Hierarchical Restructuring of Graph Neural Networks for Robust Learning under Edge Perturbations

Anonymous Author(s)
Affiliation
Address
email

Abstract

Graph Neural Networks (GNNs) are central to graph representation learning, yet their robustness is challenged by structural perturbations in graphs, leading to suboptimal analytical outcomes. These perturbations, common in real-world graphs due to factors like adversarial interferences, result in noisy and incomplete data. Addressing this issue, we propose the hierarchical restructuring (HR) framework[1], utilizing a hierarchical Bayesian model to capture these latent disruptions. Our framework is uniquely adaptable as a plug-in for various GNN variants, optimizing a hierarchical variational lower bound alongside downstream task training. The HR-enhanced models show superior performance in node-level classification, graph classification, and spatial-temporal graph classification tasks. The results indicate accuracy gains in a range of 3% to 21% under a 90% perturbation ratio for node classification tasks and up to 38% for graph classification tasks under a 50% perturbation ratio. These findings underscore the effectiveness of our framework in enhancing the robustness and accuracy of GNNs in the presence of structural perturbations.

1 Introduction

Graph Neural Networks (GNNs) have carved a niche in the realm of managing intricate graph-structured data, systematically dealing with datasets where relationships and relational properties are pivotal. Existing within the paradigm of the Message Passing Network (MPN) [1], GNNs ingeniously innovate node representations through a meticulous aggregation of neighboring information. However, their broad application is somewhat hampered when it comes to real-world scenarios, where accurate and complete graph structures might not always be available or directly observable and when facing with adversarial attack or structure perturbations [2, 3, 4].

In Figure 1, we demonstrate the susceptibility of Graph Neural Networks (GNNs) to structural perturbations within the context of semi-supervised node classification. Specifically, the G_1 and G_2 is perturbed by deleting normal edge and adding noisy edge respectively. This leads to the failures of GNN, since the standard GNN learning paradigm involves aggregating information from neighboring nodes, pooling the aggregated data, and then making a prediction, is significantly disrupted by such perturbations. To counteract these perturbations, we introduce a hierarchical restructuring technique that rectifies the perturbed graph into a refined graph G_3, thereby ensuring more reliable predictions in the presence of adversarial structural changes.

A myriad of applications, ranging from molecule classification to Electroencephalogram (EEG) predictions, underscore the necessity for a GNN that can reliably handle incomplete or noisy graphs. Traditional methods like GNNExplainers [5] and filtering-based methods [6, 7], while valuable, tend

[1]Our code available at `https://anonymous.4open.science/r/HRGNN-7BDC/README.md`

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

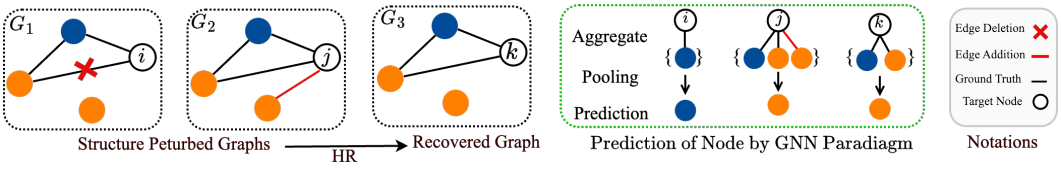


Figure 1: Hierarchical restructuring (HR) corrects structural perturbations such as edge deletions in G_1 and edge additions in G_2 into a clean graph G_3, thereby restoring GNN prediction accuracy via the aggregate–pool–predict paradigm.

to be highly dependent on original structures, offering little in the way of correcting or completing inaccurate structures. Generative methods such as GraphRNN [8], or VGAE [9] provide a possible way to tackle this issue.

Existing graph generation models primarily focus on reconstructing structural patterns while neglecting two critical aspects: 1) discriminative feature preservation for downstream tasks, and 2) explicit modeling of hidden factors underlying graph incompleteness. To address these limitations, we propose a hierarchical restructuring (HR) framework that integrates multi-level latent variables through variational inference. Our approach dynamically infers potential edges during GNN training by jointly considering observed neighbor features and latent topological patterns, thereby completing graphs while maintaining task-relevant structural information.

The hierarchical architecture employs structured variational distributions to model both global graph characteristics and local node interactions. For graph-level tasks, we model full node interdependencies, while node-level tasks adopt k-hop neighborhood constraints in subgraph sampling to balance expressiveness and computational efficiency. This adaptive mechanism enables robust graph representation learning under structural uncertainty while preventing over-smoothing. The framework simultaneously optimizes two objectives: mutual information maximization between reconstructed graphs and task labels, and parameter learning for both the generator and GNN through variational lower bound maximization.

Experimental validation demonstrates our HR framework's superior performance across diverse scenarios, including graph–level classification, node-level classification and dynamic graph analysis. Results show consistent accuracy improvements over baseline methods while maintaining stable computational efficiency, confirming its effectiveness in handling both structural incompleteness and task-specific feature preservation.

2 Related works and Basic Definitions

2.1 Graph Representation Learning by GNNs

In graph-based applications, high quality of representations of a graph including nodes, edges, and the entire graph is crucial for downstream tasks such as graph classification, node classification, link prediction, etc. GNNs is a powerful deep learning model to learn good representations from graph data. There are two mainstream graph neural networks (GNNs) in terms of convolution operator[10], i.e., spatial-based and spectral-based. Spatial-based GNNs utilize aggregating operation to filter information from node neighborhoods, such an operation is an analogy to a graph convolution, the aggregator is known as a convolution operator [11, 12, 13]. Spectral-based GNNs are based on graph spectral theory[14], pioneering works such as [15] and other following GCNs [16, 17, 18]implement the graph convolution by graph Fourier transform. Xu proposed GWNN [19], another kind of spectral-based GNNs which leverages graph wavelet transform [14] as the convolution kernel or convolution operator.

2.2 Joint Graph Structure Learning

Joint graph structure learning (GSL) integrates topology inference with downstream tasks—typically semi-supervised classification—and is broadly classified into metric learning, probabilistic learning, and direct parameter learning [20]. Unlike fully unsupervised methods (e.g., GPT-GNN), joint GSL adapts the graph structure during task training to bolster GNN robustness against perturbations

and adversarial attacks that can degrade classification performance [4, 21, 22, 23]. Metric-based approaches (e.g., [6]) distinguish and prune negative edges, whereas probabilistic methods (e.g., [2, 24, 25]) model edges as Bernoulli variables or bi-level programs to learn an optimal edge distribution. Direct-parameter techniques (e.g., [4]) apply proximal optimization to refine an adjacency matrix, and dynamic GAE variants (e.g., [26]) update embedding models during downstream training. In contrast, our hierarchical framework treats latent factors as the generative source of graph topology, enabling more expressive and robust structure learning.

2.3 Basic Definitions

In this section, we introduce notations and definitions for subsequent discussions and define the main problem of interest.

