

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, both models achieve identical classification accuracy (89.5%), yet the 1-2-GNN produces explanations with higher validity rates (82% vs. 78%), validation scores (0.735 vs. 0.681), and degree realism (0.565 vs. 0.490), demonstrating that higher-order expressiveness improves explanation quality independently of classification performance. On PROTEINS, both models achieve comparable classification (79.8% vs. 78.9%), but each excels at explaining a different class: the 1-GNN dominates Non-Enzyme explanations while the 1-2-GNN achieves higher Enzyme validity. Cross-model classification of 500 generated graphs per class reveals that the two architectures learn fundamentally different decision boundaries—on PROTEINS, graphs generated for one model are systematically misclassified by the other. These results suggest that the benefit of higher-order message passing for interpretability is dataset-dependent, and that different architectures can develop incompatible internal representations of graph structure.

Contents

1	Introduction	4
2	Background	4
2.1	Graph Neural Networks	4
2.2	The k -WL Hierarchy and k -GNNs	5
2.3	GIN-Graph	6
3	Method	6
3.1	1-GNN Architecture	6
3.2	2-GNN and the Hierarchical 1-2-GNN	7
3.3	GIN-Graph Generator	8
3.4	Training Objective	9
3.5	Dense Wrapper for Differentiable k -GNN Inference	9
3.6	Validation Score	11
4	Experimental Setup	11
4.1	Datasets	11
4.2	Model Configuration	12
4.3	Evaluation Protocol	13
5	Results	13
5.1	k -GNN Classification Performance	13
5.2	GIN-Graph Explanation Quality: MUTAG	13
5.3	GIN-Graph Explanation Quality: PROTEINS	14
5.4	Metric Decomposition	19
5.5	Training Dynamics	19
5.6	Generated Dataset Comparison	20
5.6.1	Structural Fidelity	21
5.6.2	Cross-Model Classification	21
5.6.3	Embedding Space Structure	22
6	Discussion	24
6.1	Does Higher-Order Message Passing Improve Explanations?	24
6.2	Qualitative Differences in Generated Explanations	24
6.3	The Granularity Problem	25
6.4	Limitations	25
6.5	Future Work	26

1 Introduction

Graph Neural Networks (GNNs) have become the standard approach for learning on graph-structured data, achieving strong results on tasks ranging from molecular property prediction to social network analysis. 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 [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. Model-level explanations aim to answer the question “what does a typical graph of class c look like according to the model?” by generating new graphs that maximise the model’s confidence for that class while remaining structurally realistic. This contrasts with instance-level methods that explain individual predictions by identifying important subgraphs.

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 expressiveness beyond 1-WL.

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 with $|V| = n$ and an edge set $E \subseteq V \times V$. Each node $v \in V$ carries a feature vector $\mathbf{x}_v \in \mathbb{R}^d$. We denote the adjacency matrix as

$\mathbf{A} \in \{0, 1\}^{n \times n}$ where $A_{uv} = 1$ if $(u, v) \in E$, and the feature matrix as $\mathbf{X} \in \mathbb{R}^{n \times d}$ where row v is \mathbf{x}_v .

GNNs learn node representations through iterative *message passing*. At each layer t , a node updates its representation by aggregating information from its neighbours:

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

where $\mathcal{N}(v) = \{u \in V : (u, v) \in E\}$ denotes the neighbourhood of v , $\mathbf{h}_v^{(0)} = \mathbf{x}_v$ is the initial node feature, and $\{\{\cdot\}\}$ denotes a multiset.

The choice of UPDATE and AGGREGATE functions determines the specific GNN variant. In our implementation, these are parameterised by separate linear transformations for the self-feature and the aggregated neighbour features. After T layers of message passing, a *readout* function pools all node embeddings into a single graph-level representation:

$$\mathbf{h}_G = \text{READOUT}(\{\mathbf{h}_v^{(T)} : v \in V\}). \quad (2)$$

Common choices for READOUT include sum, mean, and max pooling over the node set. We use sum pooling throughout this work, as it preserves information about both the features and the number of nodes.

2.2 The k -WL Hierarchy and k -GNNs

The *Weisfeiler–Leman* (WL) graph isomorphism test is a classical algorithm that iteratively refines node colour labels based on the multiset of neighbour colours. Two graphs are distinguished if, after some number of iterations, they produce different multisets of colours. The 1-WL test operates on individual nodes.

A fundamental result in GNN theory is that standard message-passing GNNs are *at most* as powerful as the 1-WL test [Morris et al.(2019)]: if 1-WL cannot distinguish two graphs, no message-passing GNN can either. This means there exist structurally different graphs (e.g., certain regular graphs) that no standard GNN can tell apart.

The k -WL test generalises this by operating 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)] proposed k -GNNs that achieve this higher expressiveness by performing message passing on k -element subsets of nodes (“ k -sets”) rather than individual nodes.

For a k -set $s = \{v_1, \dots, v_k\} \subseteq V$, the *local neighbourhood* is:

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

where $s \Delta s'$ denotes the symmetric difference. In words: a neighbour of s is obtained by replacing exactly one element of s with a new node that is adjacent (in the original graph G) to the removed node.

Example. Consider a 2-set $s = \{u, v\}$. Its neighbours are all pairs $\{v, w\}$ where $(u, w) \in E$ (replacing u with adjacent w) and all pairs $\{u, w\}$ where $(v, w) \in E$ (replacing v with adjacent w). This means the 2-GNN can capture pairwise relationships and bond-level patterns that are invisible to a 1-GNN operating on individual nodes.

2.3 GIN-Graph

GIN-Graph [Sun et al.(2025)] is a model-level explanation method that generates class-representative graphs using a generative adversarial approach. The key idea is to train a generator \mathcal{G} to produce graphs that satisfy two objectives simultaneously:

1. **Realism**: generated graphs should be structurally similar to real graphs from the dataset, enforced by a WGAN-GP discriminator.
2. **Class specificity**: generated graphs should be confidently classified as the target class by the pretrained GNN, enforced by a classification guidance loss.

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 the two losses through a dynamic weighting schedule that shifts focus from realism (early training) to class specificity (late training).

3 Method

This section details the mathematical formulations of our two classifier architectures (1-GNN and 1-2-GNN), the GIN-Graph generator, the training procedure, and the differentiable dense wrapper that bridges them.

3.1 1-GNN Architecture

Our 1-GNN implements message passing with separate transformations for self-features and neighbour-aggregated features. At layer t , the update rule for node v is:

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

where $\mathbf{W}_1^{(t)}, \mathbf{W}_2^{(t)} \in \mathbb{R}^{d_{t-1} \times d_t}$ are learnable weight matrices, σ is the ReLU activation function $\sigma(x) = \max(0, x)$, and $d_0 = d$ (the input feature dimension). The self-transformation $\mathbf{h}_v^{(t-1)} \mathbf{W}_1^{(t)}$ allows the model to retain and transform the node’s own representation, while the neighbour term $\sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(t-1)} \mathbf{W}_2^{(t)}$ aggregates structural context.

This can equivalently be written in matrix form for the entire graph:

$$\mathbf{H}^{(t)} = \sigma \left(\mathbf{H}^{(t-1)} \mathbf{W}_1^{(t)} + \mathbf{A} \mathbf{H}^{(t-1)} \mathbf{W}_2^{(t)} \right), \quad (5)$$

where $\mathbf{H}^{(t)} \in \mathbb{R}^{n \times d_t}$ is the matrix of all node representations at layer t , and $\mathbf{A} \in \{0, 1\}^{n \times n}$ is the adjacency matrix. The matrix product $\mathbf{A} \mathbf{H}^{(t-1)}$ computes the sum of neighbour features for every node simultaneously.

After $T = 3$ layers with hidden dimension $d_1 = d_2 = d_3 = 64$, the graph embedding is obtained via sum pooling:

$$\mathbf{h}_G^{(1)} = \sum_{v \in V} \mathbf{h}_v^{(T)} \in \mathbb{R}^{d_T}. \quad (6)$$

This embedding is fed through a two-layer classifier MLP with ReLU activation and dropout:

$$\hat{y} = \text{softmax}\left(\mathbf{W}_o \sigma(\mathbf{W}_c \mathbf{h}_G^{(1)})\right), \quad (7)$$

where $\mathbf{W}_c \in \mathbb{R}^{d_T \times d_T}$ and $\mathbf{W}_o \in \mathbb{R}^{d_T \times C}$ with C being the number of classes.

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

The 2-GNN operates on 2-sets (unordered node pairs) rather than individual nodes. For each pair $s = \{u, v\}$ with $u < v$ (canonical ordering), the initial feature vector incorporates the 1-GNN embeddings of both nodes and an *isomorphism type* indicator:

$$\mathbf{f}_s^{(0)} = [\mathbf{h}_u^{(T)} \parallel \mathbf{h}_v^{(T)} \parallel \mathbb{K}[(u, v) \in E]] \in \mathbb{R}^{2d_T+1}, \quad (8)$$

where \parallel denotes vector concatenation. The isomorphism type $\mathbb{K}[(u, v) \in E] \in \{0, 1\}$ indicates whether u and v are connected by an edge. This is crucial because it allows the 2-GNN to distinguish between pairs of connected nodes and pairs of unconnected nodes—a distinction that carries structural meaning (e.g., in a molecule, bonded vs. non-bonded atom pairs).

