functions and the distance function as a heatmap. Each cell (k, l) in the heatmap represents the value $\rho\left(1/(1+l)\right) \cdot f_k(1)$. This figure illustrates the interplay the model has learned between the graph's structure and the attributes of its nodes. As the task involves binary classification, positive values in the heatmap contribute to classifying a molecule as mutagenic, whereas negative values indicate non-mutagenic properties.

This heatmap illustrates how atoms at specific distances influence the final outcome. For instance, it shows that the model has learned that the presence of a Ca atom (cell (Ca, 0)) or its proximity (cell (Ca, 1)) contributes to mutagenicity. The visualizations can also be used for debugging purposes. This can be crucial, for example, to ensure that the model is free from biases or to identify any discrepancies with existing scientific knowledge. If it is already known that Ca atoms actually have a negative effect on mutagenicity, users could identify and correct this misalignment in the model's learning. Additionally, this detailed understanding allows users to select models that not only achieve high accuracy on the given samples but also align with prior knowledge, optimizing both performance and reliability.

Interpreting multiclass prediction tasks poses significant challenges, as noted by Zhang et al. [6]. In this context, we showcase the interpretability of GNAN using the PubMed dataset [48]. This dataset comprises 19,717 scientific publications related to diabetes archived on PubMed and categorized into three distinct classes (type-1 diabetes, type-2 diabetes, and gestational diabetes). The dataset's citation network includes 44,338 links. Each publication, represented as a node, is characterized by a TF/IDF weighted word vector derived from a dictionary containing 500 unique words.

As there are three classes, we trained the GNAN model such that the output of the distance and feature functions are of dimension three. In this setting it is interesting to compare the three functions, corresponding to the three classes and therefore we draw them on the same figure [6]. Figure 3 shows that the model uses only the local neighborhood of each node, and as nodes become more distanced, the information between them is less used. We also observe a difference between the classes; while for type 2 diabetes, the longer the distance, the less their information is used (converges to 0), for type 1 and gestational diabetes, nodes of long-distance have a negative effect.

In Figure 4, we display the feature shape functions for nine selected features, demonstrating GNAN's capability to learn complex, non-monotone functions such as those seen in the 'diet' and 'hepat'
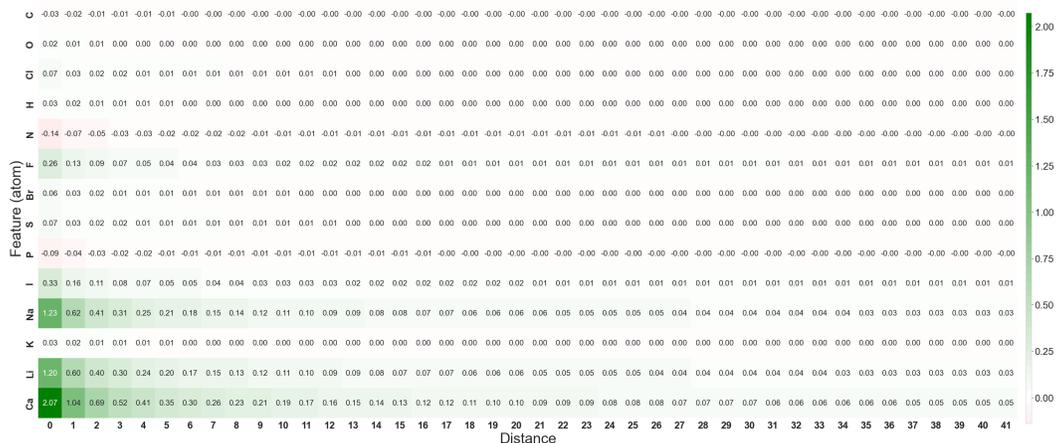
Figure 2: Visualization of products of the outputs of the distance function and the feature functions, trained on Mutagenicity. Each cell shows the exact contribution, positive or negative, of features at a certain distance to the prediction. Positive values (green) contribute to classifying a molecule as mutagenic, and negative values (red) contribute to classifying a molecule as non-mutagenic.
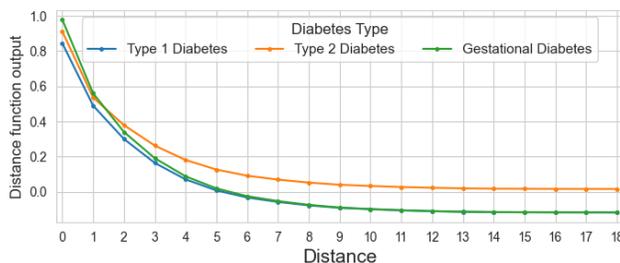

Figure 3: Visualization of the distance shape function learned on the PubMed dataset. As the output of the function is of dimension three, we plot it as three shape functions, one for each class. We plot them on the same figure to compare them. The shape functions show that the model uses only the local neighborhood of each node. It also shows a difference between the classes; while for type 2 diabetes, the longer the distance, the less their information is used (converges to 0), for type 1 and gestational diabetes, nodes of long-distance have a negative effect.

features. Observing these shape functions across the three classes simultaneously allows for an understanding of how different feature values are utilized by the model to distinguish among the classes. For instance, the shape function for the 'insulin' feature reveals that the absence of this word in a document (i.e., feature values close to zero) does not significantly indicate the document's class. However, as the frequency of 'insulin' increases within the document, its impact on the prediction becomes more pronounced, though this effect varies distinctly between type 1 & 2 diabetes and gestational diabetes.

To visualize the contribution of a feature value at a specific distance, we employ a heatmap for each class, evaluating the products between the outputs of the feature function over the input range ($[0, 1]$) and the output of the corresponding distance function. Figure 5 exemplifies this visualization technique with the 'children' feature. It is insightful to observe that the presence of the word 'children' influences the predictions differently across the diabetes types. The model has learned that papers concerning type 1 diabetes seldom mention 'children', nor do related papers. In contrast, the term frequently appears in the context of gestational diabetes.