Let $G = (\mathbf{A}, \mathbf{X})$ denote an attributed graph with N nodes, where $\mathbf{A} \in \{0,1\}^{N\times N}$ represents the adjacency matrix, and $\mathbf{X} \in \mathbb{R}^{N\times C}$ denotes the feature matrix with each row corresponding to a C-dimensional attribute vector. Typically, for a given graph $G = (\mathbf{A}, \mathbf{X})$, a 2-layer Graph Convolutional Network (GCN), denoted simply by f, can be used to learn node representations as follows:

$$\mathbf{H} = f_\theta(\mathbf{A}, \mathbf{X}) = \sigma(\mathbf{A}\sigma(\mathbf{A}\mathbf{X}\mathbf{W}_1)\mathbf{W}_2), \tag{1}$$

where $\mathbf{H} \in \mathbb{R}^{N\times F}$ is the matrix of node representations with each row H_i representing node i in an F-dimensional hidden space, σ is a nonlinear function such as sigmoid or ReLU, and $\theta = (\mathbf{W}_1, \mathbf{W}_2)$ are the learnable parameters of the GCN.

We can employ $\mathbf{H}$ for node-level or graph-level tasks. For node classification, let $\mathcal{Y}$ represent the set of classes, with each node label $y_i \in \mathcal{Y}$. A learnable matrix $\mathbf{W}_3$ can project $\mathbf{H}$ into a $|\mathcal{Y}|$-dimensional space, followed by a softmax function to calculate the probability of each class, i.e., $\hat{Y} = \text{Softmax}(\mathbf{H}\mathbf{W}_3)$. For graph classification, an additional pooling layer is applied to $\mathbf{H}$, i.e., $\hat{Y} = \text{Softmax}(\text{Pool}(\mathbf{H})\mathbf{W}_3)$, where $\text{Pool}(\cdot)$ could be max pooling, average pooling, etc.

In practice, however, $\mathbf{A}$ is often noisy or influenced by hidden factors, and GCNs are sensitive to structural perturbations, leading to suboptimal downstream performance. Thus, our method aims to jointly learn a new structure $\mathbf{A}^*$, such that a GCN using $\mathbf{A}^*$ and $\mathbf{X}$ yields an optimal $H^* = f(\mathbf{A}^*, \mathbf{X})$ for downstream tasks, enhancing the robustness of GCNs against structural perturbations.

3 hierarchical restructuring

In this section, we present a novel hierarchical variational inference method designed for generating hidden graphs, namely hierarchical restructuring (HR), which can be particularly beneficial for downstream tasks. Even our method can be extended to both graph level and node level tasks, we use graph classification as a prime example to provide a clear illustration of its potential applications.

3.1 Variational Inference on Graphs

Before introduce our method, we first discuss the application of variational inference to graph generation, particularly focusing on the Variational Graph Auto-Encoder (VGAE). The VGAE learns a distribution over graph structures for the link prediction task. The generative process is formalized as:

$$P(\mathbf{A}|\mathbf{Z}) = \prod_{i=1}^{N}\prod_{j=1}^{N} p(\mathbf{A}_{ij}|\mathbf{z}_i, \mathbf{z}_j), \tag{2}$$

where $p(\mathbf{A}_{ij} = 1|\mathbf{z}_i, \mathbf{z}_j) = \sigma(\mathbf{z}_i^\top \mathbf{z}_j)$. The latent variables $\mathbf{Z}$ are inferred from a variational distribution $q(\mathbf{Z}|\mathbf{X}, \mathbf{A})$, which should capture the complex dependencies introduced by the observed data $(\mathbf{X}, \mathbf{A})$. The optimization objective for VGAE is as follows:

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}[\log p(\mathbf{A}|\mathbf{Z})] - \text{KL}[q(\mathbf{Z}|\mathbf{X}, \mathbf{A})\|p(\mathbf{Z})], \tag{3}$$

where $\text{KL}[q\|p]$ signifies the Kullback-Leibler divergence between two distributions q and p, and $p(\mathbf{Z})$ is the Gaussian prior defined as $p(\mathbf{Z}) = \prod_i p(\mathbf{z}_i) = \prod_i \mathcal{N}(\mathbf{z}_i|\mathbf{0}, \mathbf{I})$.

However, this model assumes that nodes are independent, and the variational distribution is merely an approximation to the standard Gaussian, which limits its ability to model the intricate dependencies

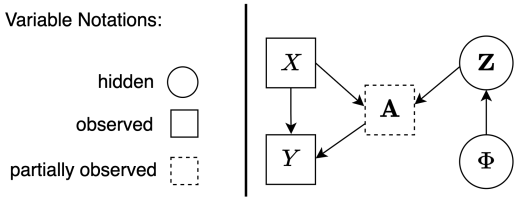


Figure 2: The graphical model of hidden graph generation for downstream task. The directed arrows denote the dependency relations. The label $\mathbf{Y}$ is dependent on $\mathbf{X}$ and $\mathcal{A}$, and $\mathcal{A}$ is dependent on $\mathbf{X}$ and hidden factors Z which is dependent on another prior Φ.

within the graph. To illustrate, consider a triad of nodes i, j, and k, with j serving as a bridge between i and k, which are not directly linked. One would expect $p(\mathbf{A}_{ij} = 1|\mathbf{z}_i, \mathbf{z}_j)$ to be high, indicating proximity between $\mathbf{z}_i$ and $\mathbf{z}_j$ in the latent space, and similarly for $\mathbf{z}_k$ and $\mathbf{z}_j$. However, this proximity should not necessarily translate to $\mathbf{z}_i$ and $\mathbf{z}_k$ being close, as this would contradict the absence of a direct link between i and k, which limits its predictive capacity for more complex graph structures.

3.2 Hierarchical Variational Inference on Graphs

In this section, we delve into the hierarchical variational model by introducing a hidden graph $\mathbf{A}$ and a hidden prior Φ, integrating this generative model within a discriminative framework.

Figure 2 illustrates the hierarchical graphical model that includes hidden graphs $\mathbf{A}$ and hidden factors $\mathbf{Z}$ for the downstream task. In this model, the structure $\mathcal{A}$ depends on the observed data $\mathbf{X}$ and a hidden factor $\mathbf{Z}$, which in turn relies on another hidden variable Φ. The $\mathbf{A}$ is partially observed; for a given sample $\mathbf{A}_i$, it is actually sampled from the conditional distribution $p(\mathbf{A}|\mathbf{X}, \mathbf{Z})$, with $\mathbf{Z}$ being unobserved, resulting in an incomplete $\mathbf{A}_i$. A complete $\mathbf{A}_i$ would be drawn from the conditional distribution over all possible $\mathbf{Z}$, that is, $\mathbf{A}_i \sim \mathbb{E}_{\mathbf{Z}\sim p(\mathbf{Z}|\Phi)} p(\mathbf{A}|\mathbf{X}, \mathbf{Z})$.

By incorporating another hidden variable Φ, the distribution of $\mathbf{Z}$ becomes more flexible and could be any implicit or explicit distributions with more representative power, rather than the normal distribution in Eq. (3). For example, by leveraging the framework of Semi-Implicit Variational Inference (SIVI), the model gains the capability to approximate complex distributions of $\mathbf{Z}$ that are not easily expressed with explicit density functions.

