applied sciences

MDPI

Article

Heterogeneous Graph Purification Network: Purifying Noisy Heterogeneity without Metapaths

Sirui Shen [1,2,3,4], Daobin Zhang [1,2,*], Shuchao Li [1,2], Pengcheng Dong [1,2], Qing Liu [1,2], Xiaoyu Li [1,2] and Zequn Zhang [1,2]

[1] Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100190, China; shensirui20@mails.ucas.ac.cn (S.S.)
[2] Key Laboratory of Network Information System Technology (NIST), Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100190, China
[3] University of Chinese Academy of Sciences, Beijing 100190, China
[4] School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 100190, China
* Correspondence: dbzhang@mail.ie.ac.cn

Abstract: Heterogeneous graph neural networks (HGNNs) deliver the powerful capability to model many complex systems in real-world scenarios by embedding rich structural and semantic information of a heterogeneous graph into low-dimensional representations. However, existing HGNNs encounter great difficulty in balancing the ability to avoid artificial metapaths with resisting structural and informational noise in a heterogeneous graph. In this paper, we propose a novel framework called **H**eterogeneous **G**raph **P**urification **N**etwork (HGPN) which aims to solve such dilemma by adaptively purifying the noisy heterogeneity. Specifically, instead of relying on artificial metapaths, HGPN models heterogeneity by subgraph decomposition and adopts inter-subgraph and intra-subgraph aggregation methods. HGPN can learn to purify noisy edges based on semantic information with a parallel heterogeneous structure purification mechanism. Besides, we design a neighborhood-related dynamic residual update method, a type-specific normalization module and cluster-aware loss to help all types of node achieve high-quality representations and maintain feature distribution while preventing feature over-mixing problems. Extensive experiments are conducted on four common heterogeneous graph datasets, and results show that our approach outperforms all existing methods and achieves state-of-the-art performances consistently among all the datasets.

Keywords: graph neural network; heterogeneous graph representation learning; heterogeneous graph purification; graph representation

Citation: Shen, S.; Zhang, D.; Li, S.; Dong, P.; Liu, Q.; Li, X.; Zhang, Z. Heterogeneous Graph Purification Network: Purifying Noisy Heterogeneity without Metapaths. *Appl. Sci.* **2023**, *13*, 3989. https://doi.org/10.3390/app13063989

Academic Editors: Barbara Strug and Grażyna Ślusarczyk

Received: 15 February 2023
Revised: 12 March 2023
Accepted: 17 March 2023
Published: 21 March 2023

1. Introduction

In recent years, graph representation learning has become increasingly popular due to the universal ability of graph-structured data to model many real-world complex systems, such as citation networks [1], social networks [2], recommendation systems [3] and so on. Now Graph Neural Network (GNN) is recognized as the most advanced technique for graph representation learning following the framework of message passing. GNNs can effectively make use of both network structure and node attributes to generate high quality representations. Most of the previous GNNs [4–6] mainly focus on homogeneous graphs, wherein all nodes and edges are treated as the same type. However, in realistic scenarios, heterogeneous graphs which contain various types of nodes and edges are actually much more general and common. Lots of useful semantic and structural information the heterogeneity brings will be lost if we simply adopt homogeneous GNNs on heterogeneous graphs, which require specifically designed models.

Various models have been proposed to make GNNs suitable for heterogeneous graphs. The majority of existed methods [2,7] adopt metapaths, which are predefined semantic

patterns designed by human prior knowledge, to extract clean and useful information from the noisy and complex heterogeneity. These metapath-based models firstly aggregate neighbor features at the scope of each metapath rather than the original graph to generate semantic representations, and then fuse these semantic representations to generate the final node representations. These methods have good interpretability and are widely used in practice, but face some inherent flaws: (1) High demand while less generality. Metapaths not only require researchers to have specific domain knowledge, but also need to be redesigned for different datasets and applications. (2) Limited expressive power. The model's learning capability is upper-bounded by artificial metapaths, which is contrary to the original purpose of deep learning to learn from data automatically.

Considering the shortcomings of metapaths, some works make attempts to design metapath-free methods, e.g., [8,9]. These models aggregate messages from a node's local neighborhood like GNNs, and model heterogeneity via applying extra modules to embed semantic information based on type-specific weight parameters into propagated messages. However, since these models choose to take the original noisy graph as input, they meet some tricky problems that heterogeneous graphs often inevitably possess: Information Lacking (IL) and Information Confusion (IC). IL means most types of node suffer from missing attributes and labels. IC means some structures of the original graph are noise and even harmful to specific downstream tasks. Take the OGB-MAG [10] citation network dataset for example. It has four types of nodes: paper, author, field of study and institution while only paper nodes have initial attributes and labels. Thus, obtaining good representations for most nodes in a graph can be challenging due to limited self-information and direct supervision. In this particular task, the goal is to predict whether a paper node belongs to a conference or journal venue. The edge type "affiliated with" between authors and institutions may have a less positive relationship to this task. Incorporating information from the IC and IL may help to reduce the effects of heterogeneity. However, most existing heterogeneous GNNs simply aggregate the representations of different types of nodes together during the message passing process. This can lead to the loss of the original distribution of different types of node features and cause information over-mixing, which is a serious problem. Based on this analysis, we can explain the surprising finding in [11] that heterogeneous GNNs do not necessarily outperform homogeneous GNNs in a fair comparison. In general, we summarize the main dilemma that existing heterogeneous GNN methods face: they cannot balance the ability of avoiding artificial design with resisting structural and informational noise. In this paper, we attempt to solve this dilemma by proposing a framework called the **H**eterogeneous **G**raph **P**urification **N**etwork (HGPN) with the aim to purify the noisy heterogeneity from both structure and feature. Specifically, HGPN avoids using any hand-engineered metapaths and we model heterogeneity by splitting the original graph into edge type-specific subgraphs and adopting node type-specific parameters by the inter-subgraph aggregation method. A parallel heterogeneous edge purification mechanism is designed during the heterogeneous message propagation process to filter irrelevant edge structures in a learning manner. Then, we present a heterogeneous neighborhood-related residual update method to discriminate and help nodes with low quality attributes automatically. Finally, a novel type-specific normalization and cluster loss function are proposed so that the representation of nodes with IL problems can achieve basic constraint optimization as well as preventing feature over-mixing. Extensive experiments are conducted on various common heterogeneous graph datasets. The results show that our approach outperforms all existing methods consistently among all the datasets.

The core contributions of our work are listed as follows:

(1) We analyze and summarize the main dilemma on which present heterogeneous GNN research should focus. Thus, a novel HGPN framework is proposed to solved the dilemma, which can jointly handle edge and node heterogeneity with inter and intra subgraph aggregation methods without any artificial metapaths.

(2) To solve the mentioned IC problem, HGPN uses a new designed parallel heterogeneous structure purification mechanism so that the model can learn to filter noisy edge structures based on semantic information and downstream tasks.
(3) To solve the mentioned IL and information over-mixing problem, we present the neighborhood-related residual update method, a novel type-specific normalization and cluster loss function to help all types of nodes achieve high-quality representations.
(4) HGPN achieves SOTA performance with the proof of extensive experiment and analysis on four real world heterogeneous graph datasets. Remarkably, the accuracy of HGPN on the large OGB-MAG dataset exceeds the current SOTA method by 2.07% and 2.22% under different training settings.

The rest of this paper is organized as follows: Section 2 summarizes previous research related to the studied problem. Section 3 formalizes the studied problem. Section 4 presents the framework and introduces each component of our model. Section 5 evaluates the proposed model through experiments. Finally, Section 6 concludes the entire paper.

2. Related Work

2.1. Graph Embedding Learning

Graph embedding learning aims to embed nodes and edges into a low-dimensional dense vector space, and a great number of efforts have been made on this subject over the past decades. Early methods based on manifold learning mainly focused on graph reconstruction, including Locally Linear Embedding (LLE) [12] and Laplacian Eigenmaps (LE) [13]. Inspired by the success of the word2vec [14] model in natural language processing, more advanced methods were proposed to learn representations of nodes in the network, such as DeepWalk [15], Node2Vec [16], and metapath2vec [17]. These methods use several random walk strategies to get node sequences that can preserve structural information of the original graph. Then, the skip-gram model is applied to capture structural similarity among node sequences based on the distributional hypothesis. However, these methods only consider the topology structure of graphs and cannot utilize node or edge features if provided, making them less adaptable in many realistic scenarios. Thus, these methods are gradually being replaced by more advanced graph neural networks that can take advantage of both node features and the graph structure simultaneously.