It is possible to construct confidence intervals for GAMs using the bootstrap method[49, 50]. We present such example with additional visualization examples in the appendix.
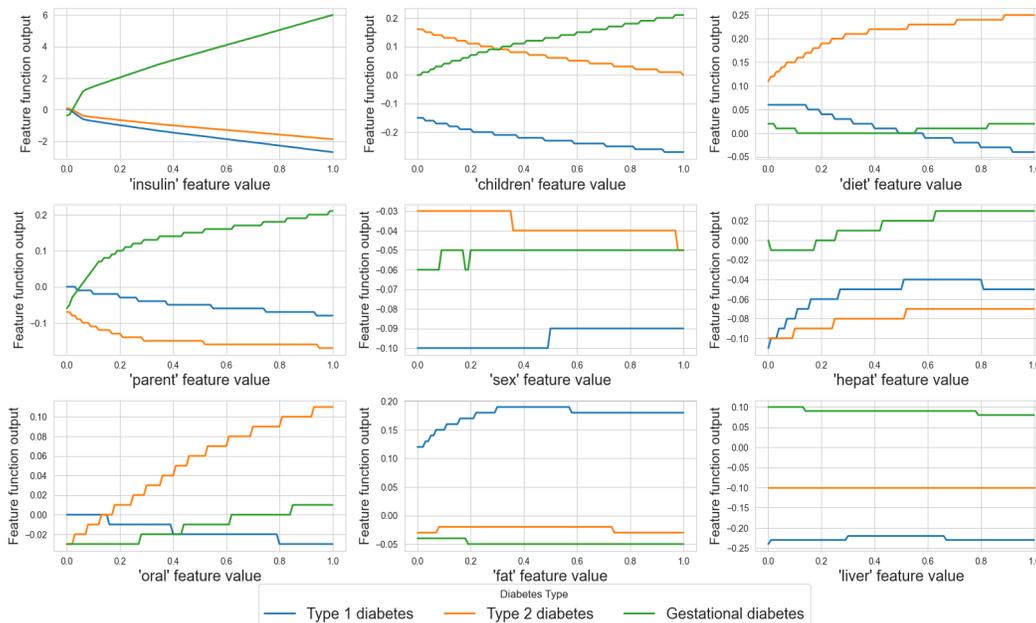
Figure 4: Visualization of nine features' shape functions, learned over the PubMed dataset.
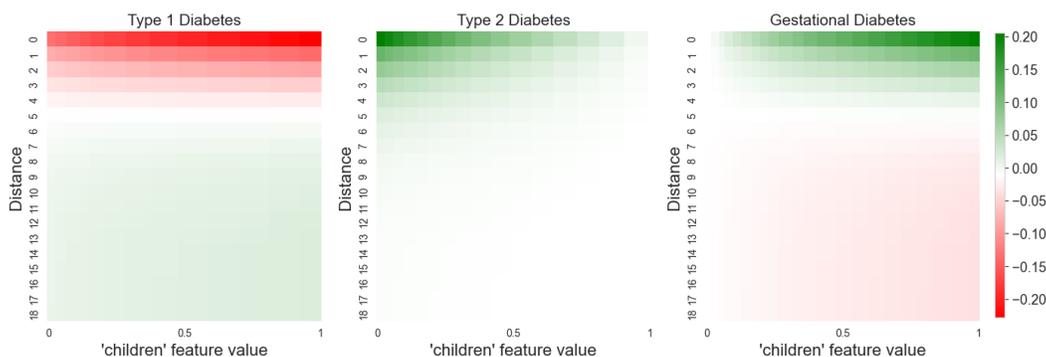

Figure 5: Visualization of the products between the outputs of the 'children' feature function over the input range $[0, 1]$ and the outputs of the distance function, learned over the PubMed dataset.

4.1 Local Explanations

Up to this point, we have demonstrated how to visualize GNAN for providing global explanations. These explanations offer a comprehensive visualization of the entire model. Now, we shift our focus to local explanations, i.e., those relevant to particular examples of interest. To illustrate this, we employ the same Mutagenicity model that was visualized globally to explain specific samples within the data. Using Equation 3, we compute the importance of each node and visualize two molecules from the dataset, where the area of each node corresponds to its importance.

Figure 6 presents two such examples. In Figure 6(a) the carbon (red) atoms play the most significant role, and a carbon ring (red cycle) is highlighted. In Figure 6(b), a group of NO_2 (grey and green subgraph) is shown to be relatively important for predicting the molecule as mutagenic. Both carbon rings and NO_2 groups are well-known for their mutagenic effects [51], making them frequently discussed in the literature on explainable GNNs [39, 41].

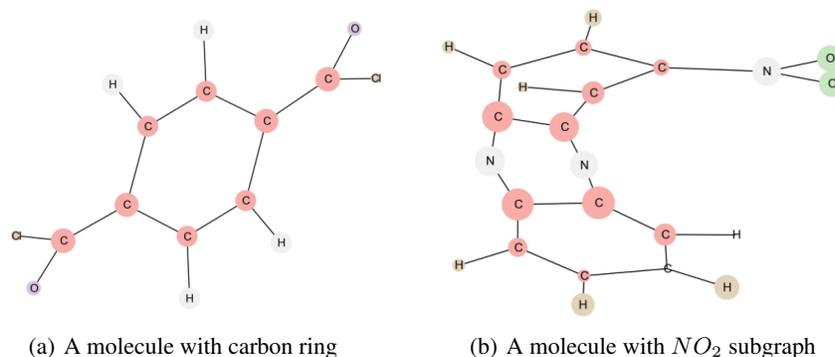(a) A molecule with carbon ring   (b) A molecule with NO_2 subgraph

Figure 6: Local explanations of two molecules from the Mutagenicity datasets, through visualizations of the molecules. The area of each atom corresponds to the node importance according to Equation 3.