Under this graphical model, we reformulate the target discriminative model as:

$$\begin{aligned} p(\mathbf{Y}|\mathbf{X}) = \mathbb{E}_{\Phi}\mathbb{E}_{\mathbf{Z}}\mathbb{E}_{\mathbf{A}}\left[p(\mathbf{Y}|\mathbf{A}, \mathbf{X})\right], \quad \text{where} \\ \mathbf{A} \sim p(\mathbf{A}|\mathbf{Z}, \mathbf{X}), \\ \mathbf{Z} \sim p(\mathbf{Z}|\Phi), \\ \Phi \sim p(\Phi). \end{aligned} \tag{4}$$

The introduction of Φ as a higher-order latent variable also facilitates a more nuanced prior over $\mathbf{Z}$, enhancing the model's ability to maintain dependencies between latent variables and to generalize better on unseen data [27]. This is especially critical in tasks that require robustness to noise and the ability to generalize from limited observations, as is often the case in graph-structured data.

Now, we consider a graph classification task in our hierarchical model. Given a dataset $\{G_i = (\mathbf{A}_i, \mathbf{X}_i)\}_{i=1}^n$ with corresponding labels $\{y_i\}_{i=1}^n$, our objective is to maximize the likelihood of the correct labels given the graphs. Under our hierarchical model, the log likelihood function can be written as:

$$\begin{aligned} \underline{\mathcal{L}} &= \log p(\mathbf{Y}|\mathbf{X}) = \log \prod_{i=1}^{n} p(\mathbf{Y}_i|\mathbf{X}_i) \\ &= \sum_{i=1}^{n} \log \int\int \sum_{\mathbf{A}} p(\mathbf{Y}_i|\mathbf{A}, \mathbf{X}_i) p(\mathbf{A}|\mathbf{Z}, \mathbf{X}_i) \\ &\qquad \times p(\mathbf{Z}|\Phi) p(\Phi) d\mathbf{Z} d\Phi \end{aligned} \tag{5}$$

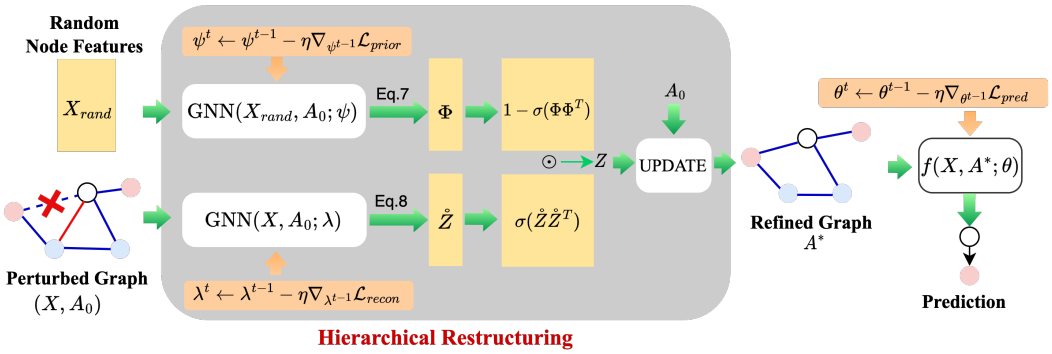


Figure 3: Overview of hierarchical restructuring (HR) in node classification task. The procedure initiates with a perturbed graph and random node features, undergoing successive refinements via HR to construct a robust graph, leading to precise node classification.

This integral is generally intractable. Therefore, we approximate it using hierarchical variational inference, and obtain a lower bound of Eq. (5).

We obtain a new objective, namely *Hierarchical Variational Objective (HVO)*, which is a lower bound of $\underline{\mathcal{L}}$ in Eq. (2) as follows (The derivation can be found in supplementary.):

$$\begin{aligned}\underline{\mathcal{L}} \geq & \mathcal{L}_{\mathbf{HVO}} \\ = & \mathbb{E}_{q(\mathbf{Z}|\Phi)}\mathbb{E}_{q(\mathbf{A}|\mathbf{Z},\mathbf{X})}\left[\log p(\mathbf{Y}|\mathbf{A},\mathbf{X})\right] \\ & - \mathbb{E}_{q(\mathbf{Z}|\Phi)}\left[\mathrm{KL}(q(\mathbf{A}|\mathbf{Z},\mathbf{X})||p(\mathbf{A}|\mathbf{Z},\mathbf{X}))\right] \\ & - \mathrm{KL}(q(\mathbf{Z}|\phi)||p(\mathbf{Z})).\end{aligned} \tag{6}$$

The variational distributions $q(\mathbf{Z}|\Phi)$ and $q(\mathbf{A}|\mathbf{Z},\mathbf{X})$ are parameterized to facilitate efficient approximation of the intractable posterior distributions. This approach is grounded in the Evidence Lower BOund (ELBO) principle, where we aim to maximize the ELBO as a proxy for the log likelihood. The ELBO is given by the expectation of the log likelihood minus the KL divergence terms, which act as a regularization by penalizing the divergence of the variational distributions from the true posteriors.

The first expectation term corresponds to the expected log likelihood of the observed data under the variational distribution. This term encourages the model to fit the data well.

The second term $\mathbb{E}_{q(\mathbf{Z}|\Phi)}\left[\mathrm{KL}(q(\mathbf{A}|\mathbf{Z},\mathbf{X})||p(\mathbf{A}|\mathbf{Z},\mathbf{X}))\right]$ and the third term $\mathrm{KL}(q(\mathbf{Z}|\Phi)||p(\mathbf{Z}))$ represent the KL divergence between the variational and true posteriors for $\mathbf{A}$ and $\mathbf{Z}$ respectively. These terms ensure that the variational distributions remain close to the true posterior distributions, thereby enforcing a form of regularization.

In following sections, we first introduce the inference model of HVO, and elaborate how to optimize the $\mathcal{L}_{HVO}$ and training procedures, as depicted in the Figure 3.

4 Inference of hierarchical restructuring

To facilitate the prediction model via HR, we take HR as a plugin tool, which is composed of two main components, i.e., *Piro Sampling* and *Graph Sampling*. The last step is to enhance the GNN model with the hidden graph generated.

The *Prior Sampling* module is to sample the prior random variable Φ. The distribution $p(\Phi)$ could be chosen implicitly or explicitly. Thanks to the powerful expressiveness of neural networks, the distribution could be parameterized by a neural network such as GCN, i.e., $p(\Phi; \psi)$ where ψ is learned by the neural network in an amortized way given the observed data, which is also known as an Encoder. In our setting, since the prior Φ is independent of the features $\mathbf{X}$, we thus only take the initial structure $\mathbf{A}$ as the input of the GCN. For an explicit distribution such as Gaussian, the inference procedure with a reparameterization of PSampling is as follows:

$$\Phi = \mathrm{GCN}_{\mu}(\mathbf{A}_0, \mathbf{X}_{rand}) + \epsilon \cdot \mathrm{GCN}_{\sigma}(\mathbf{A}_0, \mathbf{X}_{rand}), \tag{7}$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, $\mathbf{X}_{rand}$ is the node feature matrix with all features as random noise.