Message passing on 2-sets follows the neighbourhood definition from Equation (3). For a 2-set $s = \{u, v\}$, the local neighbourhood decomposes into two cases:

$$\mathcal{N}_L(\{u, v\}) = \underbrace{\{\{v, w\} : w \neq u, (u, w) \in E\}}_{\text{Case 1: replace } u \text{ with } w} \cup \underbrace{\{\{u, w\} : w \neq v, (v, w) \in E\}}_{\text{Case 2: replace } v \text{ with } w}. \quad (9)$$

In Case 1, we remove u from the pair and add a new node w that must be adjacent to u in the original graph. In Case 2, we remove v and add w adjacent to v . This ensures that neighbourhood relationships in the 2-GNN reflect the edge structure of the underlying graph.

The 2-GNN layer update rule mirrors the 1-GNN structure:

$$\mathbf{f}_s^{(\ell+1)} = \sigma \left(\mathbf{f}_s^{(\ell)} \mathbf{W}_3^{(\ell)} + \sum_{s' \in \mathcal{N}_L(s)} \mathbf{f}_{s'}^{(\ell)} \mathbf{W}_4^{(\ell)} \right), \quad (10)$$

where $\mathbf{W}_3^{(\ell)}, \mathbf{W}_4^{(\ell)}$ are the 2-GNN weight matrices at layer ℓ . We apply $T_2 = 2$ layers.

The **hierarchical 1-2-GNN** combines both levels. It first runs the 1-GNN (Equation 4) for $T_1 = 3$ layers to obtain node embeddings, then uses these embeddings to initialise 2-GNN features (Equation 8), and runs $T_2 = 2$ layers of 2-GNN message passing. The final graph representation concatenates both readouts:

$$\mathbf{h}_G = [\mathbf{h}_G^{(1)} \parallel \mathbf{h}_G^{(2)}] \in \mathbb{R}^{2d_T}, \quad (11)$$

where the 1-GNN readout is $\mathbf{h}_G^{(1)} = \sum_{v \in V} \mathbf{h}_v^{(T_1)}$ and the 2-GNN readout pools over all pairs:

$$\mathbf{h}_G^{(2)} = \sum_{\{u, v\} \in \binom{V}{2}} \mathbf{f}_{\{u, v\}}^{(T_2)} \in \mathbb{R}^{d_T}. \quad (12)$$

The concatenated embedding \mathbf{h}_G is then fed through the same classifier structure as Equation (7), but with input dimension $2d_T$ instead of d_T .

Scalability. The number of 2-sets grows as $\binom{n}{2} = \frac{n(n-1)}{2}$, which becomes prohibitive for large graphs. For PROTEINS (up to 620 nodes, yielding up to 191,890 pairs), we process all pairs using a numba-accelerated sparse kernel with CSR adjacency representation, achieving efficient $O(\text{degree})$ neighbour lookup instead of $O(N)$ dense masks. This enables exact computation over the full pairwise structure without sampling.

3.3 GIN-Graph Generator

The generator \mathcal{G} maps a latent noise vector $\mathbf{z} \in \mathbb{R}^{d_z}$ to a graph (\tilde{A}, \tilde{X}) through a backbone MLP followed by separate output heads:

$$\mathbf{h} = \text{LeakyReLU}(\mathbf{W}_2 \text{LeakyReLU}(\mathbf{W}_1 \mathbf{z})) \in \mathbb{R}^{2d_h}, \quad (13)$$

$$\tilde{A}_{\text{raw}} = \text{sym}(\text{reshape}(\mathbf{h} \mathbf{W}_A, N \times N)), \quad (14)$$

$$\tilde{X}_{\text{raw}} = \text{reshape}(\mathbf{h} \mathbf{W}_X, N \times D), \quad (15)$$

where $d_z = 32$ is the latent dimension, $d_h = 128$ is the hidden dimension, N is the maximum number of nodes, D is the number of node feature types, and $\text{sym}(\mathbf{M}) = \frac{1}{2}(\mathbf{M} + \mathbf{M}^\top)$ ensures the adjacency logits are symmetric (undirected graphs).

Gumbel-Softmax. Graphs are inherently discrete objects: edges are either present or absent, and node types are categorical. Standard gradient-based optimisation cannot backpropagate through discrete sampling. The Gumbel-Softmax trick [Jang et al.(2017)] provides a differentiable relaxation.

For a categorical distribution with unnormalised log-probabilities (logits) π_1, \dots, π_K , the Gumbel-Softmax sample is:

$$y_i = \frac{\exp((\pi_i + g_i)/\tau)}{\sum_{j=1}^K \exp((\pi_j + g_j)/\tau)}, \quad g_i = -\log(-\log(u_i)), \quad u_i \sim \text{Uniform}(0, 1), \quad (16)$$

where $\tau > 0$ is a temperature parameter. As $\tau \rightarrow 0$, the output approaches a one-hot vector (hard categorical sample); as $\tau \rightarrow \infty$, it approaches a uniform distribution. During training, we use the *straight-through estimator*: the forward pass produces hard (discrete) samples via $\arg \max$, but the backward pass uses the continuous Gumbel-Softmax gradients. This gives the generator discrete outputs while maintaining gradient flow.

For the **adjacency matrix**, each potential edge (i, j) is a binary choice (edge/no-edge). We construct 2-class logits $[\tilde{A}_{\text{raw}}[i, j], -\tilde{A}_{\text{raw}}[i, j]]$ and apply Gumbel-Softmax with $K = 2$. To guarantee symmetry ($\tilde{A}_{ij} = \tilde{A}_{ji}$), we sample only the upper triangle and mirror it:

$$\tilde{A}_{\text{upper}} = \text{triu}(\text{GumbelSoftmax}(\tilde{A}_{\text{raw}}, \tau)), \quad \tilde{A} = \tilde{A}_{\text{upper}} + \tilde{A}_{\text{upper}}^\top. \quad (17)$$

For **node features**, each node has D possible types (e.g., 7 atom types for MUTAG). We apply Gumbel-Softmax with $K = D$ to produce one-hot feature vectors:

$$\tilde{X}[v, :] = \text{GumbelSoftmax}(\tilde{X}_{\text{raw}}[v, :], \tau) \in \{0, 1\}^D. \quad (18)$$

The temperature $\tau = 1.0$ during training (allowing exploration) and $\tau = 0.1$ during evaluation (producing sharper, more discrete outputs).

3.4 Training Objective

The total generator loss combines adversarial and GNN-guidance terms:

$$\mathcal{L}_G = (1 - \lambda(t)) \mathcal{L}_{\text{GAN}} + \lambda(t) \mathcal{L}_{\text{GNN}}, \quad (19)$$

where $\lambda(t) \in [0, 1]$ is a dynamic weight that increases over training.

WGAN-GP loss. We use a Wasserstein GAN with gradient penalty [Gulrajani et al.(2017)]. The discriminator \mathcal{D} (a GNN-based network) estimates the Wasserstein distance between real and generated graph distributions. The discriminator loss is:

$$\mathcal{L}_D = \underbrace{\mathbb{E}_{\tilde{G} \sim \mathcal{G}}[\mathcal{D}(\tilde{G})]}_{\text{score on fake}} - \underbrace{\mathbb{E}_{G \sim p_{\text{data}}}[\mathcal{D}(G)]}_{\text{score on real}} + \underbrace{\lambda_{\text{GP}} \mathbb{E}_{\hat{G}}[(\|\nabla_{\hat{G}} \mathcal{D}(\hat{G})\|_2 - 1)^2]}_{\text{gradient penalty}}, \quad (20)$$

where $\hat{G} = \epsilon G + (1 - \epsilon)\tilde{G}$ with $\epsilon \sim \text{Uniform}(0, 1)$ is a random interpolation, and $\lambda_{\text{GP}} = 10$. The gradient penalty enforces the Lipschitz constraint on \mathcal{D} , stabilising training. The generator’s adversarial loss is simply:

$$\mathcal{L}_{\text{GAN}} = -\mathbb{E}_{\tilde{G} \sim \mathcal{G}}[\mathcal{D}(\tilde{G})]. \quad (21)$$

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

$$\mathcal{L}_{\text{GNN}} = -\log p_{\Phi}(y = c \mid \tilde{G}), \quad (22)$$

where $p_{\Phi}(y = c \mid \tilde{G})$ is the softmax probability assigned to class c by the pretrained k -GNN Φ . Minimising this loss pushes the generator to produce graphs that the classifier recognises as belonging to class c .

Dynamic weighting. The balance parameter $\lambda(t)$ follows a sigmoid schedule [Sun et al.(2025)]:

$$\lambda(t) = \lambda_{\min} + (\lambda_{\max} - \lambda_{\min}) \cdot \sigma\left(k \cdot \left(\frac{2(t/T - p)}{1 - p} - 1\right)\right), \quad (23)$$

where T is the total number of training iterations, $p = 0.4$ is the fraction of training before λ begins increasing, $k = 10$ controls the steepness of the transition, and $\sigma(x) = 1/(1 + e^{-x})$ is the logistic sigmoid.

The intuition is as follows. When $t \ll pT$, the sigmoid argument is strongly negative, giving $\lambda(t) \approx 0$: the generator focuses on learning realistic graph structure (GAN loss dominates). As t passes pT , $\lambda(t)$ increases rapidly toward 1: the generator shifts to producing class-specific patterns (GNN loss dominates). Without this schedule, the generator would collapse to structurally degenerate graphs that fool the classifier but look nothing like real molecules or proteins.

