algorithms

Article

TRed-GNN: A Robust Graph Neural Network with Task-Relevant Edge Disentanglement and Reverse Process Mechanism

Menghui Xu [1], Yang Yan [2,*], Qiuyan Wang [3,4], Hanning Chen [1,5,*] and Zhao Zhang [6]

[1] School of Artificial Intelligence, Tiangong University, Tianjin 300387, China
[2] School of Information Technology and Engineering, Tianjin University of Technology and Education, Tianjin 300222, China
[3] School of Computer Science and Technology, Tiangong University, Tianjin 300387, China
[4] Fujian Key Laboratory of Financial Information Processing, Putian University, Putian 351100, China
[5] College of Artificial Intelligence, Tianjin University of Science and Technology, Tianjin 300457, China
[6] College of Mechanical Engineering, Tianjin University of Science and Technology, Tianjin 300457, China
* Correspondence: yangyan@tute.edu.cn (Y.Y.); chenhanning@tiangong.edu.cn (H.C.)

Abstract

Graph Neural Networks (GNNs) capture complex information in graph-structured data by integrating node features with iterative updates of graph topology. However, they inherently rely on the homophily assumption—that nodes of the same class tend to form edges. In contrast, real-world networks often exhibit heterophilous structures, where edges are frequently formed between nodes of different classes. Consequently, conventional GNNs, which apply uniform smoothing over all nodes, may inadvertently aggregate both task-relevant and task-irrelevant information, leading to suboptimal performance on heterophilous graphs. In this work, we propose TRed-GNN, a novel end-to-end GNN architecture designed to enhance both the performance and robustness of node classification on heterophilous graphs. The proposed approach decomposes the original graph into a task-relevant subgraph and a task-irrelevant subgraph and employs a dual-channel mechanism to independently aggregate features from each topology. To mitigate the interference of task-irrelevant information, we introduce a reverse process mechanism that, without compromising the main task, extracts potentially useful information from the task-irrelevant subgraph while filtering out noise, thereby improving generalization and resilience to perturbations. Theoretical analysis and extensive experiments on multiple real-world datasets demonstrate that TRed-GNN not only achieves superior classification performance compared to existing methods on most benchmarks, but also exhibits strong adaptability and stability under graph structural perturbations and over-smoothing scenarios.

Keywords: graph neural networks (GNNs); heterophilous graphs; disentangled representation learning; reverse process

check for updates

Academic Editor: James Jianqiao Yu

Received: 25 August 2025
Revised: 2 October 2025
Accepted: 6 October 2025
Published: 8 October 2025

Citation: Xu, M.; Yan, Y.; Wang, Q.; Chen, H.; Zhang, Z. TRed-GNN: A Robust Graph Neural Network with Task-Relevant Edge Disentanglement and Reverse Process Mechanism. *Algorithms* **2025**, *18*, 632. https://doi.org/10.3390/a18100632

1. Introduction

Graph-structured data are ubiquitous in real-world scenarios. Examples include social networks, which store and represent interpersonal relationships, interactions, and information flow [1], as well as biological networks, which describe complex relationships and interactions within biological systems [2]. Due to the inherent complexity of graph structures and the irregularity of relationships between nodes, learning from graph data has long been an important research focus. Graph Neural Networks (GNNs), particularly Graph Convolutional Networks (GCNs) have emerged as powerful algorithms for processing graph-structured data, capable of capturing rich graph representations by aggregating information from neighboring nodes. Owing to their remarkable performance, GNNs have been widely applied to a variety of graph-based learning tasks, such as node classification [3], link prediction [4], and graph classification [5].

However, many GCNs are designed under the homophily assumption, wherein nodes of the same class tend to be connected by edges. Some studies have shown that GCNs can also perform well on heterophilous graphs with low homophily rates. To learn node representations in heterophilous graphs, GNNs often need to capture long-range interactions between nodes, which typically require stacking multiple message-passing layers [6]. Nevertheless, both empirical and theoretical studies have demonstrated that GNNs tend to over-smooth node representations across layers, significantly limiting their generalization ability beyond homophily [7]. Recently, GCNs have been extended to heterophilous graphs. For example, GCN-IED updates the graph topology to introduce direct edges and employs scalable neighborhood aggregation to explore latent edges from multi-hop neighbors [8]. Similarly, LAAH leverages label information from different neighborhoods to guide adaptive aggregation. However, such approaches cannot fully avoid the influence of inter-class information, which may be inadvertently aggregated into the representations of nodes from different classes, thereby causing class indistinguishability [9].

One potential reason for the suboptimal performance of GNNs on heterophilous graphs lies in the mismatch between node labels and graph links. The former serves as the target for the classification task that GNNs aim to predict, while the latter determines how messages are propagated between nodes to achieve this objective. In homophilous graphs, the two are closely aligned, as most connected nodes belong to the same class. However, in heterophilous graphs, the motivation for establishing connections between nodes may be ambiguous with respect to the classification task. For example, in social networks, numerous connections exist between users with different interests, backgrounds, or professions [10]. This means that, during information propagation over the network, harmful or distracting information may be introduced. Existing state-of-the-art GCN models often fail to fully identify and differentiate such irrelevant or harmful information in the local neighborhood, making node representations prone to entanglement with incorrect signals and thus reducing robustness. An effective classification design should be able to identify task-irrelevant connections while extracting the most relevant information for prediction. However, in real-world datasets, node features are often noisy, and the connections between nodes may not reflect their class relationships. Existing techniques typically parameterize the weights of node pairs to indicate similarity or dissimilarity, but such approaches do not effectively assess the correlation between node connections and downstream task objectives.

In this paper, we propose an end-to-end node classification framework named TRed-GNN. We assume that the primary reason two nodes are connected is their similarity in features, and such features are not necessarily related to the target task. Some studies have divided the edges of a graph into two complementary sets, each representing potential relationships between nodes [11]. However, this division is not entirely accurate, as there may still exist useful information in connections between two task-irrelevant nodes. Therefore, we categorize node connections as task-relevant or task-irrelevant. In the implementation, we design a residual mechanism to distinguish between these types of edges. Node features are then aggregated separately over these connections to produce disentangled representations. Furthermore, a reverse process is introduced to extract latent information from task-irrelevant connections. To theoretically support our proposed algorithm, we conduct extensive experiments on real-world datasets and provide a detailed analysis of the feasibility of our model. In summary, the main contributions of this work are as follows:

- We propose TRed-GNN, a novel framework for node classification. By disentangling graph edges into task-relevant and task-irrelevant subgraphs, TRed-GNN mitigates noise from edge heterogeneity and processes information through independent channels. This dual-path design improves classification accuracy, especially on graphs with complex heterogeneous structures.
- To handle edge heterogeneity and node-level noise, TRed-GNN introduces a reverse process on the task-irrelevant subgraph. This mechanism recovers useful information while suppressing noise, effectively alleviating over-smoothing.
- We conducted systematic experiments on multiple real-world datasets (e.g., Cora, Citeseer, Chameleon) to evaluate TRed-GNN, demonstrating its superior performance over existing graph neural network methods across graphs with varying levels of homogeneity and heterogeneity.

2. Related Work

In this section, we review the fundamental research on Graph Neural Networks (GNNs) and their extensions to heterophilous graph scenarios. We also discuss recent advances in task-relevance modeling and disentangled representation learning and analyze the causes of the over-smoothing problem along with existing mitigation strategies. These studies lay the theoretical foundation for the proposed TRed-GNN framework and highlight our innovations in task-relevance modeling and robustness enhancement. As shown in Table 1, by comparing with existing methods, we clearly demonstrate the unique mechanism and advantages of the TRed GNN model in dealing with these problems.