The *Graph Sampling* module is to sample hidden graphs from $q(\mathbf{A}|\mathbf{Z}, \mathbf{X})$. We first draw $\mathbf{Z}$ from $q(\mathbf{Z}|\Phi)$, which could be chosen as a very flexible distribution. In practice, it requires the distribution to be as expressive as possible. By introducing the additional conditional variable Φ, a Gaussian distribution conditioned on Φ has much more capacity and provides more functional distribution families to choose from. To tackle the node dependence problem and facilitate training via SGD, we design a sampling strategy as follows:

$$\mathbf{Z} \sim \mathcal{N}(\mu, \sigma^2), \tag{8}$$

where $\mu = \text{GCN}_\mu(\mathbf{X}, \mathbf{A}_0)$, and $\sigma = \text{GCN}_\sigma(\mathbf{X}, \mathbf{A}_0)$. Let $\mathbf{Z}$ be a composition of $\mathring{\mathbf{Z}}$ and Φ:

$$\mathbf{Z} = (\mathring{\mathbf{Z}}, \Phi). \tag{9}$$

The last step is to generate $\mathbf{A}$ by leveraging $\mathbf{Z}$ in a deterministic way:

$$\mathbf{A} = \mathbf{P} = \text{sigmoid}(\mathring{\mathbf{Z}}\mathring{\mathbf{Z}}^\top) \odot (1 - \text{sigmoid}(\Phi\Phi^\top)), \tag{10}$$

where $\odot$ is the element-wise product operation, and $\mathbf{P}$ is a probability matrix. Note that, in a deterministic way, we directly let $\mathbf{A} = \mathbf{P}$, which is a weighted adjacency matrix, in which each element $\mathbf{A}_{ij}$ denotes the probability of the connection between node i and node j.

Instead of the implicit distribution, we can also leverage explicit distributions to generate structures. In the context of spatial-temporal tasks where a richer representation of uncertainty is required, and the conditional distribution $p(\mathbf{Z}|\Phi)$ could be constrained with an explicit distribution such as a mixture of Gaussian. In our spatial-temporal scenario, we adopt a mixture Guassian distribution and use a Gumbel-Softmax trick to facilitate the sampling retaining the differentiable. The generation process is given by:

$$\mathbf{O} = \text{softmax}\left(\frac{\log(\pi(\Phi)) + \mathbf{G}}{\tau}\right), \tag{11}$$

$$\mathbf{Z} = \sum_{k=1}^{K} \mathbf{O}^{(k)} \cdot (\boldsymbol{\mu}_k(\Phi) + \boldsymbol{\sigma}_k(\Phi) \cdot \boldsymbol{\epsilon}), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{12}$$

$$\mathbf{A} = \text{sigmoid}(\mathbf{Z}\mathbf{Z}^\top), \tag{13}$$

where $\mathbf{G}$ is a matrix of i.i.d samples from a Gumbel(0,1) distribution, τ is a temperature parameter that controls the discreteness of the output distribution, $\boldsymbol{\pi}(\Phi)$ are the softmax-normalized weights of the mixture components conditioned on Φ, $\boldsymbol{\mu}_k(\Phi)$ and $\boldsymbol{\sigma}_k(\Phi)$ are the mean and standard deviation of the k-th mixture component conditioned on Φ, respectively, and $\mathbf{O}^{(k)}$ is the weight of the k-th component in the mixture.

For *prediction* in downstream task using sampled $\overline{\mathbf{A}}$, we refine the initial graph $\mathbf{A}_0$ as follows:

$$\mathbf{A}^* = \text{UPDATE}(\overline{\mathbf{A}}, \mathbf{A}_0) = \alpha\mathbf{A}_0 + (1 - \alpha)\overline{\mathbf{A}}, \tag{14}$$

where the parameter $\alpha \in [0, 1]$ is to control the weight of of learned structure and original structure. Note that, for different tasks, it could be fixed or learnable, such as in spatial-temporal task, we utilize an attention mechanism to refine the initial graph and multiple learned graphs. The attention mechanism can be found in the work [28].

Then we can feed that refined graph $\mathbf{A}^*$ into a GNN f to encode a graph representation $\mathbf{H}^*$ for downstream task as follows:

$$\mathbf{H}^* = f(\mathbf{X}, \overline{\mathbf{A}}). \tag{15}$$

5 Training of model with hierarchical restructuring

In this section, we introduce the optimization objectives obtained by the $\mathcal{L}_{HVO}$. Note that, we elaborate the training strategy for downstream tasks in the supplementary section.

5.1 Loss Function Design

As illustrated in Figure 3, each module is parameterized by neural networks, specifically, we introduce three notations to represent all the learnable parameters set of each module, i.e., ψ, λ and θ, corresponding to Prior Sampling module, Graph Sampling module, and the prediction model f respectively. Then we design loss functions by using $\mathcal{L}_{HVO}$ to optimize these parameters.

In $\mathcal{L}_{HVO}$, the first term is the log-likelihood of the dataset predicted by a discriminative model with given $\mathbf{A}$ and $\mathbf{X}$. Maximize this term is equivalent to minimize the cross-entropy (CE) between $\mathbf{Y}, \overline{\mathbf{Y}}$ denoted as $\mathcal{L}_{pred}$:

$$\mathcal{L}_{pred} = \text{CE}(\mathbf{Y}, \overline{\mathbf{Y}}), \quad \overline{\mathbf{Y}} = f_\theta(\mathbf{A}, \mathbf{X}). \tag{16}$$

The second term of $\mathcal{L}_{HVO}$ is to minimize the the KL-divergence of the variational distribution $q(\mathbf{A}|\mathbf{Z}, \mathbf{X}; \lambda)$ and the real structure distribution $p(\mathbf{A}|\mathbf{Z}, \mathbf{X})$ by given the hidden factor $\mathbf{Z}$. Since the real distribution of structure is unknown, we instead to use a reconstruction loss $\mathcal{L}_{recon}$ to measure such divergence gap:

$$\mathcal{L}_{recon} = \text{CE}(\mathbf{A}_0 - \overline{\mathbf{A}}), \tag{17}$$

where, the $\overline{\mathbf{A}}$ is the prediction of the graph. Note that, this $\overline{\mathbf{A}}$ then is updated with the initial graph $\mathbf{A}_0$ to get the final refined graph $\mathbf{A}^* = \text{Refine}(\mathbf{A}, \mathbf{A}_0)$ as the input of f.