2.2. Graph Neural Networks

Graph Neural Networks (GNNs) extend the thought of deep learning to data with graph structure, and they have been successfully applied in various applications such as node classification [18], graph classification [19], traffic prediction [20], and recommendation systems [21]. Generally, GNNs can be unified under the message-passing framework, which propagates information among nodes and their neighbors and then updates node representations by aggregating the received messages. GNNs can be categorized as spectral methods or spatial methods. GCN [4] is one of the most popular spectral methods and simplifies the operation of graph Laplacian smoothing by proposing a localized first-order approximation. GraphSAGE [5] is a pioneer of spatial methods that makes GNNs inductive by introducing neighbor sampling strategies and utilizes flexible aggregating functions. GAT [6] based on the spatial domain adopts the attention mechanism to adaptively assign different importance to one-hop neighbors and get weighted aggregation of node features. However, most existing GNNs are only suitable for homogeneous graphs and cannot deal with the complexity of various types of nodes and edges in heterogeneous graphs.

2.3. Heterogeneous Graph Neural Networks

Recently, researchers have begun to pay a large amount of attention to applying GNNs for heterogeneous graph learning. Ref. [22] proposed R-GCN to use relation-specific projection matrices for aggregation among different relation edges. Ref. [2] presents HAN by adopting a hierarchical attention mechanism to capture both node-level and semantic-level information through multiple human-designed metapaths. MAGNN [7] enhances HAN by

considering the intermediate nodes in metapaths and using several encoders for both intra-metapath aggregation and inter-metapath aggregation. Het-GNN [23] uses a random walk strategy to sample neighbor sequences from different distances and then adopts specialized Bi-LSTMs to aggregate information. Ref. [24] develops HetSANN with type-specific graph attention layers to handle messages from various relations. Motivated by the architecture of Transformer [25], Refs. [8,25] introduces HGT to characterize the heterogeneous information with meta-relation-specific attention modules, rather than metapaths. Although the above methods have explored a lot and achieved great success, they rely highly on the quality of the original graph structure and node features. Therefore, these existing methods cannot handle the structural and informational noise of heterogeneous graph data in reality.

Considering the drawbacks of the above-mentioned methods, we believe that a good solution of HGNN should handle the heterogeneity in a natural and automatic manner and be robust to any type of potential noise in realistic graph data. This principle guides us to the design of different components in HGPN.

3. Preliminaries

3.1. Heterogeneous Graph

A heterogeneous graph is denoted as $G = (\nu, \varepsilon, A, R)$ associated with a node type mapping function $\phi : \nu \rightarrow A$ and an edge type mapping function $\psi : \varepsilon \rightarrow R$, where ν, ε, A, R respectively represent the set of nodes, edges, node types and edge types. Each node $v \in \nu$ and each edge $e \in \varepsilon$ belong to their corresponding type in A and R using ϕ and ψ. Heterogeneous graphs should meet the requirement that $|A| + |R| > 2$.

3.2. Heterogeneous Graph Representation Learning

Given a heterogeneous graph $G = (\nu, \varepsilon, A, R)$, heterogeneous graph representation learning aims to learn a function $f : \nu \rightarrow \mathbb{R}^d$ to project nodes into a d-dimensional representation space satisfying $d \ll |\nu|$. The learned representations should be capable of reflecting both node semantic information and graph structure information involved in the original graph data to facilitate different downstream tasks, such as node classification, graph classification and link prediction.

4. Proposed Method

This section first presents the framework of the proposed model and then introduces each component in detail step by step.

Figure 1 illustrates the architecture of HGPN, which consists of several key components: a subgraph decomposition method, a pre-processing and embedding layer, an inter-subgraph aggregation layer, a type-specific update layer, and a cluster-aware loss. The original heterogeneous graph is first decomposed into multiple type-specific subgraphs based on edge types. The pre-processing and embedding layer is then used to process or initialize node features and map them into a uniform shape for further use.

Next, the inter-subgraph aggregation layer utilizes heterogeneous graph attention layers to learn type-specific messages from each subgraph individually while purifying noisy edge structures. The inter-subgraph aggregation aims to fuse these multi-subgraph views into a compact one and improve the interactions of message representations across different types of relations. The type-specific update method uses the comprehensive neighbor-passed messages obtained earlier to update the representation of the target nodes. Type-specific batch normalization and neighborhood smoothness related dynamic residual connections are adopted to purify node features and maintain their original distribution.

Additionally, a cluster-aware loss is designed to ensure feature distinguishability and prevent feature over-mixing, which also contributes to feature purification. Finally, the learned node representations can be used to facilitate various downstream tasks, such as node classification, node clustering, and link prediction.

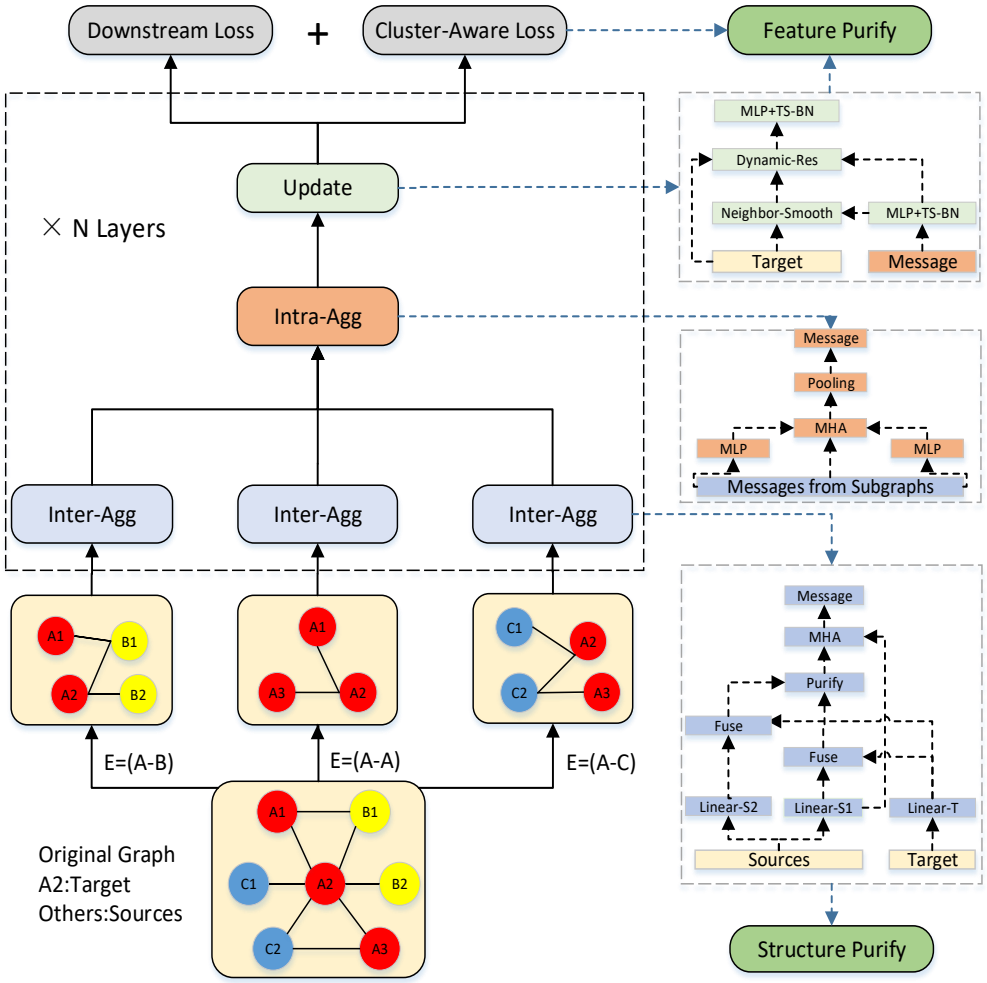

Figure 1. Our HGPN framework: (1) Subgraph Decomposition method; (2) Inter-subgraph Aggregation (Inter-Agg); (3) Inter-subgraph Aggregation(Intra-Agg); (4) Type-specific Update(Update); (5) Cluster-Aware Loss. "Sources" (A1, A3, B1, B2, C1, C2) and "Target" (A2) respectively refer to neighbor source nodes and central target nodes in the process of graph message passing. "MHA" denotes "Multi Head Attention". "TS-BN" is short for "Type-Specific Batch Normalization", while "Dynamic-Res" means "neighborhood smoothness related Dynamic Residual connection".

