# 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 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:

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

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:

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

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 $s = \{v_1, \ldots, v_k\} \subseteq V$, the *local neighbourhood* is:

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

where $s \triangle s'$ denotes the symmetric difference. 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:

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

where $\mathbf{W}_1^{(t)}, \mathbf{W}_2^{(t)} \in \mathbb{R}^{d_{t-1}\times d_t}$ are learnable weight matrices, $\sigma$ 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:

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

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:

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

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

$$\hat{y} = \operatorname{softmax}\Big(\mathbf{W}_o\, \sigma(\mathbf{W}_c\, \mathbf{h}_G^{(1)})\Big)\,, \tag{7}$$

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:

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

where $\parallel$ denotes vector concatenation. The isomorphism type $\mathbb{1}[(u,v) \in E] \in \{0,1\}$ 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:

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

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:

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

where $\mathbf{W}_3^{(\ell)}, \mathbf{W}_4^{(\ell)}$ are the 2-GNN weight matrices at layer $\ell$. 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:

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

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:

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

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 randomly sample up to 5000 pairs per graph. This preserves a representative sample of pairwise interactions while keeping computation tractable.

## 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:

$$\begin{aligned} \mathbf{h} &= \mathrm{LeakyReLU}(\mathbf{W}_2\, \mathrm{LeakyReLU}(\mathbf{W}_1 \mathbf{z})) \in \mathbb{R}^{2d_h}, && (13)\\ \tilde{A}_{\text{raw}} &= \mathrm{sym}(\mathrm{reshape}(\mathbf{h}\,\mathbf{W}_A,\ N \times N))\,, && (14)\\ \tilde{X}_{\text{raw}} &= \mathrm{reshape}(\mathbf{h}\,\mathbf{W}_X,\ N \times D), && (15) \end{aligned}$$

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 $\mathrm{sym}(\mathbf{M}) = \frac{1}{2}(\mathbf{M} + \mathbf{M}^\top)$ 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) $\pi_1, \dots, \pi_K$, the Gumbel-Softmax sample is:

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

where $\tau > 0$ is a temperature parameter. As $\tau \to 0$, the output approaches a one-hot vector (hard categorical sample); as $\tau \to \infty$, 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:

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

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:

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

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:

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

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:

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

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:

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

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

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

where $p_\Phi(y = c \mid \tilde{G})$ is the softmax probability assigned to class $c$ by the pretrained $k$-GNN $\Phi$. 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)]:

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

where $T$ is the total number of training iterations, $p = 0.4$ is the fraction of training before $\lambda$ 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 $\mathbf{A} \in [0,1]^{B\times N\times N}$:

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

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 $\mathbf{H}^{(T_1)} \in \mathbb{R}^{B\times N\times d_T}$:

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

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:

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

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

The constraints $w \neq j$ in $\mathrm{agg}_1$ and $w \neq i$ in $\mathrm{agg}_2$ 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 $\mathrm{agg}_1$ and $w = j$ from $\mathrm{agg}_2$, 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:

$$\mathbf{F}^{(\ell+1)} = \sigma\Big(\mathbf{F}^{(\ell)}\,\mathbf{W}_3^{(\ell)} + \mathrm{agg}_1^{(\ell)} + \mathrm{agg}_2^{(\ell)}\Big). \tag{28}$$

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

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

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)]:

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

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:

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

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:

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

where $\bar{d}_{\tilde{G}} = 2|\tilde{E}|/|\tilde{V}|$ is the average degree of the generated graph, and $\mu_d, \sigma_d$ 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** $(\kappa)$. We additionally report:

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

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 $\sim$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 100 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 $\lambda_{\text{GP}} = 10$ and trains the discriminator once per generator update ($n_{\text{critic}} = 1$). The dynamic weighting uses $p = 0.4$, $k = 10$, $\lambda_{\min} = 0$, $\lambda_{\max} = 1$.

