Hierarchical k-GNN Explanations via Generative Interpretation Networks

Comparing 1-GNN and 1-2-GNN for Model-Level Graph Classification Interpretability

[Author Name]

February 2026

Abstract

Graph Neural Networks (GNNs) are powerful tools for graph classification, yet their decision processes remain opaque. Higher-order k-GNNs, which operate on k-element subsets of nodes, provably increase expressive power beyond the Weisfeiler–Leman hierarchy, but it is unclear whether this additional expressiveness translates into more interpretable explanations. We investigate this question by comparing a standard 1-GNN with a hierarchical 1-2-GNN using the GIN-Graph framework for model-level explanation generation. Our pipeline trains k-GNN classifiers on the MUTAG and PROTEINS benchmarks, then uses a Wasserstein GAN with gradient penalty to generate class-representative explanation graphs guided by the pretrained classifier. We introduce a fully differentiable dense wrapper that maintains gradient flow through both node features and adjacency matrices for hierarchical models. On MUTAG, the 1-2-GNN achieves higher classification accuracy (89.5% vs. 84.2%) and produces explanations with substantially higher validity rates (88% vs. 78%), validation scores (0.80 vs. 0.69), and embedding similarity (0.995 vs. 0.964). On PROTEINS, the 1-GNN outperforms in both classification and explanation quality, suggesting that the benefit of higher-order message passing is dataset-dependent. We discuss limitations including undertrained checkpoints, single-seed evaluation, and the granularity problem in model-level explanations.

Contents

1 Introduction

Graph Neural Networks (GNNs) have become the standard approach for learning on graph-structured data, achieving state-of-the-art results on tasks ranging from molecular property prediction to social network analysis [Kipf and Welling(2017), Xu et al.(2019)]. Despite their effectiveness, GNNs suffer from the same interpretability challenges as other deep learning models: their predictions are difficult to explain in human-understandable terms.

The expressiveness of standard message-passing GNNs is bounded by the 1-dimensional Weisfeiler–Leman (1-WL) graph isomorphism test [Xu et al.(2019), Morris et al.(2019)]. Morris et al. [Morris et al.(2019)] proposed k-GNNs that operate on k-element subsets of nodes, achieving expressiveness equivalent to the k-WL test. In theory, higher-order k-GNNs can distinguish graph structures that standard GNNs cannot.

A natural question arises: *does this additional structural expressiveness translate into better model-level explanations?* If a model captures richer structural patterns, the explanations it produces—representative graphs that characterise each class—should be more informative and structurally valid.

We investigate this question using the GIN-Graph framework [Sun et al.(2025)], which generates model-level explanation graphs through a GAN-based approach guided by a pretrained classifier. Specifically, we compare:

- A **1-GNN**: standard node-level message passing, equivalent to 1-WL;
- A **1-2-GNN**: hierarchical architecture combining node-level (1-GNN) and pairwise (2-GNN) message passing, achieving 2-WL expressiveness.

Our contributions are:

1. A fully differentiable dense wrapper enabling GIN-Graph training with hierarchical k-GNN classifiers, maintaining gradient flow through adjacency matrices at both the 1-GNN and 2-GNN levels.
2. A corrected validation metric that computes actual cosine similarity between generated and real graph embeddings, replacing the original proxy of using prediction probability.
3. An empirical comparison of explanation quality across two benchmark datasets (MUTAG, PROTEINS), showing that the benefit of higher-order message passing for interpretability is dataset-dependent.

2 Background

2.1 Graph Neural Networks

A graph $G = (V, E)$ consists of a node set V and edge set $E \subseteq V \times V$. Each node $v \in V$ may carry a feature vector $\mathbf{x}_v \in \mathbb{R}^d$. GNNs learn node representations through iterative message passing: at each layer t, a node aggregates information from its neighbours:

$$\mathbf{h}_v^{(t)} = \mathrm{UPDATE}\big(\mathbf{h}_v^{(t-1)},\ \mathrm{AGGREGATE}\big(\{\mathbf{h}_u^{(t-1)} : u \in \mathcal{N}(v)\}\big)\big)\,, \tag{1}$$

where $\mathcal{N}(v)$ denotes the neighbourhood of v and $\mathbf{h}_v^{(0)} = \mathbf{x}_v$.

