Interpretable and Parameter Efficient Graph Neural Additive Models with Random Fourier Features

Thummaluru Siddartha Reddy, Vempalli Naga Sai Saketh, Mahesh Chandran
{Thummaluru.Siddarthareddy, nagasaisaketh.vempalli, Mahesh.Chandran}@fujitsu.com
Fujitsu Research of India, Bangalore.

Abstract

Graph Neural Networks (GNNs) excel at jointly modeling node features and topology, yet their *black-box* nature limits their adoption in real-world applications where interpretability is desired. Inspired by the success of interpretable Neural Additive Models (NAM) for tabular data, Graph Neural Additive Network (GNAN) extends the additive modeling approach to graph data to overcome limitations of GNNs. While being interpretable, GNAN representation learning overlooks the importance of local aggregation and more importantly suffers from parameter complexity. To mitigate the above challenges, we introduce Graph Neural Additive Model with Random Fourier Features (G-NAMRFF), a lightweight, self-interpretable graph additive architecture. G-NAMRFF represents each node embedding as the sum of feature-wise contributions where contributions are modeled via a *Gaussian process* (GP) with a graph- and feature-aware kernel. Specifically, we construct a kernel using Radial Basis Function (RBF) with graph structure induced by Laplacian and learnable Finite Impulse Response (FIR) filter. We approximate the kernel using Random Fourier Features (RFFs) which transforms the GP prior to a Bayesian formulation, which are subsequently learnt using a single layer neural network with size equal to number of RFF features. G-NAMRFF is light weight with $168\times$ fewer parameters compared to GNAN. Despite its compact size, G-NAMRFF matches or outperforms state-of-the-art GNNs and GNAN on node and graph classification tasks, delivering real-time interpretability without sacrificing accuracy [1].

1 Introduction

Graph Neural Networks (GNNs) are powerful and topology-aware deep representation learning architectures that leverage the structure of graphs as an inductive bias [12, 20, 32]. While GNNs have demonstrated impressive performance on various downstream machine learning tasks such as node and graph classification, they remain largely as a *black-box* models [5, 3]. As a result, the learned representations often lack *interpretability* which limits their adoption in regulated domains like healthcare, finance, and defense [31, 5]. Interpretability in machine learning focuses on designing models whose representations directly reveal how each feature contributes to the output [28]. This stands in contrast to *post hoc* explainable models which seeks to generate explanations for the model inference after the model has been trained [3, 13].

Additive models are the cornerstone for interpretable representation learning for tabular data. Notable examples of such models includes Neural Additive Model (NAM) [2], Neural Basis Model (NBM) [24] and Gaussian Process Additive Model (GPAM) [41]. The key innovation of this additive model lies in its ability to handle each feature independently by learning a separate univariate function for every feature and then summing these contributions to obtain the final prediction. Although all these models share a common trait i.e., additive structure, they differ in how each feature's

[1] Code to reproduce the results is available at https://github.com/FujitsuResearch/GNAM-RFF

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

univariate function is represented and learned. For example, NAM approximates each feature's effect with a dedicated deep neural network (DNN), yielding high flexibility but also posing significant computational overhead. In contrast, NBM and GPAM represent each feature using a fixed dictionary of basis functions (e.g., splines, Fourier bases, or Gaussian process kernels) and learn only the linear combination of weights, thereby retaining interpretability while dramatically reducing training time.

Building on the success of additive models for tabular data, efforts have been made to extend it to graph data. GNAN is the first graph neural additive model that incorporates graph structure as an inductive bias into a neural additive models [5], thus making it a self-interpretable *glass-box* model by design, in contrast to the conventional and well-studied *post-hoc* models designed to explain *black-box* GNNs outputs [31, 3, 37, 22, 17]. GNAN disentangles feature effects and graph topology by learning separate univariate functions, one per each feature and one for neighborhood weighting via DNNs, respectively. The neighborhood weighting is done through a distance measure that spans the entire graph. However, this design has two key drawbacks. Firstly, modeling each feature and neighbourhood with separate neural networks with multiple layers leads to a large trainable parameters with a significant training time. Secondly, for a graph with N nodes and E-edges, GNAN aggregation over all nodes requires $\mathcal{O}(N + E)$ distance computations per node, leading to quadratic scaling of the neighbourhood weighting function with graph size. Additionally, distant, non-neighbouring nodes influence the embeddings through the weighting function in a non-trivial way, introducing opacity to the interpretation unlike localized aggregation.

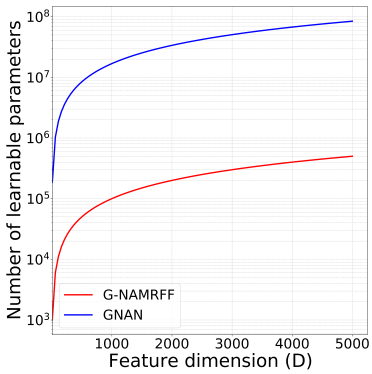


Figure 1: Parameter complexity.

In this work, we introduce the Graph-aware Neural Additive Model with Random Fourier Features (G-NAMRFF), a lightweight and inherently interpretable *glass-box model* designed for representation learning with graphs. Similar to NAM, G-NAMRFF represents each node prediction as the sum of individual feature contributions; however, it replaces deep neural networks with a Gaussian Process (GP) prior characterized by a kernel explicitly designed to be both graph- and feature-aware. Specifically, we propose a smoothness-promoting kernel that encourages neighboring nodes with similar attributes to yield similar outputs. In particular, the proposed kernel can be expressed as a Radial Basis Function (RBF) where inputs are projected features that capture one-hop neighbourhood in the graphs. Based on Bochner's theorem, we approximate the kernel effectively using RFFs, transforming the GP priors into a linear Bayesian formulation. To effectively capture multi-hop graph information, we embed the node features using Finite Impulse Response (FIR) filters with trainable coefficients. These coefficients, trained jointly with univariate functions help uncover critical graph structural information by highlighting which hops significantly influence the node predictions. With the equivalent Bayesian linear model established, we learn the model parameters using a single-layer network whose complexity depends solely on the number of RFFs, thereby ensuring G-NAMRFF's computational efficiency and lightweight nature. Figure 1, illustrates the parameter complexity comparison between our proposed model and existing method, clearly demonstrating the reduced parameter requirements of our approach. More importantly, by feeding RFFs with structure-aware features computed via learnable filter coefficients, our model seamlessly integrates graph topology and feature mapping into a single module unlike GNAN.

We summarize the main contributions of this paper as follows:

- We propose G-NAMRFF, a light-weight, interpretable graph neural additive model with RFFs and minimal parameter overhead. G-NAMRFF is a unified approach that seamlessly integrates topology and feature learning in one framework providing feature level interpretability through univariate functions and clear hop-level interpretability through the learned FIR coefficients.
- We provide theoretical guarantees that establish the robustness and permutation equivariance properties of G-NAMRFF.
- The proposed G-NAMRFF is parameter-efficient with $168\times$ fewer parameters compared to GNAN and consistently outperforms it on both node- and graph-classification tasks, and also

matches or outperforms state-of-the-art black-box GNNs, demonstrating its suitability for sensitive and performance critical applications.

Notation: Throughout this paper, we represent matrices and vectors as $\mathbf{X}$ (boldface capital letters) and $\mathbf{x}$ (boldface lowercase letters), respectively. The (i, j)-th element of a matrix is indexed by $x_{i,j}$. Identity matrix of dimension M is denoted by $\mathbf{I}_M$.

2 Preliminaries

In this section, we set up the mathematical background required for the development of the proposed model.

2.1 Graph Neural Additive Network

Consider a graph $\mathcal{G}$ with N nodes, with the feature vector associated with each node being $\mathbf{x} \in \mathbb{R}^D$; the feature matrix as $\mathbf{X} \in \mathbb{R}^{N\times D}$.The `GNAN` models each feature independently using a `DNN` and weights it with a learnable function that captures underlying graph topology. Formally, the embedding of k-th component of node i is given by,

$$z_{i,k} = \sum_{j=1}^{N} \frac{1}{\#\mathrm{dist}_i(j,i)} \rho\left(\frac{1}{1+\mathrm{dist}(j,i)}\right) f_k(x_{j,k}), \tag{1}$$

Here $f(.)$, $\rho(.)$ are the learnable non-linear functions. $\mathrm{dist}(i, j)$ captures the distance between the nodes and $\#\mathrm{dist}_i(i, j)$ is the number of nodes at $\mathrm{dist}(i, j)$ from node i. From (1), it is clear that embedding of the i-th node requires computing the distance of the i-th node to all the nodes of $\mathcal{G}$, which makes (1) ineffective to handle large graphs.

2.2 Gaussian Process

A Gaussian Process `(GP)` [26] is a probabilistic framework that defines a distribution over functions, where any finite set of function values follows a multivariate Gaussian distribution. It is completely specified by a mean function, $\mathbf{m}(\mathbf{x})$ and a covariance kernel $\mathbf{K}_\theta(\mathbf{x}, \mathbf{x}')$, where $\mathbf{x}$ and $\mathbf{x}'$ are data points on $\mathbb{R}^D$. We denote the `GP` prior as $f(\mathbf{x}) \sim \mathcal{GP}\left(\mathbf{m}(\mathbf{x}), \mathbf{K}_\theta(\mathbf{x}, \mathbf{x}')\right)$, where $\mathbf{K}_\theta(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\theta^2}\right)$ with θ being the bandwidth.