3.5 Dense Wrapper for Differentiable k -GNN Inference

A key technical challenge is that the pretrained k -GNN uses sparse PyTorch Geometric message-passing operations (scatter-add on edge lists), while the generator outputs dense adjacency matrices $\tilde{A} \in [0, 1]^{N \times N}$ that require gradient flow for backpropagation. We implement a fully differentiable *dense wrapper* that re-implements the k -GNN forward pass using dense matrix operations while reusing the pretrained weights (which are frozen).

Dense 1-GNN. The node-level update (Equation 5) translates directly to dense batched computation. For a batch of B graphs with node features $\mathbf{X} \in \mathbb{R}^{B \times N \times d}$ and adjacency $\mathbf{A} \in [0, 1]^{B \times N \times N}$:

$$\mathbf{H}^{(t)} = \sigma\left(\mathbf{H}^{(t-1)}\mathbf{W}_1^{(t)} + \mathbf{A}\mathbf{H}^{(t-1)}\mathbf{W}_2^{(t)}\right), \quad (24)$$

where the batch dimension is broadcast. The matrix product $\mathbf{A}\mathbf{H}^{(t-1)}$ replaces the sparse scatter-add operation and is fully differentiable with respect to the continuous \mathbf{A} . Crucially, the weights $\mathbf{W}_1^{(t)}, \mathbf{W}_2^{(t)}$ are copied from the pretrained sparse model and kept frozen.

Dense 2-GNN. The 2-GNN component requires constructing pair features and aggregating over 2-set neighbourhoods, both as dense operations. We represent all $\binom{N}{2}$ pair features as an $N \times N$ tensor (using only the upper triangle for the canonical pairs).

First, we construct the pair features from the 1-GNN node embeddings $\mathbf{H}^{(T_1)} \in \mathbb{R}^{B \times N \times d_T}$:

$$\mathbf{F}[i, j] = [\mathbf{h}_{\min(i, j)} \parallel \mathbf{h}_{\max(i, j)} \parallel A_{ij}] \in \mathbb{R}^{2d_T+1}, \quad (25)$$

where the canonical ordering (min, max) ensures that $\mathbf{F}[i, j] = \mathbf{F}[j, i]$, matching the unordered pair semantics. The isomorphism type A_{ij} is the continuous adjacency value from the generator, maintaining differentiability.

For the neighbourhood aggregation, recall from Equation (9) that there are two cases. Using the transformed features $\mathbf{g} = \mathbf{F}\mathbf{W}_4^{(\ell)} \in \mathbb{R}^{B \times N \times N \times d_T}$, the aggregation for pair (i, j) is:

$$\text{agg}_1[i, j] = \sum_{\substack{w=1 \\ w \neq j}}^N A_{iw} \cdot \mathbf{g}[j, w] \quad (\text{replace } i \text{ with } w \text{ adjacent to } i), \quad (26)$$

$$\text{agg}_2[i, j] = \sum_{\substack{w=1 \\ w \neq i}}^N A_{jw} \cdot \mathbf{g}[i, w] \quad (\text{replace } j \text{ with } w \text{ adjacent to } j). \quad (27)$$

The constraints $w \neq j$ in agg_1 and $w \neq i$ in agg_2 are essential: without them, the summation includes the self-pair $\{j, j\}$ or $\{i, i\}$, which has cardinality 1 and is not a valid 2-set. The self-loop exclusion ($A_{ii} = 0$) already removes $w = i$ from agg_1 and $w = j$ from agg_2 , but the opposite boundary terms must be explicitly masked. In practice, we zero out the diagonal of \mathbf{g} (setting $\mathbf{g}[k, k] = \mathbf{0}$ for all k) before the summation, which eliminates both boundary terms simultaneously.

These sums are computed efficiently via Einstein summation (`einsum`) over the masked \mathbf{g} . The key insight is that multiplying by A_{iw} (a continuous value from the generator) acts as a soft attention over potential neighbours, and gradients flow through \mathbf{A} into the generator. The full 2-GNN layer update is then:

$$\mathbf{F}^{(\ell+1)} = \sigma\left(\mathbf{F}^{(\ell)}\mathbf{W}_3^{(\ell)} + \text{agg}_1^{(\ell)} + \text{agg}_2^{(\ell)}\right). \quad (28)$$

After T_2 layers, the 2-GNN readout sums over the upper-triangular entries:

$$\mathbf{h}_G^{(2)} = \sum_{i < j} \mathbf{F}^{(T_2)}[i, j]. \quad (29)$$

Both the 1-GNN and 2-GNN dense paths are fully differentiable through \mathbf{A} , enabling the generator to learn how edge structure affects the classifier’s output at both the node and pair level.

3.6 Validation Score

We evaluate generated explanations using a composite validation score [Sun et al.(2025)]:

$$v = (s \cdot p \cdot d)^{1/3}, \quad (30)$$

comprising three components. The geometric mean ensures that all three aspects must be satisfied: a graph cannot achieve a high score if it fails on any single component.

Embedding similarity (s). The cosine similarity between the generated graph’s embedding (computed via the dense wrapper) and a *class centroid*—the mean embedding of all real graphs in the target class, computed via the sparse k -GNN on the training set:

$$s = \cos(\mathbf{h}_{\tilde{G}}, \boldsymbol{\mu}_c) = \frac{\mathbf{h}_{\tilde{G}} \cdot \boldsymbol{\mu}_c}{\|\mathbf{h}_{\tilde{G}}\| \|\boldsymbol{\mu}_c\|}, \quad \text{where } \boldsymbol{\mu}_c = \frac{1}{|\mathcal{G}_c|} \sum_{G \in \mathcal{G}_c} \mathbf{h}_G. \quad (31)$$

This measures whether the generated graph “looks like” a real class member in the model’s learned representation space. The centroid $\boldsymbol{\mu}_c$ is computed once and stored in the GIN-Graph checkpoint.

Note on the corrected metric. The original GIN-Graph implementation used the prediction probability p as a proxy for embedding similarity (effectively setting $s = p$), which collapsed the validation score to $v = (p^2 \cdot d)^{1/3}$. We corrected this by computing actual cosine similarity via the dense wrapper’s `get_embedding()` method.

Prediction probability (p). The softmax probability assigned to the target class by the pretrained classifier: $p = p_{\Phi}(y = c \mid \tilde{G})$.

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

$$d = \exp\left(-\frac{(\bar{d}_{\tilde{G}} - \mu_d)^2}{2\sigma_d^2}\right), \quad (32)$$

where $\bar{d}_{\tilde{G}} = 2|\tilde{E}|/|\tilde{V}|$ is the average degree of the generated graph, and μ_d, σ_d are the mean and standard deviation of average degrees across real graphs in the target class. A graph with average degree close to the class mean receives $d \approx 1$; one with atypical degree is penalised exponentially.

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 (κ). We additionally report:

$$\kappa = 1 - \min\left(1, \frac{n_{\tilde{G}}}{\bar{n}_c}\right), \quad (33)$$

where $n_{\tilde{G}}$ is the number of active (non-isolated) nodes in the explanation and \bar{n}_c is the average number of nodes in class c . Values near 0 indicate coarse-grained explanations (graph-sized); values near 1 would 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 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 cap the graph size at 50 nodes (the median protein graph has ~ 26 nodes), as the explanations highlight key substructures rather than reproducing entire large graphs.

4.2 Model Configuration

All k -GNN models use a hidden dimension of $d_T = 64$. The 1-GNN uses $T_1 = 3$ message-passing layers; the 1-2-GNN uses 3 layers for the 1-GNN component and $T_2 = 2$ layers for the 2-GNN component. Both use dropout of 0.5 and are trained with Adam (lr = 0.01) for 300 epochs with cross-entropy loss. Data is split 80/20 into train/test with a fixed seed of 42.

The GIN-Graph generator uses a latent dimension $d_z = 32$, hidden dimension $d_h = 128$, and is trained for 300 epochs with Adam (lr = 0.001) and batch size 64. WGAN-GP uses $\lambda_{GP} = 10$ and trains the discriminator once per generator update ($n_{critic} = 1$). The dynamic weighting uses $p = 0.4$, $k = 10$, $\lambda_{min} = 0$, $\lambda_{max} = 1$.

Computational cost. Table 2 reports wall-clock times on CPU. The 2-GNN component incurs substantial overhead due to the $O(n^2)$ pair construction and message passing. On PROTEINS, the 1-2-GNN is $\sim 326\times$ slower per forward pass than the 1-GNN, reflecting the cost of processing all $\binom{n}{2}$ node pairs for graphs with up to 620 nodes. GIN-Graph generation uses the dense wrapper, where the 1-2-GNN overhead is $\sim 36\text{--}74\times$ at the fixed generation size ($N = 28$ for MUTAG, $N = 50$ for PROTEINS).

Table 2: Wall-clock inference times on CPU. k -GNN: sparse forward pass (batch of 64 graphs). GIN-Graph: dense wrapper generation (100 graphs).

Dataset	Model	k -GNN (ms/batch)	GIN-Graph gen (s/100)
MUTAG	1-GNN	1.9	0.02
MUTAG	1-2-GNN	99.5 (52 \times)	1.48 (74 \times)
PROTEINS	1-GNN	6.8	0.04
PROTEINS	1-2-GNN	2214 (326 \times)	1.43 (36 \times)

4.3 Evaluation Protocol

For each model-dataset-class combination, we generate 100 explanation graphs at evaluation temperature $\tau = 0.1$ and evaluate them using the validation score (Equation 30). We report:

- **Validity rate**: fraction of explanations passing degree and score thresholds;
- **Mean validation score**: averaged over all 100 generated graphs;
- **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 3 summarises the classification results. On MUTAG, both models achieve identical best test accuracy of 89.5%, indicating that the additional pairwise message passing does not provide a classification advantage on this small molecular dataset. On PROTEINS, both models achieve comparable classification accuracy (79.8% vs. 78.9%) after 300 epochs of training, with the 1-2-GNN now processing all $\binom{n}{2}$ node pairs via a numba-accelerated kernel instead of sampling a subset. The 1-2-GNN’s best accuracy occurs at epoch 75, showing that the model benefits from extended training when given access to the full pairwise structure.