5   Empirical evaluation

In this section, we evaluate GNAN on real-world graph and node labeling tasks, including large-scale, long-range, and heterophily datasets.[3]. We compare GNAN to multiple commonly used black-box GNNs including GraphConv [52], GraphSAGE [30], Graph Isomorphism Network (GIN) [33], the expressive version of the Graph Attention Network (GATv2) [29, 53], the Graph Transformer (GTransformer) [54]. We also evaluate the FSGNN model, which disentangles the node features from the graph structure [35]. The information on the hyper-parameters tuned for each baseline can be found in the Appendix. We used the following common benchmarks:

Node labeling tasks *Cora, Citeseer, PubMed, ogb-arxiv* [55, 56] are paper citation networks where the goal is to classify papers into one of several topics. The ogb-arxiv dataset is a large-scale network. *Cornell [57] & Tolokers [58]* are heterophilious datasets. Cornell is a web-link network with the task of classifying nodes into one of five categories. Tolokers dataset is based on data from the Toloka crowdsourcing platform. The nodes represent tolokers (workers) who have participated in at least one of 13 selected projects. An edge connects two tolokers if they have worked on the same task. The goal is to predict which tolokers have been banned in one of the projects. Node features are based on the worker's profile information and task performance statistics.

Graph labeling tasks *NCI1, Proteins, Mutagen & PTC* [59] are datasets of chemical compounds. In each dataset, the goal is to classify compounds according to some property of interest. Thr μ ,α ,α_{HOMO} [60] datasets are long-range molecular property prediction regression tasks, over the large-scale QM9 molecular dataset.
Additional data information, including the data statistics, can be found in the Appendix.

Protocol For all tasks, we used existing splits, protocols, and metrics, as commonly used in the literature for each dataset. The complete protocols for each dataset are given in detail in the Appendix. The metrics we report are: For Cornell, Cora, Citeseer, PubMed, ogb-arxiv, Mutagenicity, PTC, NCI, and Proteins, we report accuracy. For μ, α and α_{HOMO} we report MAE. For Tolokers we report ROC-AUC. For the node labeling tasks, we used the pre-defined splits in the data and followed the common protocols for each dataset. The results are an average of the test set using 5 or 10 random seeds. For the Proteins and NCI1 tasks, we followed the splits and the nested-cross-validation protocol from [61]. The final reported result on these datasets is an average of 30 runs (10-folds and 3 random seeds). For NCI1 and PTC we followed the splits and protocol from [39] and report the average accuracy and std of a 10-fold nested cross-validation.

Results The results are presented in Table 1. GNAN performed as the best or second-best model in 9 out of the 13 tasks we evaluated. In GNAN, each node gathers information from all others, ensuring complete information flow, while the ρ function modulates influence based on distance.

---

[3]The implementation can be found at `https://github.com/mayabechlerspeicher/Graph-Neural-Additive-Networks---GNAN`

Table 1: Evaluation of GNAN on node (top) and graph (bottom) tasks. The best and second-best models are marked in cyan and violet colors, respectively. We report accuracy and std for all tasks, except for the Tolokers dataset where we report ROC-AUC and std, and the μ, α, α_{HOMO} datasets where we report MAE and std.

Model	Cornell	Tolokers	Cora	Citeseer	PubMed	ogb-arxiv
GraphConv	65.9 ± 0.5	83.5 ± 0.7	81.3 ± 1.1	75.9 ± 2.0	85.9 ± 0.5	72.4 ± 0.1
GraphSAGE	75.9 ± 5.0	82.4 ± 0.4	81.4 ± 0.7	76.4 ± 0.8	88.4 ± 0.4	71.7 ± 0.2
GIN	69.0 ± 1.3	81.0 ± 0.4	80.0 ± 1.2	77.1 ± 1.9	85.3 ± 0.9	73.8 ± 1.4
GATv2	72.5 ± 0.7	83.8 ± 1.1	83.1 ± 0.9	73.9 ± 1.5	84.4 ± 0.5	74.0 ± 2.1
GTransformer	70.5 ± 1.7	83.3 ± 0.9	80.7 ± 0.5	76.0 ± 0.9	85.3 ± 1.6	73.1 ± 0.2
FSGNN	86.0 ± 4.1	83.1 ± 0.6	83.0 ± 1.3	76.2 ± 1.3	85.0 ± 1.3	72.9 ± 1.7
GNAN	85.7 ± 4.8	84.5 ± 0.9	81.1 ± 1.5	75.8 ± 0.6	86.9 ± 1.2	74.1 ± 1.5

Model	μ	α	α_{HOMO}	Proteins	Mutagen	PTC	NCI1
GraphConv	2.91 ± 0.1	4.37 ± 0.5	1.46 ± 0.1	73.1 ± 1.6	64.3 ± 1.7	63.9 ± 5.0	76.5 ± 1.2
GraphSAGE	3.55 ± 0.2	4.51 ± 0.7	1.44 ± 0.2	73.0 ± 4.5	64.1 ± 0.3	67.1±12.6	76.0 ± 1.8
GIN	2.60 ± 0.1	4.67 ± 0.5	1.42 ± 0.1	73.3 ± 4.0	69.4 ± 1.2	55.6±11.1	80.0 ± 1.4
GATv2	2.72 ± 0.1	4.39 ± 0.6	1.41 ± 0.1	73.5 ± 2.8	72.0 ± 0.9	59.5 ± 2.1	80.4 ± 1.6
GTransformer	2.90 ± 0.3	4.30 ± 0.5	1.41 ± 0.2	73.9 ± 1.5	73.1 ± 0.9	55.9 ± 3.5	80.5 ± 1.1
FSGNN	3.57 ± 0.3	4.50 ± 0.4	1.44 ± 0.3	72.9 ± 2.1	66.9 ± 1.5	60.3 ± 7.2	79.7 ± 1.1
GNAN	2.55 ± 0.1	4.28 ± 0.9	1.40 ± 0.1	73.2 ± 3.1	72.2 ± 1.0	64.9 ± 7.1	76.9 ± 1.2

Consequently, GNAN avoids the computational bottlenecks encountered by some message-passing GNNs [25]. Particularly in the long-range tasks μ, α, and α_{HOMO}, GNAN outperformed all other evaluated baselines, aligning with findings by Alon and Yahav [25] that emphasize the benefits of capturing long-range information. While intelligibility sometimes comes at the cost of accuracy, our findings suggest that enhancing intelligibility does not necessarily result in significant accuracy loss. It may appear surprising that GNAN, despite its limited capacity, matches the accuracy of more expressive GNNs. However, prior research indicates that even limited-capacity GNNs, such as linear GNNs, can achieve high accuracy on various real-world datasets [62, 61, 63], suggesting that some real-world graph problems are simpler than anticipated. Our results corroborate these observations.

