HINPool: A Unified Heterogeneous Graph Pooling Framework for Accurate Molecular and Protein Property Prediction

Ming-Yi Hong[1, 2*], You-Chen Teng[1], Shao-En Lin[1], Chih-Yu Wang[2*], Che Lin[1*†]

[1]National Taiwan University
[2]Academia Sinica

Abstract

Graph pooling has gained significant progress in recent years as an effective solution for graph-level property classification tasks. With the emergence of research on Heterogeneous Information Networks (HINs), this paper argues that graph-level datasets for graph classification should be treated as HINs rather than homogeneous graphs to enhance information aggregation. We propose HINPool, a novel and general graph pooling framework for graph-level property classification with HINs. First, we devise a systematic HIN construction procedure from the original data to capture complex interactions. Next, we introduce a type-aware heterogeneous graph pooling method featuring a Type-Aware Selector (TAS) to select essential nodes and a Readout Aggregator (RA) to fuse critical information into a graph-level representation. Finally, a cross-layer fusion function is applied to combine the output embeddings from each graph pooling layer, creating a final graph representation for downstream classification tasks. Our approach achieves near state-of-the-art performance on widely used graph classification benchmark datasets, demonstrating significant improvements in four out of five datasets. This work redefines the strategy for graph-level property classification with HGNNs and heterogeneous graph pooling to model intricate relationships, enhancing performance without requiring extensive domain-specific knowledge.

Code & Appendix —
https://github.com/ntuidssplab/HINPool

1 Introduction

Graph neural networks (GNNs), such as Graph Convolutional Networks (GCN) (Kipf and Welling 2017) and Graph Attention Networks (GAT) (Veličković et al. 2018), have shown exceptional performance in tasks involving graph-structured data, including community analysis (Shchur and Günnemann 2019), chemical bond analysis (Stokes et al. 2020), and recommender systems (Stokes et al. 2020). These tasks, which include node classification, link prediction, and graph classification, present unique challenges. However, previous works primarily focus on node and link-level tasks, with a lack of attention given to graph-level predictions. Graph classification, in particular, requires not only analyzing individual node information but also determining a representation for the entire graph. To address this, graph pooling has been developed to aggregate node embeddings generated by GNNs into a comprehensive graph-level representation.

Recent advancements in graph pooling techniques aim to reduce graph size and enhance representations. These methods can be categorized into global (Zhang et al. 2018; Lee, Lee, and Kang 2019) and hierarchical (Ying et al. 2018; Gao and Ji 2019; Amouzad et al. 2024) approaches. However, traditional graph pooling techniques typically adopt a homogeneous approach, treating all nodes as the same type and using homogeneous GNNs like GCN or GAT to generate node embeddings, neglecting various node and edge types when designing graph pooling. For instance, datasets used in graph classification tasks, such as molecular property prediction (Debnath et al. 1991; Wale, Watson, and Karypis 2008) and protein structure prediction (Borgwardt et al. 2005; Dobson and Doig 2003), often consist of homogeneous graphs, where atom types or protein structures are embedded as node features. Treating these as the same node type can result in the loss of important information, reducing classification accuracy.

Heterogeneous Information Networks (HINs) (Sun and Han 2012), with multiple node and edge types, offer a more suitable approach to modeling complex real-world data, such as citation networks or molecular structures. Heterogeneous Graph Neural Networks (HGNNs) have been developed for tasks such as node classification (Cen et al. 2019) and link prediction (Schlichtkrull et al. 2018), but their potential for graph classification remains underexplored, particularly for whole-graph classification (Hong et al. 2024).

To address these limitations and fully leverage HGNN for graph-level property classification, we introduce HINPool, a general framework designed to bridge the gap between graph pooling techniques and HINs. This framework incorporates any suitable HGNNs to produce robust node embeddings and employs node-type-aware pooling techniques to coarsen the graph, extracting and retaining crucial information. Our experimental results demonstrate that modeling molecular and protein structures as HINs yields a more representative approach to graph classification.