The third term of $\mathcal{L}_{HVO}$ is to minimize the KL-diverngence of the variational distribution $q(\mathbf{Z}|\Phi; \psi)$ and real hidden factor distribution $p(\mathbf{Z})$ which is also unknown. Similar to VGAE, we let the $p(\mathbf{Z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. Since the $q(\mathbf{Z}|\Phi; \psi)$ is a semi-implicit variational distribution, it is not easy to convergent to a simple solution. A regularization term using the L1 norm for prior Φ is required to encourage sparsity in the matrix $\Phi\Phi^\top$. This term is motivated by the desire to ensure that the latent representation captures only the most significant interactions, reflecting the sparse nature of real-world graphs. This loss function is given by:

$$\mathcal{L}_{prior} = \text{KL}(q(\mathbf{Z}|\Phi; \psi)||\mathcal{N}(\mathbf{0}, \mathbf{I})) + \alpha\|\Phi\Phi^\top\|_1 \tag{18}$$

$\mathcal{L}_{recon}$ is to measure the matching loss between generative hidden graphs and the ground truth graphs. Even the ground truth graphs are not fully observed, instead, the observation A_0 is perturbed by some hidden factors which are included in our model, consequently, the reconstruction loss can learn the conditional distribution of the hidden graphs.

6 Experimental Settings

6.1 Datasets

Table 1: Details of graph-level datasets [29].

Dataset	Graphs	Classes	Average nodes	Features
MUTAG	188	2	17.9	7
PROTEINS	1,113	2	39.1	3
ENZYMES	600	6	32.6	3
NCI1	4,110	2	29.8	37
AIDS	2000	2	15.69	38

Table 2: Details of node-level datasets.

Dataset	Nodes	Edges	Features	Labels
Cora	2,708	5,429	1,433	7
CiteSeer	3,327	4,732	3,703	6

Graph-level Classification: 5 benchmark datasets from TUDataset [29] covering biochemical and social networks (Table 1). These span diverse domains including molecular graphs (MUTAG, NCI1), protein structures (PROTEINS, ENZYMES), and medical compounds (AIDS).

Node-level Classification: Standard citation networks Cora and CiteSeer [30] containing sparse node features. This task focuses on a single large graph with localized structural noise, where perturbations only affect k-hop neighborhoods (k=2 in our experiments).

Spatial-Temporal Classification: TUSZ EEG seizure corpus [31] containing 3,050 clinical seizures across 7 types. Following [28], we construct dynamic functional connectivity graphs from 20

EEG channels using sliding windows. The latent neural connectivity patterns critical for seizure classification must be inferred from raw signals.

6.2 Baseline Methods

In our graph classification tests, we evaluate the Graph Isomorphism Network (**GIN**) [32] with and without our hierarchical restructuring (HR), benchmarking against key models in the field. The Graph Convolutional Network (**GCN**) [18] is included as a foundational model in graph representation learning. We also consider the Graph Attention Network (**GAT**) [33], known for its attention-based layers that dynamically weigh node significance, and the Relational Graph Convolutional Network (**RGCN**) [34], which models node representations as gaussian distributions to counter adversarial attacks. Additionally, the **GCN-Jaccard** [35] method, which preprocesses networks to remove edges connecting dissimilar nodes, the **Pro-GNN** framework [4], optimizing a structural matrix through proximal gradient descent and the **BetaGNN**[36], which uses a weighted ensemble, combining any GNN with a multi-layer perceptron for preserving clean data structure and performance. For spatial-temporal dataset analysis, our comparisons include baselines such as Support Vector Machines (**SVM**), Convolutional Neural Networks (**CNN**) based model namely SeizureNet[37], **GNN** based model following [38], and the **Transformer** based model proposed by [39]. The details of hypeparameter settings can be found in supplementary.

7 Performance Results and Analysis

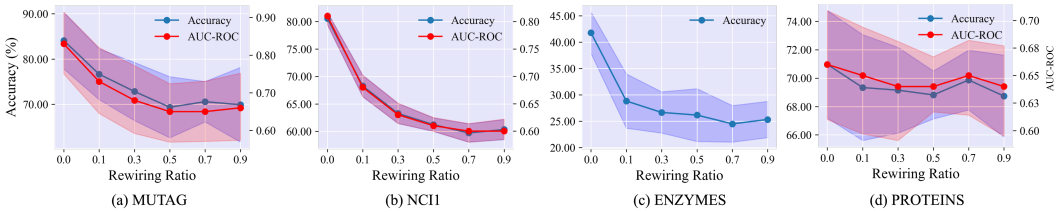


Figure 4: Test accuracy and AUC-ROC versus graph rewiring ratio for GCN, showing an overall inverse relationship—select examples are shown; see Table 3 for full results.

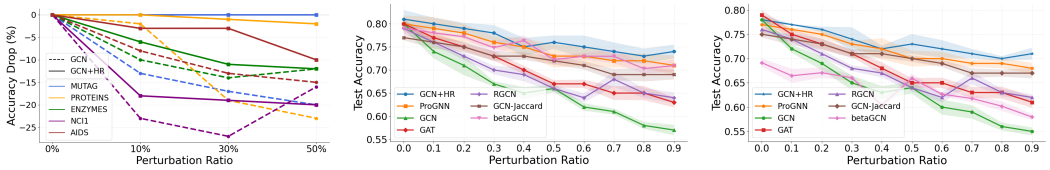
Figure 4 shows performance under increasing rewiring ratios $r = |E_R|/|E|$, where E_R denotes the set of rewired edges ($E_R \cap E = \emptyset$). Most datasets degrade rapidly under perturbation; for instance, NCI1 and ENZYMES experience near-random accuracy at just 10% rewiring, demonstrating the fragility of learned representations to structural corruption.

7.1 Robustness Analysis in Graph Classification

Graph classification demands representations that capture both local and global structures, node attributes, and their interactions [40, 41, 42]. Figure 5(a) compares standard GCN and our HR-enhanced GCN under four noise levels (0%, 10%, 30%, 50% rewiring), demonstrating that GCN-HR consistently achieves higher accuracy with smaller drops under increasing perturbation. Notably, on MUTAG, GCN-HR sustains over 87% accuracy at 50% rewiring—far exceeding the 70.7% of the clean-graph GCN—suggesting that HR not only bolsters robustness but also uncovers more informative latent structures. Across all datasets, GCN-HR exhibits greater stability, underscoring its ability to preserve task-relevant graph signals despite structural noise.

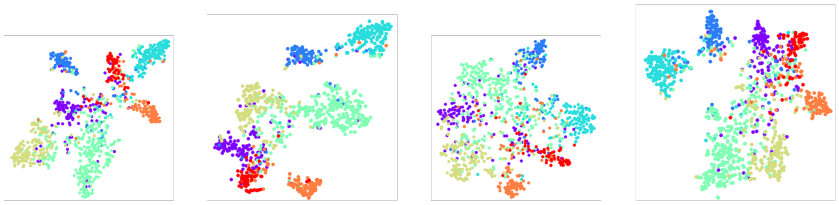
Table 3: Performance comparison of GCN and GCN+HR on various graph classification datasets under multiple perturbation rates (0%, 10%, 30%, 50%).