6 Conclusion

In this work, we introduced the Graph Neural Additive Network (GNAN), a novel extension of the interpretable class of Generalized Additive Models, to accommodate graph data. GNAN is inherently interpretable, and provides both global and local explanations directly from its architecture, eliminating the need for post-hoc interpretations. This direct interpretability enhances the transparency of the model and is particularly useful in high-stakes applications where understanding model decisions is crucial. Furthermore, GNAN demonstrates competitive performance with popular GNNs, showing that intelligibility does not necessarily entail a significant degradation in accuracy.

It is possible to enhance GNAN in several ways. To generate smooth shape functions, one could integrate techniques from the recently proposed Kolmogorov–Arnold Networks [46] or adaptive activations for graphs [64]. Increasing the capacity of GNAN is feasible by learning individual distance functions for each feature. Exploring reduced capacity is also intriguing, particularly in scenarios with many features, where it may be beneficial to employ regularization to limit the number of shape functions used. Additionally, applying these techniques to biological network datasets, such as protein interactions, could be a valuable tool to support scientific discoveries. These and other directions are left for future studies.

Acknowledgements

References

[1] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

[2] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, 2019.

[3] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI magazine*, 38(3):50–57, 2017.

[4] Andrew Selbst and Julia Powles. "meaningful information" and the right to explanation. In *conference on fairness, accountability and transparency*, pages 48–48. PMLR, 2018.

[5] European Parliament. European parliament legislative resolution of 13 march 2024 on the proposal for a regulation of the european parliament and of the council on laying down harmonised rules on artificial intelligence (artificial intelligence act). *OJ*, 2024. URL `https://www.europarl.europa.eu/doceo/document/TA-9-2024-0138_EN.pdf`.

[6] Xuezhou Zhang, Sarah Tan, Paul Koch, Yin Lou, Urszula Chajewska, and Rich Caruana. Axiomatic interpretability for multiclass additive models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 226–234, 2019.

[7] Blair Bilodeau, Natasha Jaques, Pang Wei Koh, and Been Kim. Impossibility theorems for feature attribution. *Proceedings of the National Academy of Sciences*, 121(2):e2304406120, 2024.

[8] Daniel Vale, Ali El-Sharif, and Muhammed Ali. Explainable artificial intelligence (xai) post-hoc explainability methods: Risks and limitations in non-discrimination law. *AI and Ethics*, 2(4): 815–826, 2022.

[9] Rebecca Wexler. When a computer program keeps you in jail. *The New York Times*, 13:1, 2017.

[10] Michael McGough. How bad is sacramento's air, exactly? google results appear at odds with reality, some say. *Sacramento Bee*, 7, 2018.

[11] Marzyeh Ghassemi, Luke Oakden-Rayner, and Andrew L Beam. The false hope of current approaches to explainable artificial intelligence in health care. *The Lancet Digital Health*, 3 (11):e745–e750, 2021.

[12] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[13] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[14] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. Evaluating feature importance estimates. *arXiv preprint arXiv:1806.10758*, 2, 2018.

[15] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[16] Waddah Saeed and Christian Omlin. Explainable ai (xai): A systematic meta-survey of current challenges and future opportunities. *Knowledge-Based Systems*, 263:110273, 2023.

[17] Gabriel Laberge, Yann Batiste Pequignot, Mario Marchand, and Foutse Khomh. Tackling the xai disagreement problem with regional explanations. In *International Conference on Artificial Intelligence and Statistics*, pages 2017–2025. PMLR, 2024.

[18] Valentina Franzoni. From black box to glass box: advancing transparency in artificial intelligence systems for ethical and trustworthy ai. In *International Conference on Computational Science and Its Applications*, pages 118–130. Springer, 2023.

[19] Trevor Hastie and Robert Tibshirani. Generalized Additive Models. *Statistical Science*, 1 (3):297 – 310, 1986. doi: 10.1214/ss/1177013604. URL `https://doi.org/10.1214/ss/ 1177013604`.

[20] Trevor Hastie and Robert Tibshirani. Generalized additive models for medical research. *Statistical methods in medical research*, 4(3):187–196, 1995.

[21] Trevor Hastie and Robert Tibshirani. Generalized additive models: Some applications. *Journal of the American Statistical Association*, 82(398):371–386, 1987. doi: 10.1080/01621459.1987. 10478440.

[22] Thomas W Yee and Neil D Mitchell. Generalized additive models in plant ecology. *Journal of vegetation science*, 2(5):587–602, 1991.

[23] Robert A Rigby and D Mikis Stasinopoulos. Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 54(3):507–554, 2005.

[24] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1721–1730, 2015.

[25] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications, 2021.

[26] Rishabh Agarwal, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Ben Lengerich, Rich Caruana, and Geoffrey Hinton. Neural additive models: Interpretable machine learning with neural nets, 2021.

[27] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL `https: //openreview.net/forum?id=SJU4ayYgl`.

[28] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry, 2017.

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

[30] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018.

[31] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation?, 2021.

[32] Chloe Wang, Oleksii Tsepa, Jun Ma, and Bo Wang. Graph-mamba: Towards long-range graph sequence modeling with selective state spaces, 2024.

[33] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL `https: //openreview.net/forum?id=ryGs6iA5Km`.

[34] Aseem Baranwal, Kimon Fountoulakis, and Aukosh Jagannath. Optimality of message-passing architectures for sparse graphs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=d1knqWjmNt`.

[35] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. Improving graph neural networks with simple architecture design, 2021.