4.1. Heterogeneity Modeling

The heterogeneity of graph structured data can be classified into two types: edge heterogeneity and node heterogeneity. In contrast to existing metapath-free models that aggregate on the original structure with relation-specific (R-GCN) or meta-relation-specific (HGT) parameters, we propose a decoupled solution that addresses both types of heterogeneity. Firstly, we decompose the original heterogeneous graph into subgraphs that contain a single edge type. Then, we introduce an inter-subgraph aggregation module to learn node representations from a simplified view. To handle node heterogeneity, we use node type-specific projection matrices in the inter-subgraph aggregation module. Finally, we employ an attention-based intra-subgraph aggregation module to aggregate information from subgraphs with different edge types.

We list three advantages of our solution to model the heterogeneity as follows:

1. The heterogeneity of edge types is naturally exploited without the need to introduce additional parameters for edges of different relations. With N types of nodes corresponding to N^2 orders of magnitude of edge types, it is more sensible not to set

different parameters for different edge types, while the increase in the number of parameters brought about by setting different parameters for different types of nodes is acceptable.
2. It prevents the aggregation of too much neighbor information for target nodes at one time, thus alleviating the well-known problem of over-squashing [26] in GNNs. This problem arises when the fixed embedding dimension cannot retain information from nodes that increase exponentially with neighbor hops, leading to issues of feature over-mixing and information loss.
3. The distribution of information from different relationships can vary significantly, and using the common graph attention module based on the $Softmax$ function directly without subgraph decomposition can result in the loss of information with relatively small numerical scales due to the amplification effect of the exponential function.

4.2. Pre-Processing and Embedding Layer

As the initial features of different types of nodes vary greatly, with some lacking initial features and others having different initial dimensions, a certain embedding pre-processing layer is required. In large datasets with rich structure information, nodes without initial features can be initialized first using traditional methods such as metapath2vec [17] and TransE [27] to generate features, while small datasets can simply use one-hot type ID vector to be initialized. After generating initial attributes for each node, we use a single-layer feedforward network for each node i to map all node features to a feature space with the same hidden dimension:

$$X_i^{'} = Relu(W_{\phi(i)} X_i + E_{\phi(i)}) \tag{1}$$

where X_i is the initial features while weight $W_{\phi(i)}$ and type embedding $E_{\phi(i)}$ are learnable parameters specific to node type $\phi(i)$.

4.3. Inter-Subgraph Aggregation

After single-relation subgraph decomposition, in the subgraph containing edge type $\psi(e)$, we assume the target node type is $\phi(d)$ and the source node type is $\phi(s)$. In the l-th layer inner-subgraph aggregation module, firstly we adopt the type-dependent matrices $W^l_{\phi(s)}$ and $W^l_{\phi(d)}$ to project nodes' representation X_s^{l-1} and X_d^{l-1} into the same semantic space for the aggregation calculation, as follows:

$$\begin{aligned} X_s^l &= W^l_{\phi(s)} X_s^{l-1} \\ X_d^l &= W^l_{\phi(d)} X_d^{l-1} \end{aligned} \tag{2}$$

In traditional heterogeneous GNNs, the model would indiscriminately aggregate all the source node information X_s^{l-1} to the target node X_d^{l-1}, which is likely to cause harm to the downstream task in heterogeneous graph data with high noise structures. To give the model the ability to automatically learn to filter out noisy information and improve robustness, a parallel structure purification mechanism is introduced in the module. Firstly, we use another set of type-dependent projection matrices $W^{l,m}_{s,\phi(s)}$ and $W^{l,m}_{d,\phi(d)}$ to project the nodes into another semantic space for determining relevance:

$$\begin{aligned} X_s^{l,m} &= W^{l,m}_{s,\phi(s)} X_s^{l-1} \\ X_d^{l,m} &= W^{l,m}_{d,\phi(d)} X_d^{l-1} \end{aligned} \tag{3}$$

Then, the filter score $F^l_{s,d}$ between each node pair $X_s^{l,m}$ to $X_d^{l,m}$ can be calculated by an additive attention layer, as follows:

$$F^l_{s,d} = tanh(W^l LeakyReLU(X_s^{l,m} + X_d^{l,m})) - \gamma_{\psi(e)} \tag{4}$$

where $\gamma_{\psi(e)}$ is a learnable threshold corresponding to the edge type of subgraph $\psi(e)$. Each $\gamma_{\psi(e)}$ is initialized to 0. We introduce the parameter $\gamma_{\psi(e)}$ because the quality of different edge types often varies significantly in practice. By incorporating this parameter with minimal increase in parameters, our model can learn to differentiate between noisy edge structures, taking into account edge heterogeneity. This allows us to obtain the final mask matrix M^l, which ensures that the source node s will not have any effect on the target node d if $F^l_{s,d} < 0$:

$$M^l_{s,d} = \begin{cases} F^l_{s,d} & \text{if } F^l_{s,d} \geq 0 \\ F^l_{s,d} - \infty & \text{if } F^l_{s,d} < 0 \end{cases} \tag{5}$$

Since the importance of different neighbor nodes cannot be treated equally, we apply attention operators for message aggregation along with the predefined mask matrix F^l. Precisely, we calculate the attention score $A^l_{s,d}$ of source node s to target node d by:

$$A^l_{s,d} = \frac{exp(a^T LeakyReLU(X^l_s + X^n_d) + M^l_{s,d})}{\sum_{s \in N_{\psi(e)}(d)} exp(a^T LeakyReLU(X^l_s + X^l_d) + M^l_{s,d})} \tag{6}$$

where $N_{\psi(e)(d)}$ denotes the neighbor nodes of target node d with edge type $\psi(e)$. Finally, we can compute a weighted average of the transformed features of the neighbor nodes as the aggregated representation $Z^l_{e,d}$, using the normalized attention scores $A^l_{s,d}$:

$$Z^l_{e,d} = \sum_{s \in N_{\psi(e)}(d)} A^l_{s,d} X^l_s \tag{7}$$

The multi-head attention mechanism [25] can make the training process more stable and allow attention to be integrated and learned in parallel across multiple subspaces, and it has already demonstrated strong performance in models of several domains. Thus, we apply the multi-head attention mechanism here, as shown, where $||$ means concatenation and H means the number of attention heads:

$$Z^l_{e,d} = ||^H_{h=0} Z^{l,h}_{e,d} \tag{8}$$

4.4. Intra-Subgraph Aggregation

After the edge-specific inter-subgraph aggregation, target nodes connected with R kinds of edges can get R different views of propagated messages. Then, we adopt an intra-subgraph aggregation module to fuse these multi relation views of neighbor messages and generate a comprehensive one. In reality, the importance of messages for target nodes from different subgraphs usually differs a lot. Therefore, we establish an attention-based aggregation method instead of simple pooling operations so that our model can learn to assign appropriate weights to different subgraphs' information automatically.

For the list of propagated messages $Z^l = [z^l_1, z^l_2, \ldots z^l_R]$ obtained from different subgraphs at the l-th layer, we compute the compositive message representation Z^l_c using a simplified self-attention style module:

$$K^l = Q^l = a^T LeakyReLU(W^l Z^l) \tag{9}$$

$$Z^l_c = Mean(Softmax(\frac{K^l (Q^l)^T}{\sqrt{d}}) Z^l) \tag{10}$$

It is worth noting that our proposed implementation of self-attention has two small changes compared to the previous method widely used in the Transformer [25]: (1) The calculation of K and Q uses a structure similar to a two-layer MLP (Multi-Layer Perceptron) instead of a single linear layer, which achieves stronger expressive ability. Introducing a^T allows our model to learn the dimensional importance of message representation, while it is only a vector of size $d \times 1$ rather than the usual $d \times d$ sized projection matrix, where d is the dimension of the node representation. Therefore, the number of parameters can be

much smaller than that of a real two-layer MLP. (2) Due to the improvement of the first point, there is no problem of parameter degradation that occurs in the original self-attention design when $K = Q$, so it is possible to let $K = Q$ to reduce the number of parameters. We also remove the V matrix in classic self-attention to further reduce cost. Experiments have shown that our implementation of self-attention can effectively reduce the number of model parameters without any decrease in model performance.