Table 3: k -GNN classification results. Best test accuracy reported over all epochs.

Dataset	Model	Params	Final Test Acc	Best Acc	Best Epoch
MUTAG	1-GNN	21,570	81.6%	89.5%	84
MUTAG	1-2-GNN	50,370	76.3%	89.5%	18
PROTEINS	1-GNN	21,058	75.8%	79.8%	120
PROTEINS	1-2-GNN	49,858	73.1%	78.9%	75

5.2 GIN-Graph Explanation Quality: MUTAG

Table 4 presents the GIN-Graph results on MUTAG. All configurations were fully trained for 300 epochs, enabling fair cross-model comparison on both classes.

On class 1 (Non-Mutagen), the 1-2-GNN produces explanations with higher validity (82% vs. 78%), higher validation scores (0.735 vs. 0.681), and notably higher degree scores (0.565 vs. 0.490), indicating more structurally realistic graphs. This advantage is particularly striking because both models achieve identical classification accuracy (89.5%), isolating the effect of higher-order message passing on explanation quality from classification performance.

On class 0 (Mutagen), the 1-2-GNN outperforms the 1-GNN (41% vs. 32% validity, 0.434 vs. 0.320 validation score), though both models struggle with low validity rates. De-

Table 4: GIN-Graph explanation quality on MUTAG. All models trained for 300 epochs.

Class	Model	Valid%	Val Score	s	p	d
0 (Mutagen)	1-GNN	32%	0.320	0.700	1.000	0.210
0 (Mutagen)	1-2-GNN	41%	0.434	0.771	1.000	0.314
1 (Non-Mutagen)	1-GNN	78%	0.681	0.935	1.000	0.490
1 (Non-Mutagen)	1-2-GNN	82%	0.735	0.933	1.000	0.565

spite Mutagen being the majority class ($125/188 = 66.5\%$), the difficulty likely stems from structural diversity: mutagenic compounds exhibit varied functional groups—nitro, amino, halogen—making a single representative graph harder to produce than the more structurally uniform Non-Mutagen class. Both models achieve perfect prediction probability ($p = 1.0$), with the 1-2-GNN achieving higher embedding similarity (0.771 vs. 0.700) and degree scores (0.314 vs. 0.210).

Qualitatively, the two models generate strikingly different atom type distributions for class 0 (Figure 1). The 1-GNN produces C/N/O-dominated structures (49% C, 25% N, 25% O) consistent with nitrogen-containing mutagenic motifs, while the 1-2-GNN generates halogen-heavy graphs dominated by fluorine and oxygen (36% F, 34% O, 20% Cl)—suggesting that the pairwise message passing leads the model to associate mutagenicity with halogenated structures rather than nitroaromatic ones. This divergence reveals that the two architectures learn qualitatively different internal representations of the same class.

Figure 2 shows the top-ranked explanation graphs for both models on MUTAG class 1.

The top 1-2-GNN explanations achieve near-perfect validation scores (0.986), with consistent graph sizes (24–27 nodes, 27–30 edges). The 1-GNN explanations achieve similarly high scores (top score 0.984), with comparable structural variation.

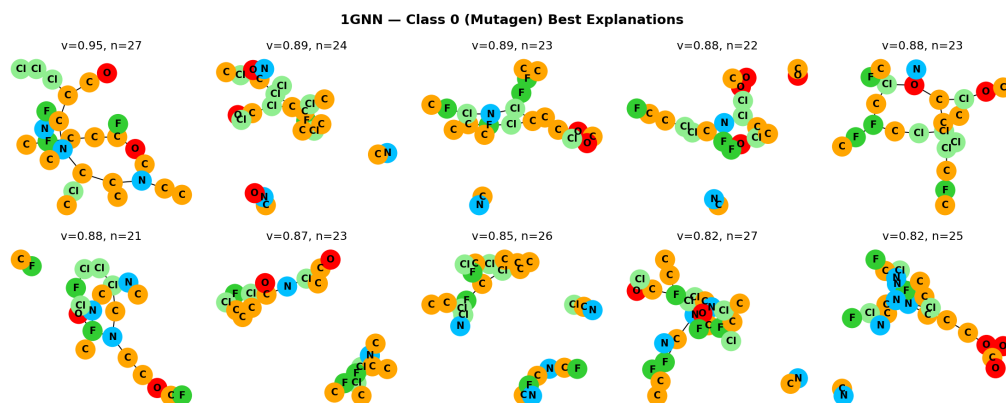
5.3 GIN-Graph Explanation Quality: PROTEINS

Table 5 presents results on PROTEINS. All configurations were fully trained for 300 epochs. Unlike MUTAG, the two models show complementary strengths: each excels at explaining a different class.

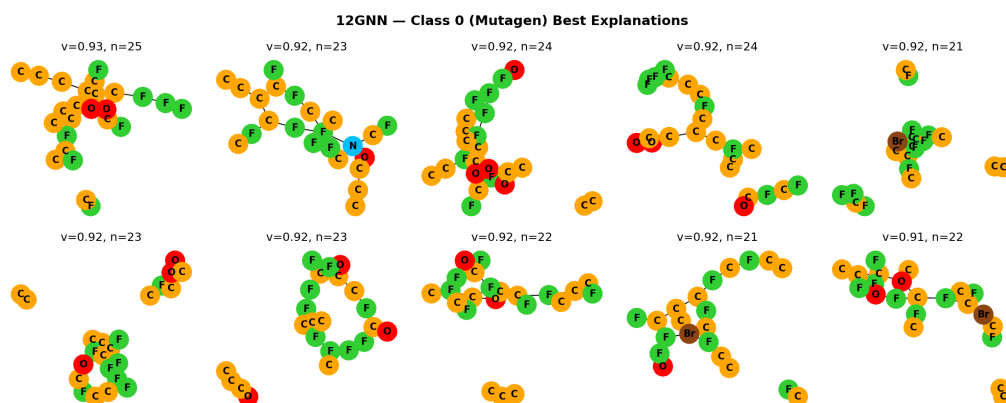
Table 5: GIN-Graph explanation quality on PROTEINS. All models trained for 300 epochs.

Class	Model	Valid%	Val Score	s	p	d
0 (Non-Enzyme)	1-GNN	100%	0.982	0.989	0.984	0.974
0 (Non-Enzyme)	1-2-GNN	100%	0.579	0.637	0.305	0.998
1 (Enzyme)	1-GNN	27%	0.251	0.250	0.923	0.976
1 (Enzyme)	1-2-GNN	100%	0.558	0.182	0.975	0.981

The 1-GNN dominates class 0 (Non-Enzyme) with near-perfect scores across all components ($v = 0.982$, $s = 0.989$, $p = 0.984$, $d = 0.974$), while the 1-2-GNN struggles with very low prediction probability ($p = 0.305$), indicating its classifier tends to predict class

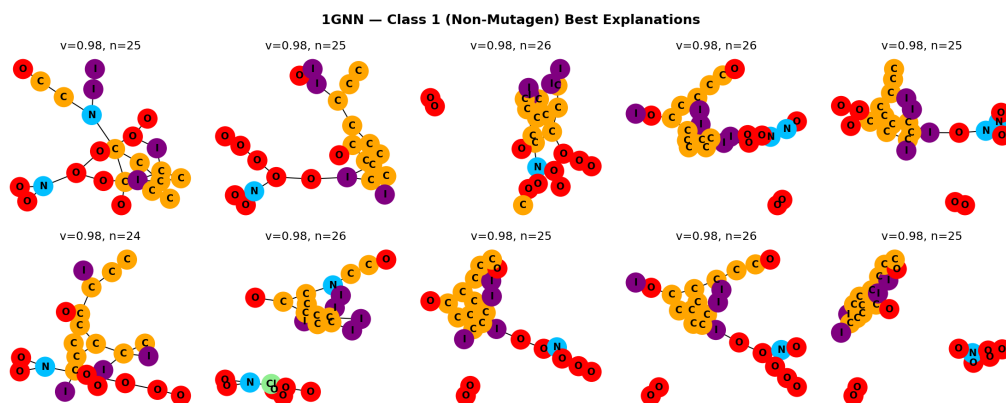


(a) 1-GNN explanations for Mutagen

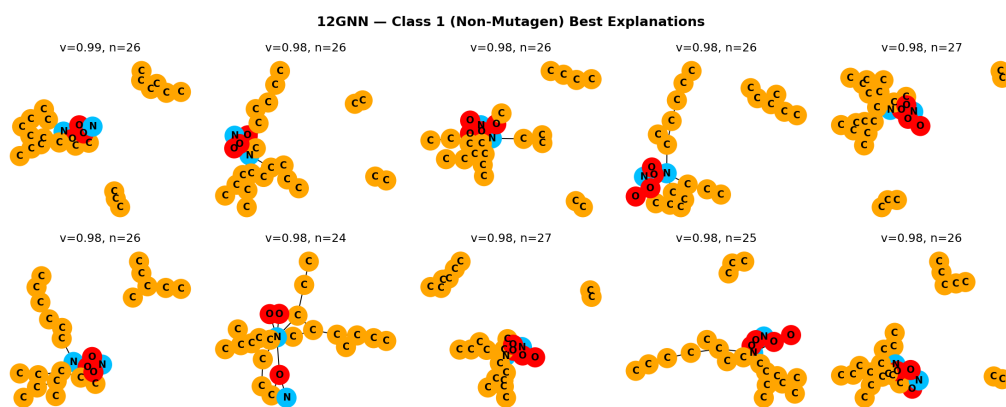


(b) 12-GNN explanations for Mutagen

Figure 1: Top-ranked GIN-Graph explanations on MUTAG class 0 (Mutagen). The 1-GNN generates C/N/O-rich structures (orange/blue/red), while the 12-GNN produces halogen-heavy graphs dominated by F and O (green/red). Node colours indicate atom types (see Figure 3).