## 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 2 summarises the classification results. On MUTAG, the 1-2-GNN outperforms the 1-GNN by 5.3 percentage points (89.5% vs. 84.2%), consistent with the theoretical advantage of higher-order message passing on molecular graphs where pairwise bond patterns carry important structural information. On PROTEINS, the result is reversed: the 1-GNN achieves 78.0% while the 1-2-GNN reaches only 65.0%. The 1-2-GNN has 2.3× more parameters (50,370 vs. 21,570), and on the relatively simple 3-dimensional node features of PROTEINS, 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.

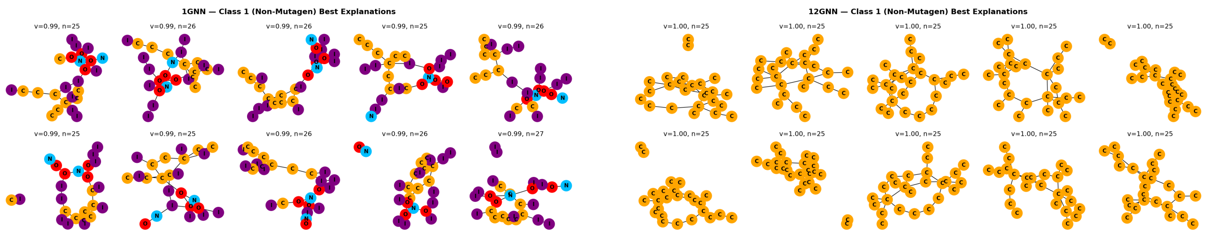
Table 3: GIN-Graph explanation quality on MUTAG. †: only 49 training epochs (under-trained).

| **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** |

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). The 1-GNN explanations, while still high quality (top score 0.991), show slightly more structural variation.



(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).

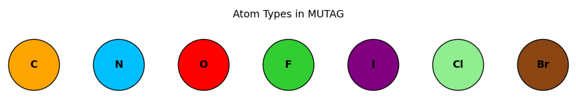


Figure 2: MUTAG atom type colour legend. Atoms: C (orange), N (blue), O (red), F (green), I (purple), Cl (light green), Br (brown).

## 5.3 GIN-Graph Explanation Quality: PROTEINS

Table 4 presents results on PROTEINS. 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 (undertrained).

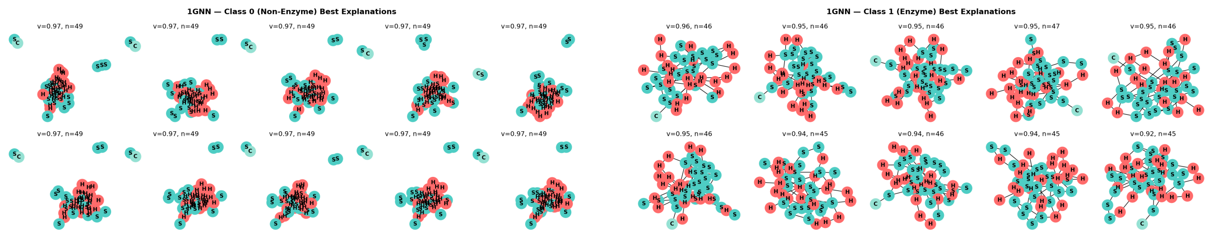
| Class | Model | Epochs | Valid% | Val Score | $s$ | $p$ | $d$ |
|---|---|---|---|---|---|---|---|
| 0 (Non-Enzyme) | 1-GNN | 300 | **100%** | **0.967** | 0.998 | 0.916 | **0.990** |
| 0 (Non-Enzyme) | 1-2-GNN† | 29 | 99% | 0.768 | 0.998 | 0.590 | 0.797 |
| 1 (Enzyme) | 1-GNN | 300 | **100%** | **0.883** | 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 underlying classifier performance (65.0% accuracy).

## 5.4 Metric Decomposition

Table 5 provides a detailed comparison of the three validation score components for the fully trained configurations (300 epochs).