For graph-level tasks, a readout function pools all node embeddings into a single graph representation:

$$\mathbf{h}_G = \mathrm{READOUT}\big(\{\mathbf{h}_v^{(T)} : v \in V\}\big)\,. \tag{2}$$

2.2 The k-WL Hierarchy and k-GNNs

The Weisfeiler–Leman (WL) graph isomorphism test iteratively refines colour labels based on neighbourhood multisets. The k-WL test operates on k-tuples of nodes, achieving strictly increasing discriminative power: for all $k \geq 2$, $(k+1)$-WL is strictly more powerful than k-WL [Cai et al.(1992)].

Morris et al. [Morris et al.(2019)] showed that standard message-passing GNNs are at most as powerful as the 1-WL test, and proposed k-GNNs that achieve k-WL expressiveness by operating on k-element subsets ("k-sets") of nodes. For a k-set $s = \{v_1, \ldots, v_k\} \subseteq V$, the *local neighbourhood* is defined as:

$$\mathcal{N}_L(s) = \{s' \subseteq V : |s'| = k,\ |s \triangle s'| = 2,\ \text{the differing elements are adjacent in } G\}\,, \tag{3}$$

where $s \triangle s'$ denotes the symmetric difference. Intuitively, a neighbour of s is obtained by replacing one element with an adjacent node.

2.3 GNN Explainability

GNN explanation methods can be categorised along two axes [Yuan et al.(2022)]:

Instance-level methods explain individual predictions. GNNExplainer [Ying et al.(2019)] identifies a subgraph and feature subset that maximise mutual information with the prediction. PGExplainer [Luo et al.(2020)] learns a global edge mask predictor.

Model-level methods explain the overall behaviour of a trained model by generating representative input patterns for each class. XGNN [Yuan et al.(2020)] uses reinforcement learning to grow graphs that maximise class prediction. GIN-Graph [Sun et al.(2025)] employs a generative adversarial approach, combining a WGAN-GP for structural realism with classifier guidance for class specificity.

We adopt the model-level approach because it directly reveals what structural patterns each model associates with a given class, enabling comparison between architectures.

2.4 GIN-Graph Overview

GIN-Graph [Sun et al.(2025)] trains a generator $\mathcal{G}$ to produce graphs that are simultaneously realistic (via a WGAN-GP discriminator) and class-specific (via a pretrained GNN classifier Φ). The generator maps noise $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to a graph $(\tilde{A}, \tilde{X})$ using Gumbel-Softmax for differentiable discrete sampling. The training objective balances adversarial and classification losses through a dynamic weighting schedule.

3 Method

3.1 1-GNN Architecture

Our 1-GNN implements standard message passing with separate self and neighbour transformations. At layer t:

$$\mathbf{h}_v^{(t)} = \sigma\left(\mathbf{h}_v^{(t-1)}\mathbf{W}_1^{(t)} + \sum_{u\in\mathcal{N}(v)} \mathbf{h}_u^{(t-1)}\mathbf{W}_2^{(t)}\right), \tag{4}$$

where $\mathbf{W}_1^{(t)}, \mathbf{W}_2^{(t)} \in \mathbb{R}^{d_t \times d_{t+1}}$ are learnable weight matrices and σ is the ReLU activation. After $T = 3$ layers, the graph embedding is obtained via sum pooling:

$$\mathbf{h}_G^{(1)} = \sum_{v\in V} \mathbf{h}_v^{(T)}. \tag{5}$$

3.2 2-GNN and the Hierarchical 1-2-GNN

The 2-GNN operates on 2-sets (unordered node pairs). For each pair $s = \{u, v\}$ with $u < v$, the initial feature incorporates the 1-GNN embeddings and an isomorphism type indicator:

$$\mathbf{f}_s^{(0)} = \left[\mathbf{h}_u^{(T)} \parallel \mathbf{h}_v^{(T)} \parallel \mathbb{1}[(u, v) \in E]\right] \in \mathbb{R}^{2d_T+1}, \tag{6}$$

where $\parallel$ denotes concatenation and $\mathbb{1}[(u, v) \in E]$ is the isomorphism type indicating whether the pair is connected.

The 2-GNN neighbourhood follows the Morris et al. definition. For $s = \{u, v\}$:

$$\mathcal{N}_L(s) = \{\{v, w\} : w \neq u,\ (u, w) \in E\} \cup \{\{u, w\} : w \neq v,\ (v, w) \in E\}. \tag{7}$$

That is, we replace one element of the pair with an adjacent node. Message passing on 2-sets then follows the same form as Equation (4), operating on $\mathbf{f}_s^{(t)}$ with 2-GNN weight matrices.