2.3 Kernel Approximation with Random Fourier Features

Random Fourier Features `(RFFs)` based kernel approximation is a popular technique for addressing the scalability of kernel-based methods such as Gaussian Processes [25]. The key idea behind `RFFs` is to approximate a shift-invariant kernel by expressing it as an inner product of low-dimensional feature mappings and is built on Bochner's theorem:

Theorem 2.1 (Bochner's Theorem and RFF Approximation [25])**.** *Let $\mathbf{K}_\theta \in \mathbb{R}^{N\times N}$ denote a bounded, shift-invariant and positive definite kernel (i.e., $\mathbf{K}_\theta$ is a Fourier transform of non negative measure). $\mathbf{K}_\theta$ when properly scaled can be approximated using `RFF` as*

$$\mathbf{K}_\theta(\mathbf{x}, \mathbf{x}') \approx \mathbf{\Phi}_{\mathbf{a}}^T(\mathbf{x})\mathbf{\Phi}_{\mathbf{a}}(\mathbf{x}'), \qquad \mathbf{\Phi}_{\mathbf{a}}(\mathbf{x}) = \left\{\sqrt{\frac{2}{M}}\cos(2\pi\mathbf{a}_{\mathrm{m}}^{\mathrm{T}}\mathbf{x} + \mathrm{b}_{\mathrm{m}})\right\}_{m=1}^{M}, \tag{2}$$

where $\mathbf{\Phi}_{\mathbf{a}}$ is a feature map termed as random Fourier features with M being the number of RFF features, $\mathbf{a}_m \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{\boldsymbol{D}})$ and $b_m \in \mathrm{Unif}(0, 2\pi)$ are drawn from normal and uniform distributions.

This approximation enables a scalable reformulation of the `GP` prior into linear Bayesian model. Specifically, prior can be modeled as $f(\mathbf{x}) = \mathbf{\Phi}_{\mathbf{a}}^T(\mathbf{x})\mathbf{w}, \quad$ where $\mathbf{w} \in \mathbb{R}^{M\times 1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_M)$.

3 Graph-aware Neural Additive Model with Random Fourier Features

In this section, we present `G-NAMRFF`, a lightweight, self-interpretable graph neural additive model that integrates graph topology through `GP` priors and `RFF`.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be weighted and undirected graph with N nodes and the associated graph Laplacian as $\mathbf{L}_\mathcal{G} \in \mathbb{R}^{N\times N}$. We assume that each node $i \in \mathcal{V}$ is associated with a D dimensional feature $\mathbf{x}_i \in \mathbb{R}^{D\times 1}$ and represent the feature matrix as $\mathbf{X} \in \mathbb{R}^{N\times D}$.

Given $\mathbf{X}$ and $\mathbf{L}_\mathcal{G}$, `G-NAMRFF` models each feature contribution $f_k : \mathbb{R} \to \mathbb{R}$ through a Gaussian process whose covariance kernel enforces smoothness with respect to both the feature values and the underlying graph topology. In particular, `G-NAMRFF` obtains the contribution of each feature to node prediction y_i as

$$y_i \;=\; \sum_{k=1}^{D} f_k\big(x_{i,k}\big), \qquad\qquad f_k \sim \mathcal{GP}(0, \mathbf{K}_\mathcal{G}(x_{i,k}, x_{j,k})) \tag{3}$$

where $\mathbf{K}_\mathcal{G}(.,.)$ is a graph covariance kernel. The embedding of node i `i.e.,` $\mathbf{z}_i \in \mathbb{R}^D$ is obtained by stacking $[f_1(x_{i,1}), f_2(x_{i,2}), \ldots, f_D(x_{i,D})]$. In what follows, we first define the graph covariance kernels that enforce smoothness assumption, which are shift invariant and can be approximated using `RFFs`. Then, leveraging this approximation we propose the light-weight model.

3.1 Gaussian Process Priors with Graph-aware Kernel

To encode the prior knowledge that adjacent nodes with similar feature values should output similar function values i.e., function should vary smoothly across the nodes, we impose the `GP` prior with the graph aware covariance kernel as

$$\mathbf{K}_\mathcal{G}(x_{i,k}, x_{j,k}) = \exp\left(\frac{-a_{i,j}(x_{i,k} - x_{j,k})^2}{2\Theta^2}\right), \tag{4}$$

where $\Theta > 0$ is the hyperparameter that controls the width of the function and $a_{i,j} > 0$ corresponds to i, j-element of weight matrix $\mathbf{A}$ with weights assigned based on the distance/similarity between the nodes. Observe that if the nodes are close in the graph and features are similar the kernel outputs high covariance, whereas for distant nodes with features being dissimilar $\mathbf{K}_\mathcal{G}$ yields low covariance. Thus, the proposed kernel $\mathbf{K}_\mathcal{G}$ is both feature and graph-aware. Imposing such a prior enforces $f_k(x_{i,k})$ to vary smoothly across the *connected* nodes. Note that for a fixed $a_{i,j}$, kernel defined in (4) is shift invariant with respect to nodes i, j locally, *i.e.,* $\mathbf{K}_\mathcal{G}(x, y) = \mathbf{G}_\mathcal{G}(x - y)$, with $\mathbf{G}_\mathcal{G}(t) = \exp(-\frac{a_{i,j}t^2}{2\Theta^2})$. Shift invariance of the kernel (Stationarity is required for the approximation of kernels by Bochner's Theorem (see Theorem 2.1) which in turn aids in formulating the light-weight model.

Observe that Eq.(4) can be equivalently expressed as $\mathbf{K}_\mathcal{G}(x_{i,k}, x_{j,k}) = \exp\left(\frac{-(\sqrt{a_{i,j}}x_{i,k} - \sqrt{a_{i,j}}x_{j,k})^2}{2\Theta^2}\right)$. Let us call projected features as $\tilde{x}_{i,k} = \sqrt{a_{i,j}}x_{i,k}$ and $\tilde{x}_{j,k} = \sqrt{a_{i,j}}x_{j,k}$. By rewriting, we obtain kernel function with graph-aware inputs

$$\mathbf{K}_\mathcal{G}(x_{i,k}, x_{j,k}) = \exp\left(\frac{-(\tilde{x}_{i,k} - \tilde{x}_{j,k})^2}{2\Theta^2}\right), \tag{5}$$

In other words, the $\mathbf{K}_\mathcal{G}$ is modified `RBF` with inputs as filtered features based on one-hop neighbourhood since the projected variables use weighted graph adjacency.

Remarks:

- Although the proposed kernel function $\mathbf{K}_\mathcal{G}$ is *shift invariant* and *graph-aware*, it is apparent that it only captures the information from one hop and may fail to capture the information from larger depths (see the importance in section 6).
- More importantly, to study the interpretability, it is also important to analyze for a fixed node i which neighbors are influencing the predictions.

To address the aforementioned challenges, we reformulate the kernel to incorporate information from deeper neighborhoods. Specifically, we make the kernel inputs topology-aware by replacing the original features with graph-filtered representations. This yields an RBF kernel whose inputs are graph-aware feature embeddings. Notably, this formulation preserves both stationarity globally and positive definiteness, ensuring that the resulting covariance function remains valid under the Gaussian process framework and RFF approximation.

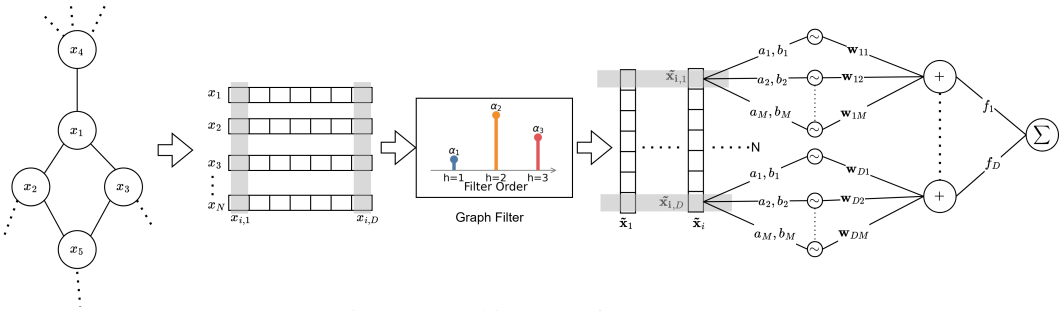


Figure 2: Architecture of G-NAMRFF.

3.1.1 Topology-Aware Random Fourier Features with Learnable FIR Filters

We aggregate multi-hop information through learnable FIR filters, represented as polynomial functions of the graph shift operator-either the adjacency matrix or the graph Laplacian. Specifically, employing the graph Laplacian as the shift operator, the filter is implemented as:

$$\mathbf{H} = \sum_{h=0}^{R} \alpha_h \tilde{\mathbf{L}}_{\mathcal{G}}^{h}, \tag{6}$$

where $\tilde{\mathbf{L}}_{\mathcal{G}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ is the normalized graph Laplacian and R is the filter order which is a hyperparameter that controls how multi-hop information is aggregated. In our framework, we set the filter parameters as learnable. We emphasize that instead of fixing the filter parameters, making them learnable enhances the interpretability (see Section 6). In particular, one can inspect the filter coefficients and infer information such as the hops that are contributing to the node embedding. Using Eq.(6), the filtered response is given by $\tilde{\mathbf{X}} = \mathbf{H}\mathbf{X} \in \mathbb{R}^{N\times D}$, with $\tilde{\mathbf{X}}$ is now the smoothened graph features encoding much richer neighborhood beyond one-hop. Note that modifying the kernel input by the filtered response transforms $\mathbf{K}_{\mathcal{G}}$ to be structure aware RBF that is shift invariant and also non negative. Following Bochners theorem, it can be expressed as $\mathbf{K}_{\mathcal{G}}(x_{i,k}, x_{j,k}) = \mathbf{\Phi}_{\mathbf{a}}^{T}(\tilde{x}_{i,k})\mathbf{\Phi}_{\mathbf{a}}(\tilde{x}_{j,k})$, where $\tilde{x}_{i,k}$ corresponds to (i,k)-th element of $\tilde{\mathbf{X}}$.