(a) 1-GNN explanations for Non-Mutagen



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

Figure 2: Top-ranked GIN-Graph explanations on MUTAG class 1 (Non-Mutagen). Node colours indicate atom types (see Figure 3).

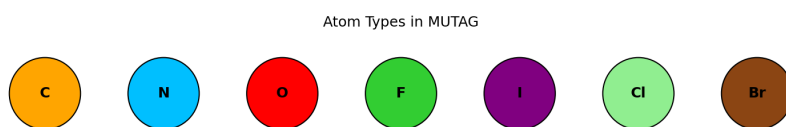


Figure 3: MUTAG atom type colour legend. Atoms: C (orange), N (blue), O (red), F (green), I (purple), Cl (light green), Br (brown).

1 regardless of generated structure. However, on class 1 (Enzyme), the situation reverses: the 1-2-GNN achieves 100% validity with high prediction probability ($p = 0.975$), while the 1-GNN achieves only 27% validity with low embedding similarity ($s = 0.250$), indicating the generated Enzyme graphs diverge from the real embedding centroid. Both models now achieve comparable classification accuracy (79.8% vs. 78.9%), so this complementary pattern reflects genuine differences in what the two architectures learn about protein structure rather than a simple classifier quality gap.

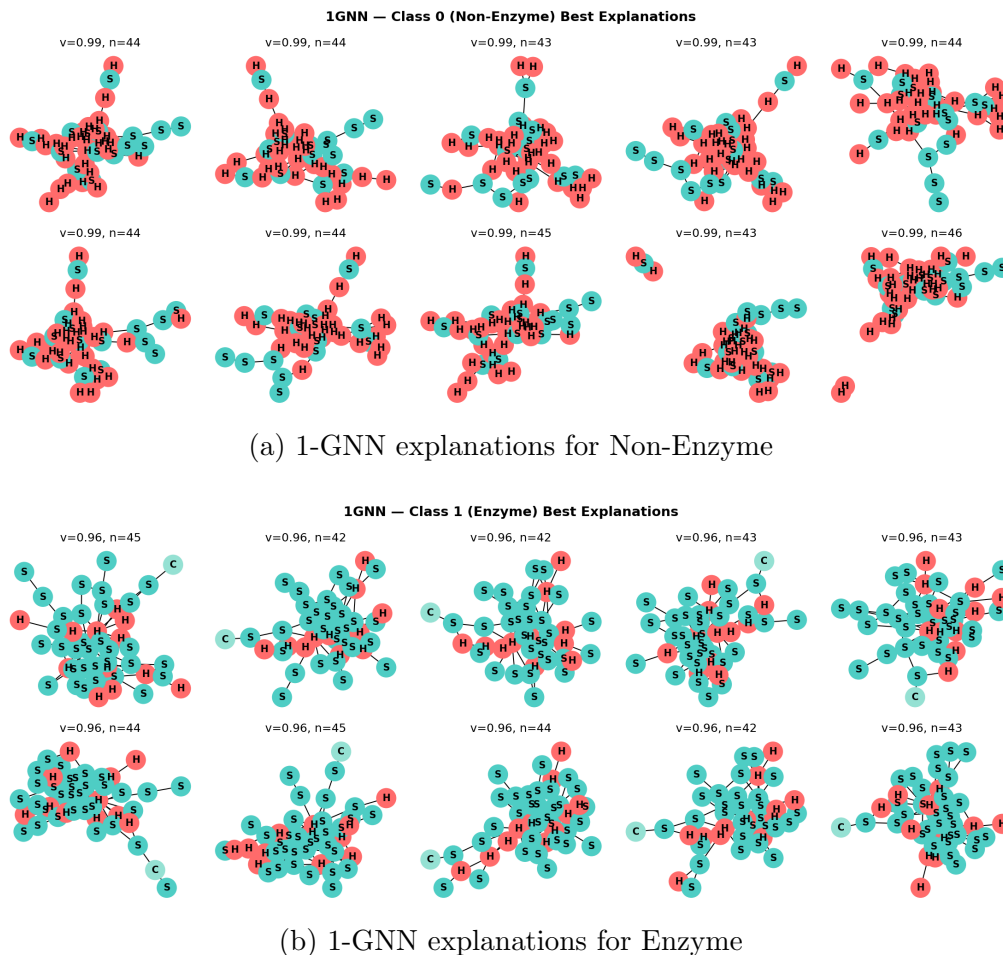
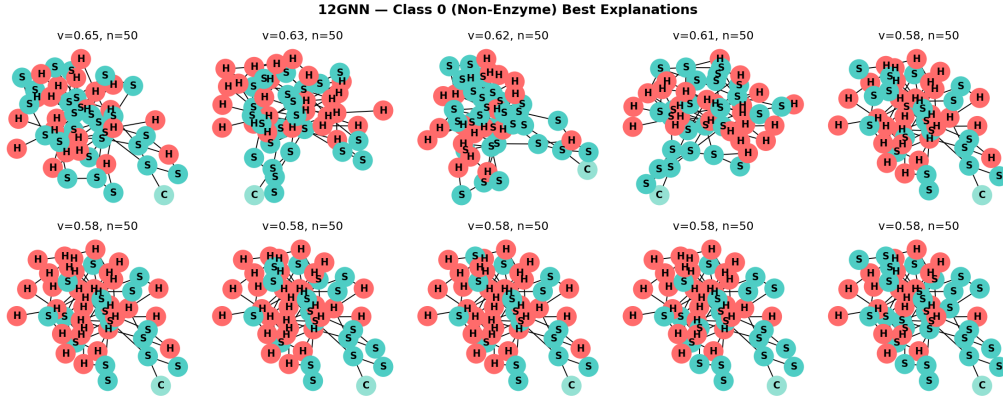
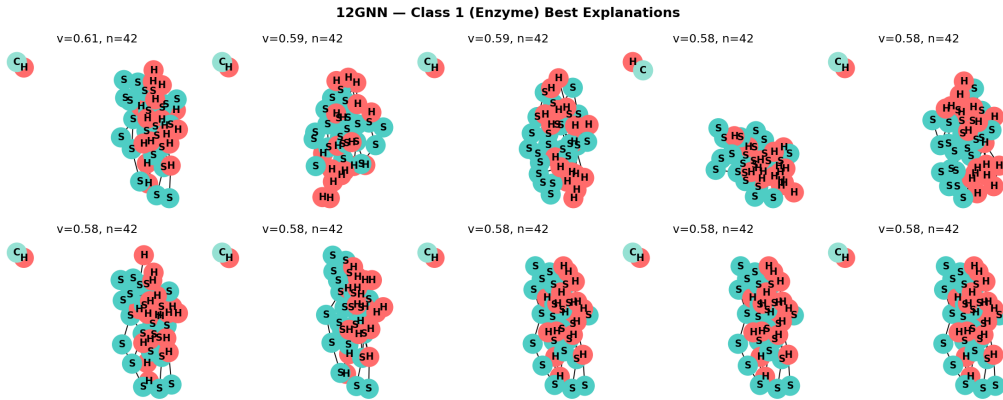


Figure 4: Top-ranked GIN-Graph explanations on PROTEINS (1-GNN). Node colours indicate secondary structure types: helix (H, pink), sheet (S, teal), coil/turn (C, green).

The class-specific strengths are visually evident in the explanation figures. The 1-GNN class 0 explanations (Figure 4) show high-quality graphs with validation scores $v = 0.99$, while its class 1 explanations are limited to the 27 valid graphs out of 100 generated. The 1-2-GNN explanations (Figure 5) achieve 100% validity on both classes, but with lower overall scores due to the prediction probability bottleneck on class 0 and the embedding similarity bottleneck on class 1. This complementary pattern—each model excelling on a different class—suggests that the node-level and pairwise representations capture different aspects of protein structure.



(a) 1-2-GNN explanations for Non-Enzyme



(b) 1-2-GNN explanations for Enzyme

Figure 5: Top-ranked GIN-Graph explanations on PROTEINS (1-2-GNN). Class 0 explanations achieve validation scores $v = 0.58$ – 0.65 with 50 nodes, while class 1 explanations achieve $v = 0.56$ – 0.61 with 42 nodes.

5.4 Metric Decomposition

Table 6 provides a detailed comparison of the three validation score components across all configurations, now with all models trained for 300 epochs.

Table 6: Validation score decomposition for all GIN-Graph models (300 epochs).