The **hierarchical 1-2-GNN** first runs the 1-GNN (Equation 4) for $T_1 = 3$ layers, then uses the resulting node embeddings to initialise 2-GNN features (Equation 6), and runs $T_2 = 2$ layers of 2-GNN message passing. The final graph representation concatenates both readouts:

$$\mathbf{h}_G = \left[\mathbf{h}_G^{(1)} \parallel \mathbf{h}_G^{(2)}\right], \quad \text{where } \mathbf{h}_G^{(2)} = \sum_{s\in\binom{V}{2}} \mathbf{f}_s^{(T_2)}. \tag{8}$$

For large graphs, the number of pairs $\binom{n}{2}$ can be prohibitive. We randomly sample up to 5000 pairs per graph to maintain tractability.

3.3 GIN-Graph Generator

The generator $\mathcal{G}$ maps a latent vector $\mathbf{z} \in \mathbb{R}^{d_z}$ to a graph $(\tilde{A}, \tilde{X})$ through a backbone MLP followed by separate heads for the adjacency matrix and node features:

$$\mathbf{h} = \text{MLP}(\mathbf{z}), \tag{9}$$

$$\tilde{A} = \text{GumbelSoftmax}(\text{sym}(\mathbf{h}\mathbf{W}_A),\ \tau) \in \{0,1\}^{N\times N}, \tag{10}$$

$$\tilde{X} = \text{GumbelSoftmax}(\mathbf{h}\mathbf{W}_X,\ \tau) \in \{0,1\}^{N\times D}, \tag{11}$$

where N is the maximum number of nodes, D is the number of node feature types, $\text{sym}(\cdot)$ symmetrises the matrix, and τ is the temperature parameter.

Gumbel-Softmax. To enable gradient-based optimisation through discrete graph structures, we use the Gumbel-Softmax trick [Jang et al.(2017)]. For a categorical distribution with logits $\boldsymbol{\pi}$:

$$y_i = \frac{\exp((\pi_i + g_i)/\tau)}{\sum_j \exp((\pi_j + g_j)/\tau)}, \quad g_i \sim \text{Gumbel}(0,1). \tag{12}$$

At low temperature $\tau \to 0$, the output approaches a one-hot vector. We use the straight-through estimator: forward pass uses hard (discrete) samples, backward pass uses the continuous relaxation.

For the adjacency matrix, each entry is a binary choice (edge/no-edge). We sample the upper triangle and mirror it to ensure symmetry.

3.4 Training Objective

The total loss combines adversarial and GNN-guidance terms:

$$\mathcal{L} = (1 - \lambda(t))\,\mathcal{L}_{\text{GAN}} + \lambda(t)\,\mathcal{L}_{\text{GNN}}, \tag{13}$$

where $\lambda(t)$ is a dynamic weight that increases over training.

WGAN-GP loss. The discriminator $\mathcal{D}$ is trained with the Wasserstein distance and gradient penalty [Gulrajani et al.(2017)]:

$$\mathcal{L}_{\text{GAN}} = \mathbb{E}_{\tilde{G}\sim\mathcal{G}}[\mathcal{D}(\tilde{G})] - \mathbb{E}_{G\sim p_{\text{data}}}[\mathcal{D}(G)] + \lambda_{\text{GP}}\,\mathbb{E}_{\hat{G}}[(\|\nabla_{\hat{G}}\mathcal{D}(\hat{G})\|_2 - 1)^2], \tag{14}$$

where $\hat{G}$ is a random interpolation between real and generated graphs, and $\lambda_{\text{GP}} = 10$.

GNN guidance loss. We maximise the pretrained classifier's confidence on the target class c:

$$\mathcal{L}_{\text{GNN}} = -\log p_\Phi(y = c \mid \tilde{G}). \tag{15}$$

Dynamic weighting. Following the GIN-Graph paper, $\lambda(t)$ follows a sigmoid schedule:

$$\lambda(t) = \lambda_{\min} + (\lambda_{\max} - \lambda_{\min}) \cdot \sigma\left(k \cdot \left(\frac{2(t/T - p)}{1 - p} - 1\right)\right), \tag{16}$$

where T is the total iterations, $p = 0.4$ controls when the transition begins, $k = 10$ controls its steepness, and $\sigma(\cdot)$ is the logistic sigmoid. Early training ($t \ll pT$) focuses on graph realism ($\lambda \approx 0$); late training shifts to class specificity ($\lambda \to 1$).