Leveraging the kernel approximation technique (Bochner's theorem), the GP prior defined in 3 can be equivalently expressed in linear Bayesian formulation as

$$f_k(x_{i,k}) = \mathbf{\Phi}_{\mathbf{a}}^{T}(\tilde{x}_{i,k})\mathbf{w}_k, \qquad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_M) \tag{7}$$

where $\mathbf{\Phi}_{\mathbf{a}}(\tilde{x}_{i,k}) = \left\{\sqrt{\frac{2}{M}}\cos(2\pi a_m \tilde{x}_{i,k} + b_m)\right\}_{m=1}^{M}$ with $\mathbf{\Phi}_{\mathbf{a}}(.)$ is a feature map, which we refer to as graph-aware Random Fourier Features (G-RFFs), $a_m \sim \mathcal{N}(0,1)$ and $b_m \in \text{Unif}(0, 2\pi)$ are drawn from normal and uniform distributions. To summarize, G-NAMRFF, learns contribution of features to node-i prediction as

$$y_i = \sum_{k=1}^{D} \mathbf{\Phi}_{\mathbf{a}}^{T}(\tilde{x}_{i,k})\mathbf{w}_k, \qquad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_M). \tag{8}$$

RFF Linear Layer
Note that under modified prior, each f_k is a linear transformation whose weights can be obtained by *single-layer neural network* with exactly M learnable parameters per feature. Also, embedding the structural information into features and representing the univariate function using graph-aware RFFs results in a light-weight model with number of trainable parameters dependent on number of RFFs and filter order (more discussion in section 4). We conclude this section with some important observations: (a) for $R = 1$ and $\alpha_0 = 0$, the present approach resembles the first approach where features are transformed based on one-hop neighbourhood. (b) The filter order and number of RFF features required are less compared to hidden units for all the graph datasets (more details in Appendix C.3).

3.2 Implementation Details

In Fig. 2, we present end to end architecture of G-NAMRFF for obtaining the prediction of i-th node as a summation of univariate function outputs. Considering $\mathbf{X} \in \mathbb{R}^{N\times D}$, we pass each feature

Algorithm 1 Algorithm for G-NAMRFF

Inputs: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$, normalized Laplacian $\tilde{\mathbf{L}}_{\mathcal{G}}$, RFF dimension M, bandwidth Θ, filter order R.
Trainable parameters: $\{\alpha_h\}_{h=0}^{R}$, weight vectors $\{\mathbf{w}_k \in \mathbb{R}^M\}_{k=1}^{D}$.
Output: Node predictions $\{y_i\}_{i=1}^{N}$.
Filtering: Compute filtered features $\tilde{\mathbf{X}} = \sum_{h=0}^{R} \alpha_h \, \tilde{L}_{\mathcal{G}}^{h} \, \mathbf{X}$.
RFF computation: Sample frequencies $\{a_{k,m}\}_{m=1}^{M} \sim \mathcal{N}(0,1)$ and phases $\{b_{k,m}\}_{m=1}^{M} \sim \mathrm{Uniform}(0, 2\pi)$ for each k. RFF map: $\mathbf{\Phi_a}(x) = \sqrt{\frac{2}{M}}\big[\cos(a_{k,1}x + b_{k,1}), \ldots, \cos(a_{k,M}x + b_{k,M})\big]^{\top}$
for each node $i = 1, \ldots, N$ **do**
    for each feature $k = 1, \ldots, D$ **do**
        Obtain feature embedding: $f_k(x_{i,k}) = \mathbf{\Phi}_{\mathbf{a}}^{\top}(\tilde{x}_{i,k})\, \mathbf{w}_k$.
    Prediction: $y_i \leftarrow \sum_{k=1}^{D} f_k(x_{i,k})$.
return $\{y_i\}_{i=1}^{N}$.

column through the graph filter parameterized by shared filter coefficients. The magnitude of the filter coefficients reflects the importance of the neighbourhood. In other words, they explain which hop information plays a key role in the prediction. We also emphasize that sharing filter parameters reduces the complexity without compromising the accuracy (more details in Section 6). In the following step, for the target node i, we extract D-features and feed them to RFF layer where it undergoes the cosine transformation with the sampling parameters $\{a_m, b_m\}_{m=1}^{M}$ shared across the features. The output of the RFF layer is the M-dimension RFF vector ($\mathbf{\Phi_a}$) for each feature. Each of these undergoes a linear transformation using a single layer NN with size as number of RFF features. The final predictions are obtained depending on the downstream tasks:
For node classification: For node classification task, the predictions obtained following (8) are passed through activation function (for ex., sigmoid or softmax).
For graph classification: We obtain the representation for the complete graph by sum pooling as $\mathbf{z}_{\mathcal{G}} = \sum_{i=1}^{N} \mathbf{z}_i$. Then the prediction follows by passing representation through an activation function.
Handling multi-class datasets: If the dataset contains C-classes, we project the weights of a single layer RFF to the number of classes thereby enabling class-wise interpretability (more details in Section 6). We present detailed step by step process in Algorithm 1.

4 Theoretical Characterization and Learnable Parameters Computation

In this section, we discuss some important properties of G-NAMRFF *i.e.*, permutation equivariance and robustness to the perturbations, and then discuss parameter complexity.

Theorem 4.1 (*Permutation Equivariance*). *Let $\mathcal{P} = \{\, \mathbf{P} \in \{0,1\}^{N \times N} : \mathbf{P}^{\top}\mathbf{P} = \mathbf{P}\mathbf{P}^{\top} = \mathbf{I}_N\}$ be the set of all $N \times N$ permutation matrices. Then under the permutation of the graph Laplacian $\mathbf{L}_{\mathcal{G}}$ and node-feature matrix $\mathbf{X}$ by any $\mathbf{P} \in \mathcal{P}$, the outputs from G-NAMRFF also modifies as $\mathbf{y}_{perm} = \mathbf{P}\mathbf{y}$, where $\mathbf{y} \in \mathbb{R}^N$ is the predictions across all the nodes.*

The proof for Theorem 4.1 is relegated to Appendix A. Observe that Theorem 4.1 guarantees any simultaneous permutation of node indices in the Laplacian and feature matrix results in the same permutation of the learned embeddings. This symmetry is essential for ensuring that the model's predictions depend only on the graph's structure, not on an arbitrary ordering of its nodes.

Theorem 4.2 (*Robustness to perturbation of graph Laplacian*). *Let $\widehat{\mathbf{L}}_{\mathcal{G}} = \mathbf{L}_{\mathcal{G}} + \Delta\mathbf{L}_{\mathcal{G}}$ be the Laplacian of the perturbed graph, with $\|\Delta\mathbf{L}_{\mathcal{G}}\|_2 \leq \epsilon$, and assume that the RFF map $\mathbf{\Phi}_a(.)$ is C_{RFF}-Lipschitz continuous. Then each node prediction satisfies $|\widehat{y}_i - y_i| \leq C\,K\,\epsilon\,D\,\|\mathbf{X}\|_2$, where $C = C_{\mathrm{RFF}}\,\big(\max_k \|\mathbf{w}_k\|_2\big), K = \frac{1}{4}\|\boldsymbol{\alpha}\|_1\,(R^2-1)\left(\frac{R-1}{R+1}\right)^{R}$ are constants.*

The proof for Theorem 4.2 is relegated to Appendix A. It emphasizes the error i.e, difference between the predictions from G-NAMRFF is bounded linearly with the norm of the error term $\Delta\mathbf{L}_{\mathcal{G}}$. Hence, the

node predictions are robust to small changes in the graph structure. The proof follows by leveraging the perturbation bounds on FIR filters and then invoking the Lipschitz continuity of RFF features.

4.1 Parameter Complexity

Recall, GNAN employs a deep network f_k for each of D features plus an additional DNN for ρ to encode the graph structure. If each of these $D+1$ is an L layer feed forward neural network with H_u hidden units, the total number of parameters to be learnt are $(D+1) \times (H_u^2 \times (L-1) + (L+2) \times H_u + 1)$.

For G-NAMRFF, the only learnable parameters are filter coefficients and weights of dimension M. Therefore, total number of learnable parameters are $D \times M + R + 1$, where R is the filter order. For numerical comparison, considering the values given in GNAN [5] with $H_u = 64$, $L = 5$ and $D = 100$, for the proposed method with $M = 100$ and $R = 5$, the number of parameters are approximately $168\times$ fewer. This reduction in model size, translates to arguably smaller run times of G-NAMRFF (see discussion on run times in Appendix C.2).

5 Related Works

The existing works that are closely related to ours can be broadly classified into 3 categories as follows:

Black-box GNN **Architectures**: Graph representation learning with GNNs has become ubiquitous across a wide range of applications. Broadly speaking, existing *black-box* GNNs can be categorized into two types: (a): Message–passing models, which iteratively aggregate and update node features via learned message functions [12, 35, 11]. (b): Spectral-convolutional architectures, which perform graph filtering in the Fourier domain to capture global structure [40, 6, 20, 42, 34]. Additionally, to better weigh the influence of individual neighbors while aggregating messages, attention-based extensions such as Graph Attention Networks (GAT) [32], and more recently, graph transformer variants [39, 33] have been proposed. [33, 39, 32]. Finally, several works employ Gaussian processes over graphs to capture predictive uncertainty directly [7, 10]. [7, 10]. Despite these advances, all these approaches model a *black-box* function, and their learned embeddings must be supplemented by separate interpretability techniques.

Interpretable Additive Models: Additive modeling is the workhorse principle of interpretable architectures for tabular data since the inception of Generalized Additive Models (GAMs) [14] to recent methods such as GPAM [41] [15, 2, 24, 8]. However, these methods were developed primarily for tabular inputs, and naive application of them to graph-structured data fails to exploit the relational inductive biases inherent in the graph topology. To address this gap, [5] extends generalized additive modeling to graphs by learning node-wise contributions aggregated across neighborhoods. As discussed earlier, this approach suffers with high parameter complexity and employs aggregation schemes that do not fully capture the influence of local graph structure.