Dataset	Class	Model	s (emb. sim)	p (pred. prob)	d (degree)
MUTAG	0 (Mutagen)	1-GNN	0.700	1.000	0.210
MUTAG	0 (Mutagen)	1-2-GNN	0.771	1.000	0.314
MUTAG	1 (Non-Mut.)	1-GNN	0.935	1.000	0.490
MUTAG	1 (Non-Mut.)	1-2-GNN	0.933	1.000	0.565
PROTEINS	0 (Non-Enz.)	1-GNN	0.989	0.984	0.974
PROTEINS	0 (Non-Enz.)	1-2-GNN	0.637	0.305	0.998
PROTEINS	1 (Enzyme)	1-GNN	0.250	0.923	0.976
PROTEINS	1 (Enzyme)	1-2-GNN	0.182	0.975	0.981

On MUTAG class 1, the degree score is the primary differentiator: the 1-2-GNN generates graphs whose average degree better matches the target class distribution (0.565 vs. 0.490). Both models achieve perfect prediction probability ($p = 1.0$) and comparable embedding similarity ($s = 0.933$ vs. 0.935).

On MUTAG class 0, the 1-2-GNN outperforms across all components: higher embedding similarity (0.771 vs. 0.700), higher degree score (0.314 vs. 0.210), and equal prediction probability ($p = 1.0$). Both models achieve perfect prediction probability on MUTAG, indicating that the classifiers provide strong gradient signal to the generators.

On PROTEINS, each model excels on a different class. The 1-GNN dominates class 0 with near-perfect scores ($s = 0.989$, $p = 0.984$, $d = 0.974$), while the 1-2-GNN dominates class 1 validity (100% vs. 27%) with high prediction probability ($p = 0.975$) but very low embedding similarity ($s = 0.182$). The 1-2-GNN struggles on class 0 with low prediction probability ($p = 0.305$), meaning its classifier tends to predict class 1 regardless of input, while the 1-GNN struggles on class 1 with low embedding similarity ($s = 0.250$), indicating the generated Enzyme graphs diverge from the real embedding centroid. The degree scores remain uniformly high across all PROTEINS configurations ($d > 0.97$), confirming that structural realism is not the bottleneck.

5.5 Training Dynamics

Figure 6 shows a representative training curve from the MUTAG 1-2-GNN class 1 experiment. The three loss curves illustrate the interplay between the WGAN-GP adversarial training and the GNN guidance objective over 300 epochs.

During the first ~ 120 epochs ($\lambda \approx 0$), the generator loss and discriminator loss dominate: the generator learns to produce structurally realistic graphs that fool the discriminator, without pressure to target a specific class. As the dynamic weighting $\lambda(t)$ increases (Equation 23), the GNN guidance loss takes over, steering the generator toward class-specific patterns. This transition is visible as the GNN loss drops sharply while the adversarial losses plateau.

The training behaviour varies across configurations. On MUTAG, all four generators converge smoothly, consistent with the small graph sizes ($N = 28$) and low feature dimensionality (7 atom types). On PROTEINS, the training dynamics are more complex: the 1-2-GNN class 0 generator never achieves high prediction probability ($p = 0.305$ at evaluation), suggesting that the GNN guidance signal is insufficient to push the generator into the target class region of the 1-2-GNN’s decision space. In contrast, the 1-GNN class 0 generator achieves near-perfect alignment ($p = 0.984$), indicating that the 1-GNN’s decision boundary is easier for the generator to exploit.

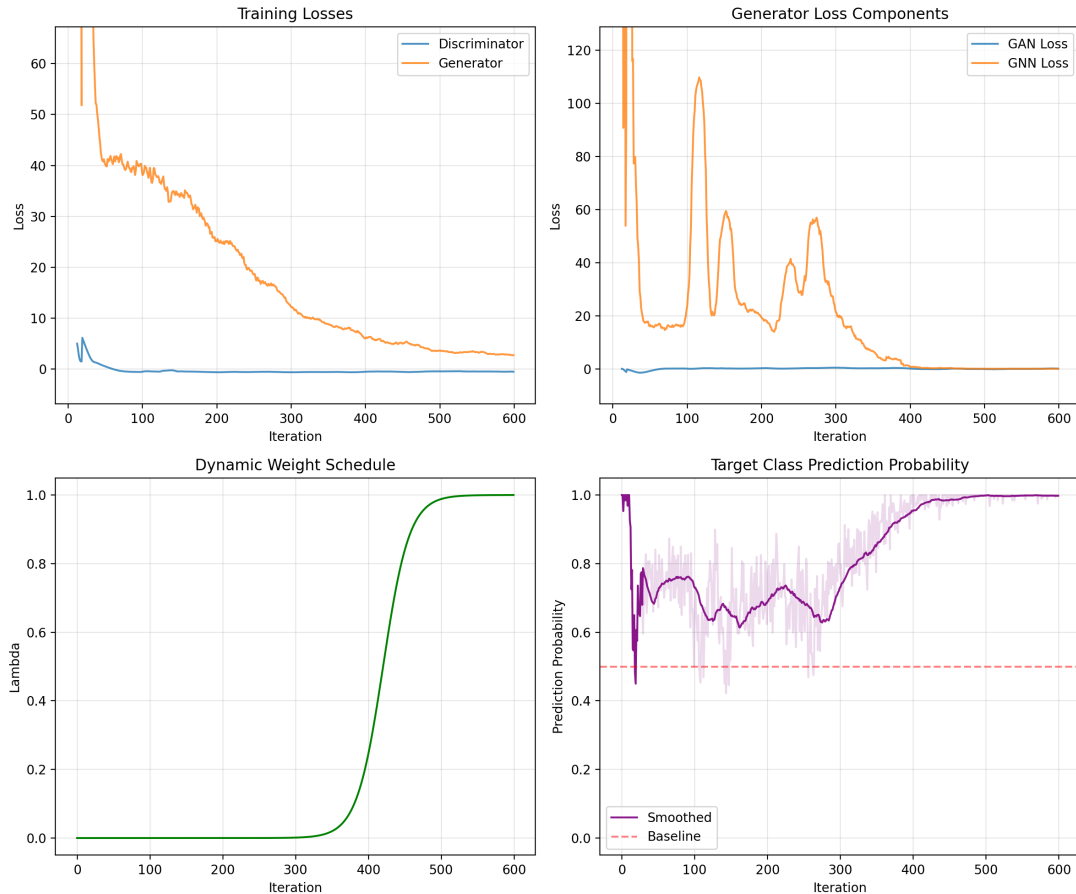


Figure 6: Training curves for GIN-Graph on MUTAG 1-2-GNN class 1, showing generator loss, discriminator loss, and GNN guidance loss over 300 epochs. The dynamic weighting schedule shifts emphasis from adversarial realism (early epochs) to class-specific GNN guidance (later epochs).

5.6 Generated Dataset Comparison

To evaluate the fidelity and cross-model transferability of generated explanations at scale, we generate 500 graphs per class from each GIN-Graph generator and compare the resulting synthetic datasets against the real data. This population-level analysis complements the per-graph metrics reported in Sections 5.2–5.4 by revealing systematic differences between what each model has learned. No retraining is performed—all graphs are generated from the existing checkpoints.

5.6.1 Structural Fidelity

Table 7 summarises the structural comparison. Degree distributions match closely across all configurations: Kolmogorov–Smirnov statistics range from 0.11 to 0.30, with generated mean degrees within 1% of real values. This confirms that the WGAN-GP component captures local connectivity patterns regardless of the guiding classifier.

Graph sizes diverge more substantially. On MUTAG ($N = 28$), generated graphs average 23–26 nodes versus 14–20 in the real dataset. On PROTEINS ($N = 50$), class 0 sizes match better (44–50 generated vs. 49 real), but class 1 graphs are roughly twice the real Enzyme size (42–44 vs. 24). This reflects the fixed generator size and the tendency of Gumbel-Softmax to activate most node slots, consistent with the granularity limitation discussed earlier.

Node type distributions reveal how each model represents class identity. On MUTAG, both models associate mutagenicity (class 0) with halogens: the 1-GNN generates 16.8% F and 14.9% Cl (vs. 0.9% and 3.0% real), and the 1-2-GNN generates 34.2% F. For Non-Mutagen (class 1), the 1-2-GNN closely matches the real distribution (78.6% C vs. 73.2%, 14.0% O vs. 17.1%), while the 1-GNN produces atypical oxygen- and iodine-heavy structures (36.3% O, 16.7% I vs. 17.1% and 0.05% real). On PROTEINS, all generators produce reasonable secondary structure distributions, with the 1-2-GNN class 1 generator best matching real proportions (H: 43.9% vs. 40.7%, S: 54.2% vs. 56.2%).

Table 7: Structural comparison: 500 generated vs. real graphs per configuration. D_{KS} = Kolmogorov–Smirnov statistic on degree distributions (all $p < 10^{-7}$). \bar{d} = mean degree, \bar{n} = mean graph size.

Configuration	$D_{KS} \downarrow$	$\bar{d}_{\text{real}} / \bar{d}_{\text{gen}}$	$\bar{n}_{\text{real}} / \bar{n}_{\text{gen}}$
<i>MUTAG</i>			
1-GNN class 0	0.113	2.09 / 2.07	13.9 / 24.0
1-GNN class 1	0.123	2.24 / 2.23	19.8 / 25.8
1-2-GNN class 0	0.121	2.09 / 2.06	13.9 / 23.4
1-2-GNN class 1	0.157	2.24 / 2.22	19.8 / 26.0
<i>PROTEINS</i>			
1-GNN class 0	0.170	3.80 / 3.80	48.7 / 44.3
1-GNN class 1	0.296	3.64 / 3.63	23.7 / 43.8
1-2-GNN class 0	0.204	3.80 / 3.80	48.7 / 50.0
1-2-GNN class 1	0.131	3.64 / 3.61	23.7 / 41.7

5.6.2 Cross-Model Classification

The most revealing analysis classifies each set of 500 generated graphs with *both* k-GNN classifiers, testing whether explanations produced by one model are recognisable by the other. Table 8 reports target class agreement rates, and Figure 7 visualises the full classification breakdown.