3.5 Dense Wrapper for Differentiable k-GNN Inference

A key technical challenge is that the pretrained k-GNN uses sparse PyTorch Geometric operations, while the generator outputs dense adjacency matrices requiring gradient flow. We implement a fully differentiable dense wrapper.

Dense 1-GNN. Equation (4) is rewritten in matrix form for a batch of graphs:

$$\mathbf{H}^{(t)} = \sigma\Big(\mathbf{H}^{(t-1)}\mathbf{W}_1^{(t)} + \mathbf{A}\,\mathbf{H}^{(t-1)}\mathbf{W}_2^{(t)}\Big), \tag{17}$$

where $\mathbf{A} \in [0,1]^{N\times N}$ is the continuous adjacency matrix from the generator. The matrix multiplication $\mathbf{A}\,\mathbf{H}^{(t-1)}$ replaces the sparse message-passing operation and is fully differentiable with respect to $\mathbf{A}$.

Dense 2-GNN. For the 2-GNN component, we construct pair features as a dense $N \times N \times (2d_T + 1)$ tensor:

$$\mathbf{F}[i,j] = \big[\mathbf{h}_{\min(i,j)} \parallel \mathbf{h}_{\max(i,j)} \parallel A_{ij}\big], \tag{18}$$

where the canonical ordering ensures consistent feature construction. The neighbourhood aggregation uses Einstein summation:

$$\mathrm{agg}_1[i,j] = \sum_w A_{iw} \cdot \mathbf{g}[j,w], \tag{19}$$

$$\mathrm{agg}_2[i,j] = \sum_w A_{jw} \cdot \mathbf{g}[i,w], \tag{20}$$

where $\mathbf{g} = \mathbf{F}\mathbf{W}_2$ are the transformed pair features. These two aggregation cases correspond to replacing the first or second element of the pair, following the Morris et al. neighbourhood definition (Equation 7). Both operations are fully differentiable through the continuous adjacency matrix.

3.6 Validation Score

We evaluate generated explanations using a composite validation score:

$$v = (s \cdot p \cdot d)^{1/3}, \tag{21}$$

comprising three components:

Embedding similarity (s). The cosine similarity between the generated graph's embedding (computed via the dense wrapper) and a class centroid (mean embedding of real target-class graphs computed via the sparse k-GNN):

$$s = \frac{\mathbf{h}_{\tilde{G}} \cdot \boldsymbol{\mu}_c}{\|\mathbf{h}_{\tilde{G}}\|\,\|\boldsymbol{\mu}_c\|}, \quad \boldsymbol{\mu}_c = \frac{1}{|\mathcal{G}_c|}\sum_{G\in\mathcal{G}_c} \mathbf{h}_G, \tag{22}$$

where $\mathcal{G}_c$ is the set of real graphs in class c.

Prediction probability (p). The softmax probability assigned to the target class by the pretrained classifier.

Degree score (d). A Gaussian kernel measuring how realistic the graph's average degree is relative to the target class:

$$d = \exp\left(-\frac{(\bar{d} - \mu_d)^2}{2\sigma_d^2}\right), \tag{23}$$

where $\bar{d}$ is the average degree of the generated graph, and μ_d, σ_d are the mean and standard deviation of average degrees for real graphs of the target class.

A generated graph is considered **valid** if its average degree falls within 3 standard deviations of the class mean and its validation score exceeds 0.5.

Granularity. We also report a granularity metric $\kappa = 1 - \min(1, n/\bar{n}_c)$, where n is the number of active nodes and $\bar{n}_c$ is the average number of nodes in class c. Values near 0 indicate coarse-grained explanations (full-graph-sized), while values near 1 indicate fine-grained substructure highlights.

4 Experimental Setup

4.1 Datasets

We evaluate on two standard graph classification benchmarks (Table 1).

Table 1: Dataset statistics. GIN Gen Size is the maximum number of nodes used for explanation generation.