[36] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. Sign: Scalable inception graph neural networks, 2020.

[37] Maya Bechler-Speicher, Ido Amos, Ran Gilad-Bachrach, and Amir Globerson. Graph neural networks use graphs when they shouldn't. In *Forty-first International Conference on Machine Learning*, 2024.

[38] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *Journal of Machine Learning Research*, 23(89):1–64, 2022.

[39] Maya Bechler-Speicher, Amir Globerson, and Ran Gilad-Bachrach. Tree-g: Decision trees contesting graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11032–11042, 2024.

[40] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network, 2020.

[41] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks, 2019.

[42] Kenza Amara, Rex Ying, Zitao Zhang, Zhihao Han, Yinan Shan, Ulrik Brandes, Sebastian Schemm, and Ce Zhang. Graphframex: Towards systematic evaluation of explainability methods for graph neural networks, 2022.

[43] Jun Yin, Chaozhuo Li, Hao Yan, Jianxun Lian, and Senzhang Wang. Train once and explain everywhere: Pre-training interpretable graph neural networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=enfx8HM4Rp.

[44] Junchi Yu, Tingyang Xu, Yu Rong, Yatao Bian, Junzhou Huang, and Ran He. Graph information bottleneck for subgraph recognition, 2020.

[45] Ying-Xin Wu, Xiang Wang, An Zhang, Xiangnan He, and Tat-Seng Chua. Discovering invariant rationales for graph neural networks, 2022.

[46] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks, 2024.

[47] Jeroen Kazius, Ross McGuire, and Roberta Bursi. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of medicinal chemistry*, 48(1):312–320, 2005.

[48] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[49] David Louis Borchers, Stephen Terrence Buckland, IG Priede, and S Ahmadi. Improving the precision of the daily egg production method using generalized additive models. *Canadian Journal of Fisheries and Aquatic Sciences*, 54(12):2727–2742, 1997.

[50] Wolfgang Härdle, Sylvie Huet, Enno Mammen, and Stefan Sperlich. Bootstrap inference in semiparametric generalized additive models. *Econometric Theory*, 20(2):265–300, 2004.

[51] AK Debnath, RL Lopez de Compadre, G Debnath, AJ Shusterman, and C Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34(2): 786—797, February 1991. ISSN 0022-2623. doi: 10.1021/jm00106a046. URL https://doi.org/10.1021/jm00106a046.

[52] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks, 2021.

[53] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks?, 2022.

[54] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification, 2021.

[55] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings, 2016. URL https://arxiv.org/abs/1603.08861.

[56] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs, 2020. URL https://arxiv.org/abs/2005.00687.

[57] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1e2agrFvS.

[58] Daniil Likhobaba, Nikita Pavlichenko, and Dmitry Ustalov. Toloker Graph: Interaction of Crowd Annotators, 2023. URL https://github.com/Toloka/TolokerGraph.

[59] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. URL www.graphlearning.io.

[60] Raghunathan Ramakrishnan, Pavlo Dral, Matthias Rupp, and Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 08 2014. doi: 10.1038/sdata.2014.22.

[61] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification, 2022.

[62] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/wu19e.html.

[63] Chenxiao Yang, Qitian Wu, Jiahua Wang, and Junchi Yan. Graph neural networks are inherently good generalizers: Insights by bridging gnns and mlps, 2023.

[64] Krishna Sri Ipsit Mantri, Xinzhi Wang, Carola-Bibiane Schönlieb, Bruno Ribeiro, Beatrice Bevilacqua, and Moshe Eliasof. Digraf: Diffeomorphic graph-adaptive activation function, 2024. URL https://arxiv.org/abs/2407.02013.

[65] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[66] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric, 2019.

[67] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tJbbQfw-5wv.

[68] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[69] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks, 2020. URL https://arxiv.org/abs/2005.10203.

A Efficient GNAN implementation with tensor products

We formulate GNAN case of classification with C classes. For regression, the exact formulation holds with $C = 1$. We use the same notation as in the main text, only now the output of the feature and distance function is of dimension $1 \times C$,

For the sake of notation, we assume that every tensor that its last dimension is of size C, is permuted to have the last dimension as its first dimension, without stating it explicitly. This is necessary to achieve a valid tensor multiplication.

We denote with M the matrix of the transformed distances that is outputted by applying ρ, i.e., $M_{i,j} = \rho(\frac{1}{1+dist(j,i)})$, $M \in \mathbb{R}^{C \times N \times N}$.

We denote with F the matrix of the transformed feature is outputted by applying the corresponding f_k for feature l of each node in the graph, i.e., $F_{ik} = f_k(x_l^i)$, $F \in \mathbb{R}^{C \times N \times d}$

Both for node and graph tasks, we first computes the matrix $M \cdot F \in \mathbb{R}^{C \times N \times d}$

The rest of the computation then depends on the task.

A.1 Node Tasks

We aggregate the transformed features weighted by the transformed distances. This is done by summing over the rows of $M \otimes F$:

$$\phi(i) = [M \otimes F \otimes \mathbb{1}_{C \times d \times 1}]_i = \sum_{k=1}^{d} \sum_{j=1}^{N} \rho(\frac{1}{1 + dist(j,i)}) \otimes f_k([\mathbf{x}_j]_k)$$

A.2 Graph Tasks

For graph classification, we set ρ and f to output a scalar and aggregate the transformed features over the nodes, i.e., the row of $M \cdot F$, to form a fixed-size vector of size d.

$$\bar{\Phi}(G) = \mathbb{1}_{C \times 1 \times N} \otimes M \otimes F \in \mathbb{R}^{C \times 1 \times d}$$

Then, we can apply another *readout NAM [26]*:

$$\Phi(G) = \bar{F}(\bar{\Phi}(G))$$

Such that $\bar{F}$ is the transformed features using the function $\{\bar{f}_k\}_{k=1}^{d}$ such that $\bar{f}_k : \mathbb{R} \to \mathbb{R}^{1 \times C}$

We can also simply sum over the outputs. In that case, we will set f and m to output a vector of dimension C:

$$\Phi(G) = \sum_{i=1}^{N} \phi(i) = \mathbb{1}_{C \times 1 \times N} \otimes M \otimes F \otimes \mathbb{1}_{C \times d \times 1}$$

B Extensions and ablations

In Section 3 we mentioned several possible extensions for GNAN. Here, we discuss them in detail.