On MUTAG, three of four configurations show high cross-model agreement: 1-GNN class 0 at 89.8%, 1-GNN class 1 at 93.0%, and 1-2-GNN class 0 at 100%. The exception is striking: the 1-2-GNN’s Non-Mutagen graphs (class 1) are classified as *Mutagen* by the 1-GNN in

99.8% of cases. These graphs—dominated by conventional C/N/O structures (78.6% C, 7.3% N, 14.0% O)—are precisely the carbon-backbone molecules that the 1-GNN’s simpler decision boundary associates with mutagenicity. The 1-2-GNN, using pairwise structural information, has learned to distinguish these from Mutagen compounds by features invisible to the 1-GNN.

On PROTEINS, the disagreement is even more extreme. The 1-GNN’s Non-Enzyme graphs achieve 100% same-model agreement but 0% cross-model agreement—the 1-2-GNN classifies every single one as Enzyme. Conversely, the 1-2-GNN’s Non-Enzyme graphs achieve 0% same-model agreement (reflecting the known low prediction probability, $p = 0.305$) but 99.8% agreement from the 1-GNN. This symmetric inversion reveals that the two classifiers have learned fundamentally opposing decision boundaries for the Non-Enzyme class: features that define “Non-Enzyme” for one model define “Enzyme” for the other. The Enzyme results mirror this pattern, with the 1-2-GNN’s Enzyme graphs receiving only 0.4% agreement from the 1-GNN.

Table 8: Cross-model classification of generated graphs (500 per class). Same = target class agreement by the generating model’s classifier; Cross = agreement by the other model’s classifier.

Generated by	Target class	Same	Cross
<i>MUTAG</i>			
1-GNN	Mutagen (c0)	100%	89.8%
1-GNN	Non-Mutagen (c1)	100%	93.0%
1-2-GNN	Mutagen (c0)	100%	100%
1-2-GNN	Non-Mutagen (c1)	99.6%	0.2%
<i>PROTEINS</i>			
1-GNN	Non-Enzyme (c0)	100%	0%
1-GNN	Enzyme (c1)	100%	99.8%
1-2-GNN	Non-Enzyme (c0)	0%	99.8%
1-2-GNN	Enzyme (c1)	100%	0.4%

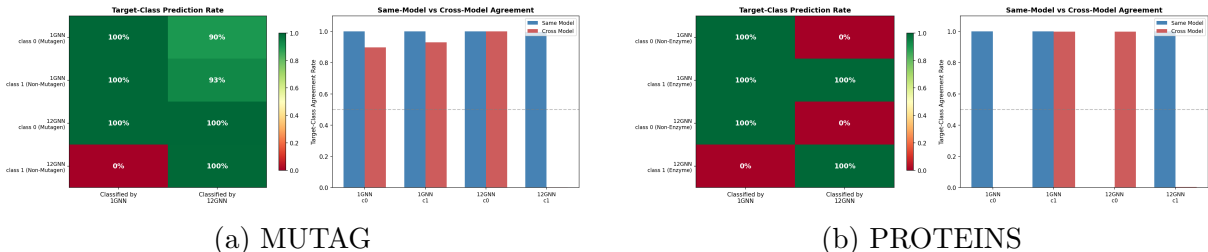


Figure 7: Cross-model classification of generated graphs. Each cell shows the fraction classified into the target class by each model. High same-model agreement (diagonal) with low cross-model agreement (off-diagonal) indicates that the two architectures have learned different decision boundaries.

5.6.3 Embedding Space Structure

Figures 8 and 9 show t-SNE projections of real and generated graph embeddings in each model’s learned representation space.

On MUTAG (Figure 8), generated graphs cluster near their target class in both models’ embedding spaces, consistent with the high same-model prediction rates. The 1-GNN’s space shows clear class separation, with generated class 0 and class 1 graphs falling within the corresponding real clusters.

On PROTEINS (Figure 9), the embedding structure reflects the complementary strengths identified earlier. In the 1-GNN’s space, generated class 0 graphs form a tight cluster near real Non-Enzyme embeddings, while generated class 1 graphs scatter more diffusely—consistent with the 1-GNN’s high class 0 quality ($v = 0.982$) and poor class 1 validity (27%). In the 1-2-GNN’s space, the low prediction probability on class 0 ($p = 0.305$) is visible as misalignment between generated and real Non-Enzyme regions. These visualisations confirm that the two models not only disagree on class boundaries but organise their embedding spaces around different structural features.

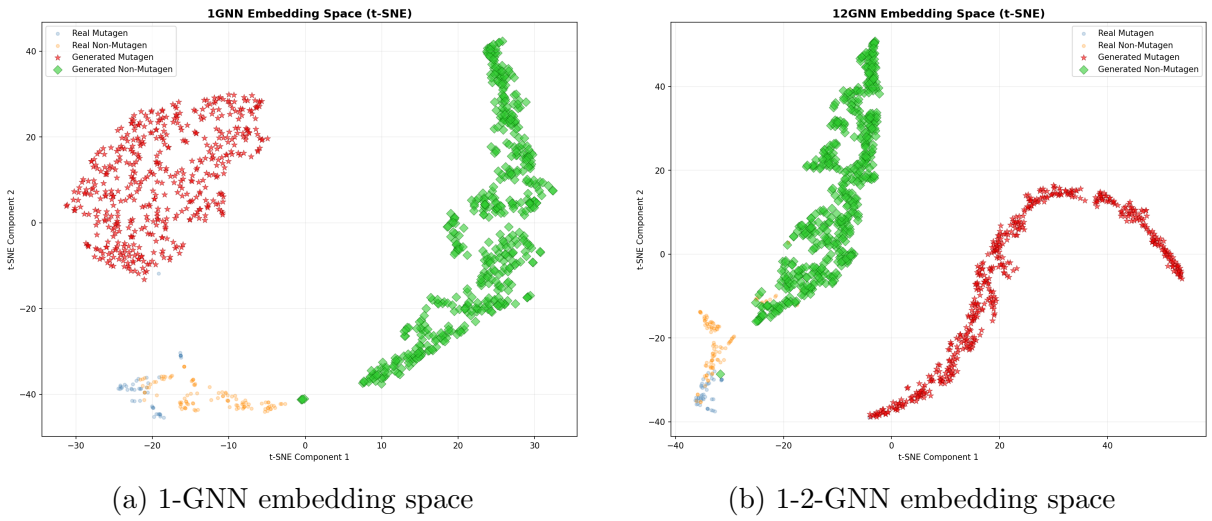


Figure 8: t-SNE projections of MUTAG real (small dots) and generated (large markers) graph embeddings. Colours distinguish Mutagen (class 0) from Non-Mutagen (class 1).

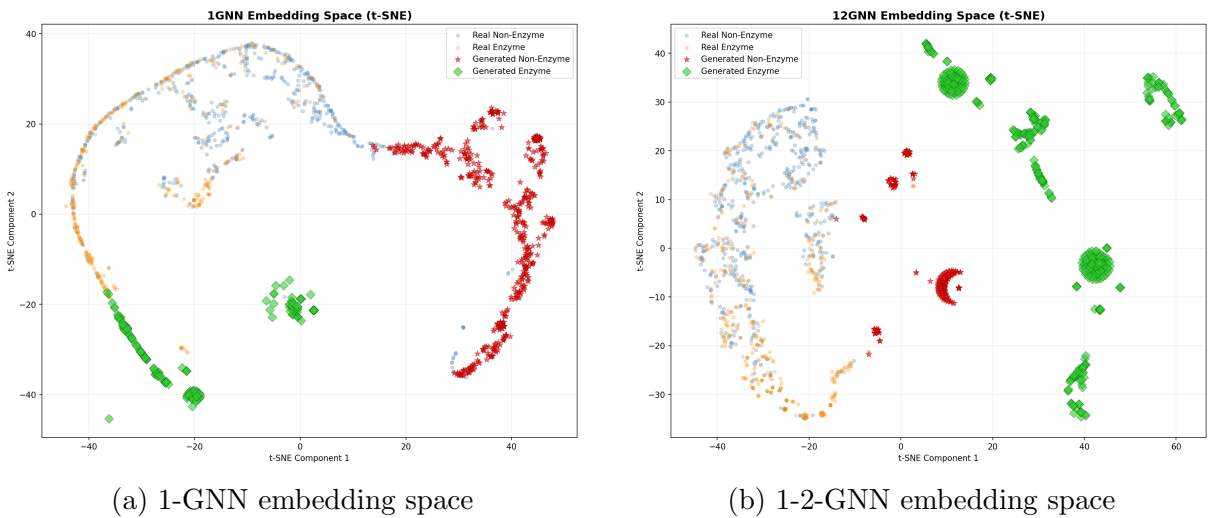


Figure 9: t-SNE projections of PROTEINS real (small dots) and generated (large markers) graph embeddings. Colours distinguish Non-Enzyme (class 0) from Enzyme (class 1).