*Data Science Degree Program, National Taiwan University and Academia Sinica.
†Corresponding author: chelin@ntu.edu.tw.
Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Our main contributions are summarized as follows:

- We introduce HINPool, a unified framework that integrates HGNN encoders with type-aware graph pooling to enable graph-level prediction over HINs. It effectively captures type semantics and is well suited to inherently heterogeneous tasks, such as molecular and protein graph classification.
- We propose a novel type-aware pooling architecture composed of a Type-Aware Selector (TAS), which identifies structurally and semantically essential nodes, and a Readout Aggregator (RA), which consolidates key information across layers into a compact graph-level embedding. A cross-layer fusion module further enhances representation by integrating embeddings from all pooling layers.
- We conduct extensive experiments across five benchmark graph datasets, validating the generalizability of HINPool. Ablation studies confirm the utility of each component, and results demonstrate that modeling datasets as HINs yields better information integration than conventional homogeneous approaches.
- To the best of our knowledge, HINPool is the first general-purpose type-aware graph pooling method for HIN classification. It consistently outperforms state-of-the-art models by fully exploiting node-type heterogeneity, highlighting its broad applicability and robustness.

2 Related Work

Homogeneous Graph Pooling

Homogeneous graph pooling is a technique designed to aggregate node embeddings in graphs where all nodes and edges are of a single type. By summarizing local and global information, these pooling methods generate graph-level representations. DiffPool (Ying et al. 2018) introduced a learnable cluster assignment matrix that softly assigns nodes to different clusters, gradually coarsening the graph. At the final layer, nodes are assigned to a single cluster, generating the final graph-level embedding vector. SAGPool (Lee, Lee, and Kang 2019) employs GNNs to produce self-attention scores for each node, using a Top-K technique to drop nodes with lower scores, either globally or hierarchically. HGP-SL (Zhang et al. 2019) introduced a structure learning mechanism to address the disconnection of nodes resulting from the node-dropping process. Building on HGP-SL, MVPool (Zhang et al. 2021) proposed a multiple-view pooling operator that leverages global graph information to generate more robust node ranks, enhancing node representation.

Graph U-Nets (Gao and Ji 2019), which is inspired by the success of U-Net (Ronneberger, Fischer, and Brox 2015), adapt the U-Net architecture for graph data, stacking pooling and unpooling layers, respectively representing downsampling and upsampling. The pooling layers perform node-dropping according to their scores from a projection layer, while the unpooling layers restore the pooled graph to its higher-resolution structure using positional information stored during pooling.

GIUNet (Amouzad et al. 2024) follows the encoder-decoder architecture of Graph U-Nets but replaces GCN (Kipf and Welling 2017) with the Graph Isomorphism Network (GIN) (Xu et al. 2018) to capture richer structural information. Additionally, GIUNet introduced two novel pooling mechanisms to determine node significance.

Nevertheless, the above pooling methods are developed for homogeneous graphs and neglect the inherent heterogeneity present in many graph datasets, which may result in information loss. This makes HINPool essential as it bridges traditional graph pooling techniques and the complex heterogeneity in graphs.

Heterogeneous Graph Neural Network

Existing HGNNs primarily focus on node-level tasks such as node classification and link prediction (Cen et al. 2019; Schlichtkrull et al. 2018). These methods can be categorized into three main approaches: (i) metapath-based (Wang et al. 2019; Fu et al. 2020; Chang et al. 2022; Yang et al. 2023; Zhang et al. 2020), which requires predefined metapaths to aggregate the metapath-based neighbors' information, (ii) transformer-based (Hu et al. 2020; Yun et al. 2019), which adopts transformers and subgraph sampling to deal with large-scale academic HINs, and (iii) shared feature space designs (Lv et al. 2021; Hong et al. 2023), which are relatively simple models utilizing node feature projection layers and learnable edge-type embeddings. SimpleHGN (Lv et al. 2021), for instance, is a representative shared feature space model that utilizes GAT as the backbone and introduces type-aware linear layers to map all node features for each node type in HINs. It also includes several enhancements, such as learnable edge-type embeddings, residual connections, and L2 normalization of output embeddings. Nevertheless, these studies dedicated to HGNNs focus on node classification and link prediction, leaving heterogeneous graph classification tasks unexplored.

In light of this research gap, our proposed framework aims to extend graph-pooling technology to heterogeneous graph pooling that can directly handle real-world HIN data. We utilize type-aware linear layers and shared feature space HGNNs to generate node embeddings and aggregate to a graph-level embedding for graph classification, emphasizing a general approach to modeling the HINs we construct.