Table 1. Comparative table: task relevance, heterophily handling, and over-smoothing mitigation.

Methods	**Task Relevance Handling**	**Heterophily Handling**	**Over-Smoothing Mitigation**
GCN	Does not explicitly handle task relevance. All neighbors are treated equally.	Assumes homophily (same class nodes are connected), struggles with heterophily.	Over-smoothing becomes a major issue in deeper layers.
GAT	Uses attention to weigh the importance of neighbors, but does not separate task-relevant and irrelevant edges.	Self-attention helps focus on more relevant neighbors, but still fails with heterophily in certain cases.	Attention may reduce over-smoothing, but it is still problematic with deep layers.
FactorGNN	Factorizes node representations, but does not explicitly separate task-relevant information.	Handles heterophily with factorization, but no direct mechanism for isolating task-irrelevant information.	Does not address over-smoothing effectively.
MixHop	Mixes features from different neighborhood levels, but task relevance is not explicitly modeled.	Captures higher-order neighbors but may struggle with heterophily where connections do not align with node labels.	The mixing of high-order features leads to over-smoothing as layers increase.
FAGCN	Uses frequency-based aggregation to improve task relevance, but lacks explicit separation of relevant and irrelevant edges.	Improves handling of heterophilous graphs but does not model task relevance clearly.	Frequency aggregation reduces over-smoothing but can still be an issue with deeper layers.
ACM-GNN	Adapts the aggregation process but does not explicitly split task-relevant from irrelevant edges.	Self-adaptive channel mixing improves performance on heterophilous graphs but does not isolate irrelevant edges.	Combines different filter types but does not completely mitigate over-smoothing.
TRed-GNN	Explicitly separates task-relevant and task-irrelevant edges, ensuring that only relevant information is used.	Uses a reverse process to recover useful information from task-irrelevant edges, specifically designed for heterophilous graphs.	Uses a reverse diffusion process to recover information and prevent over-smoothing, ensuring effective message passing even in deep layers.

2.1. Graph Neural Networks

GNNs are a class of neural networks that operate directly on graph structures. The core idea of most GNNs is to utilize the information from a node's neighborhood to construct its task-specific representation. Based on this principle, numerous variants have been developed, such as GCN [12], GAT [13], and GraphSAGE [14]. Among them, Graph Convolutional Networks primarily operate in the spectral and spatial domains. The spectral approach defines graph convolution as a filtering operation on graph signals in the frequency domain [15], while the spatial approach interprets it as an aggregation operation between nodes [16]. In the spectral domain, convolution operations are defined via the graph Fourier transform based on spectral graph theory. For example, BernNet employs Bernstein polynomials to construct graph filters capable of learning arbitrary filter types, enabling convolution to better adapt to different graph structures [17]. ChebNet further approximates graph convolution using Chebyshev polynomials, which allows efficient signal processing in the frequency domain while avoiding the computationally expensive eigen-decomposition operation [18]. On the other hand, spatial-based GNNs perform convolution operations directly on the spatial structure of the graph. For instance, GAT applies an attention mechanism to learn the weights between each node and its neighbors, automatically assigning importance to different neighbors [13]. GraphSAGE proposes a framework for sampling and aggregating neighbor features [14]. However, these methods are generally designed under the homophily assumption and largely ignore the structural relationships in heterophilous graphs, which can lead to non-smooth variations of node features and labels across the graph.

2.2. GNN for Heterophilous Graphs

In recent years, GCNs have been increasingly extended to heterophilous graphs, where nodes tend to connect with others that have dissimilar features. Heterophilous graphs emphasize the dissimilarity between a node and its neighbors, indicating that edges may form between nodes of different types. To adapt GNNs to heterophilous graphs, several studies have exploited remote information beyond the immediate neighborhood. For example, Geom-GCN proposes a two-level aggregation method that leverages geometric neighborhood selection and multi-region information aggregation to address the issue of information mixing in conventional GNNs on heterophilous graphs [19]. H2GCN incorporates higher-order neighborhood aggregation and skip connections to better accommodate heterophilous structures [20]. ACM-GNN enhances the flexibility of convolution through an adaptive channel-mixing mechanism, enabling the model to select appropriate information propagation strategies based on the characteristics of the graph, thereby improving performance across different types of graphs [21]. GEN estimates a graph structure suitable for GNN learning to compute node embeddings, incorporating multi-order neighborhood information and using Bayesian inference as guidance [22].

However, these methods often overlook the underlying motivation for the connection between two nodes and fail to associate such connections with the learning task. This motivates the need for a model that explicitly distinguishes between task-relevant and task-irrelevant connections. Such a design would allow us to leverage this information as guidance, separating and extracting additional information from task-irrelevant connections with respect to the final prediction objective, thereby improving the performance of GNNs on heterophilous graphs.

2.3. Consider Task Relevance and Disentanglement Representation Learning

The concept of task relevance has been employed in many areas of GNN research to address various problems, such as contrastive learning [23], structure learning [24,25], and topology denoising [26,27]. Given that this work focuses on classification tasks, we next review studies in this domain to highlight the unique contribution of TRed-GNN to task-relevance modeling in GNN development. For example, GCN-LPA introduces task-relevant structural information via label propagation to improve classification performance and interpretability [28]. PGIB incorporates prototype learning into the information bottleneck framework to identify critical subgraphs in the input graph that are relevant to the classification task, using them as key prototypes [29]. These methods focus on selecting task-relevant edges to facilitate the extraction of task-related information, but they lack mechanisms to handle task-irrelevant information, and their modeling of task relevance is typically one-sided and static. In contrast, TRed-GNN adopts a disentangled paradigm that separates both the network topology and node features into task-relevant and task-irrelevant components. Specifically, it explicitly models task-irrelevant information to reduce noise while enhancing feature extraction within task-relevant information. By analyzing the graph topology from a more comprehensive perspective, our approach improves classification performance.

Disentangled representation learning aims to learn interpretable and independent representations, where each dimension or subspace corresponds to a distinct factor of the data [30]. In recent years, disentangled representation learning has been increasingly applied to GNNs to address graph-related tasks. For example, IPGDN enhances model performance by promoting independence among decomposition factors and integrating global graph information [31]. DisenGCN introduces a neighborhood routing mechanism that iteratively partitions each node's neighborhood into different segments, thereby disentangling node information [32]. FactorGNN takes a generative perspective, decoupling node representations into multiple factor graphs to capture higher-order semantics [33]. IDGCL employs multi-channel disentangled modeling to accurately distinguish multiple independent factors in a graph, effectively improving interpretability in graph classification tasks [34]. It is worth noting that our model shares certain similarities with FactorGNN in that both decompose the original network topology into multiple subgraphs. However, there are fundamental differences between the two. First, FactorGNN merely partitions neighboring nodes into multiple semantic factors without explicitly modeling task relevance. Second, it allows an edge to belong to multiple subgraphs, which may lead to potential overlaps. In contrast, TRed-GNN adopts an edge-separation strategy to generate two complementary subgraphs representing task-relevant and task-irrelevant topologies. It then introduces a reverse process mechanism within the subgraphs to enhance classification capability and generalization.