Readout layer for graph tasks   In graph tasks, after aggregating the node representations, it is possible to apply another transformation before aggregating over the entries of the graph representations. There may be many ways to do so, and we did not explore all of them. We did explore an application of another set of feature functions to each feature or the graph representation vector. This approach increases the capacity of the model in the cost of interpretability. This is because the set of addition feature functions should be plotted separately, and the product between the feature function and the distance functions does not affect the final output directly but rather through another feature function. Empirically, we observed this approach did not improve performance with respect to the performance reported in Section 5.

Splines It is possible to learn splines for the activations in each feature network to achieve smoother shape functions [46]. We note that the Tolokers example presented in Section C shows that the learned feature shape function is smooth, although we use ReLU activations. Nonetheless, in other cases, such as in the PubMed example presented in the main paper, many of the learned feature functions are step functions. Therefore, it is likely that the model could benefit from spline activations, to smooth its shape functions.

Normalization In GNAN we normalize the weight of nodes of distance l with the number of nodes of distance l, so that the cumulative weight of nodes of distance l will be $\rho(1/(1+l))$. We examined the effect of removing this normalization. We observed that without normalization, the loss scale is drastically larger. Therefore, more epochs are required to fit the data. As a result, for the fixed number of epochs we used in our experiments (1000), without normalization, the accuracy decreases.

C Additional explanation examples

In the main paper, we presented two examples of explanations over two datasets with different properties. In this section, we present additional explanations and examples we could not fit into the main text due to space limitations.

C.1 Confidence Intervals

In our main paper, we discussed the construction of confidence intervals for Generalized Additive Models (GAMs) using the bootstrap method. Here, we provide a specific example using the Mutagenicity dataset.

We computed 95% confidence intervals by applying the bootstrap method with 200 resamples. This involved resampling the original dataset with replacement 200 times and calculating the statistic of interest for each resample. The resulting bootstrap estimates were sorted, and the 2.5th and 97.5th percentiles were taken as the lower and upper bounds of the confidence interval, respectively. Figure 7 presents the distance function with its confidence intervals.
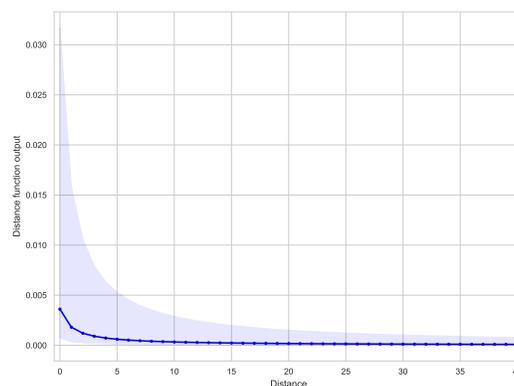

Figure 7

C.2 Additional PubMed heatmaps

In the main text, we presented the heatmaps for the 'children' feature. Here we provide additional heatmaps for additional features: the 'fat' feature and the 'young' feature, as presented in Figures 8 and 9.

C.3 Tolokers - Binary classification with binary and continuous features

The Tolokers dataset is based on data from the Toloka crowdsourcing platform. The nodes represent tolokers (workers) who have participated in at least one of 13 selected projects. An edge connects two tolokers if they have worked on the same task. The goal is to predict which tolokers have been
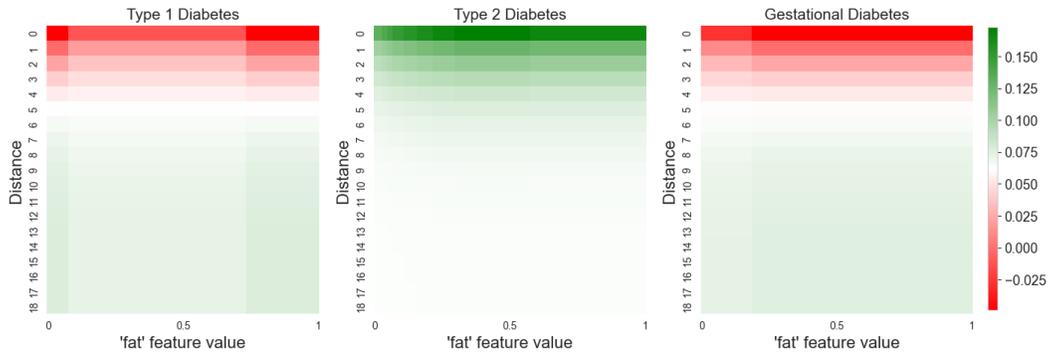
Figure 8: Visualization of the products between the outputs of the 'fat' feature function over the input range [0, 1] and the outputs of the distance function, learned over the PubMed dataset
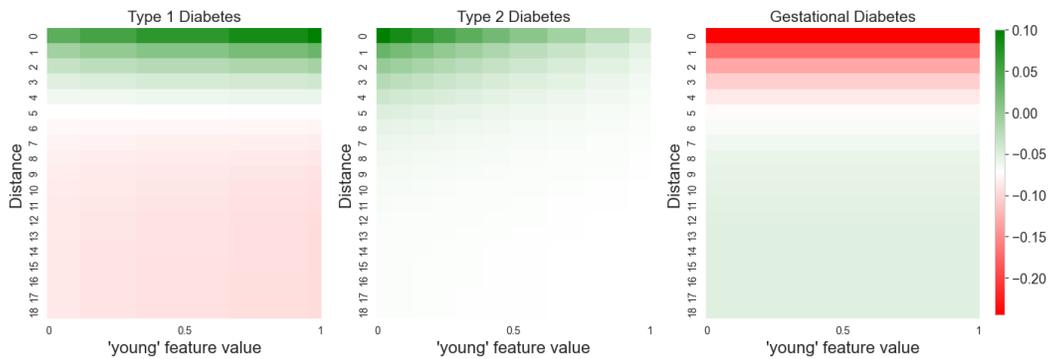