Heterogeneous Graph Pooling

In recent studies, several approaches have begun to leverage the heterogeneity of HINs by incorporating multiple node and edge types to better capture the complex relationships within the data. These works demonstrate promising results, suggesting that modeling graph heterogeneity with HINs can improve performance on graph classification tasks.

For example, UaG (Wu et al. 2021) models each user as a personalized heterogeneous graph based on their unique behaviors with a heterogeneous graph pooling algorithm inspired by DiffPool. It learns separate GNNs for different node types to cluster nodes, allowing for personalized and accurate user modeling. However, UaG has only been applied to E-commerce datasets, raising concerns about its generalizability to different application fields.

Similarly, MPMol (Ji et al. 2023) treats a molecule as a heterogeneous graph and employs numerous predefined

metapaths to capture information related to chemical functional groups. While MPMol effectively captures specific domain knowledge, it relies heavily on extensive domain expertise. This reliance makes it unsuitable for other molecular or protein datasets where an additional node property is absent.

Despite these advancements, these methods still struggle to aggregate information across heterogeneous nodes and edges without losing essential structural and semantic details. This gap highlights the need for further investigation into HINs in graph classification to fully leverage their potential.

To address these limitations, we propose a general heterogeneous graph pooling framework that effectively handles existing homogeneous graph classification datasets, which should naturally be represented as HINs. Our framework pushes the boundaries of graph classification by accounting for the inherent heterogeneity that most existing methods overlook. This approach also enhances performance without requiring extensive domain-specific knowledge.

3 Proposed Method: HINPool

In this section, we present the HINPool framework and the architecture of Type-aware Heterogeneous Graph Pooling (THeGP), as illustrated in Figure 1. In a THeGP layer, the HGNN generates node embeddings, the Type-aware Selector coarsens the graph, and the Readout Aggregator aggregates information into a single graph representation, considering the differences between node types. Finally, we introduce the cross-layer fusion technique that integrates semantic information from the multiple pooling stages to form a comprehensive graph representation for graph classification.

Preliminary

A typical HIN is denoted as $G = (V, E, T_v, T_e, \Phi, \Psi)$, where V is the set of nodes and E is the set of edges. Each node $v \in V$ has a node type $\Phi(v) \in T_v$, and each edge $e \in E$ has an edge type $\Psi(e) \in T_e$, where T_v and T_e denote the sets of node types and edge types.

In HINs, node types are assigned to node categories with corresponding item attributes, and edge types denote relationships between nodes, with edge features representing their characteristics. The corresponding node feature for each node type t is denoted X_t.

HIN construction

We propose a systematic procedure for constructing Heterogeneous Information Networks (HINs) from raw graph data, such as MUTAG (Debnath et al. 1991), NCI1, NCI109 (Wale, Watson, and Karypis 2008), ENZYMES (Dobson and Doig 2003), and PROTEINS (Borgwardt et al. 2005). For each dataset, we construct HINs based on the underlying property or structure of nodes and edges. Specifically, for molecules, the node and edge types indicate atom types and bond types. For proteins, node types represent secondary structural elements (SSEs) of AAs — either helix, sheet, or turn, while edge identifications represent the connections between AAs. Additional implementation details are provided in Appendix A. This approach enables the capture of complex interactions, enhancing performance in graph prediction tasks while minimizing the need for extensive domain-specific knowledge. The derived HINs can be easily applied to the graph pooling method introduced later.

Type-aware Heterogeneous Graph Pooling (THeGP)

Heterogeneous Graph Neural Network (HGNN) In our work, we leverage HGNNs as robust graph encoders for HINs, as shown by the purple blocks in Figure 1, effectively integrating features across various node types. Edge types represent semantic interactions between nodes, which are the main characteristics we consider crucial in HINs. By utilizing learnable embeddings for each edge type, we can enhance the representation capabilities within HINs, enabling more complex interaction modeling. The node embeddings are generated as Eq. (1) within each layer n:

$$Z_t^n = HGNN_t(H_t^{n-1}), \tag{1}$$

where H_t^{n-1} is the input node representations from the previous THeGP layer and $H_t^0 = X_t$. The output Z_t^n represents the generated node embeddings for node type t. In this study, we use SimpleHGN (Lv et al. 2021) as the HGNN encoder. Note that the encoder can be replaced by other HGNNs, highlighting the generalizability of our framework.