2.4. The Problem of Over-Smoothing in GNNs

Many empirical and theoretical studies have shown that GNNs tend to smooth node representations across layers, ultimately leading to learned representations that become overly similar. During the aggregation process, information from neighboring nodes becomes increasingly alike, and as this balance is reached, node representations become indistinguishable. However, existing studies have proposed various hypotheses to address this issue. For instance, CPGNN introduces a learnable compatibility matrix to capture information from non-adjacent but label-consistent nodes [35]. FSGNN proposes soft feature selection, which adaptively aggregates features from neighboring and multi-hop nodes [36]. LRGNN employs a low-rank approximation to compute a label relationship matrix, which is then used for signed message passing [37]. Despite these advances, many methods still cause global node representations to converge toward similarity, which degrades classification performance. To address this, Ordered GNN retains non-aggregated information by extracting and preserving it separately and proposes ordering message passing to prevent the mixing of messages from different hops [38]. FAGCN introduces a self-gating mechanism that adaptively leverages both low-frequency and high-frequency signals [39]. ACM-GCN uses a combination of low-pass and high-pass filters to adaptively capture node-level local information [21]. GRAND strengthens the understanding of over-smoothing by drawing analogies between GNN architectures and the heat diffusion equation [40]. DropEdge randomly removes edges from the graph to cut off message propagation between adjacent nodes [41]. While these methods may mitigate the over-smoothing problem to some extent, learning discriminative representations between adjacent nodes remains challenging. In our approach, we introduce a reverse process mechanism to alleviate over-smoothing. Notably, we apply this mechanism only to the task-irrelevant subgraph. This is because the task-relevant subgraph is used for the main task learning and should directly participate in forward information propagation, whereas the task-irrelevant subgraph is not intended to enhance classification performance directly. Instead, it serves to extract useful information from the latent structure and filter out noise without interfering with the main task learning, thereby improving robustness and generalization.

3. Notations and Preliminaries

In this section, we represent an undirected graph as $G = (V, E)$, where V denotes the set of nodes and $E \subseteq |V| \times |V|$ denotes the set of edges. Let $X \in \mathbb{R}^{n\times d}$ denote the node feature matrix, where d is the dimensionality of node features and n is the number of nodes. We use $X_{[i,:]}$ to denote the i-th row of X, corresponding to node v_i. Let $A \in \mathbb{R}^{n\times n}$ be the adjacency matrix of G, where $A_{i,j} = 1$ indicates that there exists an edge between node i and node j, and $A_{i,j} = 0$ otherwise. The degree matrix D is obtained by summing each row of A and placing the results along the diagonal. The adjacency matrix with self-loops is defined as $\tilde{A} = A + I$, and the corresponding degree matrix is $\tilde{D} = D + I$. In this paper, we decompose the original graph into two subgraphs: a task-relevant subgraph and a task-irrelevant subgraph, whose adjacency matrices are denoted as A_R and A_{IR}, respectively. For node classification tasks, each node is assigned with a label c_i out of $C \leq N$ classes and have a ground truth one-hot vector $y_i \in \mathbb{R}^C$. In this context, real-world graphs can be divided into homophilous and heterophilous ones based on the extent of similarity in class labels among connected nodes. The homophily ratio is shown in Table 2.

Table 2. Statistics of experimental datasets.

Dataset	Nodes	Edges	Features	Classes	Homophily (%)
Cora	2708	1433	1433	7	81
Citeseer	3327	3703	3703	6	74
Chameleon	2277	31,421	2325	5	23
Squirrel	5201	198,493	2089	5	22
Film	7600	26,752	931	5	22
Cornell	183	280	1703	5	31
Wisconsin	251	466	1703	5	21
Texas	183	295	1703	5	11

4. The Proposed Method

In this section, we present an end-to-end graph learning framework, TRed-GNN, which extends Graph Neural Networks to heterophilous graphs. An overview of the model is shown in Figure 1. The core idea is to decompose the original GNN graph structure into two subgraphs and process them separately, using the graph topology to guide task-relevant and task-irrelevant node features.

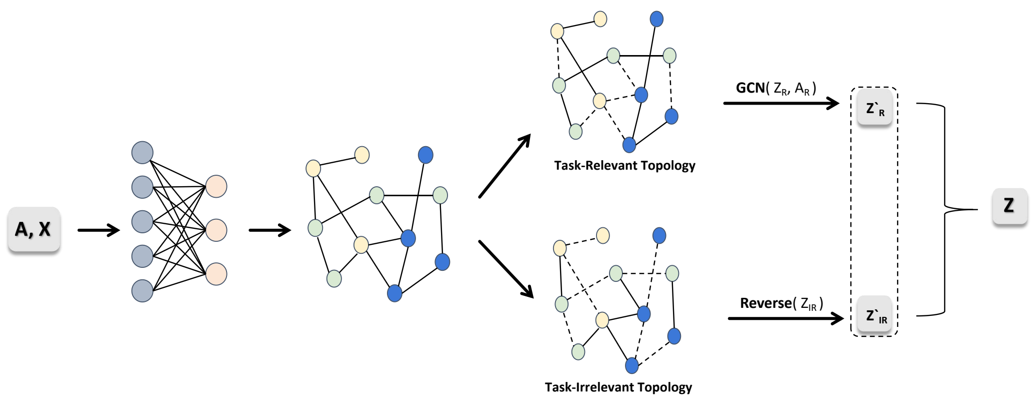


Figure 1. Illustration of TRed-GNN framework where A and X denote the adjacency matrix and feature matrix of nodes, respectively. First, X is passed through a fully connected layer to learn node representations more suitable for the task. Then, it is mapped to different latent subspaces via separate channels, R and IR. Next, an edge splitting operation is performed to divide the original graph edges into two complementary sets. Node information can then be aggregated separately on the different edge sets to produce disentangled representations for the subsequent edge splitting in the next layer. By extracting latent information from Z'_{IR}, the task is then predicted by combining it with Z'_R.

4.1. Dynamically Update Graph Topology

Before updating the graph topology, the original node features X may be irregularly distributed and not directly related to the task. Therefore, we first apply a fully connected layer to learn node representations that are more suitable for the task. Specifically, the feature matrix X is passed through a fully connected neural network (FCNN) parameterized by f_θ, as formulated below:

$$H^0 = f_\theta(X) \tag{1}$$

where f_θ denotes a FCNN, which is then applied to the feature matrix using the ReLU activation function.

Next, the updated feature matrix is fed into two separate channels to extract task-relevant and task-irrelevant information from the nodes. We project the above feature matrix into different subspaces:

$$Z_s^{(0)} = \sigma(W_s^T H^0 + b_s) \tag{2}$$

where $W_s \in \mathbb{R}^{d \times d}$ and $b_s \in \mathbb{R}^d$ are learnable parameters in channel $s \in \{R, IR\}$, d denotes the dimension of the node hidden states, and σ represents the non-linear activation function.

First, we assume that the connection between two nodes is primarily due to their similarity in certain features. However, such feature similarity may be relevant to the current learning task, irrelevant, or even harmful. Based on this assumption, we can adopt a flexible approach by assigning continuous weights ranging from 0 to 1 to soften the node connections, reflecting the varying degrees of task relevance or irrelevance of each edge. However, determining A_R and A_{IR} independently based on node similarity metrics may fail to fully capture the complex interactions between the two channels and may reduce attention to topological distinctions. To address this issue, for an edge $A_{(i,j)}$, we parameterize the difference between $A_{R(i,j)}$ and $A_{IR(i,j)}$ by solving the following equation:

$$\begin{cases} \mathbf{A}_{R(i,j)} - \mathbf{A}_{IR(i,j)} = \alpha_{i,j} \\ \mathbf{A}_{R(i,j)} + \mathbf{A}_{IR(i,j)} = 1 \end{cases} \tag{3}$$

where $A_{R(i,j)} = \frac{1+\alpha_{i,j}}{2}$ and $A_{IR(i,j)} = \frac{1-\alpha_{i,j}}{2}$, with $-1 \leq \alpha_{i,j} \leq 1$. To effectively quantify the interaction between the task-relevant and task-irrelevant aspects of each edge, we introduce a residual mechanism:

$$\alpha_{i,j} = \tanh\left(g\left[Z_{R[i,:]} \oplus Z_{IR[i,:]} \oplus Z_{R[j,:]} \oplus Z_{IR[j,:]}\right]^T\right) \tag{4}$$

where g is a learnable convolution function, and tanh is the hyperbolic tangent activation function, which constrains the output within the range $[-1, 1]$.