6 Discussion

6.1 Does Higher-Order Message Passing Improve Explanations?

Our results suggest a nuanced answer: *it depends on the dataset, and each architecture may excel at explaining different classes.*

On MUTAG, both models achieve identical classification accuracy (89.5%), yet the 1-2-GNN produces consistently better explanations on both classes: 82% vs. 78% validity and 0.735 vs. 0.681 validation score on class 1, and 41% vs. 32% validity and 0.434 vs. 0.320 on class 0. This is a particularly clean comparison because the equal classification performance isolates the effect of higher-order message passing on explanation quality. The pairwise message passing enables the generator to produce graphs whose average degree better matches real molecules (degree score 0.565 vs. 0.490 on class 1) and with higher embedding fidelity on class 0 (0.771 vs. 0.700). The consistent improvement across both classes demonstrates that the 1-2-GNN’s richer structural representations translate into more informative model-level explanations, even when both models classify equally well.

On PROTEINS, both models achieve comparable classification (79.8% vs. 78.9%) after processing all node pairs via numba-accelerated sparse computation, but each excels on a different class. The 1-GNN dominates class 0 (Non-Enzyme) with near-perfect validation scores ($v = 0.982$), while the 1-2-GNN achieves far higher validity on class 1 (Enzyme) at 100% versus 27%. This complementary pattern suggests that the two architectures learn different aspects of protein structure: the 1-GNN’s node-level representations better capture Non-Enzyme patterns, while the 1-2-GNN’s pairwise representations better characterise Enzyme structures, though with low embedding similarity ($s = 0.182$) indicating the generated graphs remain distant from the real embedding centroid.

The cross-model classification analysis (Section 5.6) quantifies this divergence. On MUTAG, the 1-GNN’s generated graphs are largely recognised by the 1-2-GNN (90–93% cross-model agreement), but the 1-2-GNN’s Non-Mutagen graphs are almost entirely misclassified by the 1-GNN (0.2% agreement). On PROTEINS, the divergence is near-total: each model’s generated Non-Enzyme graphs receive 0% agreement from the other model. This confirms that the two architectures have learned fundamentally different decision boundaries, not merely different feature weightings within a shared decision space.

6.2 Qualitative Differences in Generated Explanations

Beyond the quantitative metrics, the generated explanations reveal qualitative differences in what each model has learned. On MUTAG class 0 (Mutagen), the 1-GNN generates molecules dominated by carbon, nitrogen, and oxygen (49% C, 25% N, 25% O)—atom types associated with classical mutagenic motifs such as nitroaromatic groups. The 1-2-GNN, by contrast, produces halogen-heavy structures dominated by fluorine and oxygen (36% F, 34% O, 20% Cl), suggesting that its pairwise message passing leads it to associate mutagenicity with halogenated compounds. Both represent valid mutagenic subclasses, but the divergence demonstrates that the two architectures learn fundamentally different internal representations of the same chemical class.

On MUTAG class 1 (Non-Mutagen), both models converge on carbon-dominated back-

bones, though the 1-GNN includes substantial iodine presence while the 1-2-GNN produces more chemically conventional C/N/O structures. This convergence on the majority class is consistent with the higher validity rates: the Non-Mutagen class has more uniform structural patterns that the generator can capture.

On PROTEINS, both models generate visually similar secondary structure distributions (mixtures of helix, sheet, and coil nodes), which is expected given that the node features encode only three secondary structure types. The qualitative differences here are not in node composition but in the class-specific patterns: the 1-GNN produces diverse, high-quality Non-Enzyme explanations but struggles to generate valid Enzyme graphs (27% validity), while the 1-2-GNN achieves full validity on both classes but with lower overall fidelity. This suggests the two architectures focus on different structural aspects of protein graphs.

6.3 The Granularity Problem

All generated explanations have granularity $\kappa \approx 0$, meaning they are as large as typical graphs in the dataset. This is a structural 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 ~ 18 nodes), the explanations use 20–27 nodes. For PROTEINS ($N = 50$, average graph ~ 26 nodes), they use 45–50 nodes.

Model-level explanations are inherently coarse-grained: they answer “what does a typical class member look like?” rather than “which substructure drives the prediction?” This is a property of the method, not a flaw—the two questions require different explanation approaches.

6.4 Limitations

Several limitations qualify our findings:

Single seed, no error bars. All experiments use a single data split (seed 42) without cross-validation. With only 38 test graphs on MUTAG, the identical 89.5% best accuracy for both models could be coincidence—a difference of one misclassified graph changes accuracy by 2.6 percentage points. Similarly, the validity differences on MUTAG (82% vs. 78%, i.e. 4 graphs out of 100) may not be statistically significant without variance estimates across seeds. The PROTEINS results are more robust (224 test graphs), but the complementary class-specific pattern could also reflect data split artefacts.

No cross-validation. Best test accuracy is reported as the maximum over epochs (early stopping by oracle), which overestimates generalisation performance. A more rigorous evaluation would use a held-out validation set for model selection.

Classifier-generator coupling and the $p = 0.305$ failure. GIN-Graph explanations are shaped by how the classifier’s internal representations interact with the generator’s optimisation landscape. The most revealing failure is the 1-2-GNN class 0 (Non-Enzyme) generator on PROTEINS, where the prediction probability never sustains values above 0.5 during training—only 0.9% of 2700 training iterations achieve $p > 0.5$, with a final mean of $p = 0.305$.

We hypothesise that the root cause is an *embedding dimensionality mismatch*. The 1-2-GNN classifier operates on a concatenated embedding $\mathbf{h}_G = [\mathbf{h}_G^{(1)} \parallel \mathbf{h}_G^{(2)}] \in \mathbb{R}^{128}$, double the 1-GNN’s \mathbb{R}^{64} . The generator MLP, however, uses the same hidden dimension ($d_h = 128$) in both cases, meaning it must navigate a higher-dimensional classifier landscape with no additional capacity. Furthermore, the 2-GNN component’s response to adjacency changes is mediated by pair-level aggregation (Equations 26–27), creating a more complex gradient landscape than the 1-GNN’s simple matrix multiplication $\mathbf{A}\mathbf{H}\mathbf{W}$. The generator’s GNN guidance gradient must propagate through both the 1-GNN and 2-GNN readouts, and small changes to \tilde{A} produce correlated but non-identical effects at both levels, potentially creating competing gradient directions.

The 1-GNN class 1 (Enzyme) failure (27% validity, $s = 0.250$) has a different character: $p = 0.923$ indicates the generator *can* find the target class region, but the generated graphs diverge from the real Enzyme embedding centroid. This is a fidelity problem (the generated graphs are atypical Enzyme structures) rather than a guidance problem (the classifier does recognise them as Enzyme).

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.

No baselines. We do not compare with instance-level explanation methods (GNNEExplainer, PGExplainer) or other model-level approaches. GIN-Graph is the only model-level method we are aware of that generates full graphs via GAN, so direct model-level baselines are limited. However, comparing the *utility* of model-level vs. instance-level explanations on the same classifiers would contextualise our findings.

6.5 Future Work

Several directions could strengthen this investigation:

- Use cross-validation and multiple random seeds for robust accuracy and explanation quality estimates. The current single-seed results, while consistent across configurations, may not generalise.
- Investigate the complementary class-specific strengths on PROTEINS—the 1-GNN excels at Non-Enzyme while the 1-2-GNN excels at Enzyme explanations—to understand how the two architectures encode different aspects of protein structure.
- Extend to the 1-2-3-GNN architecture, which adds 3-set (triplet) message passing for even higher expressiveness, to test whether the interpretability benefit continues up the k -WL hierarchy.
- Incorporate richer structural evaluation metrics beyond average degree, such as clustering coefficients, motif frequencies, or subgraph counts, to better capture the qualitative differences between generated explanations.
- Apply the framework to larger and more diverse datasets where the expressiveness gap between 1-WL and higher-order tests is more pronounced.

7 Conclusion

We compared a standard 1-GNN with a hierarchical 1-2-GNN for model-level explanation generation using the GIN-Graph framework, with all configurations fully trained for 300 epochs across both datasets and both classes.

On MUTAG, both models achieve identical classification accuracy (89.5%), yet the 1-2-GNN produces consistently better explanations across both classes (82% vs. 78% validity, 0.735 vs. 0.681 validation score on class 1; 41% vs. 32% validity on class 0). This controlled comparison—where classification performance is held constant—provides evidence that higher-order message passing improves the structural quality of model-level explanations independently of classification accuracy.

On PROTEINS, both models achieve comparable classification (79.8% vs. 78.9%), but each excels at explaining a different class: the 1-GNN dominates Non-Enzyme explanations ($v = 0.982$), while the 1-2-GNN achieves higher validity on Enzyme explanations (100% vs. 27%). Cross-model classification of 500 generated graphs per class reveals that this complementary pattern reflects fundamentally different learned decision boundaries: on PROTEINS, graphs generated for one model are systematically misclassified by the other, with near-0% cross-model agreement on the Non-Enzyme class. This suggests that the benefit of higher-order message passing for interpretability is both dataset- and class-dependent, and that the two architectures develop incompatible internal representations of graph structure.

Our fully differentiable dense wrapper enables GIN-Graph explanation generation for hierarchical k -GNN architectures while maintaining gradient flow through both node features and adjacency matrices. Additionally, our corrected validation metric—computing actual cosine similarity between generated and real graph embeddings rather than using prediction probability as a proxy—provides a more faithful measure of explanation quality. These technical contributions open 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.

- [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.