Similar to the module in Section 4.4, the multi-head attention mechanism is adopted here as follows, where H denotes the number of attention heads:

$$Z_c^l = ||_{h=0}^{H} Z_h^l \tag{11}$$

4.5. Type-Specific Update

After the inner-subgraph and intra-subgraph aggregation modules, we need to use the comprehensive message Z_i^l to update the original node representation. First, we use the following formula to transform the message back to the representation space of target node type:

$$M_i^l = GeLU(BN_{\phi(i)}(W_{u,\phi(i)}^L Z_i^l)) \tag{12}$$

where $W_{u,\phi(i)}$ is the transform matrix corresponding to node type $\phi(i)$ and $GeLU$ means gelu activation function [28]. It is worth noting that we utilize a distinct batch normalization for each type of node, rather than a default shared one or layer normalization, and we name it type-specific batch normalization. The reason for this design is that the distribution of representations for different types of nodes is usually very different. Using the same batch normalization layer directly would lead to grossly inaccurate estimates of the node feature statistics, which may significantly reduce the effectiveness of our model. At the same time, the type-specific batch normalization also helps to maintain node representations of the same type in the same distribution space, serving to purify the node representation.

Residual connection [29] has been widely used among models in various domains and proved to be very effective in alleviating problems such as gradient disappearance and oversmoothing in GNNs [30]. In contrast to the normal one with fixed residual weight, we design a novel and neighborhood smoothness-related dynamic residual connection scheme for an adapted version in the field of heterogeneous graph networks:

$$\lambda_{\phi(i)}^l = sigmoid(\gamma_{\phi(i)} \times \frac{|X_i^{l-1} - M_i^l|_F}{\sqrt{d}}) \tag{13}$$

$$X_i^l = (1 - \lambda_{\phi(i)}^l) X_i^{l-1} + \lambda_{\phi(i)}^l M_i^l \tag{14}$$

where $|\ |_F$ means Frobenius norm and $\gamma_{\phi(i)}$ is a learnable coefficient specific to the type of node $\phi(i)$. X_i^{l-1} is the node representation from the previous layer and d is the dimension of node embedding. We define neighborhood smoothness of nodes by calculating the Euclidean difference between the passed message and the original node representation. The distance value is divided by $\sqrt{d}$ to avoid gradient vanishing in *sigmoid* function. The basic design idea of our proposed dynamic residual connection is that the update weight of each node should be positively correlated to its neighborhood smoothness. The interpretation is that propagated messages can be considered as the information complement from neighboring nodes to central target nodes. And we believe that nodes with lower neighborhood smoothness represent higher original feature noise according to the widely accepted graph smoothness assumption, especially for most types of nodes without initial features. So, they should rely more on the information provided by the neighboring nodes for feature purification in the method of representation update. Meanwhile, if the neighborhood smoothness is already very high, it means that the original information of the node is sufficient and the additional information brought by the graph structure has limited value. Thus, a smaller update weight should be used to prevent the problem of over-smoothing in GNN training. Overall, our design can provide a two-fold positive effect. In addition,

the evaluation metrics of neighborhood smoothing should not be consistent for different types of nodes, so the node category-related parameter $\gamma_{\phi(i)}$ is introduced for modeling type difference.

After the dynamic residual update method, we let X_i^l continuously come through a two layer MLP with normal residual connection and type-specific batch normalization, which is similar to the Transformer [25] architecture:

$$X_i^{l+1} = X_i^l + BN_{\phi(i)}(MLP_{\phi(i)}(X_i^l)) \tag{15}$$

This design is motivated by the challenge of effectively scaling the number of GNN layers in large-scale graphs. Due to limitations such as sampling mechanisms, memory constraints, and the over-smoothing problem, it can be difficult to make the GNN deep enough. To ensure sufficient expressive power, we address this challenge by increasing the depth of the non-linear transformation in the update module.

4.6. Cluster-Aware Loss

In many downstream tasks, only a few target nodes are labeled, making it difficult to optimize the representations of other types of nodes directly through supervised constraints. Additionally, frequent information passing between different types of nodes in HGNNs can lead to feature over-mixing, causing the loss of nodes' original representation distribution. To address these challenges, we propose a simple and fundamental constraint goal: we aim to maintain the original feature distribution of each node category by ensuring that the node features of different categories have clustering properties in the representation space, rather than being mixed together. Building upon this goal, we propose the following loss design scheme, where $B_{\phi(i)}$ represents the center of the learnable node type anchor corresponding to each node type $\phi(i)$, and $Norm_2$ denotes the l_2 normalization operation.

$$L_{agg} = \frac{1}{|\nu|} \sum_{i \in \nu, j \in A, j \neq \phi(i)} exp(min[(X_i - B_{\phi(i)})^2 - (X_i - B_j)^2, 0]) \tag{16}$$

$$L_{push} = \frac{1}{|A|^2 - |A|} \sum_{i,j \in A, i \neq j} exp(Norm_2(B_i) \cdot Norm_2(B_j)) \tag{17}$$

The idea of L_{agg} is that the distance from each node to its own class's center anchor should be less than the distance to other classes' center anchors, thus providing an inter-class clustering effect. To prevent the problem of representation collapsing, which means that the representation space of the nodes within the same class collapse into too small a space, we introduce a stop mechanism to let the optimization end when the distance difference $(X_i - B_{\phi(i)})^2 - (X_i - B_j)^2$ is larger than zero. Meanwhile, L_{push} aims to extend the distance between different class center anchors, thus making it easier to separate the representation distribution of each class. It also enlarges the range of space that each class of node representation can occupy, which is helpful to relieve the representation collapsing problem. In order to minimize L_{agg} and maximize L_{push} at the same time, we develop the final $L_{cluster}$, similar to the formula of InfoNCE loss [31]:

$$L_{cluster} = -log \frac{L_{agg}}{L_{agg} + L_{push}} \tag{18}$$

The optimization target of our model HGPN is the combination of downstream task loss and $L_{cluster}$. Take the supervised node classification task as an example. The final loss L can be computed as the weighted sum of cross-entropy loss and $L_{cluster}$:

$$L = L_{sup} + \lambda_L L_{cluster} = -\sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \cdot log(p_{i,c}) + \lambda_L L_{cluster} \tag{19}$$

where λ_L is a weighted coefficient to balance these two parts of loss.

4.7. Additional Training Skills

4.7.1. Label as Features

Researchers [32,33] have discovered that GNNs can benefit from labels beyond supervision by adding labels to the node feature. One of the simplest but effective ways to do this is to dynamically concatenate the one-hot representation of the label for the nodes which are not in training batches:

$$X_i = \begin{cases} X_i||0 & \text{if } x \in V_{batch\_train} \\ X_i||Y_i & \text{if } x \notin V_{batch\_train} \end{cases} \tag{20}$$

To avoid data leakage, we ensure that the true validation and test labels are not observed during both training and testing. Furthermore, we do not provide the central node's own label to prevent any potential bias. In cases where the true label of a node cannot be obtained, we use an all-zero vector as its label representation to match the dimension of its neighbor nodes' representations.

4.7.2. Multi-Stage Pseudo Label Training

Recent studies [34,35] have discovered that multi-stage training techniques can improve the performance of GNNs. This technique involves selecting test nodes with sufficiently confident predictions at the end of each training stage and adding these nodes to the training set. The labels for these added nodes are generated using the previous stage's model, and the updated training set is used to continue training the model. This approach allows for the full utilization of label information in the validation and test sets, and eliminates input inconsistencies between training and inference when the "Label as Features" training method is also employed. Let T_1 be the initial training set, and K be the number of training stages. After training in stage k, nodes with maximum prediction scores above the predefined threshold η_k are added to the training set for the next training stage $k+1$. The process can be formulated as follows:

$$T_{k+1}(1 \leq k \leq K) = T_k \cup T_{N_i} \; s.t. \; P_{N_i} > \eta_k \tag{21}$$

We usually set K = 2 or 3 to balance performance and training cost.