Post-hoc GNN **Explainers**: Explainable models for graph data are mostly *post-hoc* methods that interpret pretrained GNNs by identifying critical substructures or features after model training. For instance, models like GNN-Explainer [37], PG-Explainer [22] uncover the important subgraphs and corresponding node features for explaining the feature predictions, whereas Graph-LIME [17] extends LIME [27] to GNN architectures to quantify each feature contribution to a node-level decision. Recently, GRAPHTRAIL[4] enhances GNN interpretability by translating learned representations into concise, human-readable Boolean formulas. A comprehensive study of post-hoc explainable models are detailed in [31, 38].

6 Numerical Experiments and Discussion on Interpretability

In this section, we evaluate G-NAMRFF performance and demonstrate its ability to learn interpretable representations on node and graph classification tasks.

6.1 Interpretability on Node Classification Task

To begin with, we analyze the univariate functions learnt on PubMed [29] dataset with node-classification task. PubMed is a multiclass citation-network dataset where nodes are research articles

categorized into three diabetes classes: Type 1, Type 2 and gestational [5]. To reveal how individual features influence the model's predictions, we plot each learned function against its corresponding raw input values. In particular, as G-NAMRFF produces three outputs per feature (one score for each class), we plot three separate curves for each feature as shown in Fig. 3. At any given feature value, the curve with the highest output indicates the class that the feature most strongly supports.

In Fig 3(a), we show the influence of feature "children" on model output. It can be observed that model learnt a stronger contribution toward Type 1 diabetes, indicating that the presence of this term in an article increases the likelihood of it being classified under this category. This observation aligns with clinical understanding, as Type 1 diabetes is autoimmune in nature with children being susceptible[30], leading to increased mention of pediatric-related terms in relevant literature. Similarly, in Fig. 3(c), we show the influence of BMI (Body Mass Index), where it is clear that model associates predominantly with Type 2 diabetes. This clearly aligns with the clinical findings that elevated BMI is a major risk factor for Type 2 diabetes [1]. In Fig. 3(b), we show the influence of "sex (gender)," where model shows relatively lower contributions to both Type 1 and Type 2 diabetes, which is consistent with the fact that sex plays a mere role for diagnosis of Type 1 and Type 2 diabetes. However, in the case of gestational diabetes, which by definition is diagnosed during pregnancy, the model attributes higher importance.

Although these explanations merely match model prediction and interpretation with ground truth, for more challenging and high-stakes scenarios, a subject matter expert would better assess the correctness of such plots. Doing so, we can truly utilize the transparency of the model to increase the belief in such predictions and also use the feedback to improve the model further.

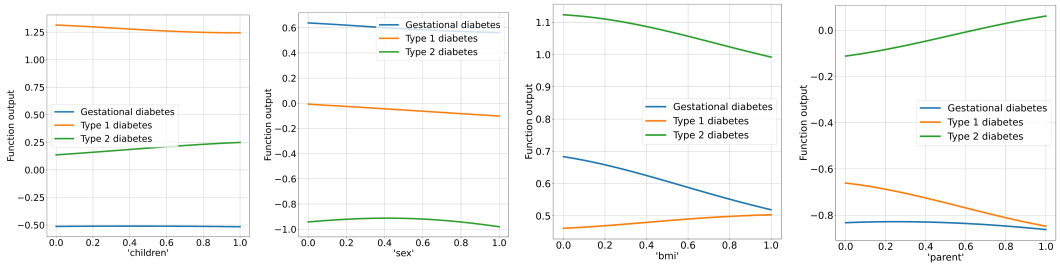


Figure 3: PubMed dataset: Univariate function outputs on different features.

6.2 Empirical Results on Node Classification Task

Having demonstrated G-NAMRFF's ability to learn interpretable representations, we empirically evaluate its performance on medium-scale and large scale datasets namely Cora, Citeseer, PubMed [36], Cornell[23], ogbn-arxiv and ogbn-products [16]. In Table 1, we report mean classification accuracy along with the standard deviation over five independent runs with different random seeds. Hyperparameter details (e.g., number of RFFs (M), scale (Θ), filter order (R), and optimizer settings) are provided in the Appendix C.1.

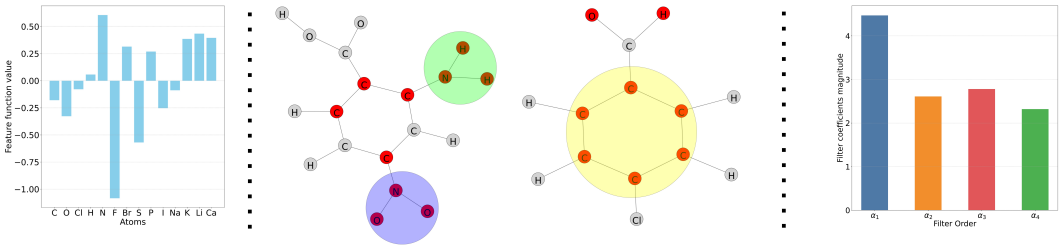
Across all six datasets, G-NAMRFF matches or exceeds the performance of state-of-the-art *black-box* GNNs, confirming that interpretability need not come at the cost of accuracy. We also observe that incorporating information from multiple hops (G-NAMRFF) outperforms the one-hop variant (G-NAMRFF ($R = 1$)), highlighting the value of deeper neighborhood aggregation. Finally, among interpretable models, G-NAMRFF outperforms the state-of-the-art models in addition to scaling effectively. In particular, on ogbn-products, both NAM and GNAN encounter out-of-memory errors, whereas G-NAMRFF trains successfully. A comprehensive study on hyperparameters is presented in the Appendix C.3.

6.3 Interpretability on Graph Classification Task

In this section, we discuss the interpretability on the graph classification task with the widely studied Mutagenicity dataset [18, 9]. The dataset includes 4,337 molecules represented as graphs and each graph is associated with binary labels. In contrast to node classification, here we present both global level and local level interpretations. Recall that we obtain the graph-level embeddings by sum pooling node embeddings. To begin with, we discuss the feature level importance.

Table 1: Performance comparison on node classification task.

Model	Cora	Citeseer	Pubmed	Cornell	ogbn-arxiv	ogbn-products
GCN [20]	81.23 ± 1.1	71.20 ± 1.7	78.50 ± 1.3	65.90 ± 0.5	71.74 ± 0.3	75.64 ± 0.3
GAT [32]	80.32 ± 2.3	70.26 ± 2.3	77.12 ± 2.4	72.50 ± 0.7	71.95 ± 0.6	79.45 ± 0.5
GraphSAGE [12]	79.94 ± 3.4	65.12 ± 1.9	78.25 ± 1.2	75.90 ± 5.0	71.49 ± 0.2	75.63 ± 0.3
Graph Transformer [39]	80.70 ± 0.5	76.00 ± 0.9	78.80 ± 1.4	70.50 ± 1.7	70.13 ± 0.5	74.74 ± 0.5
NAM [2]	51.35 ± 2.3	55.40 ± 1.9	58.16 ± 2.3	59.15 ± 2.6	56.12 ± 3.4	OOM
GPAM [41]	59.96 ± 3.2	60.30 ± 3.9	62.30 ± 3.7	60.12 ± 3.6	62.35 ± 4.2	60.13 ± 3.9
GNAN [5]	77.89 ± 5.1	65.23 ± 3.7	75.13 ± 2.4	71.76 ± 4.2	69.56 ± 0.9	OOM
G-NAMRFF ($R = 1$)	75.32 ± 1.8	67.12 ± 1.1	75.12 ± 3.8	64.12 ± 2.9	66.94 ± 1.6	55.73 ± 1.8
G-NAMRFF	79.84 ± 1.7	69.45 ± 2.5	77.30 ± 1.4	73.54 ± 4.9	70.02 ± 3.9	72.13 ± 0.4



(a) (b) (c)

Figure 4: (a). Feature level explanation on Mutagenicity dataset. (b). Local structure level explanation on Mutagenicity dataset. (c). Learnt FIR filter coefficients.

Focusing on mutagenic class, we obtain the feature level scores globally by taking the sum of each atom's contribution across the graphs. In Fig. 4(a), we present per-atom importance for the mutagenic class. Notably, Nitrogen, Oxygen, and Lithium atoms exhibit the highest positive contributions towards predicting mutagenicity. This interpretability not only helps us understand *why* the model makes a given prediction, but also directs domain experts to inspect and, if necessary, adjust the features or model. For example, if the model misclassifies a molecule, one can review the atom importance scores to assess which atomic features drove the error and representation learning can be modified accordingly.

Although feature-level importance gives insight into individual atom contributions, understanding which substructures are responsible for mutagenicity is crucial for molecular datasets. Using our model, we obtain *local explanations* by aggregating the feature-level attributions along the neighborhood to highlight the subgraph responsible for a prediction.

In Fig. 4(b), we visualize these extracted substructures for two representative molecules classified as mutagenic. In the first molecule, the nitro (NO_2) and amino (NH_2) groups emerge as the dominant substructures each exhibiting strong positive attribution toward the mutagenic label. In the second, the aromatic carbon ring itself drives the mutagenicity score. These findings align closely with prior toxicology studies [9, 37], which identify NH_2, NO_2, and polyaromatic rings as key mutagenic motifs. Importantly, our model reveals that hydrogen and oxygen atoms directly bonded to nitrogen also carry a significant contribution towards a mutagenic class, whereas isolated hydrogens on carbon rings do not play that significant role. This observation supports the global atom-importance ranking in Fig. 4(a), where nitrogen outranks hydrogen. The study presented again emphasizes the `G-NAMRFF` ability in global-level and local-level attributions. Recall that each FIR filter coefficient α_h quantifies how much the model weights information from its h-hop neighborhood. To demonstrate this on the Mutagenicity dataset, in Fig. 4(c), we plot the filter coefficients for a fixed filter order. It can be observed that there is a clear decay in magnitude as h increases, confirming that our model naturally prioritizes close neighbors and progressively down-weights more distant nodes. As explained earlier, these learned coefficients thus offer direct, interpretable insight into the receptive field of each atom.