Figure 9: Visualization of the products between the outputs of the 'young' feature function over the input range [0, 1] and the outputs of the distance function, learned over the PubMed dataset

banned in one of the projects. Node features are based on the worker's profile information and task performance statistics. Each node in the graph is associated with nine features. There are 4 continuous features in the range [0, 1] and 5 are binary features. Figure 10 shows the shape functions of the features learned by GNAN. Figure 11 shows the distance shape function learned by GNAN. In Figures 13 and 12 we present the heatmaps of the cross product of the shape functions and the distance function.
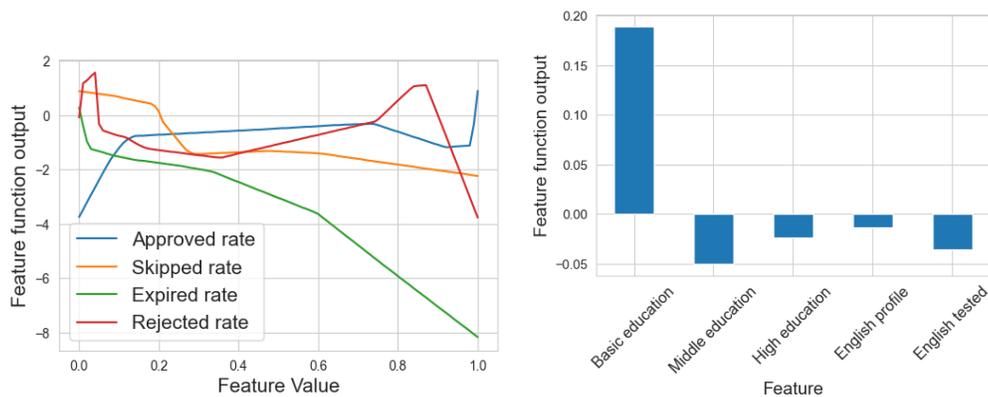

Figure 10: Visualization of the feature functions learned over the Tolokers dataset.
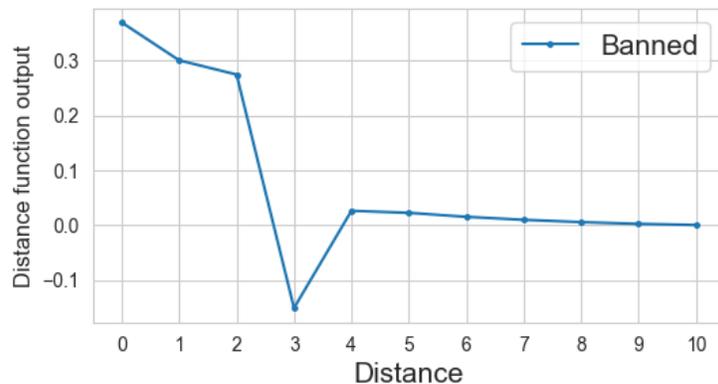
Figure 11: isualization of the distance shape function learned on the Tolokers dataset.
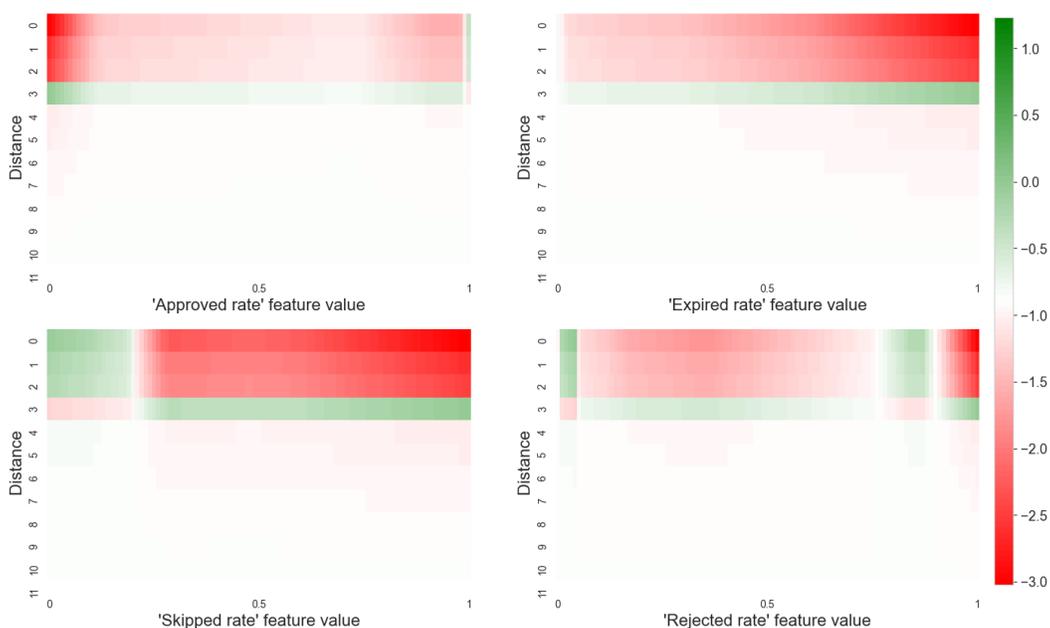

Figure 12: Visualization of the products between the outputs of the continuous feature functions over the input range $[0, 1]$ and the outputs of the distance function, learned over the Tolokers dataset.

D   Additional experimental details

All our baselines are implemented using PyTorch [65] and PyTorch-Geometric [66].

D.1   Dataset information

Here we provide additional information about the datasets used in Section 5. The data statistics are given in Table 2.

Proteins [59] is a dataset of chemical compounds consisting of 1113 graphs, respectively. The goal in the first two datasets is to predict whether a compound is an enzyme or not, and the goal in the last datasets is to classify the type of an enzyme among 6 classes.