5. Experiments

5.1. Dataset and Evaluation Metrics

We evaluate our HGPN model on four real-world heterogeneous graph datasets, ranging from different scales: ACM, DBLP, IMDB and OGB-MAG [10]. We list the statistics of these datasets in Table 1.

Table 1. Statistics of datasets used in this paper.

Dataset	Nodes	Node Types	Edges	Edge Types	Classes	Target
ACM	10,942	4	547,842	8	3	paper
DBLP	26,128	4	239,566	6	4	author
IMDB	21,420	4	86,642	6	5	movie
OGB-MAG	1,939,743	4	42,222,014	8	349	paper

DBLP: a subset of the DBLP computer science bibliography, comprising 14,328 papers, 4057 authors, 20 venues, and 8789 terms. The initial attributes of paper nodes are bag-of-words vectors of their keywords, while the initial attributes of author nodes are bag-of-words representations of their affiliates, titles, and keywords extracted from their published papers. For the attributes of terms, no computer science specialized pre-trained word vectors were used, while the attributes of venues were represented using one-hot vectors. The authors in the dataset are the target nodes and are grouped into four research areas based on the conferences to which they submitted.

ACM: a subset extracted from ACM citation network, which comprises 4019 papers, 7167 authors and 60 subjects. The attributes of the papers are bag-of-words representations of their keywords, and the attributes of the authors are bag-of-words representations of affiliates, titles and keywords extracted from their published papers. For subjects, their attributes are bag-of-words representations of keywords from their connected papers. The target nodes are papers, which are divided into three classes according to the conference at which they were published.

IMDB: a benchmark dataset in movie recommendation, which includes 4080 movies, 5075 actors and 2001 directors. We collected a subset of IMDB in five subjects of movies: action, comedy, romance, drama and thriller. The target node is the movie node and the label is the subject of movie. The movie's attribute is a bag-of words vector represented by plots.

OGB-MAG: a heterogeneous academic network extracted from the Microsoft Academic Graph (MAG), which includes four types of nodes: papers (P), authors (A), fields (F), and institutions (I). The network comprises four types of directed edges: an author is "affiliated with" an institution, an author "writes" a paper, a paper "cites" a paper, and a paper "has a topic of" a field of study. To account for bidirectional relationships, we include related reverse edges for all directed edges. The papers in the network are published across 349 different venues, and the task is to predict the venue for each paper. Each paper node has an associated Word2Vec [14] attribute. We use the metapath2vec [17] model to generate features for other types of nodes that do not have input features.

For all four datasets, the task is node classification. Consistent with previous studies, we evaluate the ACM, DBLP, and IMDB datasets using the Macro-F1 and Micro-F1 metrics, while the OGB-MAG dataset is evaluated using the accuracy metric.

5.2. Methods for Comparison

We choose the following nine state-of-the-art methods for comparison:

HAN [2]: introduces a hierarchical attention mechanism to aggregate both node-level and semantic-level information based on multiple artificially selected metapaths.

MAGNN [7]: makes several improvements on HAN by using metapath-based graphs to replace metapath instances and considering information of all nodes rather than only the head and tail node in metapaths.

HetSANN [24]: leverages type-specific attention layers to directly encode node representations with the consideration of heterogeneous graph structures.

R-GCN [22]: extends the homogeneous GCN to knowledge graphs with multiple relations by employing relation-specific projection matrices.

HGT [8]: proposes a novel heterogeneous mutual attention mechanism to capture the information of different meta-relation triplets inspired by the Transformer architecture.

HGB [11]: adopts the multi-layer GAT [6] as backbone with enhancements from the redesign of learnable edge-type embedding, residual connections, and l_2 normalization.

NARS [36]: decouples the message passing and representation update process in GNNs and trains on neighbor-averaged features for randomly-sampled subgraphs.

HSAGN [35]: enhances NARS [36] with an attention aggregation mechanism and label model.

GAMLP [37]: learns to capture the underlying correlations between different scales of knowledge with two novel attention mechanisms and knowledge within node labels.

Here, we compared our proposed models to a total of nine baseline methods, including three metapath-based models (HAN, MAGNN, and HetSANN), three metapath-free models (R-GCN, HGT, and HGB), and three decoupled methods (NARS, HSAGN, and GAMLP), in order to comprehensively evaluate their performance. It should be noted that some of the baseline models are difficult to apply to large-scale datasets such as OGB-MAG due to challenges in choosing and sampling metapaths. Therefore, we only compared our models with those that are available in the online leaderboard for the OGB-MAG dataset.

5.3. Experimental Settings

For all datasets, we set the dimension of the node embedding to 512 and the number of heads in multi-head attention to 8 in the embedding layer for all the compared methods. We use a HGPN with three layers on the ACM, DBLP, and IMDB datasets, while a HGPN with two layers is used for the OGB-MAG dataset. Nodes without initial attributes are initialized with a one-hot id vector in ACM, DBLP, and IMDB, while in OGB-MAG, we adopt the unsupervised metapath2vec method to generate 128-dimensional embeddings. We set the initial learning rate to 0.001, with a cosine annealing learning rate scheduler and a fixed dropout rate of 0.5. The model parameters are optimized via the AdamW [38] optimizer with a weight decay of 0.01. The loss balance parameter λ_L is set to 0.3. We set the confident threshold to 0.6 for models that use multi-stage pseudo label training techniques. We implement our HGPN model with PyTorch [39] and Deep Graph Library (DGL) [40] and our model is trained on one Tesla V100 GPU with 32 GB graphics memory.

We implemented all the compared methods using the full-batch training form on medium-scale datasets including ACM, DBLP, and IMDB. However, when training on large-scale datasets such as OGB-MAG, we found it difficult to follow the same training form due to memory limitations. Therefore, we utilized the concept of neighbor sampling strategy [5], which allowed us to train HGNN methods in the mini-batch training form while inferring on the original whole graph. Specifically, we independently sampled a fixed number of edges for all types of edges in the original graph to generate node-specific subgraph batches. For HGT, we used the standard sampling strategy instead of the one mentioned in its original paper to ensure fair comparison. All the models were trained with a fixed 200 epochs, and an early stopping strategy was employed with a patience of 20. This means the training process would be terminated if the evaluation metrics on the validation set did not exceed the previous best value for over 20 epochs. We set the hyperparameters and model with the best performance on the validation set for testing. To ensure a more stable comparison, each model was tested at least 10 times to minimize the impact of random seeds.

5.4. Overall Performance

5.4.1. Results on Medium-Scale Datasets

The results of the experiment on medium-scale datasets ACM, DBLP and IMDB are shown in Table 2. As we can see, metapath-free methods (i.e., HGT, HGB and HGPN) tend to exhibit better performances. This finding is aligned with [11], which suggests that metapaths may not be necessary for heterogeneous graph learning. Remarkably, our proposed HGPN consistently outperforms previous SOTA methods across all datasets on all metrics. In particular, HGPN achieves the greatest improvements on the IMDB dataset. IMDB is believed to be more difficult for the reason that all models perform worse on IMDB and simple methods like MLP and metapath2vec can also obtain comparable results on ACM and DBLP. This indicates that our HGPN can better extract useful information from heterogeneity due to our model structure and novel design of different purification methods.

Table 2. Overall performances on ACM, DBLP and IMDB. The numbers in bold mean the best results. All methods are tested 10 times and we report the average performances.