Type-Aware Selector (TAS) To capture higher-order semantics, we design a node selector that takes the different properties of distinct node types into account, as the green blocks in Figure 1. In TAS, the graph with HGNN output embeddings is coarsened into a subgraph with a condensed resolution and a larger receptive field, as in Figure 2.

TAS comprises three main modules: score projection, top-k node selection, and embedding attention. Eq. (2) describes the procedure of score projection.

$$S_t^n = MLP_t(Z_t^n). \tag{2}$$

Given the node embeddings Z_t^n of each node type t in layer n, presented with different colors in Figure 2, they are projected into node scores S_t^n by type-aware MLPs with dimension $\in \mathcal{R}^{1\times N_t}$, with N_t being the number of nodes for type t.

To preserve the most important information from the input graph, nodes with higher scores within each node type are selected by the following operation:

$$Z_t^{n\prime} = TopK(Z_t^n, K) \tag{3}$$

Note that K determines the pooling ratio within layers that is applied to all nodes with type t. In Figure 2, the colors of indices of selected nodes are retained, and the indices are used to extract the node embeddings and the adjacency matrix to form a coarsened graph for the next layer. We further discuss the impact of different compositions of pooling ratios in Section 4.

Moreover, we multiplied the embeddings $Z_t^{n\prime}$ of the selected nodes by their projected scores $S_t^{n\prime}$ to enhance the expressiveness as follows:

$$H_t^n = Z_t^{n\prime} \cdot S_t^{n\prime}, \tag{4}$$

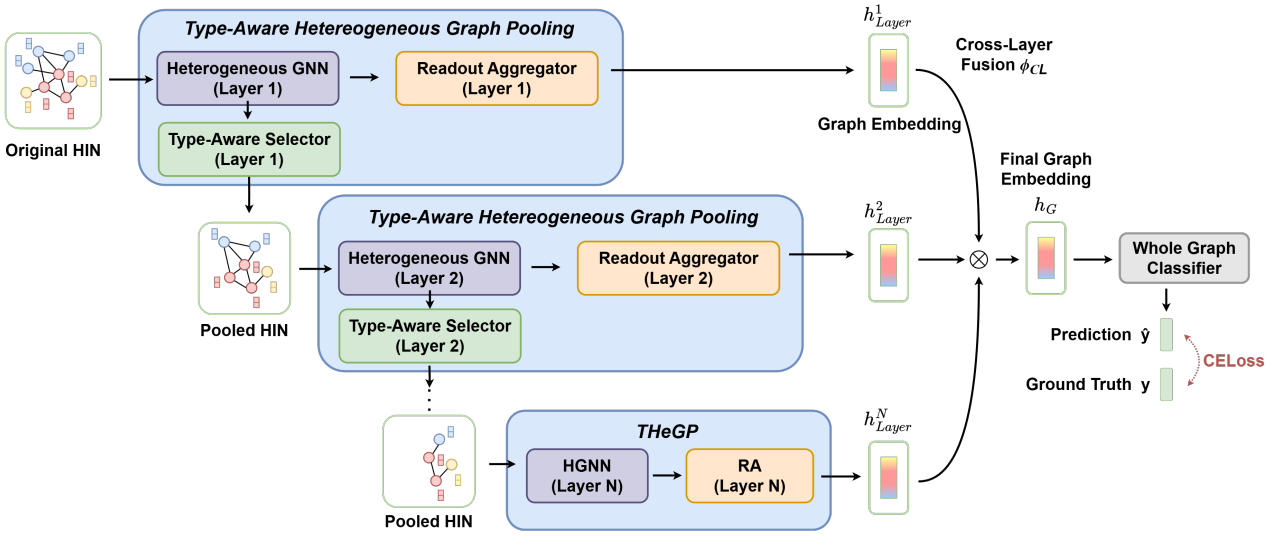

Figure 1: HINPool architecture. The core consists of N layers of the THeGP module, which includes an HGNN, a Readout Aggregator (RA), and a Type-Aware Selector (TAS). The HGNN generates node embeddings, while the RA and TAS take these embeddings to reduce graph size and produce layerwise graph-level embeddings. A cross-layer fusion function combines output embeddings from each THeGP layer into a final graph representation for downstream classification tasks.