6.4 Empirical Evaluation on Graph Classification Task

In this section, we report `G-NAMRFF` performance on several standard graph-classification benchmarks: Proteins, NCI1, Mutagenicity, MUTAG, and PTC [16]. We compare the proposed model against both *black-box* `GNNs` and interpretable models. In Table 2, we present the mean graph classifica-

Table 2: Performance comparison on graph classification task.

Model	Proteins	Mutag	Mutagenicity	NCI1	PTC
GCN [20]	70.97±4.6	68.07±6.3	75.69±0.9	66.35±1.3	56.98±5.8
GAT [32]	69.92±4.0	67.20±3.4	69.40±1.2	66.12±2.1	55.60±11.1
GraphSAGE [12]	67.35±2.3	64.12±2.4	69.25±3.9	65.56±3.9	57.12±4.9
Graph Transformer [39]	69.76±3.2	66.30±5.3	73.10±0.9	68.24±3.4	55.90±3.5
NAM [2]	62.45±4.2	63.12±9.1	67.35±2.5	57.15±1.2	54.97±7.5
GPAM [41]	65.68±4.1	64.30±8.4	65.46±2.2	53.80±2.7	52.65±8.1
GNAN [5]	59.64±2.4	67.35±3.9	66.64±4.7	50.87±1.4	55.07±5.2
G-NAMRFF (R=1)	67.83±4.4	71.20±6.7	68.98±3.4	63.65±2.8	55.65±5.6
G-NAMRFF	69.94±3.7	79.81±5.3	71.70±2.0	66.10±1.7	61.91±3.4

tion accuracies from 10-fold cross-validation, each repeated over three random seeds. The results highlights the strength of proposed model in obtaining the interpretable graph embeddings without compromising the accuracy. Detailed hyperparameter settings and ablation studies are provided in Appendix C.1 and C.3.

7 Limitations and Future Directions

While `G-NAMRFF` models the prediction output as a sum of independent feature-wise contributions, this formulation overlooks potential correlations among features. A straightforward extension that explicitly models both individual and pairwise feature interactions would substantially increase the parameter count, compromising the model efficiency. Hence designing a parameter-efficient formulation that can capture feature dependencies and higher-order correlations without sacrificing interpretability remains an important direction for future research. Further, recall that the smoothness prior imposed on the function limits its effectiveness on heterophilic graphs, where connected nodes often have dissimilar labels. Therefore, extending the current approach to better handle heterophilic data presents an interesting and promising direction for future research.

8 Conclusions

In this work, we introduced `G-NAMRFF`, a self-interpretable, lightweight additive model for graphs. We proposed a unified interpretable architecture that integrates both graph topology and node features into univariate functions via `Random Fourier Features`. Further, we provided theoretical guarantees showing that `G-NAMRFF` is robust to graph perturbations and permutational equivariant. Our analysis on parameter complexity revealed that `G-NAMRFF` requires $168\times$ fewer parameters than `GNAN`. Finally, through experiments on node and graph classification benchmarks, we demonstrated that although being light-weight and interpretable, `G-NAMRFF` does not compromise predictive performance.

References

[1] Ali Abbasi, Dorota Juszczyk, Cornelia HM van Jaarsveld, and Martin C Gulliford. Body mass index and incident type 1 and type 2 diabetes in children and young adults: a retrospective cohort study. *Journal of the Endocrine Society*, 1(5):524–537, 2017.

[2] Rishabh Agarwal, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Ben Lengerich, Rich Caruana, and Geoffrey E Hinton. Neural additive models: Interpretable machine learning with neural nets. *Advances in neural information processing systems*, 34:4699–4711, 2021.

[3] 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. *arXiv preprint arXiv:2206.09677*, 2022.

[4] Burouj Armgaan, Manthan Dalmia, Sourav Medya, and Sayan Ranu. Graphtrail: Translating gnn predictions into human-interpretable logical rules. *Advances in Neural Information Processing Systems*, 37:123443–123470, 2024.

[5] Maya Bechler-Speicher, Amir Globerson, and Ran Gilad-Bachrach. The intelligible and effective graph neural additive network. *Advances in Neural Information Processing Systems*, 37:90552–90578, 2024.

[6] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Graph neural networks with convolutional arma filters. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3496–3507, 2021.

[7] Viacheslav Borovitskiy, Iskander Azangulov, Alexander Terenin, Peter Mostowsky, Marc Deisenroth, and Nicolas Durrande. Matérn gaussian processes on graphs. In *International Conference on Artificial Intelligence and Statistics*, pages 2593–2601. PMLR, 2021.

[8] Chun-Hao Chang, Rich Caruana, and Anna Goldenberg. Node-gam: Neural generalized additive model for interpretable deep learning. *arXiv preprint arXiv:2106.01613*, 2021.

[9] Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin 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, 1991.

[10] Jinyuan Fang, Qiang Zhang, Zaiqiao Meng, and Shangsong Liang. Structure-aware random fourier kernel for graphs. *Advances in Neural Information Processing Systems*, 34:17681–17694, 2021.

[11] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. Sign: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198*, 2020.

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

[13] Vikas Hassija, Vinay Chamola, Atmesh Mahapatra, Abhinandan Singal, Divyansh Goel, Kaizhu Huang, Simone Scardapane, Indro Spinelli, Mufti Mahmud, and Amir Hussain. Interpreting black-box models: a review on explainable artificial intelligence. *Cognitive Computation*, 16 (1):45–74, 2024.

[14] Trevor Hastie and Robert Tibshirani. Generalized additive models. *Statistical science*, 1(3): 297–310, 1986.

[15] Trevor J Hastie. Generalized additive models. *Statistical models in S*, pages 249–307, 2017.

[16] 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. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[17] 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*, 35(7):6968–6972, 2022.

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

[19] Henry Kenlay, Dorina Thanou, and Xiaowen Dong. On the stability of polynomial spectral graph filters. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5350–5354. IEEE, 2020.

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

[21] Anna-Maria Lampousi, Sofia Carlsson, and Josefin E Löfvenborg. Dietary factors and risk of islet autoimmunity and type 1 diabetes: a systematic review and meta-analysis. *EBioMedicine*, 72, 2021.

[22] 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.

[23] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

[24] Filip Radenovic, Abhimanyu Dubey, and Dhruv Mahajan. Neural basis models for interpretability. *Advances in Neural Information Processing Systems*, 35:8414–8426, 2022.

[25] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, volume 20, 2007.

[26] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*. MIT press, 2006.

[27] 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.

[28] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

[29] 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.

[30] AG Unnikrishnan, Eesh Bhatia, Vijayalakshmi Bhatia, Sanjay Kumar Bhadada, Rakesh Kumar Sahay, Arun Kannan, V Kumaravel, Dipti Sarma, Bantwal Ganapathy, Nihal Thomas, et al. Type 1 diabetes versus type 2 diabetes with onset in persons younger than 20 years of age: results from an indian multicenter study. *Annals of the New York Academy of Sciences*, 1150(1): 239–244, 2008.

[31] 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.

[32] Petar Velickovi'c, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[33] Chloe Wang, Oleksii Tsepa, Jun Ma, and Bo Wang. Graph-mamba: Towards long-range graph sequence modeling with selective state spaces. *arXiv preprint arXiv:2402.00789*, 2024.

[34] Xiyuan Wang and Muhan Zhang. How powerful are spectral graph neural networks. In *International conference on machine learning*, pages 23341–23362. PMLR, 2022.

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

[36] 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.

[37] 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.

[38] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(5): 5782–5799, 2022.

[39] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. *Advances in neural information processing systems*, 32, 2019.

[40] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23, 2019.

[41] Wei Zhang, Brian Barr, and John Paisley. Gaussian process neural additive models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 16865–16872, 2024.

[42] Hao Zhu and Piotr Koniusz. Simple spectral graph convolution. In *International conference on learning representations*, 2021.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

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 abstract and introduction clearly state our two key contributions: (1) a lightweight model with far fewer parameters than the state of the art interpretable GNN model, and (2) unified self-interpretable structure via graph-aware Random Fourier Features and learnable FIR filters. These two contributions are strongly supported by the emphasis on parameter-complexity analysis.

   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 acknowledges that the proposed model obtains predictions by assuming feature-wise independence, thereby overlooking potential correlations among features. This design choice is intentional, as the primary objective is to develop a lightweight and interpretable model without compromising predictive performance. Extending the current framework to efficiently capture feature correlations while maintaining interpretability is identified as an important direction for future work.

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: [Yes]

   Justification: The outline of proofs and assumptions are discussed in the main paper whereas, complete proof for the Theorems are provided in the Appendix.

   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 detailed hyperparameter analysis and step by step process followed to generate the results are detailed in the Appendix.

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: We have included the repository link in the abstract.

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, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have followed the standard splits available from the open benchmark graph datasets and wherever they are not available we have given detailed explanation in the Appendix. Detailed hyperparameter analysis is given in the Appendix.

   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: Mean accuracies along with standard deviations are reported. We clearly mentioned the factors of variablity.

   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: All the details regarding the compute workers and runtime plots are mentioned in the Appendix.

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: Experimentation is mostly performed on open benchmark datasets which are widely acceptable.

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: [NA]

Justification: There is no direct societal impact of the work.

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: This paper doesnot have such risks.

    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: Properly acknowledged the assets used.