Methods	DBLP		IMDB		ACM	
	Macro-F1	**Micro-F1**	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro-F1**
RGCN	91.52 ± 0.50	92.07 ± 0.50	58.85 ± 0.26	62.05 ± 0.15	91.55 ± 0.74	91.41 ± 0.75
HetSANN	78.55 ± 2.42	80.56 ± 1.50	49.47 ± 1.21	57.68 ± 0.44	90.02 ± 0.35	89.91 ± 0.37
HAN	91.67 ± 0.49	92.05 ± 0.62	57.74 ± 0.96	64.52 ± 0.50	90.89 ± 0.43	90.79 ± 0.43
MAGNN	93.28 ± 0.51	93.76 ± 0.45	56.46 ± 3.20	64.67 ± 1.67	90.88 ± 0.64	90.77 ± 0.65
HGT	93.01 ± 0.23	93.49 ± 0.25	63.00 ± 1.19	67.20 ± 0.57	91.12 ± 0.76	91.00 ± 0.76
HGB	94.01 ± 0.24	94.46 ± 0.22	63.53 ± 1.36	67.36 ± 0.57	93.42 ± 0.44	93.35 ± 0.45
HGPN (ours)	**95.02** ± 0.18	**95.48** ± 0.29	**67.55** ± 0.46	**69.34** ± 0.56	**94.12** ± 0.24	**94.21** ± 0.18

5.4.2. Results on Large-Scale Dataset

The large-scale dataset OGB-MAG is considered to have a more realistic value due to its size and number of classes that are closer to industrial application scenarios. Results on OGB-MAG are listed in Table 3. Since some of the compared baselines use additional training skills on the OGB-MAG dataset, we split the experiment into three parts according to different training settings for fair comparison. The symbol "-" means we follow the base training settings. "Label input" means label information is treated as model input ,while "label input + multi-stage" denotes that multi-stage pseudo label training is adopted after "label input". The details of these skills are described in Section 4.7. We report the results on both validation and test sets.

From the table, we can make three observations: (1) Additional use of label information does have a positive effect on GNNs. (2) HGPN invariably achieves better performances under three different training settings, improving 2.07% on the base setting, 1.6% on the "label input" setting and 2.22% on the "label input + multi-stage" setting. Such improvement can be attributed to HGPN's ability to purify both structure and feature noises. (3) Decoupled methods (NARS, GAMLP and HSAGN) are not necessarily superior to traditional sampling-based methods on large-scale graphs. Previous researchers [37,41] claim that the decouple GNN methods are more suitable to large-scale graphs because they can have a larger receptive field without the limitation of graphic memory. However, our HGPN with only 2 layers of sampling nodes beats all the decoupled methods, which demonstrates that sampling-based methods with a proper design can also become very competitive in graphs with millions of nodes.

Table 3. Overall performances on OGB-MAG dataset. The numbers in bold mean the best results. Vacant positions ("-") mean that the method does not adopt any additional training skills. We run each method 10 times and report the average performances.

Methods	Test Accuracy	Valid Accuracy	Training Skills
R-GCN	47.37 ± 0.48	48.35 ± 0.36	-
HetGNN	47.81 ± 0.29	49.12 ± 0.25	-
HGT	49.82 ± 0.13	51.24 ± 0.46	-
R-HGNN	52.04 ± 0.26	53.61 ± 0.22	-
NARS	52.40 ± 0.16	53.72 ± 0.09	-
HGPN (Ours)	**54.47** ± 0.15	**55.72** ± 0.17	-
GAMLP	53.96 ± 0.18	55.48 ± 0.08	label input
HSAGN	53.95 ± 0.14	55.52 ± 0.16	label input
HGPN (Ours)	**55.56** ± 0.17	**57.43** ± 0.21	label input
GAMLP + RLU	55.90 ± 0.27	57.02 ± 0.41	label input + multi-stage
HSAGN + SLE	54.40 ± 0.15	57.87 ± 0.12	label input + multi-stage
HGPN (Ours)	**58.12** ± 0.11	**59.22** ± 0.14	label input + multi-stage

5.5. Ablation Study

In this section, we remove or replace various novel designs of HGPN to analyze the impact of each component. The study is conducted on OGB-MAG without any additional training skills due to its higher difficulty and larger scale.

Table 4 presents the results of our ablation study on the HGPN model design, where we consider five variants of HGPN. Three of them remove the heterogeneous structure purification" module, the adaptive residual update" module, and the "cluster-aware loss" module, respectively. One uses mean-pooling instead of an attention mechanism for intra-subgraph aggregation. The last variant adopts R-GCN style of modeling heterogeneity to use a weight matrix for each relation rather than subgraph decomposition.

As shown in Table 4, each novel method proposed in HGPN has a positive impact on final performance. Specifically, the cluster-aware loss" module brings the least increase, which can be explained by the fact that type-specific projection matrices and batch normalization can similarly help maintain distinct node representation. For the cluster-aware

loss" module and the "heterogeneous structure purification" module, we provide detailed analysis of their effects in Section 5.6.

We also execute experiments on the choices of normalization, the results of which are listed in Table 5. As we can see, our type-specific batch normalization achieves the best scores, layer normalization drops 0.83% while normal batch normalization [42] experiences a significant decrease of up to 10.95%. It demonstrates the significance of maintaining the respective representation distribution of nodes from different types. By comparison, layer normalization [43] does not contribute to this goal, while normal batch normalization is counterproductive to mix the representation of different types of nodes, leading to a serious decrease in the quality of performance.

Table 4. Results of ablation study on HGPN's novel design in OGB-MAG dataset. We use normal residual connection and mean-pooling instead of "Adaptive Residual Update" and "Attention Aggregation". "Subgraph Decomposition" means we adopt R-GCN style of modeling heterogeneity.

Methods	Accuracy	Decrease
HGPN	54.47 ± 0.15	0
- Structure Purification	53.87 ± 0.12	-0.60
- Adaptive Residual	53.84 ± 0.18	-0.63
- Cluster-Aware Loss	54.08 ± 0.14	-0.39
- Attention Aggregation	53.31 ± 0.1	-1.16
- Subgraph Decomposition	52.74 ± 0.23	-1.73

Table 5. Results of ablation study on normalization choices on OGB-MAG dataset. "Normal Batch Norm" means we use the same batch normalization layer for different types of nodes.

Methods	Accuracy	Decrease
Type-specific Batch Norm	54.47 ± 0.15	0
Layer Norm	53.64 ± 0.12	-0.83
Normal Batch Norm	43.52 ± 0.35	-10.95

5.6. Further Analysis

5.6.1. Parameter Sensitivity Analysis

In this section, we analyze the sensitivity of different hyperparameters to investigate the robustness of our HGPN model. Experiments are also conducted on OGB-MAG and the results are reported in Figure 2.

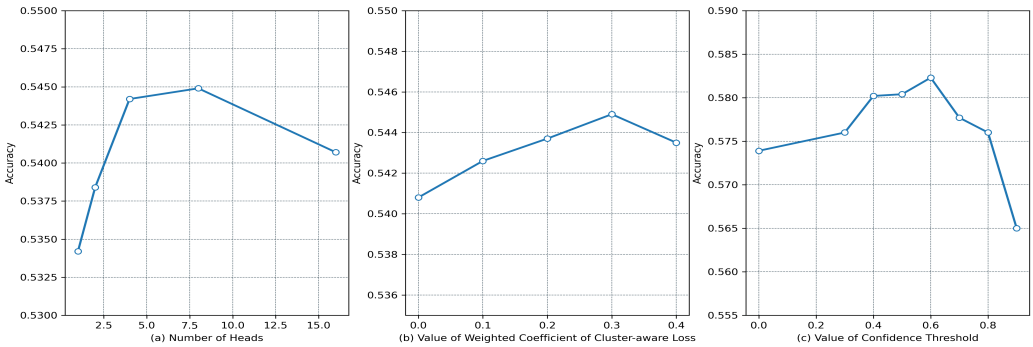

Figure 2. Parameter Sensitivity Analysis Results on OGB-MAG.

Number of attention heads: We fix the dimension of node representation to 512 and vary the number of attention heads from 1 to 16. Figure 2a shows the results. The best performance is achieved when setting the number of heads to 8, which reveals the effect of the multi-head attention mechanism to make training more stable. Thus, we use the same setting of attention heads and representation dimension on all compared baselines.

Value of weighted coefficient of cluster-aware loss: We investigate the effect of value of weighted coefficient of cluster-aware loss and the results are reported in Figure 2a. HGPN can obtain the top score when the value of the weighted coefficient is set to 0.3. We also find that the change of the weighted coefficient of cluster-aware loss has a slight influence on the final results. One plausible explanation is that cluster-aware loss is not a very strong constraint for HGPN and it is relatively easy to converge at an early stage.

Value of confidence threshold: We investigated the influence of the confidence threshold value under the "label input + multi-stage" training setting and presented the results in Figure 2c. We observed that the optimal confidence threshold value was 0.6. Furthermore, even when the confidence threshold value was set to the worst value, our proposed HGPN still outperformed existing baselines, indicating that the superiority of our model was due to its novel design rather than parameter tuning.