Dataset	Graphs	Node Feats	Max Nodes	GIN Gen	Task
MUTAG	188	7 (atom types)	28	28	Mutagenicity
PROTEINS	1113	3 (sec. struct.)	620	50	Enzyme vs. non-enzyme

MUTAG [Debnath et al.(1991)] contains 188 molecular graphs labelled by mutagenic effect on *Salmonella typhimurium*. Nodes represent atoms (C, N, O, F, I, Cl, Br) and edges represent chemical bonds. The two classes are *Mutagen* (class 0, 125 graphs) and *Non-Mutagen* (class 1, 63 graphs).

PROTEINS [Borgwardt et al.(2005)] contains 1113 protein graphs where nodes are secondary structure elements (helix, sheet, coil/turn) connected if they are neighbours in the amino acid sequence or within a distance threshold in 3D space. The classes are *Non-Enzyme* (class 0) and *Enzyme* (class 1). For GIN-Graph generation, we use a maximum of 50 nodes (the median protein graph has $\sim$26 nodes).

4.2 Model Configuration

All models use a hidden dimension of 64. The 1-GNN uses 3 message-passing layers; the 1-2-GNN uses 3 layers for the 1-GNN component and 2 layers for the 2-GNN component. Both use dropout of 0.5 and are trained with Adam (lr = 0.01) for 100 epochs. Data is split 80/20 into train/test with a fixed seed of 42.

The GIN-Graph generator uses a latent dimension of 32, hidden dimension of 128, and is trained for 300 epochs with Adam (lr = 0.001). WGAN-GP uses $\lambda_{\text{GP}} = 10$ and $n_{\text{critic}} = 1$. The Gumbel-Softmax temperature is $\tau = 1.0$ during training and $\tau = 0.1$ during evaluation for sharper discrete outputs.

4.3 Evaluation Protocol

For each model-dataset-class combination, we generate 100 explanation graphs and evaluate them using the validation score (Equation 21). We report:

- **Validity rate**: fraction of explanations passing degree and score thresholds;
- **Mean validation score**: averaged over all 100 generated graphs;
- **Mean valid score**: averaged over valid explanations only;
- **Component scores**: embedding similarity (s), prediction probability (p), degree score (d);
- **Granularity**: how fine-grained the explanations are.

5 Results

5.1 k-GNN Classification Performance

Table 2 summarises the classification results. On MUTAG, the 1-2-GNN outperforms the 1-GNN by 5.3 percentage points, consistent with the theoretical advantage of higher-order message passing on molecular graphs with rich local structure. On PROTEINS, the result is reversed: the 1-GNN achieves 78.0% while the 1-2-GNN reaches only 65.0%, below the majority-class baseline. The 1-2-GNN has 2.3× more parameters, and on the relatively simple node features of PROTEINS (3 types vs. 7), this additional capacity may lead to overfitting.

Table 2: k-GNN classification results. Best test accuracy reported over all epochs.

Dataset	Model	Params	Train Acc	Test Acc	Best Acc
MUTAG	1-GNN	21,570	86.7%	76.3%	84.2%
MUTAG	1-2-GNN	50,370	92.7%	84.2%	89.5%
PROTEINS	1-GNN	21,058	76.6%	77.1%	78.0%
PROTEINS	1-2-GNN	49,858	58.8%	62.8%	65.0%

5.2 GIN-Graph Explanation Quality: MUTAG

Table 3 presents the GIN-Graph results on MUTAG. The fair comparison is on **class 1 (Non-Mutagen)**, where both models were fully trained for 300 epochs.

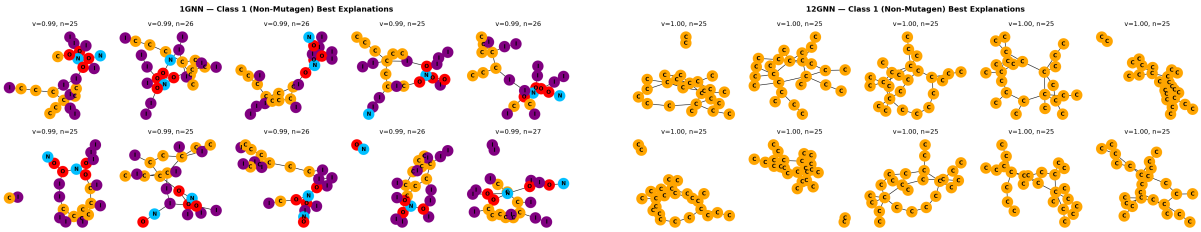
On class 1, the 1-2-GNN produces explanations with higher validity (88% vs. 78%), higher validation scores (0.799 vs. 0.687), and notably higher degree scores (0.638 vs. 0.486), indicating more structurally realistic graphs. The embedding similarity is near-perfect for both models (0.995 vs. 0.964), with the 1-2-GNN slightly closer to the class centroid.