4.2. Neighborhood Aggregation

Since the split graph topologies reveal partial relationships between nodes in different latent spaces, they can be used to aggregate information between different nodes. Specifically, we first use GCN to aggregate task-relevant node information, as given by the following equation:

$$Z_R^{(l+1)} = \sigma\left(Z_R^{(l)} W_m^{(l)} + D^{-\frac{1}{2}} A_R D^{-\frac{1}{2}} Z_R^{(l)} W_n^{(l)}\right) \tag{5}$$

where W_m and W_n are learnable weight matrix, σ is the activation function, and D is the degree matrix associated with the adjacency matrix A.

Next, we aim to learn to recover the information from the input space based on a certain output space. However, in traditional GNNs, node representations are typically obtained by aggregating information from neighboring nodes through message passing. This process tends to push the representations of adjacent nodes toward similarity, and the over-smoothing phenomenon becomes more severe as the number of graph layers increases. To alleviate this issue, we propose using a reverse diffusion model, whose core idea is to invert the diffusion process in traditional GNNs. Specifically, the reverse diffusion process attempts to "trace back" the node representations to their pre-diffusion state, thereby avoiding over-smoothing. Based on this idea, we apply the reverse diffusion process to the task-irrelevant graph topology so that it can re-extract useful information from the topological structure. The formula is as follows:

$$Z_{IR}^{(l+1)} = \mathcal{R}(Z_{IR}^{(l)}) \tag{6}$$

where $\mathcal{R}$ is a reverse-process function. In this paper, we implement it by using an MLP as the reverse mapping function through structural design.

Next, we incorporate the useful information from task-irrelevant nodes into the task-relevant node representations through weighted fusion, as expressed by the following equation:

$$Z = \lambda Z_R + \gamma Z_{IR} \tag{7}$$

where Z_R and Z_{IR} denote the task-relevant and task-irrelevant representations, respectively, and $\lambda, \gamma \in \mathbb{R}$ are hyperparameters controlling their weights.

Finally, we use the cross-entropy loss function to compute the loss and improve the model performance by minimizing the following training objective.

$$\mathcal{L} = -\frac{1}{|\mathcal{V}_{\text{trn}}|} \sum_{v_i \in \mathcal{V}_{\text{trn}}} \mathbf{y}_i^T \log(\hat{\mathbf{y}}_i) \tag{8}$$

where v_i denotes a training node, V_{trn} is the number of training nodes, $y_i \in \mathbb{R}^C$ is the ground-truth label of the i-th node (one-hot encoded), and $\hat{y}_i \in \mathbb{R}^C$ is the predicted probability distribution of the i-th node (typically obtained via softmax), where C is the number of classes.

4.3. Computational Complexity Analysis

Let the graph be $G = (V, E)$, where the adjacency matrix $A \in \mathbb{R}^{n\times n}$ and the feature matrix $X \in \mathbb{R}^{n\times d}$. In each layer, TRed-GNN requires the following operations: (1) feature projection and the reverse process MLP with complexity $\mathcal{O}(n^2 d)$; (2) edge disentanglement with residual coefficient computation $\alpha_{i,j}$, costing $\mathcal{O}(|E|d)$; (3) message passing on both task-relevant and task-irrelevant subgraphs, with a total cost of $\mathcal{O}(|E|d)$; (4) weighted fusion, which costs only $\mathcal{O}(nd)$ and can be neglected in asymptotic analysis. Therefore, the overall time complexity per layer of TRed-GNN is $\mathcal{O}(n^2 d + |E|d)$. To assess the computational efficiency of all compared methods, we provide a comparison for the time complexities of the compared algorithms, outlined in Table 3.

Table 3. A comparison for computational complexity of all compared methods, where | E | is the number of edges.

Methods	**GCN**	**GAT**	**FAGCN**	**MixHop**	**GPR-GNN**
Time complexity	$\mathcal{O}(\lvert E\rvert d)$	$\mathcal{O}(n^2 d + \lvert E\rvert d)$	$\mathcal{O}(\lvert E\rvert d)$	$\mathcal{O}(n^2 d + \lvert E\rvert d)$	$\mathcal{O}(\lvert E\rvert d)$
Methods	**SGC**	**ACM-GNN**	**FactorGNN**	**Geom-GCN**	**TRed-GNN**
Time complexity	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2 + nd)$	$\mathcal{O}(n^2 d)$	$\mathcal{O}(n^2 d)$	$\mathcal{O}(n^2 d + \lvert E\rvert d)$

5. Experiments

Datasets: In this section, we evaluate TRed-GNN on real-world datasets. We use the following real-world datasets: Cora, Citeseer, Cornell, Chameleon, Squirrel, Wisconsin, Texas, and Film.
Data Splits: For homophilous graphs, we follow the standard setting of selecting 20 nodes per class for training, 500 nodes for validation, and 1500 nodes for testing. For heterophilous graphs, we split the data into training, validation, and test sets with ratios of 60%, 20%, and 20%, respectively.
Baselines and Implementation Details: To assess the performance of our model, we compare it against several state-of-the-art GNN models and task-specific models. Specifically, the baseline models include GCN [12], GAT [13], SGC [42], GraphSAGE [14], APPNP [43], Geom-GCN [19], ACM-GCN [21], H2GCN [20], FAGCN [39], GPR-GNN [44], LRGNN [37], and MixHop [45]. For all baselines and TRed-GNN, we set $d = 64$ as the number of hidden units to ensure a fair comparison, use Adam as the optimizer, and tune hyperparameters for each dataset using Optuna on the validation set. For the multi-layer perceptron, the hidden feature dimension is set to 512, and training is performed for 200 runs. After obtaining the optimal hyperparameters, we train the model for 1000 epochs with an early stopping strategy of 100 epochs patience. The final performance is reported as the average over 10 runs with different random data splits on the test set.

5.1. Classification Results

Table 4 presents the node classification accuracy of various models on real-world datasets. We observe that, compared to the strongest baseline model, TRed-GNN achieves improvements of 4.62 percentage points, 7.85 percentage points, 4.00 percentage points, and 6.56 percentage points on the Citeseer, Wisconsin, Texas, and Squirrel datasets, respectively. On the Cora, Chameleon, and Film datasets, it yields smaller gains of 2.03 percentage points, 2.47 percentage points, and 0.57 percentage points, respectively. However, on the Cornell dataset, its performance is lower than that of the ACM-GNN model, which we attribute to the relatively low edge density and sparser topology of Cornell, limiting our model's ability to extract sufficient useful information when partitioning the graph topology.

Table 4. Performance (%) comparison of 8 real heterophilous datasets. The best and second-best results are highlighted in bold and underlined, respectively. Error reduction gives the average improvement of TRed-GNN upon baselines w/o Basic GNNs.