We did not analyze the number of layers as the limited GPU memory on OGB-MAG dataset allows sample-based models like HGPN to support no more than two layers. Nonetheless, this indicates the superior performance of our model with fewer layers compared to decoupled models such as NARS and GAMLP.

5.6.2. Detailed Analysis of Cluster-Aware Loss

Cluster-aware loss is introduced to make intra-type node representation separable while maintaining inter-type node representation space. In the following, we explore another two potential designs of cluster-aware loss and analyze the reason for our final design's advantages. The first variant removes the stop mechanism and the second variant uses the average representation of different types of nodes instead of type-specific anchors.

Table 6 presents a summary of the experiment results. Our experimental results validate the significance of our proposed design for maintaining the representation space. Specifically, we have discovered the following: (1) Removing the stopping mechanism from the model will lead to a negative impact of the cluster-aware loss on the model performance. This phenomenon can be attributed to the problem of representation collapse, wherein the inter-type distribution of node representation will decrease to a smaller space during the period of loss optimization, resulting in a loss of information in the inter-type representation diversity. However, by employing the stopping mechanism, we can achieve a balance between inter-type node diversity and intra-type node distinguishability. (2) Using average node representation as cluster anchor instead of learnable anchors will not bring any benefits in the presence of cluster-aware loss. This phenomenon can be explained by the fact that using average representation as a cluster anchor will compress the space of representation, even with the stopping mechanism in place. However, the additional learnable anchors can effectively minimize the influence of compressing inter-type representation space while utilizing very few extra parameters.

Table 6. Results of detailed analysis on cluster-aware loss module on OGB-MAG. We use the mean node representation as class centers in the "learnable anchor" variant.

Methods	Accuracy	Decrease
Ours	54.47 ± 0.15	0
- stopping mechanism	53.84 ± 0.17	-0.63
- learnable anchor	54.07 ± 0.13	-0.40
- Cluster-Aware Loss	54.08 ± 0.18	-0.39

5.6.3. Detailed Analysis of Heterogeneous Structure Purification

In Section 5.5, our ablation study has demonstrated the positive impact of our proposed heterogeneous structure purification mechanism. However, as this module introduces additional parameters and operations, further comparative experiments are necessary to identify the underlying reasons for the observed improvement. Therefore, we have designed three additional tests:

Firstly, we aim to investigate the necessity of introducing parallel projection matrices for calculating filter scores. To this end, we have created a variant that reuses the attention score as the filter score.

Secondly, we seek to exclude the possibility that the observed improvement is solely due to the introduction of extra model parameters, rather than the purification mechanism. We have achieved this by removing the mask method and simply adding filter scores to attention scores without truncation.

Finally, as DropEdge [44] has been proven to be an effective regularization technique in GNNs, we have introduced a variant that randomly masks edges with a rate of 0.3 at each iteration to verify whether our structure purification mechanism outperforms random regularization.

All experimental results can be found in Table 7.

Table 7. Results of detailed analysis of heterogeneous structure purification module on OGB-MAG. "Share Parameters" means we reuse the attention scores as filter scores. "Only add" means we add filter scores on attention scores without truncation. "Random drop" means we drop edges at random with a probability of 0.3.

Methods	Accuracy	Decrease
Ours	54.47 ± 0.15	0
Share Parameters	53.64 ± 0.16	-0.83
Only Add	53.86 ± 0.14	-0.61
Random drop	53.42 ± 0.26	-1.05
Structure Purification	53.87 ± 0.12	-0.60

Table 7 shows that our method achieves significantly better results than the other variants. The "share parameters" variant also performs similarly well, while the "only add" and "random drop" variants perform worse compared to the variant without the structure purification mechanism. These results indicate that the observed improvement is indeed due to our proposed structure purification mechanism, which enables HGPN to detect and remove noisy edges in raw graph data. We can also conclude that parallel projection matrices are essential for determining noisy edges and neighbor information aggregation, as they focus on different aspects of the data. Additionally, the random DropEdge method has been found to have a negative effect on training stability.

Table 8 presents the statistics of edges that HGPN learns to mask on OGB-MAG under two training settings, and Figure 3 shows the purification rate of each type of edge. From these statistics, we can make the following observations:

(1) HGPN tends to mask edges such as "affiliated with" and its reverse edges, which are less relevant to the task of paper venue prediction and consistent with our prior knowledge. This demonstrates that our proposed structure purification method has the ability to identify useful graph structures with realistic causal significance. The results also explain the drawbacks of DropEdge, which can only mask all types of edges equally and thus may lose more useful information that high-quality edges provide.

(2) Under the "label input" training setting, the quality of paper nodes' attributes increases due to the additional label information. As we can see, the structure purification method can automatically capture this change by reducing the masking rate of edges related to paper nodes, such as "cites", "writes", and "has topic". This phenomenon reveals that structure noise and feature noise are not independent, and feature quality is positively associated with structure quality. Therefore, our design of HGPN to both purify feature and structure can jointly help to obtain a clean and refined heterogeneous graph.

Table 8. Statistics of numbers of edges that HGPN learns to purify under 2 two training settings. "w" means "writes", "c" means "cites", "ht" means "has topic", "aw" means "affiliated with". "r-x" means the reverse type of edge type "x". "Hop i" denotes neighbours from *i*-th hop.

Edge/Num	Original Setting		Label Input Setting	
	Hop 1	**Hop 2**	**Hop 1**	**Hop 2**
w/7,145,660	69	0	1	0
r-w/7,145,660	5710	698,753	0	698,192
c/5,416,271	195,939	2058	0	0
r-c/5,416,271	0	0	0	0
ht/7,505,078	1283	7,338,815	0	205,404
r-ht/7,505,078	7,505,078	0	0	599
aw/1,043,998	833,143	1,042,902	31,500	17,070
r-aw/1,043,998	1,043,907	1168	1,043,996	1,040,857

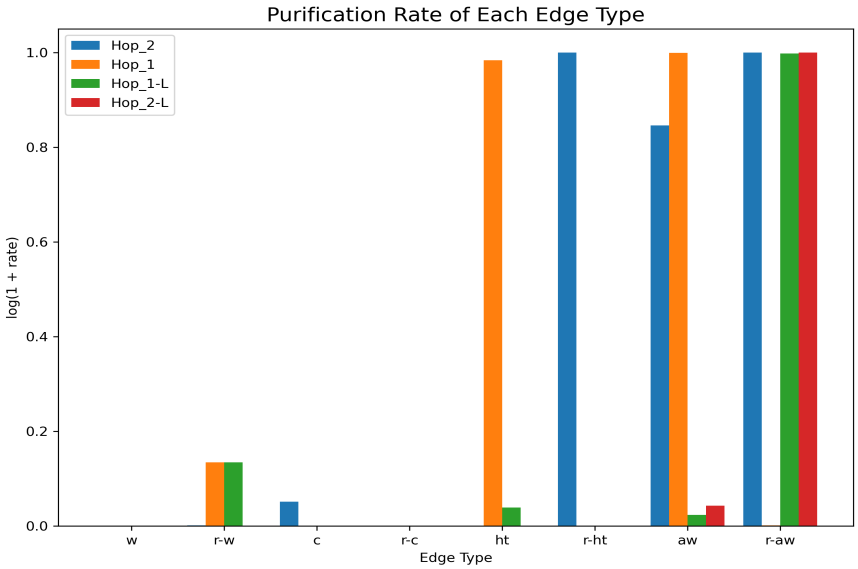

Figure 3. Purification rate of each edge type on OGB-MAG under 2 training settings. "w" means "writes", "c" means "cites", "ht" means "has topic", "aw" means "affiliated with". "r-x" means the reverse type of edge type "x". "Hop_i" denotes neighbours from i-th hop, and "Hop_i-L" denotes "Hop_i" with the training setting of "Label Input".