    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: The paper doesnot release any new assets.

    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: The current work doesnot involve any research with human subjects.

    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: Current work doesnot involve any research with human subjects.

    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.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: LLMs are only used for formatting the tables.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Appendix

We organize the content in the Appendix as follows: In section A, we present proofs that supports permutation equivariance and robustness properties of G-NAMRFF. In section B, we include an additional discussion on interpretability with node and graph classification tasks. In section C, we present an ablation studies with respect to the hyperparameters.

A Proofs for the Theoretical Characterization of G-NAMRFF

In this section, we present the proofs for the permutation equivariance and robustness properties of G-NAMRFF.

Theorem. 4.1. *(*Permutation Equivariance*) Let $\mathcal{P} = \{\, \mathbf{P} \in \{0,1\}^{N\times N} : \mathbf{P}^\top\mathbf{P} = \mathbf{P}\mathbf{P}^\top = \mathbf{I}_N\}$ be the set of all $N \times N$ permutation matrices. Then under the permutation of the graph Laplacian $\mathbf{L}_\mathcal{G}$ and node-feature matrix $\mathbf{X}$ by any $\mathbf{P} \in \mathcal{P}$, the output from G-NAMRFF also modifies as $\mathbf{y}_{perm} = \mathbf{P}\mathbf{y}$, where $\mathbf{y} \in \mathbb{R}^N$ is the predictions across all the nodes.*

Proof. Let $\mathbf{P} \in \mathcal{P}$ be the permutation matrix acting on the graph $\mathcal{G}$. Under this permutation, the graph Laplacian and feature matrix transforms as $\tilde{\mathbf{L}}_{\mathcal{G},\text{perm}} = \mathbf{P}\tilde{\mathbf{L}}_\mathcal{G}\mathbf{P}^T$ and $\mathbf{X}_{\text{perm}} = \mathbf{P}\mathbf{X}$. Let us call the prediction of the node i under the permutation as $y_{i,\text{perm}}$, whereas $\mathbf{y}_{\text{perm}}$ is obtained by stacking $y_{i,\text{perm}}$.

Under the permutations, filter output modifies as

$$\begin{aligned}\tilde{\mathbf{X}}_{\text{perm}} = \sum_{h=0}^{R}\alpha_h\tilde{\mathbf{L}}^h_{\mathcal{G},\text{perm}}\mathbf{X}_{\text{perm}} \;&= \sum_{h=0}^{R}\alpha_h\mathbf{P}\tilde{\mathbf{L}}^h_\mathcal{G}\mathbf{P}^T\mathbf{P}\mathbf{X}\\ &\overset{(a)}{=} \sum_{h=0}^{R}\alpha_h\mathbf{P}\tilde{\mathbf{L}}^h_\mathcal{G}\mathbf{X},\\ &\overset{(b)}{=} \mathbf{P}\tilde{\mathbf{X}},\end{aligned} \tag{9}$$

where (9)(a), follows from the property of permutation matrix i.e., $\mathbf{P}^T\mathbf{P} = \mathbf{I}$. Observe that from (9)$(b)$ under the permutation, filtered outputs also gets permuted. Leveraging this, we now evaluate the prediction of G-NAMRFF under permutation as

$$\begin{aligned}\mathbf{y}_{\text{perm}} &= \left[y_{1,\text{perm}};\,\ldots;\,y_{N,\text{perm}}\right],\\ &= \sum_{k=1}^{D}\underbrace{\left[\mathbf{\Phi}^T_\mathbf{a}(\tilde{x}_{1,k,\text{perm}});\ldots;\mathbf{\Phi}^T_\mathbf{a}(\tilde{x}_{N,k,\text{perm}})\right]}_{\mathbf{\Psi}_{k,\text{perm}}}\mathbf{w}_k,\\ &\overset{(a)}{=} \mathbf{P}\sum_{k=1}^{D}\left[\mathbf{\Phi}^T_\mathbf{a}(\tilde{x}_{1,k});\ldots;\mathbf{\Phi}^T_\mathbf{a}(\tilde{x}_{N,k})\right]\mathbf{w}_k,\\ &= \mathbf{P}\mathbf{y},\end{aligned} \tag{10}$$

where (10)(a), follows from (9)(b) where it can be observed that for a fixed feature k i.e., $\tilde{\mathbf{x}}_{k,\text{perm}} \in \mathbb{R}^{N\times 1} = \mathbf{P}\tilde{\mathbf{x}}_k$, therefore mappings gets reordered with $\mathbf{\Psi}_{k,perm} \in \mathbb{R}^{N\times M} = \mathbf{P}\mathbf{\Psi}_k$. Also it can be observed that $\mathbf{z}_{k,perm} = [f_k(x_{1,k}), f_k(x_{2,k}), \ldots, f_k(x_{N,k})] \in \mathbb{R}^{N\times 1} = \mathbf{P}\mathbf{z}_k$ from the permutational equivariance of $\mathbf{\Psi}_k$. □

Theorem. 4.2. *(*Robustness to perturbation of graph Laplacian*) Let $\widehat{\mathbf{L}}_\mathcal{G} = \mathbf{L}_\mathcal{G} + \Delta\mathbf{L}_\mathcal{G}$ be the Laplacian of the perturbed graph, with $\|\Delta\mathbf{L}_\mathcal{G}\|_2 \le \epsilon$, and assume that the RFF map $\mathbf{\Phi}_a(.)$ is C_{RFF}-Lipschitz continuous. Then each node prediction satisfies $|\widehat{y}_i - y_i| \le\; C\,K\,\epsilon\,D\,\|\mathbf{X}\|_2$,, where $C = C_{\text{RFF}}\left(\max_k\|\mathbf{w}_k\|_2\right), K = \frac{1}{4}\|\boldsymbol{\alpha}\|_1\left(R^2-1\right)\left(\frac{R-1}{R+1}\right)^R$ are constants.*

Proof. Before proceeding to the proof, we state the following Lemma that bounds the error between the FIR filters built with $\mathbf{L}_{\mathcal{G}}$ and $\hat{\mathbf{L}}_{\mathcal{G}}$. With slight abuse of notation from here on we represent normalized Laplacian with $\mathbf{L}_{\mathcal{G}}$ instead of $\tilde{\mathbf{L}}_{\mathcal{G}}$.

Lemma A.1. *[19] Consider a polynomial filter $\mathbf{H}(\mathbf{L}_{\mathcal{G}}) = \sum_{h=0}^{R} \alpha_h \mathbf{L}_{\mathcal{G}}^h$, and perturbed Laplacian as $\hat{\mathbf{L}}_{\mathcal{G}} = \mathbf{L}_{\mathcal{G}} + \Delta\mathbf{L}_{\mathcal{G}}$ with $\|\Delta\mathbf{L}_{\mathcal{G}}\|_2 \leq \epsilon$ then*

$$\|\mathbf{H}(\mathbf{L}_{\mathcal{G}}) - \mathbf{H}(\hat{\mathbf{L}}_{\mathcal{G}})\|_2 \leq \frac{1}{4}\|\boldsymbol{\alpha}\|_1(R^2-1)\left(\tfrac{R-1}{R+1}\right)^R \epsilon, \tag{11}$$

where $\|\boldsymbol{\alpha}\|_1 = [\alpha_0, \ldots, \alpha_R] \in \mathbb{R}^{R+1}$.

Lemma A.1 shows that the error between the outputs of two filters fed with true and perturbed Laplacian is bounded and scales linearly with the error norm. We leverage the above Lemma to bound the difference between the prediction obtained from G-NAMRFF under perturbations.

Assume that the prediction of node i under the graph perturbation as $\hat{y}_i$. To begin with we evaluate the difference between the filter outputs. Considering the filter outputs with and without perturbation of graph as $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{X}}_{\text{per}}$. We have

$$\begin{aligned} \tilde{\mathbf{X}} &= \mathbf{H}(\mathbf{L}_{\mathcal{G}})\mathbf{X}, \\ \tilde{\mathbf{X}}_{\text{per}} &= \mathbf{H}(\hat{\mathbf{L}}_{\mathcal{G}})\mathbf{X}. \end{aligned} \tag{12}$$

Then the difference between the filtered outputs are bounded as

$$\begin{aligned} \|\tilde{\mathbf{X}}_{\text{per}} - \tilde{\mathbf{X}}\|_2 &= \|\mathbf{H}(\hat{\mathbf{L}}_{\mathcal{G}})\mathbf{X} - \mathbf{H}(\mathbf{L}_{\mathcal{G}})\mathbf{X}\|_2 \\ &\overset{(a)}{\leq} \|\mathbf{H}(\hat{\mathbf{L}}_{\mathcal{G}}) - \mathbf{H}(\mathbf{L}_{\mathcal{G}})\|_2 \|\mathbf{X}\|_2 \\ &\overset{(b)}{\leq} \frac{1}{4}\|\boldsymbol{\alpha}\|_1(R^2-1)\left(\tfrac{R-1}{R+1}\right)^R \epsilon \|\mathbf{X}\|_2, \end{aligned} \tag{13}$$

where (13)(a) follows from a norm inequality and (13)(b) follows from (11). From (13)(b) it is clear that the difference of filter outputs is bounded linearly with the energy in the error term. Leveraging this we now evaluate the difference in the prediction output. We index the i,k element of $\tilde{\mathbf{X}}_{\text{per}}$ with $\tilde{x}_{i,k,\text{per}}$. Then,

$$\begin{aligned} |\hat{y}_i - y_i| &= |\sum_{k=1}^{D} \boldsymbol{\Phi}_a^T(\tilde{x}_{i,k,\text{per}})\mathbf{w}_k - \sum_{k=1}^{D} \boldsymbol{\Phi}_a^T(\tilde{x}_{i,k})\mathbf{w}_k| \\ &\leq \sum_{k=1}^{D} \|\boldsymbol{\Phi}_a^T(\tilde{x}_{i,k,\text{per}}) - \boldsymbol{\Phi}_a^T(\tilde{x}_{i,k})\|_2 \|\mathbf{w}_k\|_2 \\ &\overset{(a)}{\leq} C_{\text{RFF}} \sum_{k=1}^{D} |\tilde{x}_{i,k,\text{per}} - \tilde{x}_{i,k}| \|\mathbf{w}_k\|_2 \\ &\overset{(b)}{\leq} \sum_{k=1}^{D} C_{\text{RFF}} \max_k\{\|\mathbf{w}_k\|_2\} \underbrace{\frac{1}{4}\|\boldsymbol{\alpha}\|_1(R^2-1)\left(\tfrac{R-1}{R+1}\right)^R}_{K} \epsilon \|\mathbf{X}\|_2 \\ &\overset{(c)}{\leq} CK\epsilon D \|\mathbf{X}\|_2, \end{aligned} \tag{14}$$

where (14)(a) follows from Lipschitz continuity of $\boldsymbol{\Phi}_{\mathbf{a}}$, (14)(b), follows from (13)(b). Whereas, (14)(c) follows by considering the $C = C_{\text{RFF}} \max_k \|\mathbf{w}_k\|_2$. From (14)(c), it is clear that difference in the node predictions scales linearly with the norm of the error term $\Delta\mathbf{L}_{\mathcal{G}}$. □