Dataset	0%		10%		30%		50%	
	GCN	**HR**	**GCN**	**HR**	**GCN**	**HR**	**GCN**	**HR**
MUTAG	70.7 ± 6.89	**87.93** ± 4.91	58.59 ± 10.02	**87.85** ± 3.21	54.73 ± 9.64	**87.71** ± 4.43	50.52 ± 12.42	**88.27** ± 4.23
PROTEINS	73.28 ± 3.22	**75.52** ± 3.33	68.09 ± 2.44	75.41 ± 3.83	61.77 ± 2.80	72.64 ± 5.42	60.21 ± 4.23	73.15 ± 4.71
ENZYMES	31.72 ± 4.54	**32.48** ± 4.20	22.11 ± 4.75	**26.86** ± 3.27	18.22 ± 2.61	**21.86** ± 5.94	19.67 ± 2.15	**20.35** ± 4.17
NCI1	76.85 ± 2.78	**77.75** ± 1.57	54.24 ± 1.87	**59.89** ± 1.4	49.86 ± 0.72	**58.47** ± 1.85	61.29 ± 1.36	**57.82** ± 1.19
AIDS	90.05 ± 2.27	**91.31** ± 1.66	82.46 ± 2.11	**87.82** ± 1.93	77.43 ± 3.31	**87.82** ± 2.37	75.13 ± 1.78	**81.19** ± 2.06



(a) Graph-level tasks (b) Node-level task on Cora (c) Node-level task on CiteSeer

Figure 5: Performance comparison of GCN, GCN+HR, and various methods on five graph-level classification datasets (perturbation rates from 0% to 50%) and two node-level tasks under perturbation rates from 0% to 90%.



(a) GCN (0%) (b) GCN+HR (0%) (c) GCN (70%) (d) GCN+HR (70%)

Figure 6: Comparative 2D t-SNE visualizations of node representations on the Cora dataset using standard GCN and GCN+HR at perturbation ratios of 0% and 70%.

7.2 Robustness Analysis in Semi-Supervised Node Classification

We assess GCN-HR on semi-supervised node classification—leveraging both node features and topology with only partial labels—by varying graph rewiring ratios. As shown in Figure 5(b)–(c), GCN-HR consistently outperforms baselines, particularly under severe perturbation, where its hierarchical gating effectively suppresses structural noise and yields narrow confidence intervals indicative of stable runs. On Cora, GCN-HR maintains high accuracy across all rewiring levels; on the sparser CiteSeer graph, the margin narrows and GCN-Jaccard performs comparably, suggesting that simple similarity defenses may suffice in such settings. Complementary t-SNE plots in Figure 6 reveal that even with 70% rewiring, GCN-HR preserves clear class clusters, underscoring its robustness to extreme structural noise.

7.3 Extension to Spatial–Temporal Graph Classification

Our HR framework seamlessly extends to spatial–temporal graphs by refining latent connectivity across time, leading to notably sharper class separation and fewer misclassifications in seizure detection (Figures 7–8). These improvements—evident in darker diagonal entries of confusion matrices and more distinct t-SNE clusters—demonstrate HR's effectiveness in capturing dynamic structural patterns; additional implementation details are in the supplementary materials.

8 Conclusions

We introduce a hierarchical restructuring (HR) framework that enhances Graph Neural Networks' resilience to incomplete and dynamically perturbed graphs by (1) relaxing node-dependence assumptions in graph autoencoders to capture richer structural relationships, (2) integrating a novel variational lower bound for unified, end-to-end GNN optimization, and (3) demonstrating substantial empirical gains—up to 21% higher node-classification accuracy on Cora and CiteSeer under 90% perturbation and up to 38% improvement in graph classification on MUTAG, PROTEINS, HIV, ENZYMES, and AIDS at 50% perturbation—across both static and temporal tasks.

References

[1] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[2] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In *International conference on machine learning*, pages 1972–1982. PMLR, 2019.

[3] Jiarong Xu, Junru Chen, Siqi You, Zhiqing Xiao, Yang Yang, and Jiangang Lu. Robustness of deep learning models on graphs: A survey. *AI Open*, 2:69–78, 2021. ISSN 2666-6510. doi: https://doi.org/10.1016/j.aiopen.2021.05.002. URL https://www.sciencedirect.com/science/article/pii/S2666651021000139.

[4] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 66–74, 2020.

[5] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[6] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 779–787, 2021.

[7] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Hkx1qkrKPr.

[8] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR, 2018.

[9] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

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

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

[12] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023, 2016.

[13] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1416–1424, 2018.

[14] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[15] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[16] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34 (4):18–42, 2017.

[17] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

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

[19] Bingbing Xu, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng. Graph wavelet neural network. *arXiv preprint arXiv:1904.07785*, 2019.

[20] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. Deep graph structure learning for robust representations: A survey. *arXiv preprint arXiv:2103.03036*, 14, 2021.

[21] Yao Ma, Suhang Wang, Tyler Derr, Lingfei Wu, and Jiliang Tang. Graph adversarial attack via rewiring. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 1161–1169, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383325. doi: 10.1145/3447548.3467416. URL https://doi.org/10.1145/3447548.3467416.

[22] Xuanqing Liu, Si Si, Jerry Zhu, Yang Li, and Cho-Jui Hsieh. A unified framework for data poisoning attack to graph-based semi-supervised learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/5cde6dedeb8892e3794f22db57ada073-Paper.pdf.

[23] Felix Mujkanovic, Simon Geisler, Stephan Günnemann, and Aleksandar Bojchevski. Are defenses for graph neural networks robust? In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 8954–8968. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/3ac904a31f9141444009777abef2ed8e-Paper-Conference.pdf.

[24] Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Xingcheng Fu, Cheng Ji, and S Yu Philip. Graph structure learning with variational information bottleneck. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 4165–4174, 2022.

[25] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. Robust graph representation learning via neural sparsification. In *International Conference on Machine Learning*, pages 11458–11468. PMLR, 2020.

[26] Rui Zhang, Yunxing Zhang, Chengjun Lu, and Xuelong Li. Unsupervised graph embedding via adaptive graph learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (4):5329–5336, 2022.

[27] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

[28] Zhengdao Li, Kai Hwang, Keqin Li, Jie Wu, and Tongkai Ji. Graph-generative neural network for eeg-based epileptic seizure detection via discovery of dynamic brain functional connectivity. *Scientific Reports*, 12(1):18998, 2022.

[29] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

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

[31] Vinit Shah, Eva Von Weltin, Silvia Lopez, James Riley McHugh, Lillian Veloso, Meysam Golmohammadi, Iyad Obeid, and Joseph Picone. The temple university hospital seizure detection corpus. *Frontiers in neuroinformatics*, 12:83, 2018.

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

[33] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[34] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1399–1407, 2019.

[35] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples on graph data: Deep insights into attack and defense. *arXiv preprint arXiv:1903.01610*, 2019.

[36] Haci Ismail Aslan, Philipp Wiesner, Ping Xiong, and Odej Kao. β-gnn: A robust ensemble approach against graph structure perturbation. In *Proceedings of the 5th Workshop on Machine Learning and Systems*, pages 168–175, 2025.

[37] Umar Asif, Subhrajit Roy, Jianbin Tang, and Stefan Harrer. Seizurenet: Multi-spectral deep feature learning for seizure type classification. In *Machine Learning in Clinical Neuroimaging and Radiogenomics in Neuro-oncology: Third International Workshop, MLCN 2020, and Second International Workshop, RNO-AI 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 4–8, 2020, Proceedings 3*, pages 77–87. Springer, 2020.