Figure 1 shows the top-ranked explanation graphs for both models on MUTAG class 1.

The top 1-2-GNN explanations achieve near-perfect validation scores (0.998), with consistent graph sizes (25 nodes, 28 edges). In contrast, the 1-GNN explanations, while still high quality (top score 0.991), show slightly more structural variation.

Table 3: GIN-Graph explanation quality on MUTAG. †: only 49 training epochs (needs retraining).

Class	**Model**	**Epochs**	**Valid%**	**Val Score**	s	p	d
0 (Mutagen)	1-GNN†	49	20%	0.197	0.491	0.995	0.144
0 (Mutagen)	1-2-GNN	300	29%	0.256	0.531	1.000	0.202
1 (Non-Mutagen)	1-GNN	300	78%	0.687	0.964	1.000	0.486
1 (Non-Mutagen)	1-2-GNN	300	**88%**	**0.799**	**0.995**	1.000	**0.638**



(a) 1-GNN explanations for Non-Mutagen (b) 1-2-GNN explanations for Non-Mutagen

Figure 1: Top-ranked GIN-Graph explanations on MUTAG class 1 (Non-Mutagen). Node colours indicate atom types (see Figure 2).

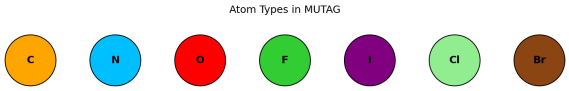


Figure 2: MUTAG atom type colour legend. Atoms: C (orange), N (blue), O (red), F (green), I (purple), Cl (light green), Br (brown).

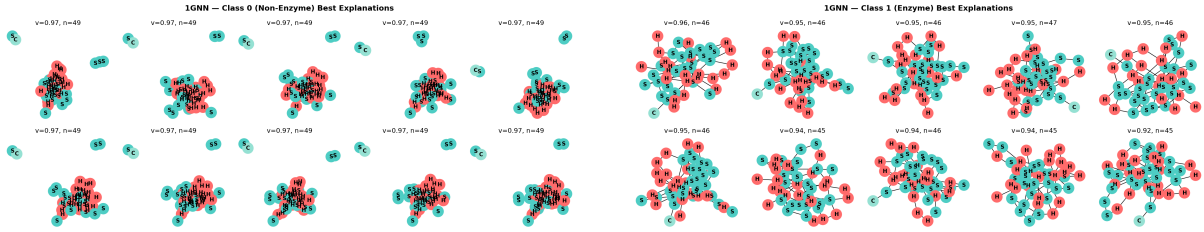
5.3 GIN-Graph Explanation Quality: PROTEINS

Table 4 presents results on PROTEINS. Here the 1-GNN produces substantially better explanations across both classes, consistent with its stronger classification performance.

Table 4: GIN-Graph explanation quality on PROTEINS. †: only 29 training epochs (needs retraining).

Class	Model	Epochs	Valid%	Val Score	s	p	d
0 (Non-Enzyme)	1-GNN	300	**100%**	**0.967**	0.998	0.916	**0.990**
0 (Non-Enzyme)	1-2-GNN†	29	99%	0.768	0.998	0.590	0.797
1 (Enzyme)	1-GNN	300	**100%**	**0.883**	0.703	1.000	**0.985**
1 (Enzyme)	1-2-GNN†	29	71%	0.568	0.999	0.410	0.571

The 1-GNN achieves 100% validity on both classes with validation scores of 0.967 and 0.883. The 1-2-GNN results are compromised by undertrained checkpoints (only 29 epochs vs. the intended 300), but the low prediction probabilities (0.590 and 0.410) reflect the weak classifier performance (65.0% accuracy).



(a) 1-GNN explanations for Non-Enzyme

(b) 1-GNN explanations for Enzyme

Figure 3: Top-ranked GIN-Graph explanations on PROTEINS (1-GNN). Node colours indicate secondary structure types: helix, sheet, coil/turn.

5.4 Metric Decomposition

Table 5 provides a detailed comparison of the three validation score components for the fully trained configurations.

Table 5: Validation score decomposition for fully trained GIN-Graph models (300 epochs).