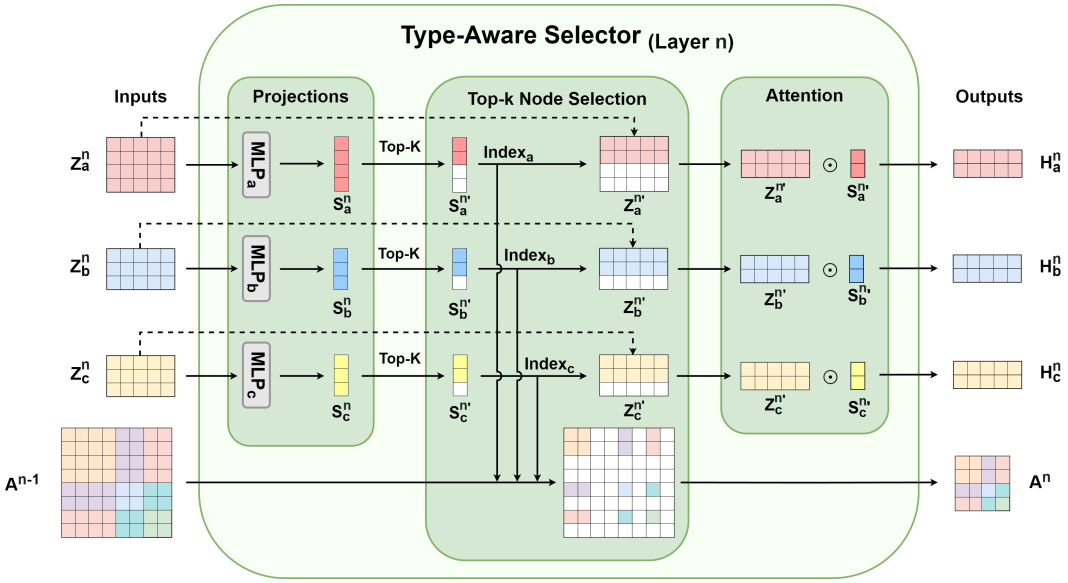

Figure 2: Type-Aware Selector. First, node embeddings Z_t^n for each node type t, generated by the HGNN layer, are input into type-aware MLPs to produce node scores S_t. The Top-K nodes within each type, based on the highest scores, are selected. Their indices are used to slice Z_t^n and the original adjacency matrix A^{n-1}. Finally, the selected node embeddings are multiplied by their respective scores to generate the attended node representations for the next THeGP layer.

where H_t^n is the output node embeddings of the layer n, $Z_t^{n\prime}$ and S_t' are the input node embeddings and node scores of selected nodes, respectively. This ensures that the model attends to nodes with higher scores in the following message-passing layer.

Readout Aggregator (RA) Parallel to the pooling layer, an aggregation of readouts is performed after HGNN obtains the node embeddings (orange block in Figure 1). This module considers the set of embeddings of different node types, and a layerwise graph-level embedding is output. The RA module contains two parts: type-aware readout fusion and mixed-type readout, as depicted in Figure 3. In the type-aware readout fusion part, node embeddings are first separated into sets by their node types, and we take the mean of each set of embeddings to form $|T_v|$ readout vectors. The

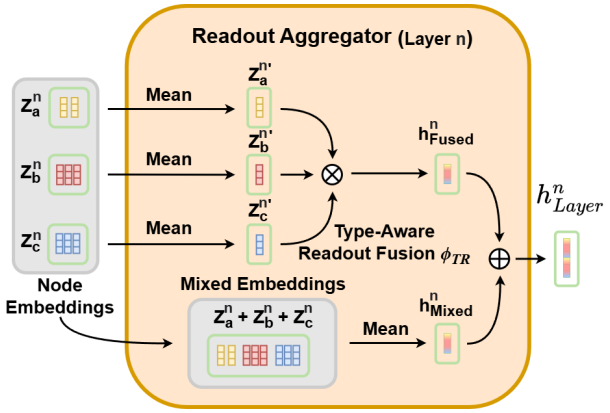

Figure 3: Readout Aggregator. Node embeddings are fused according to their node types by a fusion function ϕ_{TR}, and the global mean is concatenated after.

readout vectors are then fused as follows:

$$h^n_{Fused} = \phi_{TR}(\{Z^{n\prime}_t \mid t \in T_v\}), \tag{5}$$

where h^n_{Fused} is the fused embedding of layer n, and ϕ_{TR} denotes the fusion function.