[38] Siyi Tang, Jared Dunnmon, Khaled Kamal Saab, Xuan Zhang, Qianying Huang, Florian Dubost, Daniel L. Rubin, and Christopher Lee-Messer. Self-supervised graph neural networks for improved electroencephalographic seizure analysis. In *ICLR*. OpenReview.net, 2022.

[39] Jianzhuo Yan, Jinnan Li, Hongxia Xu, Yongchuan Yu, and Tianyu Xu. Seizure prediction based on transformer using scalp electroencephalogram. *Applied Sciences*, 12(9):4158, 2022.

[40] Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. In *International conference on machine learning*, pages 7134–7143. PMLR, 2019.

[41] Jiaxuan You, Jonathan M Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10737–10745, 2021.

[42] Zhengyang Wang and Shuiwang Ji. Second-order pooling for graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[43] Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In *International conference on machine learning*, pages 324–333. PMLR, 2016.

A Technical Appendices and Supplementary Material

A.1 Hyperparameter Settings

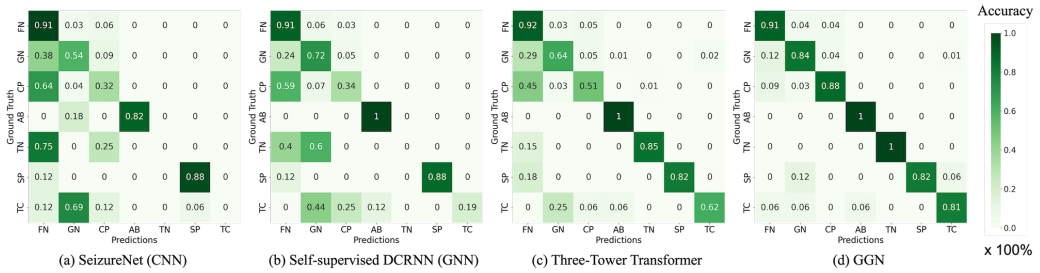
In our experiments covering both graph and node classification, we systematically explore the effects of perturbation rates, varying from 0% to 90% in increments of 10%. This perturbation analysis follows the random attack methodology as introduced in [4]. For the EEG dataset, which inherently lacks a predefined structure, we incorporate our hierarchical restructuring (HR) within a CNN-based architecture to facilitate dynamic graph generation. The specific architectural details of this model are in line with the GGN model [28].

For graph classification tasks, our experiments are conducted on a rigorously structured benchmark platform, encompassing risk assessment and model selection frameworks, alongside a 10-fold cross-validation strategy to ensure robustness and reproducibility of results. In the context of node classification within three citation networks, we maintain consistency with previously established hyperparameters for baseline models. However, we uniquely augment both the GCN and GAT models with our HR, aiming to assess the enhancements brought about by our approach. Further details regarding the hyperparameter settings for all experiments are available in our open-source code repository.

A.2 Extend HR in Spatial-Temporal Graph Classification

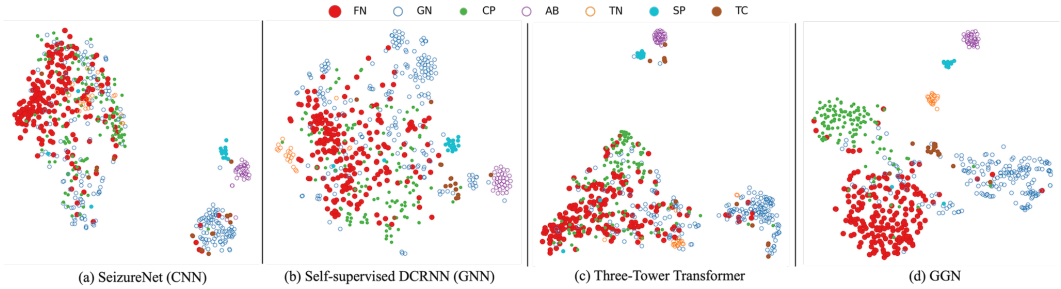
In the domain of epileptic seizure classification, where latent graph structures evolve over time yet remain unobserved, our hierarchical restructuring (HR) method proves pivotal. These structures, crucial for depicting the dynamic functional connectivity across brain regions, vary significantly across seizure types. Accurately capturing these variations is imperative. We compare our HR-augmented approach against established methods, demonstrating that our technique excels in generating representative spatial-temporal features.

Expanding on the GGN model [28], we integrate our HR module, maintaining the original architecture's CNN-based temporal encoder and GNN-based spatial decoder. The HR provides a nuanced support structure for the temporal features, enhancing the model's interpretative power. Refer to [28] for an in-depth architecture exploration. **Confusion Matrix Analysis.** In Figure 7, we analyze confusion matrices for four deep learning methods applied to seven seizure types. These



(a) SeizureNet (CNN) (b) Self-supervised DCRNN (GNN) (c) Three-Tower Transformer (d) GGN

Figure 7: Confusion matrices of seizure classification results. (a) SeizureNet (CNN) with notable misclassifications. (b) Self-supervised DCRNN (GNN) also struggles with accuracy. (c) The Three-Tower Transformer model improves classification. (d) GGN with HR achieves high accuracy with minimal confusion.



(a) SeizureNet (CNN) (b) Self-supervised DCRNN (GNN) (c) Three-Tower Transformer (d) GGN

Figure 8: Comparative 2D t-SNE visualization of graph representations for seizure classification across four different models. (a). SeizureNet (CNN) shows mixed classes. (b). Self-supervised DCRNN (GNN) has slightly improved separation. (c). Three-Tower Transformer provides better class distinction. (d). GGN with HR exhibits the most discriminant and separable class representations, outperforming the others.

methods include SeizureNet (CNN-based), Self-supervised DCRNN (GNN-based), Three-Tower Transformer, and our GGN with HR. The matrices (parts (a) to (d)) use color intensity to indicate detection accuracy, with darker shades denoting higher accuracy.

These matrices emphasize detection precision, where darker shades along the diagonal suggest accurate classification, and lighter off-diagonal shades imply misclassifications. Our GGN with HR shows notable superiority in accuracy over the others. SeizureNet's performance is hindered by static convolution kernels, leading to significant misclassifications.

In Figure 7(b), the GNN's performance is comparable to the CNN, limited by static filtering. The transformer, shown in Figure 7(c), reduces misclassifications through its attention mechanism and enhanced feature dimensions. Notably, all models accurately classify AB attacks, while FN, CP, and TN attacks are more prone to misclassification, often as FN. The transformer demonstrates fewer errors compared to CNN and GNN models, highlighting the effectiveness of our GGN in seizure detection.

t-SNE Visualization. The t-SNE technique is employed to project high-dimensional data into a two-dimensional space, facilitating intuitive visualization. In Figure 8, we introduce two dominant composite features: tSNE1 and tSNE2. Each seizure attack type is denoted by unique symbols and colors, with their proximity indicating similarity—closer points represent greater resemblance.