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).


1GNN — Class 0 (Non-Enzyme) Best Explanations
1GNN — Class 1 (Enzyme) Best Explanations

(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.

Table 5: Validation score decomposition for fully trained GIN-Graph models.

| **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 |

On PROTEINS, the 1-GNN shows an asymmetry between classes: 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 occupies a more complex region of the embedding space 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, illustrating the evolution of losses 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. When the GIN-Graph generator is guided by this richer classifier, it produces graphs whose average degree better matches real molecules (degree score 0.638 vs. 0.486) and whose embeddings lie closer to the real class centroid (0.995 vs. 0.964). 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

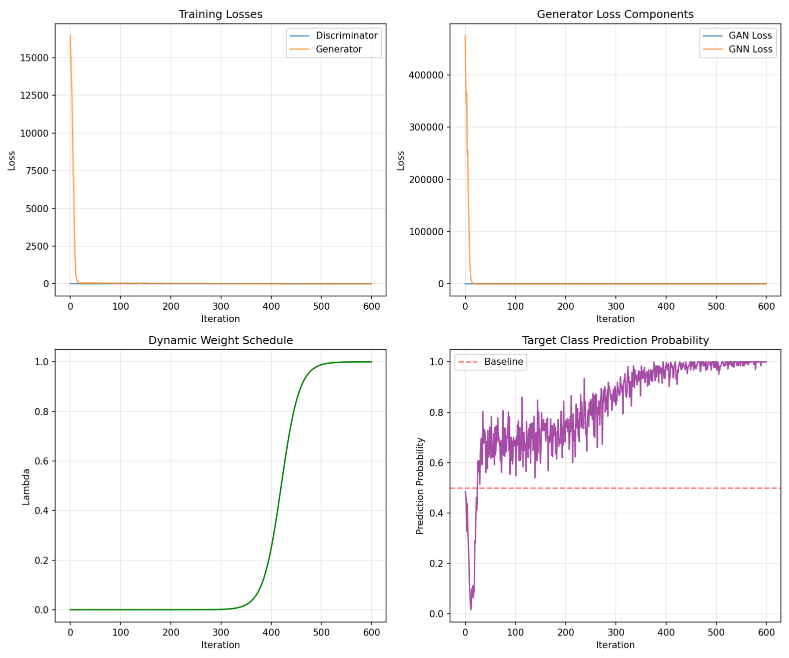


Figure 4: Training curves for GIN-Graph on MUTAG 1-2-GNN class 1, showing the evolution of generator loss, discriminator loss, and GNN guidance loss over 300 epochs.

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 $\kappa \approx 0$, 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 $\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 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.

## 6.3 Limitations

Several limitations qualify our findings:

**Undertrained checkpoints.** The MUTAG 1-GNN class 0 GIN-Graph was only trained for 49 epochs, and the PROTEINS 1-2-GNN for only 29 epochs. These incomplete runs make cross-model comparison on those configurations unreliable. A complete study would require all checkpoints trained to convergence.

**Single seed.** All experiments use a single data split (seed 42) without cross-validation. The small size of MUTAG (188 graphs, 38 in the test set) means that test accuracy can fluctuate 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 (300 epochs) for fair cross-model comparison on all dataset-class combinations.
- Use cross-validation and multiple random seeds for robust accuracy and explanation quality estimates.
- Investigate why the 1-2-GNN underperforms on PROTEINS—potential causes include overfitting due to the parameter increase, sensitivity to hyperparameters, or the low feature dimensionality of PROTEINS (3 vs. 7 for MUTAG).
- Extend to the 1-2-3-GNN architecture, which adds 3-set (triplet) message passing for even higher expressiveness.
- Incorporate richer structural evaluation metrics beyond average degree, such as clustering coefficients or subgraph counts.

# 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 when the classifier benefits from it. On PROTEINS, the 1-GNN dominates in both classification and explanation quality, 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 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.

[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.