Method	**Cora**	**Citeseer**	**Chameleon**	**Wisconsin**	**Texas**	**Squirrel**	**Cornell**	**Film**
GCN	79.74 ± 1.2	69.56 ± 1.5	67.69 ± 1.9	59.51 ± 3.3	61.74 ± 3.8	55.25 ± 1.5	52.85 ± 6.0	31.26 ± 1.1
GAT	79.13 ± 1.1	69.91 ± 1.2	67.96 ± 2.4	57.72 ± 4.5	55.43 ± 5.4	54.76 ± 2.2	51.24 ± 5.0	30.97 ± 1.3
SGC	82.21 ± 1.0	69.92 ± 1.9	67.34 ± 2.3	57.91 ± 3.5	55.42 ± 2.2	54.86 ± 1.3	50.42 ± 1.0	30.58 ± 0.8
GraphSAGE	86.88 ± 1.3	76.72 ± 1.4	62.24 ± 2.0	77.25 ± 3.1	71.84 ± 3.0	44.25 ± 1.1	62.24 ± 3.2	34.17 ± 1.2
APPNP	87.36 ± 0.6	75.29 ± 1.6	54.39 ± 1.9	45.69 ± 2.9	58.92 ± 2.5	35.11 ± 1.5	58.65 ± 2.6	26.53 ± 1.1
GeomGCN	84.83 ± 0.9	75.41 ± 1.3	60.92 ± 1.0	64.51 ± 4.1	68.38 ± 3.5	38.09 ± 1.8	59.45 ± 2.9	31.65 ± 1.5
ACM-GCN	86.71 ± 1.0	77.09 ± 1.7	66.47 ± 2.2	76.47 ± 5.9	74.05 ± 1.3	54.38 ± 4.9	**84.86 ± 1.0**	36.12 ± 1.1
H2GCN	81.46 ± 1.4	68.72 ± 2.0	62.91 ± 1.9	82.63 ± 4.0	79.81 ± 7.3	45.13 ± 1.9	79.62 ± 4.9	38.46 ± 1.0
FAGCN	82.65 ± 1.3	70.34 ± 1.6	68.09 ± 1.8	82.35 ± 4.4	80.35 ± 5.5	50.46 ± 2.6	79.42 ± 5.5	37.94 ± 1.4
GPR-GNN	81.51 ± 1.5	69.63 ± 1.7	69.62 ± 1.7	82.32 ± 4.1	81.76 ± 4.9	54.12 ± 1.6	79.95 ± 5.3	38.30 ± 1.1
LRGNN	72.65 ± 1.3	60.53 ± 1.1	77.16 ± 2.9	78.25 ± 3.2	71.14 ± 3.1	56.75 ± 2.2	56.86 ± 4.0	21.65 ± 1.3
MixHop	81.93 ± 0.6	71.45 ± 0.9	81.21 ± 2.2	79.81 ± 3.2	77.24 ± 1.9	55.31 ± 1.6	78.3 ± 2.4	37.32 ± 0.9
TRedGNN	**89.13 ± 1.1**	**80.65 ± 1.7**	**83.22 ± 2.1**	**86.48 ± 4.3**	**85.03 ± 4.0**	**60.47 ± 1.5**	76.12 ± 3.5	**38.68 ± 0.8**
w/o ZIR	76.65 ± 0.8	70.23 ± 1.2	73.67 ± 2.5	78.11 ± 4.5	75.41 ± 4.2	48.87 ± 1.9	67.21 ± 2.9	30.54 ± 1.2
w/o $\mathcal{R}$	83.22 ± 1.1	77.08 ± 1.4	79.12 ± 2.9	80.53 ± 5.1	78.24 ± 4.1	54.96 ± 1.4	73.32 ± 3.1	34.23 ± 1.3

Importantly, beyond mean accuracy, we also report 95% confidence intervals across 10 random runs to ensure fairness and statistical robustness in Table 5. The intervals show that the improvements in TRed-GNN over baselines are consistent and remain significant within the estimated uncertainty ranges. For example, on Citeseer and Texas, TRed-GNN not only achieves higher mean accuracy but also exhibits tighter confidence intervals, indicating both stability and reliability. This further demonstrates that the observed performance gains are not due to randomness but reflect the inherent advantage of disentangling task-relevant and task-irrelevant structures.

Table 5. Accuracy (%) with 95% confidence intervals across 10 runs. Bold indicates the best performance.

Method	**Cora**	**Citeseer**	**Chameleon**	**Wisconsin**	**Texas**	**Squirrel**	**Cornell**	**Film**
GCN	79.74 ± 0.74	69.56 ± 0.93	67.69 ± 1.18	59.51 ± 2.05	61.74 ± 2.36	55.25 ± 0.93	52.85 ± 3.72	31.26 ± 0.68
GAT	79.13 ± 0.68	69.91 ± 0.74	67.96 ± 1.49	57.72 ± 2.79	55.43 ± 3.35	54.76 ± 1.36	51.24 ± 3.10	30.97 ± 0.81
SGC	82.21 ± 0.62	69.92 ± 1.18	67.34 ± 1.43	57.91 ± 2.17	55.42 ± 1.36	54.86 ± 0.81	50.42 ± 0.62	30.58 ± 0.50
GraphSAGE	86.88 ± 0.81	72.02 ± 1.24	62.24 ± 1.49	62.71 ± 2.60	58.92 ± 1.55	55.25 ± 0.99	52.31 ± 0.74	30.86 ± 0.74
APPNP	87.36 ± 0.37	75.29 ± 0.99	54.39 ± 1.18	45.69 ± 1.80	58.92 ± 1.55	35.11 ± 1.12	58.65 ± 1.61	26.53 ± 0.68
GeomGCN	84.83 ± 0.56	75.41 ± 0.80	60.92 ± 0.62	64.51 ± 0.68	68.38 ± 2.17	38.09 ± 1.12	59.45 ± 1.80	31.65 ± 0.93
ACM-GCN	86.71 ± 0.62	77.09 ± 1.05	66.47 ± 1.36	76.47 ± 3.65	74.05 ± 0.80	54.38 ± 3.04	**84.86 ± 0.62**	36.12 ± 0.68
H2GCN	81.46 ± 0.87	78.62 ± 1.24	82.63 ± 2.48	82.63 ± 2.48	79.48 ± 2.29	50.43 ± 0.81	79.62 ± 3.04	38.46 ± 0.99
FAGCN	82.65 ± 0.81	70.34 ± 0.99	69.08 ± 1.12	65.32 ± 2.71	60.35 ± 3.41	50.46 ± 1.61	79.25 ± 3.41	37.94 ± 0.87
GPR-GNN	81.51 ± 0.93	69.63 ± 1.05	69.68 ± 1.05	82.32 ± 2.54	81.76 ± 3.05	55.16 ± 0.74	79.95 ± 3.47	38.31 ± 0.68
LRGNN	72.65 ± 0.81	70.53 ± 0.68	77.16 ± 1.80	78.25 ± 1.99	71.14 ± 1.92	56.75 ± 1.36	56.86 ± 2.48	21.65 ± 0.80
MixHop	81.36 ± 0.56	71.45 ± 0.56	81.03 ± 0.87	79.71 ± 2.60	77.24 ± 1.18	55.31 ± 1.05	62.34 ± 2.60	32.37 ± 0.56
TRedGNN	**89.13 ± 0.68**	**80.65 ± 1.05**	**83.22 ± 1.30**	**86.48 ± 2.67**	**85.03 ± 2.48**	**60.47 ± 0.93**	76.12 ± 2.17	**38.66 ± 0.50**
w/o ZIR	76.65 ± 0.50	70.23 ± 0.74	73.67 ± 1.55	78.11 ± 2.79	75.41 ± 2.61	48.87 ± 1.18	67.21 ± 1.80	30.54 ± 0.74
w/o $\mathcal{R}$	83.22 ± 0.68	77.08 ± 0.87	79.12 ± 1.80	80.53 ± 3.16	78.24 ± 2.54	54.96 ± 0.87	73.32 ± 1.92	34.23 ± 0.80

Overall, TRed-GNN demonstrates superior and statistically robust performance on most datasets, strongly indicating that our model can effectively reduce inter-class edge noise propagation during node classification.