B Additional Discussion on Interpretability

B.1 Interpretability on Node Classification Task

In this section, we present additional examples of the learned univariate functions on the PubMed dataset for the node classification task. Specifically, in Fig. 5, we illustrate the univariate function

outputs for the features `i.e.,` keywords as *insulin*, *fat*, *liver*, and *diet*. Recall that at any given feature value, the curve with the highest output indicates the class that the feature most strongly supports.

In Fig. 5a, we examine the influence of the presence of *diet* keyword in the document on the model's predictions. It can be observed that the model associates higher output with type 2 diabetes, suggesting that *diet* is a strong predictor for this class. This observation aligns well with existing medical literature, which indicates that type 2 diabetes is heavily influenced by dietary factors, whereas type 1 diabetes, being an autoimmune condition is less affected by diet [21]. Consistent with this, the model assigns less importance to type 1 diabetes for this feature. In Fig. 5b, we plot the influence of *fat* keyword in the document where it is clear that it contributes less significantly to the prediction of type 1 diabetes, further supporting the model's ability to capture medically relevant patterns. Similarly, in Fig. 5c and Fig. 5d we plot the impact of these features on model prediction where it is clear that presence of this word aligns prediction towards type 2 and gestational diabetes. Although for illustration purposes we have presented few examples one can follow a similar procedure to obtain the contribution of different features on different datasets.

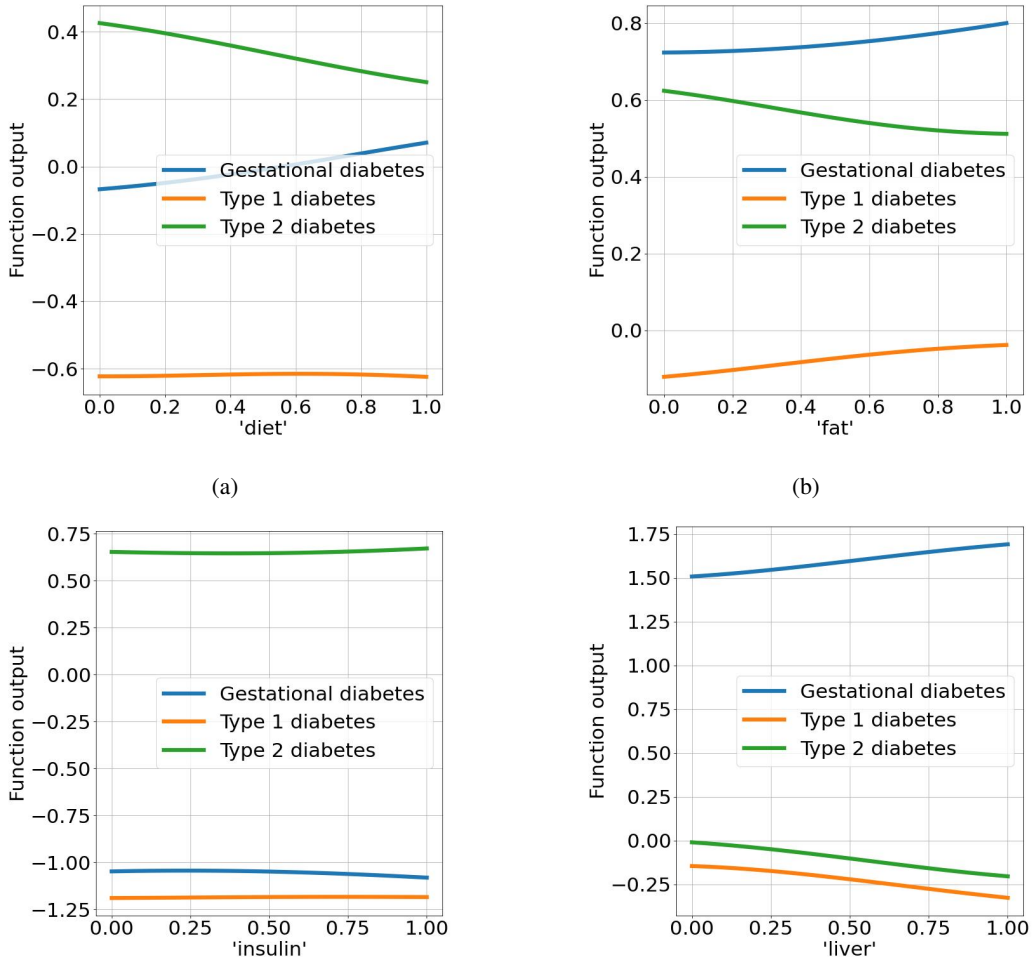
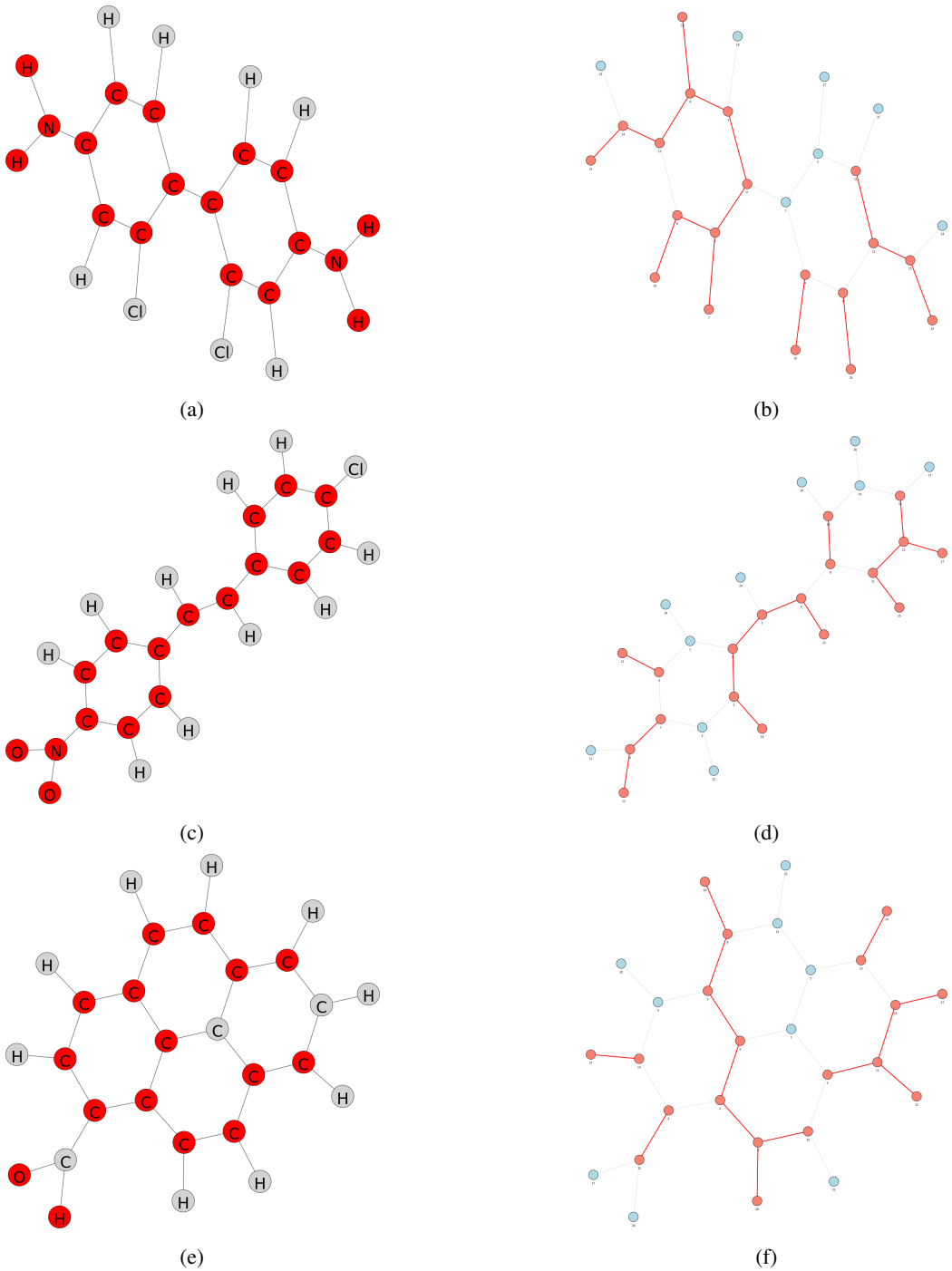


Figure 5: PubMed dataset: Univariate function outputs on features.