This dimensionality reduction method approximates the probability q_{ij} of similarity between feature i and feature j. The right-hand plots in Figure 8(a, b) show dense overlap among samples, implying confusion among certain seizure types. CNN and GNN models struggle to differentiate FN, TN, and CP attack types, as indicated by their clustering. In contrast, Figure 8(d) showcases the GGN method's distinct advantage, with minimal confusion evident along the main diagonal.

In summary, the GGN method demonstrates exceptional capability in distinguishing between attack types, a fact corroborated by testing on 1,014 validation cases. The four seizure detection methods

rank from most to least accurate as follows: GGN, Transformer, GNN, and CNN, confirming the efficacy of our proposed approach.

A.3 Detailed Explanation of Training Strategy

Our training strategy, as articulated in Algorithm 1, is fundamentally grounded in the hierarchical restructuring framework. This approach utilizes stochastic gradient descent (SGD) combined with reparameterization tricks, drawing upon methodologies from advanced machine learning research [27, 43]. The strategy is versatile, accommodating various tasks like graph and node classification with only minor adjustments required, especially concerning the loss function tailored for node-wise predictions.

The crux of our method lies in the interplay between the graph generation process, as detailed in Algorithm 2, and the optimization steps in Algorithm 1. The unique aspect of our approach is the dual-condition sampling methodology employed in Algorithm 2. This dual-mode sampling encompasses both explicit and implicit conditions, offering flexibility in handling different types of graph data.

In explicit condition sampling, latent variables $\mathbf{Z}$ and the refined graph $\mathbf{A}^*$ are directly drawn based on the specified conditions using Equations 11, 12, and 13. Conversely, the implicit condition relies on a more subtle approach, utilizing Equations 8, 9, and 10 to infer the latent variables and generate the graph. This dual-mode sampling is pivotal for adapting to various graph structures and dynamics, ensuring the robustness of our model.

During the training, as depicted in Algorithm 1, the model iteratively optimizes the parameters ψ, λ, and θ. The algorithm first generates modified adjacency matrices $\mathbf{A}_i$ through the Sampling procedure of Algorithm 2, which is intricately designed to consider both explicit and implicit conditions of the graph structure. This sampling is crucial for capturing the underlying graph dynamics and perturbations effectively.

The subsequent steps involve feeding these sampled graphs into a GNN to produce node embeddings and graph-level representations, which are then used for prediction. The optimization process is conducted in two phases: the first phase focuses on optimizing ψ and λ, and the second phase on optimizing θ. This phased approach, aided by the dual-condition sampling, allows for a more nuanced and effective learning of the graph structure and dynamics.

The hyperparameters S_1 and S_2 play a critical role in balancing the optimization of structure and parameters. Typically, setting S_1 higher than S_2 ensures a faster convergence while maintaining the structural integrity of the graph. This balance is critical for achieving a well-structured and high-performing model, as reflected in our extensive experimental validations.

In summary, the synergy between the hierarchical restructuring process (Algorithm 2) and the training optimization steps (Algorithm 1) underpins the success of our framework. This approach not only enhances the robustness and accuracy of GNNs in processing complex graph data but also sets a new benchmark in the field of graph representation learning.

Algorithm 1: Model Training based on hierarchical restructuring

Input: Dataset $\mathbb{D} = \{\mathbf{X}_i, \mathbf{A}_{i0}, y_i\}_{i=1}^n$, init. params $\{\psi, \lambda, \theta\}$, S_1, S_2
Output: Optimized params $\{\psi^*, \lambda^*, \theta^*\}$

while *not converged* **do**
  Batch $\leftarrow \{\mathbf{X}_i, \mathbf{A}_{0i}, y_i\}_{i=1}^B$
  // (1) Optimize ψ, λ
  for $s = 1$ **to** S_1 **do**
    for $i = 1$ **to** B **do**
      $\mathbf{A}_i \leftarrow \text{HR}(\psi, \lambda, \mathbf{X}_i, \mathbf{A}_{0i})$; // Alg2.
    end
    $\lambda \leftarrow \text{Optimizer}(\nabla_\lambda \mathcal{L}_{recon})$
    $\psi \leftarrow \text{Optimizer}(\nabla_\psi (\mathcal{L}_{recon} + \mathcal{L}_{prior}))$
  end
  // (2) Optimize θ
  for $s = 1$ **to** S_2 **do**
    for $i = 1$ **to** B **do**
      $\mathbf{A}_i \leftarrow \text{HR}(\psi, \lambda, \mathbf{X}_i, \mathbf{A}_{0i})$; // Alg2.
      $\mathbf{H}_i \leftarrow f_\theta(\mathbf{X}_i, \mathbf{A}_i)$
      $\overline{y}_i \leftarrow \text{Readout}(\mathbf{H}_i)$
    end
    $\theta \leftarrow \text{Optimizer}(\nabla_\theta \mathcal{L}_{pred})$
  end
end

Algorithm 2: hierarchical restructuring (HR)

Input: Parameters ψ, λ, features $\mathbf{X}$, initial adjacency matrix $\mathbf{A}_0$
Output: Refined graph $\mathbf{A}^*$

Draw prior $\Phi \sim q(\Phi; \psi)$; // Eq. (7)
if *using implicit distribution* **then**
  // Sample latent variables:
  Draw $\mathbf{Z} \sim q(\mathbf{Z}|\Phi; \lambda)$; // Eq. (8) and Eq. (9).
  // Sample graph:
  Draw $\mathbf{A} \sim q(\mathbf{A}|\mathbf{Z}, \mathbf{X}; \lambda)$; // Eq. (10).
else
  // Sample latent variables:
  Draw $\mathbf{Z} \sim q(\mathbf{Z}|\Phi; \lambda)$; // Eq. (11) and Eq. (12)
  // Sample graph:
  Draw $\mathbf{A} \sim q(\mathbf{A}|\mathbf{Z}, \mathbf{X}; \lambda)$; // Eq. (13)
end
// Refine initial graph:
$\mathbf{A}^* \leftarrow \text{UPDATE}(\mathbf{A}, \mathbf{A}_0)$
return $\mathbf{A}^*$

NeurIPS Paper Checklist

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

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

- You should answer [Yes] , [No] , or [NA] .

- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

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

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

1. **Claims**

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

   Answer: [Yes]

   Justification: We claim it in the abstract.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We claim it in the supernumeraries.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.

- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: We provide it in the Sec. 3.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer:[Yes]

   Justification: We introduce the settings in the Sec. 6.

   Guidelines:

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

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: We provide it in the github repository link.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We introduce the settings in the Sec. 6.

   Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: We provide it in the figure 4 and 5.

Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We provide it in the supplementaries.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have read it thoroughly.

   Guidelines:

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

10. **Broader impacts**

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

Answer: [Yes]

Justification: We discuss it in the abstract, introduction and conclusion sections.

Guidelines:

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

11. **Safeguards**

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

Answer: [NA] .

Justification: Our scenarios not include such risks.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: We credit it the references.

Guidelines:

- The answer NA means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes] .

    Justification: We introduce the code in the github repository link.

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

    Answer: answerNA

    Justification: We don't involve crowdsourcing nor research with human subjects.

    Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [NA]

    Justification: We don't involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

Answer: [NA]

Justification: Our research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

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