Several ϕ_{TR} are implemented in the experiment part: (i) Concatenation (Concat), (ii) Element-wise Average (EA), (iii) Element-wise Maximization (EM), (iv) Attention mechanism with weights shared across all layers (Att_shared) as Eq. 6, and (v) Attention mechanism with weights separate across all layers (Att_sep) as Eq. 7.

$$h^n_{Fused} = \sum_{t \in T_v} \alpha_t Z^{n\prime}_t, \tag{6}$$

$$h^n_{Fused} = \sum_{t \in T_v} \alpha^n_t Z^{n\prime}_t, \tag{7}$$

where α_t and α^n_t represent the trainable attention weight of node type t shared across all layers or only for layer n, respectively.

In the mixed-type mean readout part, the embeddings Z^n_t of all nodes are directly averaged regardless of node types to form the mixed-type embedding h_{Mixed} as Eq. 8, as a global view of the graph in model training.

$$h^n_{Mixed} = \frac{1}{|V|} \sum_{v \in V} Z^n_v. \tag{8}$$

After obtaining h^n_{Fused} and h^n_{Mixed}, we concatenate them to get the layerwise graph embedding h^n_{Layer} as Eq. 9.

$$h^n_{Layer} = h^n_{Fused} \,\|\, h^n_{Mixed}\,. \tag{9}$$

Cross-Layer Fusion

To enhance the training procedure by integrating the output graph-level embeddings from each layer, we design a cross-layer fusion ϕ_{CL} to ensure the semantics of different graph resolutions are considered. The graph-level embeddings are fused as follows:

$$h_G = \phi_{CL}(\{h^n_{Layer} \mid n = 1, 2, ..., N\}), \tag{10}$$

where h_G represents the final embedding of the whole graph, h^n_{Layer} is the graph-level embedding output by each HINPool layer, and ϕ_{CL} denotes the cross-layer fusion function.

Similar to readout fusion, we implemented several candidate fusion functions: (i) Concat, (ii) EA, (iii) EM, and (iv) Attention mechanism (Att). Att is presented as:

$$h_G = \sum_{n=1}^{L} \alpha^n \cdot h^n_{Layer}, \tag{11}$$

where α^n denotes the trainable attention weight for layer n.

Lastly, the fused representation h_G is passed to a decoder, which uses a linear layer to predict the label for each graph G. We train the model using cross-entropy loss for both binary and multi-class classification tasks.

4 Experiments

Graph Datasets

To validate the generalizability of HINPool, we evaluate its performance on five datasets that are commonly used in graph classification tasks, including molecular datasets, MUTAG (Debnath et al. 1991), NCI1, and NCI109 (Wale, Watson, and Karypis 2008); and protein structure datasets, PROTEINS (Borgwardt et al. 2005) and ENZYMES (Dobson and Doig 2003). The statistics of datasets are shown in Table 1. In molecular datasets, the nodes represent atoms, and the edges correspond to bonds. On the other hand, in protein structure datasets, nodes represent amino acids (AAs), with node features derived from their physical and chemical properties. Two nodes are connected by an edge if their distance is less than 6 Å.

Dataset	# Graphs	Avg. $\lvert V\rvert$	Avg. $\lvert E\rvert$	# Classes
MUTAG	188	17.93	19.79	2
NCI1	4110	29.87	32.30	2
NCI109	4127	29.68	32.13	2
PROTEINS	1113	39.06	72.82	2
ENZYMES	600	32.63	62.14	6

Table 1: Dataset statistics

Evaluation Protocol

Evaluation Metrics We use AUROC as the primary metric and accuracy as a secondary one. Since all datasets are binary classification except ENZYMES (multi-class), AUROC provides a threshold-independent evaluation.

Split Setting Most graph classification literature (Zhang et al. 2018; Amouzad et al. 2024) uses 10-fold cross-validation (CV), typically reporting the average of the highest testing scores (validation set) achieved during training. However, since 10-fold CV relies on validation set accuracies, it does not fully simulate real-world scenarios and can overestimate model performance.