Configuration	Model	s (emb. sim)	p (pred. prob)	d (degree)
MUTAG class 1	1-GNN	0.964	1.000	0.486
MUTAG class 1	1-2-GNN	0.995	1.000	0.638
PROTEINS class 0	1-GNN	0.998	0.916	0.990
PROTEINS class 1	1-GNN	0.703	1.000	0.985

The degree score is the primary differentiator between models on MUTAG: the 1-2-GNN generates graphs whose average degree better matches the target class distribution (0.638 vs. 0.486). Both models achieve near-perfect prediction probability ($p \approx 1$), so this

component does not discriminate. The embedding similarity, while high for both, is closer to 1.0 for the 1-2-GNN (0.995 vs. 0.964).

On PROTEINS, the 1-GNN shows an interesting asymmetry: class 0 (Non-Enzyme) has near-perfect embedding similarity (0.998) while class 1 (Enzyme) has lower similarity (0.703). This suggests that the Enzyme class has a more complex embedding manifold that the generator cannot fully capture.

5.5 Training Dynamics

Figure 4 shows a representative training curve from the MUTAG 1-2-GNN class 1 experiment.

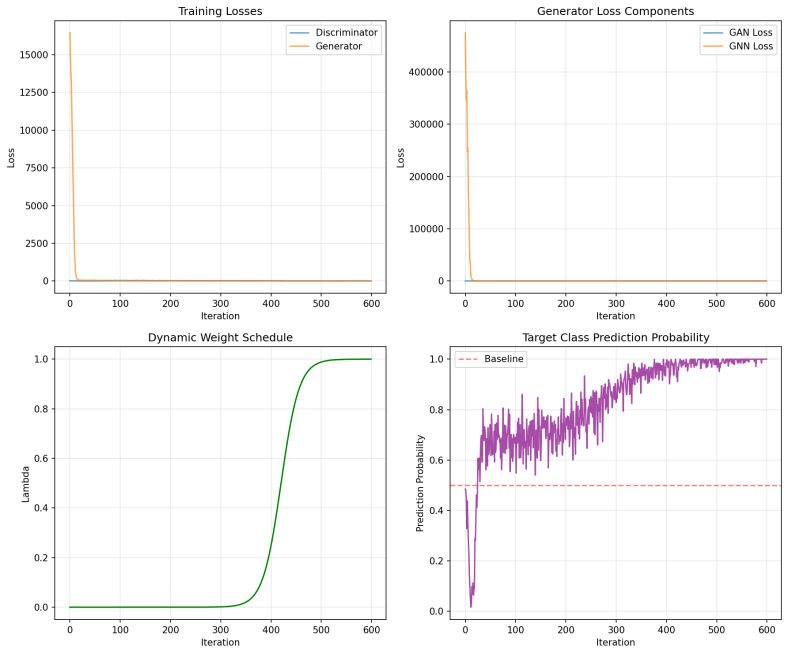


Figure 4: Training curves for GIN-Graph on MUTAG 1-2-GNN class 1, showing the evolution of generator loss, discriminator loss, and GNN guidance loss over 300 epochs.

6 Discussion

6.1 Does Higher-Order Message Passing Improve Explanations?

Our results suggest a nuanced answer: *it depends on the dataset and the classifier's performance.*

On MUTAG, the 1-2-GNN both classifies better and produces better explanations. The pairwise message passing captures bond-level patterns in molecular graphs that the 1-GNN misses, and the GIN-Graph generator leverages this richer representation to produce more structurally valid class-representative graphs. The 10 percentage point improvement in validity (88% vs. 78%) and the 0.11 improvement in validation score (0.80 vs. 0.69) are substantial.

On PROTEINS, the situation is reversed. The 1-2-GNN fails to learn effective classification (65.0%, below the 1-GNN's 78.0%), and consequently, the GIN-Graph generator cannot produce meaningful explanations from a weak classifier. This illustrates a fundamental coupling: *explanation quality is bounded by classifier quality.* A model that does not understand the data cannot produce informative explanations, regardless of its theoretical expressiveness.

6.2 The Granularity Problem

All generated explanations have granularity near 0, meaning they are as large as typical graphs in the dataset. This is a known limitation of the GIN-Graph approach: the generator produces graphs of a fixed maximum size N, and the Gumbel-Softmax sampling tends to activate most nodes. For MUTAG ($N = 28$, average graph $\sim$18 nodes), the explanations use 20–27 nodes. For PROTEINS ($N = 50$, average graph $\sim$26 nodes), they use 45–50 nodes.