NCI1 [59] is a datasets consisting of 4110 graphs, representing chemical compounds. Vertices and edges represent atoms and the chemical bonds between them. The graphs are divided into two classes
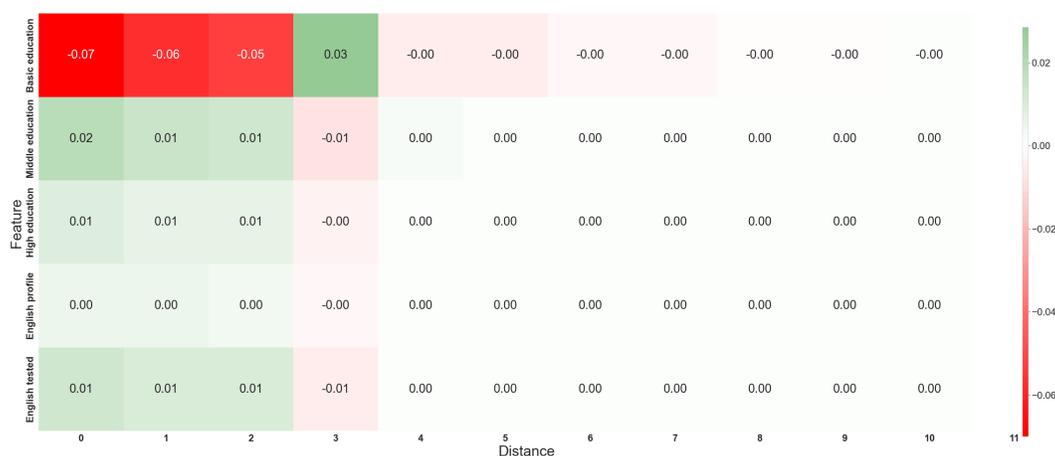
Figure 13: Visualization of products of the outputs of the distance function and the feature functions, trained on Tolokers. Each cell shows the exact contribution, positive or negative, of features at a certain distance to the prediction. Positive values (green) contribute to classifying a toloker as 'banned', and negative values (red) contribute to classifying a toloker as 'not banned'.

according to their ability to suppress or inhibit tumor growth.

Mutagenicity [59] is a dataset consisting of 4337 chemical compounds of drugs divided into two classes: mutagen and non-mutagen. A mutagen is a compound that changes genetic material such as DNA, and increases mutation frequency.

PTC [59] is a dataset consisting of 344 chemical compounds divided into two classes according to their carcinogenicity for rats.

Cornell [57] is a heterophilic webpage dataset collected from the computer science department at Cornell University. Nodes represent web pages, and edges are hyperlinks between them. The task is to classify the nodes into one of five categories.

Table 2: Statistics of the real-world datasets used in our evaluation.

Dataset	# Graphs	Avg # Nodes	Avg # Edges	# Node Features	# Classes
Proteins [59]	1,113	39.06	72.82	3	2
NCI1 [59]	4,110	29.87	32.3	37	2
Mutagenicity [59]	4,337	30.32	30.37	7	2
PTC [59]	344	14	14	19	2
QM9 [60]	130,831	18	37.3	11	-
Cora [55]	1	2,708	10,556	1,433	7
Citeseer [55]	1	3,327	9,104	3,703	6
PubMed [55]	1	19,717	88,648	500	3
ogb-arxiv [56]	1	169,343	1,166,243	128	40
Cornell [57]	1	183	295	1,703	5
Tolokers [58]	1	11758	519000	10	2

D.2 Protocols

ogb-arxive The ogb-arxive datasets are large-scale datasets provided in the Open Graph Benchmark (OGB) paper [56] with pre-defined train and test splits and different metrics and protocols for each dataset. As common in the literature when evaluating OGB datasets, we followed its pre-defined

metric and protocol. The metric used is accuracy. We ran GNAN 10 times and reported the mean accuracy and std over the runs.

Cornell For the Cornell dataset we used the splits and protocol from [57] and report the test accuracy averaged over 10 runs, using the best hyper-paremeters found on the validation set.

Tolokers For the Tolokers dataset, we followed the protocol and pre-defined splits from [58, 67]. The reported result is an average of a 10-fold nested cross-validation.

Core, Citeseer and PubMed Following [68, 30, 29], for the Core, Citeseer and Pubmed datasets we tuned the parameters on the Cora dataset using the pre-defined splits from [68]. For all these datasets we report the test accuracies averaged over 5 runs, using the parameters obtained from the best accuracy on the validation set of Cora.For Cora and Citeseer, we followed the schemes used in [69] where the used graph is the largest connected component, to increase the stability of the evaluated models.

Proteins, NCI We used 10-fold nested cross validation with the splits and protocol of Errica et al. [61]. The final reported result on these datasets is an average of 30 runs (10-folds and 3 random seeds).

Mutagenicity, PTC We use the splits and protocols from [39], and use a 10-fold nested cross-validation. The final reported test accuracies are averages over the 10 test sets of the outer 10 folds.

D.3 Hyperparameters

All GNNs (excluded GNAN) use ReLU activations with $\{3, 5\}$ layers and 64 hidden channels. They were trained with Adam optimizer over 1000 epochs and early on the validation loss with a patient of 100 steps, eight Decay of $1e - 4$, learning rate in $\{1e - 3, 1e - 4\}$, dropout rate in $\{0, 0.5\}$, and a train batch size of 32.

In GNAN, all the feature and distance networks use ReLU activations with $\{3, 5\}$ layers and $\{64, 32\}$ hidden channels. They were trained with Adam optimizer over 1000 epochs weight decay of $0, 5e - 4$, learning rate in $\{1e - 2, 1e - 3\}$, dropout rate in $\{0, 0.6\}$.

D.4 Compute resources

All experiments ran on an NVIDIA GeForce RTX 3090 GPU.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer:[Yes]

   Justification: the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: the paper discuss the limitations of the work performed by the authors

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: NA

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics .

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: the paper discuss both potential positive societal impacts and negative societal impacts of the work performed

    Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: NA

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: NA

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: NA

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: NA

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.