B.2 Interpretability on Graph Classification Task

In this section, we include more discussion on interpretability on the mutagenicity dataset. Here the main target is to give local level (structural) explanations and also compare them with the `GNN-Explainer` [37] which is a *post-hoc* explainer.

With the focus on mutagenic class, we give structure level explanations for the model prediction. In Figs. 6a, 6c and 6e we present three example molecules that are predicted as mutagenic and highlight predicted substructures responsible for the mutagenic prediction. It is clear that the NH_2, NO_2 and aromatic rings act as key contributors to mutagenic class. These explanations align with the existing studies which advocate the presence of these substructures highly influences the mutagenicity [9]. For a fair comparison we present the substructures revealed from `GNN-Explainer` in Fig. 6b, 6d and 6f. It can be observed that compared to `GNN-Explainer`, our proposed model identifies NH_2, NO_2 and aromatic rings more precisely.



(a) (b) (c) (d) (e) (f)

Figure 6: Mutagenicity dataset: Local level explanations.

C Hyperparameter Details and Ablation Studies

In this section, we first present the hyperparameter details used in our node and graph classification tasks. Then we analyze how each of these hyperparameters affects `G-NAMRFF` performance across datasets.

C.1 Hyperparameter Details

We have conducted all the experiments using an *NVIDIA* A30 GPU. In particular, we set the tuning range as follows: Number of RFFs (M)- $\{20, 40, 50, 100, 200\}$, filter order (R) - $[1, 7]$, kernel width (Θ)- $[1.0, 4.0]$, learning rate (Lr)- $[1e-4, 2e-1]$ and weight decay (Wd)- $[1e-5, 1e-2]$. The hyperparameter configurations that yielded the best validation performance are detailed in Table 3. For node classification tasks, we follow the standard dataset splits as specified in [16, 20, 23]. In the case of graph classification tasks, where no standard splits are available, we employ 10-fold cross-validation. All models are trained for 1000 epochs using the *AdamW* optimizer with randomly initialized parameters. The specific learning rates and weight decay values used across different experiments are also reported in Table 3. For node classification, we report the mean classification accuracy computed over five independent random seeds. For graph classification, we report the mean classification accuracy across 10 folds, with each fold averaged over three random seeds.

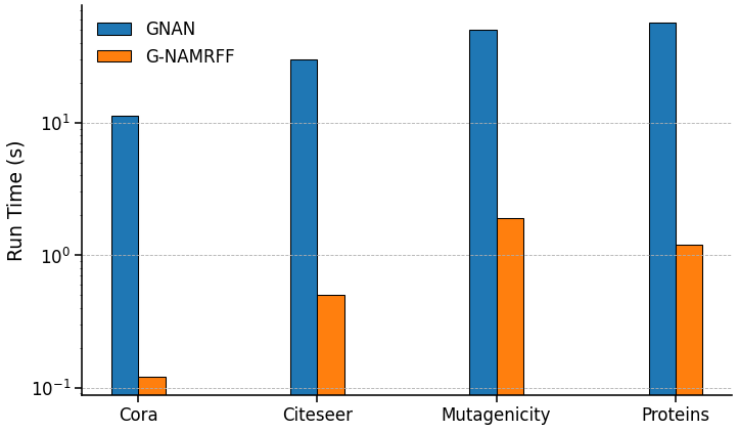


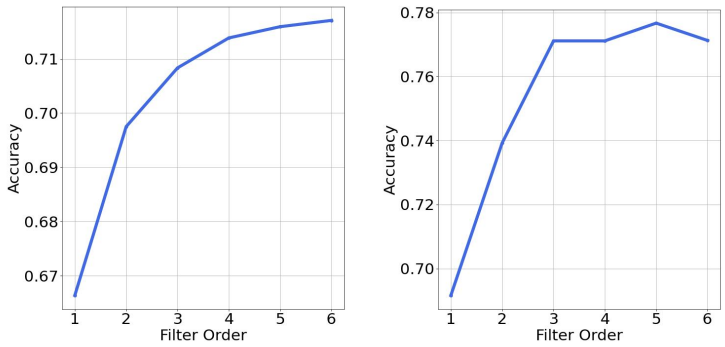
Figure 7: Run time comparison.

C.2 Run time analysis

In this section, we compare the runtime performance of `G-NAMRFF`, against `GNAN`, which is the only existing *glass-box* `GNN` architecture. In Fig. 7, we present the per-epoch runtime comparison on both node and graph classification tasks. Notably, it can be observed that even when `GNAN` is configured

Dataset	Number of RFFs (M)	Filter Order (R)	Kernel width (Θ)	Learning rate (Lr)	Weight Decay (Wd)
Cora	100	7	2.2	1.8e-1	2.4e-4
Citeseer	100	5	2.5	5.1e-1	3.1e-4
Pubmed	100	5	3.8	2.7e-2	4e-4
Cornell	100	6	2.9	7.3e-2	3.7e-4
ogbn-arxiv	100	4	3.4	5.2e-2	1.3e-4
ogbn-products	40	5	2.3	2.9e-2	5e-4
Proteins	50	5	1.9	3.2e-3	5e-3
Mutag	200	7	1.2	3.7e-2	5e-3
Mutagenicity	200	6	1.4	1.1e-2	5e-4
NCI1	50	7	3.7	1.6e-2	5e-4
PTC	50	5	1.7	2.9e-3	2.5e-5

Table 3: Hyperparameter details.



(a) Mutagenicity (b) Pubmed

Figure 8: Classification accuracy against filter order.

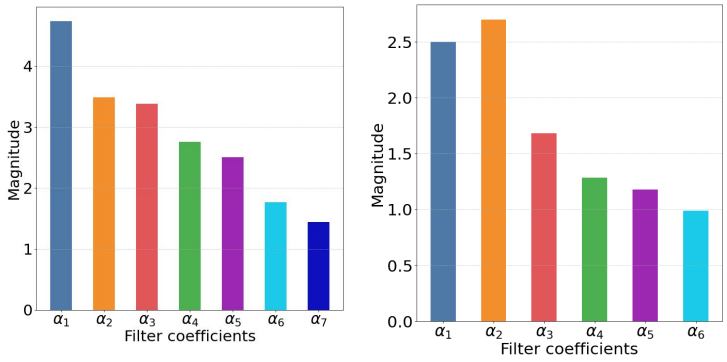
with a relatively small hidden dimension of 32, it still requires 10 to 100 times more runtime per epoch compared to `G-NAMRFF`.

Furthermore, it is worth emphasizing that on small-scale datasets such as Citeseer and Pubmed, increasing the hidden dimension to 64 can already result in out-of-memory (OOM) errors when training `GNAN`. On large-scale datasets like ogbn-products, training `GNAN` becomes entirely infeasible due to excessive memory requirements.

C.3 Ablation studies

In this section, we analyze the impact of hyperparameters on the model performance. Recall the key hyperparameters in `G-NAMRFF` includes the filter order (R) and number of `RFF` (M).
Filter Order: In Fig. 8, we illustrate the effect of the filter order on classification accuracy for two tasks: node classification on the Pubmed dataset and graph classification on the Mutagenicity dataset. Recall that a filter of order R aggregates information from R-hop neighborhoods in the graph. It can be observed that as the filter order increases, the model is able to incorporate information from larger receptive fields, leading to an initial improvement in classification accuracy. However, beyond a certain point, further increasing the filter order causes the accuracy to saturate or decline, likely due to oversmoothing. Notably, we observe that using higher-order filters ($R > 1$) significantly improves performance compared to $R = 1$, highlighting the benefit of aggregating information from larger graph neighborhoods.

To further highlight the relative importance of different node neighborhoods, we present the magnitudes of the learned filter coefficients for a fixed filter order in Fig. 9, using the Mutagenicity and Cornell datasets. It can be observed that the model assigns higher weights to closer neighborhoods, emphasizing the greater importance of immediate node interactions in the classification task.

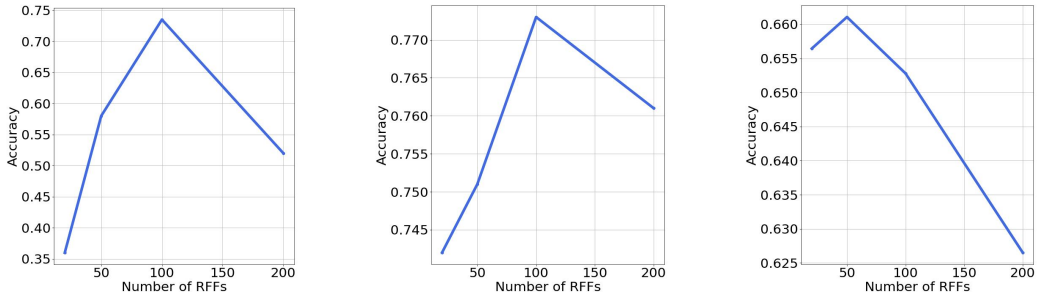


(a) Mutag (b) Cornell

Figure 9: Filter coefficients magnitude against filter order.

Random Fourier Features:

We analyze the effect of the number of `RFFs` on classification accuracy. Figures. 10a, 10b, and 10c show the performance variation on the Cornell, PubMed, and NCI1 datasets, respectively. As the number of random Fourier features (M) increases, the classification accuracy initially improves due to better kernel approximation. However, beyond a certain point, performance drops as the model may begin to overfit with larger M. Across all datasets, we observe that good performance can be achieved with relatively small values of M, which translates to a lower number of learnable parameters. Recall that the total number of parameters in `G-NAMRFF` is $D \times M + R + 1$. As observed M being small the model remains lightweight while being effective.



(a) Cornell (b) Pubmed (c) NCI1

Figure 10: Classification accuracy vs number of RFFs.