Model-level explanations are inherently coarse-grained: they represent "what a typical class member looks like" rather than "which substructure drives the prediction." For fine-grained explanations, instance-level methods like GNNExplainer would be more appropriate.

6.3 Limitations

Several limitations qualify our findings:

Undertrained checkpoints. The MUTAG 1-GNN class 0 GIN was only trained for 49 epochs, and the PROTEINS 1-2-GNN GIN for only 29 epochs. These incomplete runs make cross-model comparison on those configurations unreliable.

Single seed. All experiments use a single data split (seed 42) without cross-validation. The small size of MUTAG (188 graphs, 38 test) means that test accuracy fluctuates by several percentage points across splits.

No cross-validation. Best test accuracy is reported as the maximum over epochs (early stopping by oracle), which overestimates generalisation performance.

Classifier dependence. GIN-Graph explanations are fundamentally constrained by classifier quality. The PROTEINS 1-2-GNN comparison is confounded because the classifier itself underperforms, making it impossible to isolate the effect of higher-order message passing on explanation quality.

Evaluation metrics. The validation score $v = (s \cdot p \cdot d)^{1/3}$ can be dominated by a single low component. The degree score, which measures structural realism via average degree alone, is a coarse proxy that does not capture finer structural properties like motif frequencies or degree distributions.

6.4 Future Work

Several directions could strengthen this investigation:

- Retrain all GIN-Graph checkpoints to completion for fair cross-model comparison.

- Use cross-validation and multiple random seeds for robust accuracy estimates.
- Investigate why the 1-2-GNN underperforms on PROTEINS (potential overfitting, hyperparameter sensitivity, or dataset characteristics).
- Compare with instance-level explanation methods (GNNExplainer, PGExplainer) to assess whether higher-order models also improve local explanations.
- Extend to the 1-2-3-GNN architecture, which adds 3-set (triplet) message passing for even higher expressiveness.
- Incorporate richer structural metrics beyond average degree, such as motif counts, clustering coefficients, or graph spectral properties.

7 Conclusion

We compared a standard 1-GNN with a hierarchical 1-2-GNN for model-level explanation generation using the GIN-Graph framework. On MUTAG, the 1-2-GNN produces both better classification (89.5% vs. 84.2%) and better explanations (88% validity, 0.80 validation score vs. 78% validity, 0.69 validation score), supporting the hypothesis that higher-order message passing improves interpretability. On PROTEINS, the 1-GNN dominates, highlighting that explanation quality is fundamentally tied to classifier performance, and that the benefits of higher-order models are dataset-dependent.

Our fully differentiable dense wrapper enables, for the first time, GIN-Graph explanation generation for hierarchical k-GNN architectures while maintaining gradient flow through both node features and adjacency matrices. This technical contribution opens the door to studying interpretability across the k-WL expressiveness hierarchy.

References

[Borgwardt et al.(2005)] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56, 2005.

[Cai et al.(1992)] J.-Y. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

[Debnath et al.(1991)] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. *J. Med. Chem.*, 34(2):786–797, 1991.

[Gulrajani et al.(2017)] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of Wasserstein GANs. In *NeurIPS*, 2017.

[Jang et al.(2017)] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with Gumbel-softmax. In *ICLR*, 2017.

[Kipf and Welling(2017)] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Luo et al.(2020)] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang. Parameterized explainer for graph neural network. In *NeurIPS*, 2020.

[Morris et al.(2019)] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *AAAI*, 2019.

[Sun et al.(2025)] J. Sun, S. Pei, F. Zhang, and N. V. Chawla. GIN-Graph: A generative interpretation network for model-level explanation of graph neural networks. *Neurocomputing*, 2025.

[Xu et al.(2019)] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[Ying et al.(2019)] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. GNNExplainer: Generating explanations for graph neural networks. In *NeurIPS*, 2019.

[Yuan et al.(2020)] H. Yuan, J. Tang, X. Hu, and S. Ji. XGNN: Towards model-level explanations of graph neural networks. In *KDD*, 2020.

[Yuan et al.(2022)] H. Yuan, H. Yu, S. Gui, and S. Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE TPAMI*, 45(5):5782–5799, 2022.