5.2. Ablation Experiment

To evaluate the effectiveness of each module in our model, we conducted ablation experiments on TRed-GNN and its variants across eight real-world datasets. Specifically, we define two variants: (1) "w/o ZIR": without the task-irrelevant channel, and (2) "w/o

$\mathcal{R}''$: without the reverse process. From Figure 2, we can draw the following conclusions. First, when the distinction between task-relevant and task-irrelevant graph topologies is removed, the model's performance drops significantly. This confirms that incorporating the classification task into the topology design helps reduce the interference of irrelevant information, thereby improving the model's ability to focus on task-relevant information. Second, removing the reverse process also leads to a noticeable drop in accuracy, which validates that the reverse process effectively enhances graph structure learning.

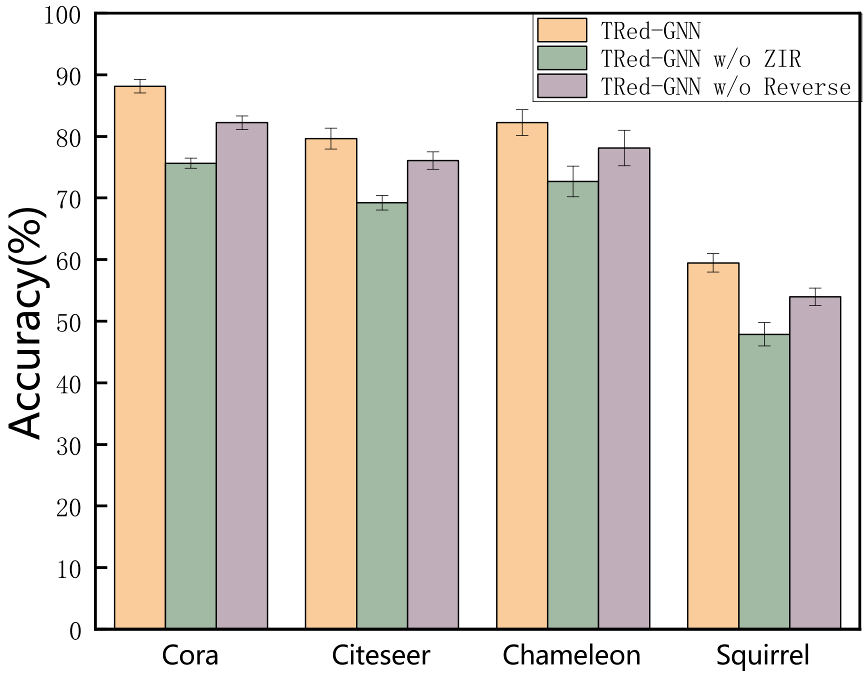
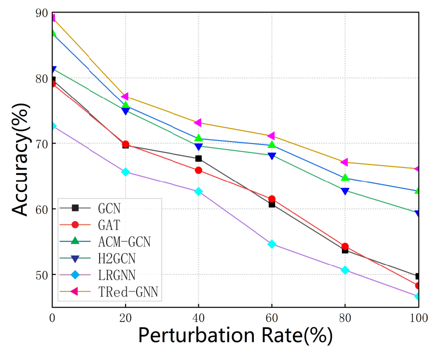


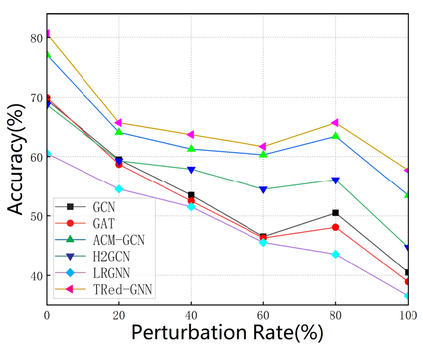
Figure 2. Ablation study of TRed-GNN on four datasets in node classification.

5.3. Robustness Analysis

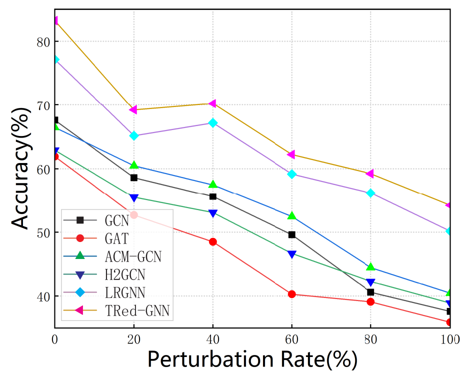
To assess the robustness of our model under graph structural perturbations, we conducted structure corruption experiments on standard datasets (Cora, Citeseer, Chameleon, and Squirrel). Specifically, we randomly rewired different proportions of edges in the original graph structure, with perturbation rates ranging from 0 percentage point to 100 percentage point in increments of 20 percentage point. This setup simulates real-world scenarios where graph structures may be incomplete, noisy, or dynamically evolving. As shown in Figure 3, we compare our model against GCN, GAT, ACM-GNN, H2GCN, and LRGNN. The results show that our model consistently exhibits superior performance. With increasing random perturbations, false edges are more likely to connect nodes from different labels, leading to erroneous message passing in conventional methods. This provides strong evidence for the ability of TRed-GNN to distinguish between task-relevant and task-irrelevant connections. Consequently, even when a large number of false edges are present in the graph topology, our model can still effectively gather neighborhood information to predict node labels. These results further demonstrate that our model has a stronger capacity to remove irrelevant edges while preserving task-relevant structures, thereby enhancing both robustness and stability.