6. Conclusions

This paper introduces HGPN, a novel framework for heterogeneous graph representation learning. Unlike existing HGNN methods that struggle with structural and informational noise and rely on human intervention, HGPN is designed to autonomously purify both types of heterogeneity without hand-engineered metapaths. HGPN employs type-specific subgraph decomposition to efficiently model heterogeneity and incorporates type-specific batch normalization and cluster-aware loss for feature purification. Additionally, the neighborhood smoothness-related dynamic residual connection improves the optimization of nodes with lower quality features. Experiments on various datasets and training settings demonstrate that HGPN consistently achieves superior performance. Further analysis confirms the effectiveness of HGPN's feature and structure purification methods. Future work will focus on exploring HGPN's causal interpretability and expanding the framework to temporal dynamic graphs for enhanced applicability in realistic scenarios.

Author Contributions: Conceptualization, S.S. and Z.Z.; methodology, S.S.; software, S.S. and S.L.; validation, S.S. and Z.Z.; formal analysis, S.S.; investigation, S.S.; resources, S.S. and D.Z.; data curation, S.S., P.D., Q.L. and X.L.; writing—original draft preparation, S.S.; writing—review and editing, S.S., D.Z., S.L., P.D., Q.L., X.L. and Z.Z.; visualization, S.S.; supervision, D.Z., S.L. and Z.Z.;

project administration, S.S. and D.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used to support this study are available online in https://ogb.stanford.edu (accessed on 2 May 2020) and https://github.com/THUDM/HGB (accessed on 2 May 2023).

Conflicts of Interest: The authors declare that they do not have any conflict of interest. This research does not involve any human or animal participation.

References

1. Sun, Y.; Han, J. Mining heterogeneous information networks: A structural analysis approach. *ACM SIGKDD Explor. Newsl.* **2013**, *14*, 20–28. [CrossRef]
2. Wang, Y.; Sun, H.; Zhao, Y.; Zhou, W.; Zhu, S. A heterogeneous graph embedding framework for location-based social network analysis in smart cities. *IEEE Trans. Ind. Inform.* **2019**, *16*, 2747–2755. [CrossRef]
3. He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; Wang, M. LIGHTGCN: Simplifying and powering graph convolution network for recommendation. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Online, 25–30 July 2020; pp. 639–648.
4. Welling, M.; Kipf, T.N. Semi-supervised classification with graph convolutional networks. In Proceedings of the 5th International Conference on Learning Representations (ICLR 2017), Toulon, France, 24–26 April 2017.
5. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1024–1034.
6. Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *Stat* **2017**, *1050*, 20.
7. Fu, X.; Zhang, J.; Meng, Z.; King, I. MAGNN: Metapath aggregated graph neural network for heterogeneous graph embedding. In Proceedings of the Web Conference 2020, Taipei, Taiwan, 20–24 April 2020; pp. 2331–2341.
8. Hu, Z.; Dong, Y.; Wang, K.; Sun, Y. Heterogeneous graph transformer. In Proceedings of the Web Conference 2020, Taipei, Taiwan, 20–24 April 2020; pp. 2704–2710.
9. Zhu, S.; Zhou, C.; Pan, S.; Zhu, X.; Wang, B. Relation structure-aware heterogeneous graph neural network. In Proceedings of the 2019 IEEE International Conference on Data Mining (ICDM), Beijing, China, 8–11 November 2019; pp. 1534–1539.
10. Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 22118–22133.
11. Lv, Q.; Ding, M.; Liu, Q.; Chen, Y.; Feng, W.; He, S.; Zhou, C.; Jiang, J.; Dong, Y.; Tang, J. Are we really making much progress? Revisiting, benchmarking and refining heterogeneous graph neural networks. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Online, 14–18 August 2021; pp. 1150–1160.
12. Roweis, S.T.; Saul, L.K. Nonlinear dimensionality reduction by locally linear embedding. *Science* **2000**, *290*, 2323–2326. [CrossRef]
13. Belkin, M.; Niyogi, P. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Adv. Neural Inf. Process. Syst.* **2001**, *14*, 585–591.
14. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 3111–3119.
15. Perozzi, B.; Al-Rfou, R.; Skiena, S. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 701–710.
16. Grover, A.; Leskovec, J. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 855–864.
17. Dong, Y.; Chawla, N.V.; Swami, A. metapath2vec: Scalable representation learning for heterogeneous networks. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 135–144.
18. Abu-El-Haija, S.; Kapoor, A.; Perozzi, B.; Lee, J. N-GCN: Multi-scale graph convolution for semi-supervised node classification. In Proceedings of the Uncertainty in Artificial Intelligence, PMLR, Online, 3–6 August 2020; pp. 841–851.
19. Zhang, M.; Cui, Z.; Neumann, M.; Chen, Y. An end-to-end deep learning architecture for graph classification. In Proceedings of the AAAI Conference on Artificial Intelligence, 2018, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
20. Jiang, W.; Luo, J. Graph neural network for traffic forecasting: A survey. *Expert Syst. Appl.* **2022**, *207*, 117921. [CrossRef]
21. Wu, S.; Sun, F.; Zhang, W.; Xie, X.; Cui, B. Graph neural networks in recommender systems: A survey. *ACM Comput. Surv.* **2020**, *55*, 1–37. [CrossRef]

22. Schlichtkrull, M.; Kipf, T.N.; Bloem, P.; van den Berg, R.; Titov, I.; Welling, M. Modeling relational data with graph convolutional networks. In Proceedings of the European Semantic Web Conference, Heraklion, Greece, 3–7 June 2018; Springer: Cham, Switzerland, 2018; pp. 593–607.
23. Zhang, C.; Song, D.; Huang, C.; Swami, A.; Chawla, N.V. Heterogeneous graph neural network. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 793–803.
24. Hong, H.; Guo, H.; Lin, Y.; Yang, X.; Li, Z.; Ye, J. An attention-based graph neural network for heterogeneous structural learning. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 4132–4139.
25. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
26. Alon, U.; Yahav, E. On the bottleneck of graph neural networks and its practical implications. *arXiv* **2020**, arXiv:2006.05205.
27. Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; Yakhnenko, O. Translating embeddings for modeling multi-relational data. *Adv. Neural Inf. Process. Syst.* **2013**, *26* , 2787–2795.
28. Hendrycks, D.; Gimpel, K. Gaussian error linear units (gelus). *arXiv* **2016**, arXiv:1606.08415.
29. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
30. Li, G.; Müller, M.; Ghanem, B.; Koltun, V. Training graph neural networks with 1000 layers. In Proceedings of the International Conference on Machine Learning, PMLR, Online, 18–24 July 2021; pp. 6437–6449.
31. van den Oord, A.; Li, Y.; Vinyals, O. Representation learning with contrastive predictive coding. *arXiv* **2018**, arXiv:1807.03748.
32. Wang, H.; Leskovec, J. Unifying graph convolutional neural networks and label propagation. *arXiv* **2020**, arXiv:2002.06755.
33. Shi, Y.; Huang, Z.; Feng, S.; Zhong, H.; Wang, W.; Sun, Y. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv* **2020**, arXiv:2009.03509.
34. Sun, K.; Lin, Z.; Zhu, Z. Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34; pp. 5892–5899.
35. Sun, C.; Gu, H.; Hu, J. Scalable and adaptive graph neural networks with self-label-enhanced training. *arXiv* **2021**, arXiv:2104.09376.
36. Yu, L.; Shen, J.; Li, J.; Lerer, A. Scalable graph neural networks for heterogeneous graphs. *arXiv* **2020**, arXiv:2011.09679.
37. Zhang, W.; Yin, Z.; Sheng, Z.; Li, Y.; Ouyang, W.; Li, X.; Tao, Y.; Yang, Z.; Cui, B. Graph attention multi-layer perceptron. *arXiv* **2022**, arXiv:2206.04355.
38. Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. *arXiv* **2017**, arXiv:1711.05101.
39. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 8024–8035 .
40. Wang, M.Y. Deep graph library: Towards efficient and scalable deep learning on graphs. In Proceedings of the ICLR Workshop on Representation Learning on Graphs and Manifolds, New Orleans, LA, USA, 6–9 May 2019.
41. Frasca, F.; Rossi, E.; Eynard, D.; Chamberlain, B.; Bronstein, M.; Monti, F. Sign: Scalable inception graph neural networks. *arXiv* **2020**, arXiv:2004.11198.
42. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, PMLR, Lille, France, 7–9 July 2015; pp. 448–456.
43. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer normalization. *arXiv* **2016**, arXiv:1607.06450.
44. Rong, Y.; Huang, W.; Xu, T.; Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv* **2019**, arXiv:1907.10903.