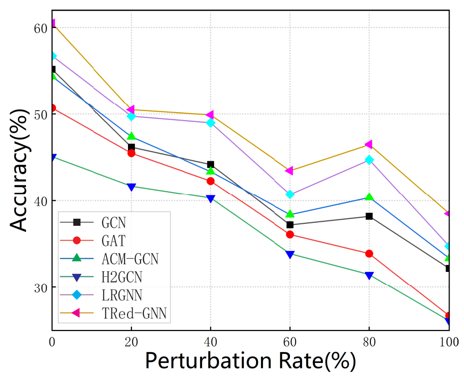
(**a**) Cora



(**b**) Citeseer



(**c**) Chameleon

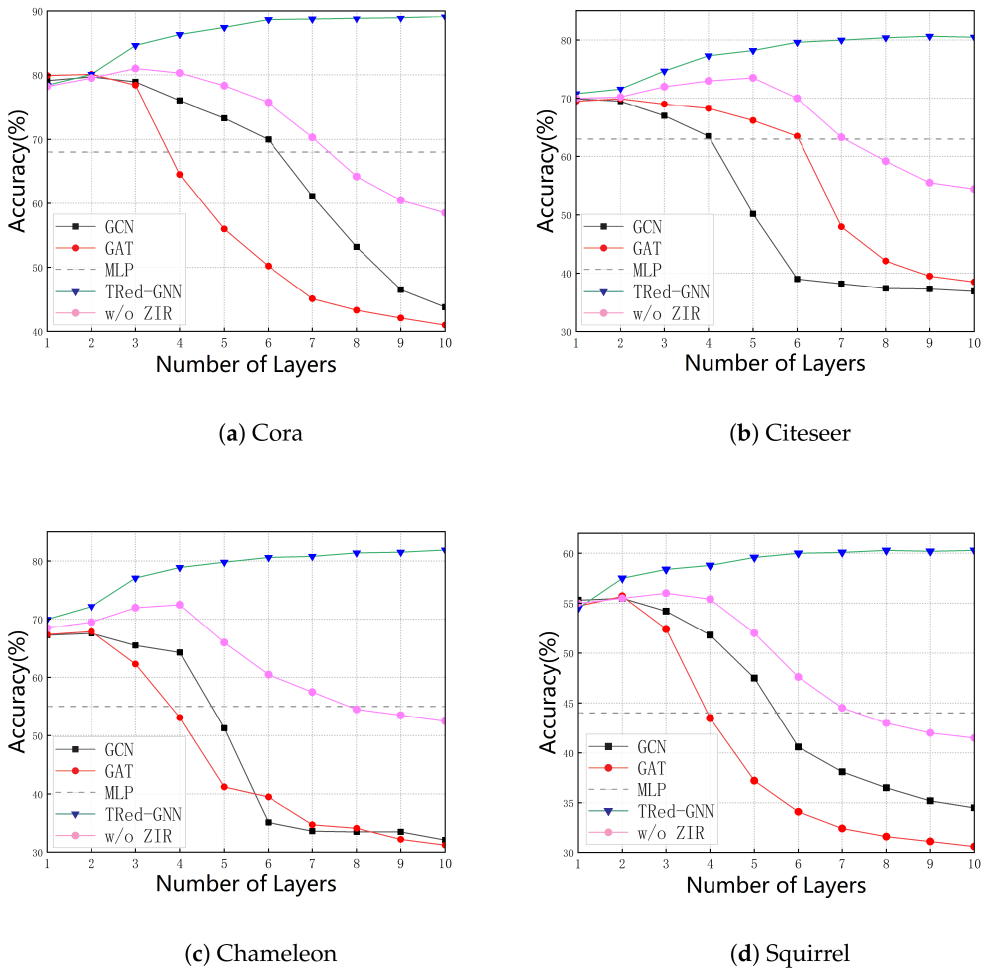


(**d**) Squirrel

Figure 3. Results of different models on perturbed homophilous graphs. TRed-GNN exhibits superior robustness against disturbances compared to other models. The line charts display the accuracy of different models under varying levels of perturbed rates in (**a**) the Cora dataset, (**b**) the Citeseer dataset, (**c**) the Chameleon dataset, and (**d**) the Squirrel dataset. TRed-GNN is able to identify the falsely injected (the task-irrelevant) graph edges, and exclude these connections from the final predictive learning, thereby displaying relative robust performance against adversarial edge attacks.

5.4. Relieve the Problem of Excessive Smoothness

To evaluate the over-smoothing behavior of TRed-GNN, we varied the number of layers in the model and compared its performance with GCN and GAT. As shown in Figure 4, when the number of layers reaches two, both GCN and GAT achieve their highest performance. However, as the number of layers increases, their accuracy gradually declines. In contrast, TRed-GNN exhibits a curve that rises steadily before leveling off. Although it starts with relatively lower accuracy, the performance of TRed-GNN improves consistently as the number of layers increases, ultimately achieving accuracy significantly higher than that of GCN and GAT. We attribute this to the fact that TRed-GNN can adaptively utilize edges from different layers to produce results aligned with the target task. Furthermore, by incorporating the reverse process to gather hidden node information, the model enriches node representations and thereby alleviates the over-smoothing problem.



(**a**) Cora (**b**) Citeseer (**c**) Chameleon (**d**) Squirrel

Figure 4. Accuracy comparison of GCN, GAT, MLP, TRed-GNN, and its variant w/o ZIR on four datasets ((**a**) Cora, (**b**) Citeseer, (**c**) Chameleon, and (**d**) Squirrel) as the number of layers increases, illustrating the impact of model depth on performance and over-smoothing.

In addition, we include the variant “w/o ZIR”, which removes the dual-channel disentanglement and propagates messages over a single graph structure. As illustrated in the figure, “w/o ZIR” performs similarly to GCN in the shallow regime, but its accuracy stabilizes at a slightly higher level when the depth increases, showing that the reverse process can still alleviate part of the over-smoothing. However, compared with the full TRed-GNN, the gap is significant, especially on heterophilous datasets such as Chameleon and Squirrel, where dual-channel disentanglement is crucial. This clearly demonstrates that separating task-relevant and task-irrelevant edges plays an indispensable role in enhancing model robustness against over-smoothing.

5.5. Limitations

While TRed-GNN demonstrates robust performance in classification tasks, several limitations and challenges must be considered when deploying the model in practical scenarios. Many real-world graph datasets exhibit sparse connectivity between nodes, particularly in highly heterogeneous graphs where limited edge connectivity may lead to less pronounced performance of TRed-GNN compared to its performance on conventional graphs. A key limitation is the model’s performance degradation on highly sparse graphs, such as the Cornell dataset. In graphs with limited edges, the model struggles to effectively distinguish between task-relevant and task-irrelevant edges due to insufficient structural information. The reverse-process mechanism may not fully compensate for this connectivity sparsity, potentially resulting in suboptimal performance, as observed in our experiments. Future work will focus on enhancing the model's capability to handle sparse graphs by incorporating additional structural priors or refining the reverse-processing mechanism for graphs with deficient edge structures.

6. Conclusions

In this paper, we proposed a novel TRed-GNN model that leverages task-relevant and task-irrelevant graph topologies, together with a reverse process mechanism, to effectively address noise interference in node representations, insufficient generalization, and the over-smoothing problem in heterophilous graphs. Extensive experiments demonstrate that the proposed method achieves excellent performance on a variety of both homophilous and heterophilous graph datasets. Furthermore, ablation studies confirm the necessity of task-relevance modeling and the reverse process mechanism. For future work, we plan to explore more fine-grained edge separation strategies and extend the framework to graph-level tasks and dynamic scenarios.

Author Contributions: Conceptualization, M.X. and Z.Z.; methodology, M.X.; software, Z.Z.; validation, M.X., Q.W. and Y.Y.; formal analysis, Y.Y.; investigation, M.X.; resources, H.C.; data curation, M.X.; writing—original draft preparation, M.X.; writing—review and editing, M.X.; visualization, M.X.; supervision, Y.Y.; project administration, H.C.; funding acquisition, H.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Open Research Fund of Fujian Key Laboratory of Financial Information Processing, Putian University (NO. JXJS202507).

Data Availability Statement: Data derived from public domain resources. The data supporting this study are openly available at https://lig-membres.imag.fr/grimal/data.html.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could be construed as influencing the work reported in this paper.

References

1. Costa, A.R.; Ralha, C.G. AC2CD: An actor–critic architecture for community detection in dynamic social networks. *Knowl.-Based Syst.* **2023**, *261*, 110202. [CrossRef]
2. Li, D.X.; Zhou, P.; Zhao, B.W.; Su, X.R.; Li, G.D.; Zhang, J.; Hu, P.W.; Hu, L. Biocaiv: An integrative webserver for motif-based clustering analysis and interactive visualization of biological networks. *BMC Bioinform.* **2023**, *24*, 451. [CrossRef] [PubMed]
3. Li, Y.; Lin, B.; Luo, B.; Gui, N. Graph representation learning beyond node and homophily. *IEEE Trans. Knowl. Data Eng.* **2022**, *35*, 4880–4893. [CrossRef]
4. Zheng, Q.; Zhang, Y. Tagnn: Time adjoint graph neural network for traffic forecasting. In Proceedings of the International Conference on Database Systems for Advanced Applications, Tianjin, China, 17–20 April 2023; Springer: Cham, Switzerland, 2023; pp. 369–379.
5. Li, W.; Wang, C.h.; Cheng, G.; Song, Q. Optimum-statistical Collaboration Towards General and Efficient Black-box Optimization. *arXiv* **2021**, arXiv:2106.09215.
6. Rusch, T.K.; Bronstein, M.M.; Mishra, S. A survey on oversmoothing in graph neural networks. *arXiv* **2023**, arXiv:2303.10993. [CrossRef]
7. Chen, D.; Lin, Y.; Li, W.; Li, P.; Zhou, J.; Sun, X. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 3438–3445. [CrossRef]
8. He, L.; Bai, L.; Yang, X.; Liang, Z.; Liang, J. Exploring the role of edge distribution in graph convolutional networks. *Neural Netw.* **2023**, *168*, 459–470. [CrossRef]
9. Liu, L.; Wang, Y.; Xie, Y.; Tan, X.; Ma, L.; Tang, M.; Fang, M. Label-aware aggregation on heterophilous graphs for node representation learning. *Displays* **2024**, *84*, 102817. [CrossRef]
10. Chen, Y.; Jiang, D.; Tan, C.; Song, Y.; Zhang, C.; Chen, L. Neural moderation of ASMR erotica content in social networks. *IEEE Trans. Knowl. Data Eng.* **2023**, *36*, 275–280. [CrossRef]

11. Guo, J.; Huang, K.; Zhang, R.; Yi, X. ES-GNN: Generalizing graph neural networks beyond homophily with edge splitting. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 11345–11360. [CrossRef]
12. Kipf, T. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv* **2016**, arXiv:1609.02907.
13. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
14. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
15. Bruna, J.; Zaremba, W.; Szlam, A.; LeCun, Y. Spectral networks and locally connected networks on graphs. *arXiv* **2013**, arXiv:1312.6203.
16. Gilmer, J.; Schoenholz, S.S.; Riley, P.F.; Vinyals, O.; Dahl, G.E. Neural message passing for quantum chemistry. In Proceedings of the International Conference on Machine Learning, Sydney, NSW, Australia, 6–11 August 2017; PMLR; pp. 1263–1272.
17. He, M.; Wei, Z.; Huang, z.; Xu, H. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 14239–14251.
18. Defferrard, M.; Bresson, X.; Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In Proceedings of the 30th International Conference on Neural Information Processing System, Barcelona, Spain, 5–10 December 2016.
19. Pei, H.; Wei, B.; Chang, K.C.C.; Lei, Y.; Yang, B. Geom-gcn: Geometric graph convolutional networks. *arXiv* **2020**, arXiv:2002.05287. [CrossRef]
20. Zhu, J.; Yan, Y.; Zhao, L.; Heimann, M.; Akoglu, L.; Koutra, D. Beyond homophily in graph neural networks: Current limitations and effective designs. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 7793–7804.
21. Luan, S.; Hua, C.; Lu, Q.; Zhu, J.; Zhao, M.; Zhang, S.; Chang, X.W.; Precup, D. Revisiting heterophily for graph neural networks. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 1362–1375.
22. Wang, R.; Mou, S.; Wang, X.; Xiao, W.; Ju, Q.; Shi, C.; Xie, X. Graph structure estimation neural networks. In Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; pp. 342–353.
23. Xu, D.; Cheng, W.; Luo, D.; Chen, H.; Zhang, X. Infogcl: Information-aware graph contrastive learning. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 30414–30425.
24. Sun, Q.; Li, J.; Peng, H.; Wu, J.; Fu, X.; Ji, C.; Yu, P.S. Graph structure learning with variational information bottleneck. *Proc. AAAI Conf. Artif. Intell.* **2022**, *36*, 4165–4174. [CrossRef]
25. Yang, M.; Shen, Y.; Qi, H.; Yin, B. Soft-mask: Adaptive substructure extractions for graph neural networks. In Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; pp. 2058–2068.
26. Zheng, C.; Zong, B.; Cheng, W.; Song, D.; Ni, J.; Yu, W.; Chen, H.; Wang, W. Robust graph representation learning via neural sparsification. In Proceedings of the International Conference on Machine Learning, Virtual Event, 13–18 July 2020; PMLR, pp. 11458–11468.
27. Luo, D.; Cheng, W.; Yu, W.; Zong, B.; Ni, J.; Chen, H.; Zhang, X. Learning to drop: Robust graph neural network via topological denoising. In Proceedings of the 14th ACM International Conference on Web Search and Data Mining, Virtual Event, 8–12 March 2021; pp. 779–787.
28. Wang, H.; Leskovec, J. Unifying graph convolutional neural networks and label propagation. *arXiv* **2020**, arXiv:2002.06755. [CrossRef]
29. Seo, S.; Kim, S.; Park, C. Interpretable prototype-based graph information bottleneck. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 76737–76748.
30. Higgins, I.; Amos, D.; Pfau, D.; Racaniere, S.; Matthey, L.; Rezende, D.; Lerchner, A. Towards a definition of disentangled representations. *arXiv* **2018**, arXiv:1812.02230. [CrossRef]
31. Liu, Y.; Wang, X.; Wu, S.; Xiao, Z. Independence promoted graph disentangled networks. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 4916–4923. [CrossRef]
32. Ma, J.; Cui, P.; Kuang, K.; Wang, X.; Zhu, W. Disentangled graph convolutional networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; PMLR, pp. 4212–4221.
33. Yang, Y.; Feng, Z.; Song, M.; Wang, X. Factorizable graph convolutional networks. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 20286–20296.
34. Li, H.; Zhang, Z.; Wang, X.; Zhu, W. Disentangled graph contrastive learning with independence promotion. *IEEE Trans. Knowl. Data Eng.* **2022**, *35*, 7856–7869. [CrossRef]
35. Zhu, J.; Rossi, R.A.; Rao, A.; Mai, T.; Lipka, N.; Ahmed, N.K.; Koutra, D. Graph neural networks with heterophily. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 11168–11176. [CrossRef]
36. Maurya, S.K.; Liu, X.; Murata, T. Simplifying approach to node classification in graph neural networks. *J. Comput. Sci.* **2022**, *62*, 101695. [CrossRef]
37. Liang, L.; Hu, X.; Xu, Z.; Song, Z.; King, I. Predicting global label relationship matrix for graph neural networks under heterophily. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 10909–10921.

38. Song, Y.; Zhou, C.; Wang, X.; Lin, Z. Ordered gnn: Ordering message passing to deal with heterophily and over-smoothing. *arXiv* **2023**, arXiv:2302.01524. [CrossRef]
39. Bo, D.; Wang, X.; Shi, C.; Shen, H. Beyond low-frequency information in graph convolutional networks. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 3950–3957. [CrossRef]
40. Chamberlain, B.; Rowbottom, J.; Gorinova, M.I.; Bronstein, M.; Webb, S.; Rossi, E. Grand: Graph neural diffusion. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; PMLR, pp. 1407–1418.
41. Rong, Y.; Huang, W.; Xu, T.; Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv* **2019**, arXiv:1907.10903.
42. Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; Weinberger, K. Simplifying graph convolutional networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; PMLR, pp. 6861–6871.
43. Gasteiger, J.; Bojchevski, A.; Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv* **2018**, arXiv:1810.05997.
44. Chien, E.; Peng, J.; Li, P.; Milenkovic, O. Adaptive universal generalized pagerank graph neural network. *arXiv* **2020**, arXiv:2006.07988.
45. Abu-El-Haija, S.; Perozzi, B.; Kapoor, A.; Alipourfard, N.; Lerman, K.; Harutyunyan, H.; Ver Steeg, G.; Galstyan, A. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; PMLR, pp. 21–29.