We follow a general train/validation/test split setting (Prechelt 2002) commonly used in node classification (Lv et al. 2021; Hong et al. 2023), NLP (Yang et al. 2021), and computer vision (Deng et al. 2009) to evaluate model performance unbiasedly. We conduct multiple experiments with varied stratified splits and random seeds. All models utilize

Models	MUTAG		NCI1		NCI109		ENZYMES		PROTEINS	
	AUROC	Acc	AUROC	Acc	AUROC	Acc	AUROC	Acc	AUROC	Acc
DiffPool	97.2±0.9	90.0±2.4	83.7±1.3	79.3±1.9	85.8±1.5	79.6±1.0	85.3±2.6	59.6±1.3	82.7±0.3	60.5±14.0
SAGPool	85.2±6.7	71.1±6.0	74.7±3.2	67.5±3.5	78.3±1.2	71.5±1.6	71.4±6.3	35.6±9.8	77.3±2.6	70.4±3.0
HGP-SL	82.5±6.9	71.1±12.6	81.1±1.4	72.9±3.7	80.9±1.4	73.0±2.4	70.8±9.5	38.3±8.8	77.5±1.1	72.7±5.6
MVPool	88.3±5.2	75.5±3.3	80.1±1.0	72.6±2.5	83.1±1.3	75.6±1.6	77.1±7.2	43.6±10.4	82.6±5.8	77.2±4.3
Graph U-Nets	88.3±4.3	80.0±7.4	75.4±0.4	68.6±1.8	76.1±0.6	67.9±1.5	76.0±1.5	35.3±1.3	89.4±0.2	79.8±1.0
GIUNet	98.0±2.8	91.1±10.6	85.4±1.2	80.2±1.9	**86.5±1.2**	**80.9±0.6**	86.2±1.1	64.3±5.0	69.2±4.2	64.6±2.9
HINPool	**100.0±0.0**	**97.7±4.9**	**87.5±0.8***	**80.3±1.0**	86.2±0.8	78.7±0.6	**92.8±1.1*****	**79.3±4.3****	**89.8±1.5**	**81.2±1.9**

Table 2: Performance comparison of baselines and HINPool. Best results per dataset are in bold, second-best are underlined. We denote * as p-value < 0.05, ** for p-value < 0.01, and *** for p-value < 0.001 compared to the second-best model.

the same dataset splits, with training/validation/testing proportions of 0.8/0.1/0.1, respectively. We report the results of the holdout testing set over 5 runs, employing 100-epoch early stopping based on validation AUROC. For the multi-class ENZYMES dataset, AUROC scores are calculated using the one-vs-rest (OVR) strategy. Hyperparameters and configurations of HINPool are detailed in Appendix B.

Results and Comparison

We compare the performance of HINPool against other state-of-the-art methods: DiffPool (Ying et al. 2018), SAGPool (Lee, Lee, and Kang 2019), HGP-SL (Zhang et al. 2019), MVPool (Zhang et al. 2021), Graph U-Nets (Gao and Ji 2019), and GIUNet (Amouzad et al. 2024). For the baselines, we follow the hyperparameter settings reported in the papers and the default settings in the source codes.

The experiment results are shown in Table 2. Compared to baseline methods, HINPool achieves near state-of-the-art performance on most datasets, with up to a 7% improvement in AUROC over the second-best model on ENZYMES. It demonstrates high stability, as indicated by the low standard deviation across all datasets. The results suggest that our approach significantly advances graph classification tasks by constructing HINs and employing type-aware techniques to capture the complex interactions within graphs, an aspect that the baseline models fail to effectively address.

Specifically, for the MUTAG dataset, where baseline models have achieved exceptionally high scores, HINPool attained a perfect AUROC score. This result highlights the effectiveness of using atom types to construct HINs. In the case of the NCI datasets, HINPool outperforms all baselines on NCI1 but does not perform as well as GIUNet on NCI109. We attribute this slight underperformance to the scarcity of samples for many node types in NCI109, which makes the direct HIN construction using atom types less effective compared to other datasets. A detailed discussion of the NCI datasets is provided in Appendix D.

For protein structure datasets, HINPool also achieves the best performance among all baselines, particularly on ENZYMES, where it significantly surpasses the second-best baseline. This indicates that incorporating the SSEs of amino acids, along with their original node features, substantially enhances the model's classification capacity.

These results validate that the HINPool framework generally pushes the limits of graph classification tasks by effectively constructing molecules and proteins as HINs according to their natural forms, where the atoms and SSEs are modeled as heterogeneous node types.

Ablation Study

We conducted ablation studies on the modules of HINPool using the MUTAG, PROTEINS, and ENZYMES datasets. We observe the impact of each module in the framework, including: (i) **w/o Cross Layer Fusion**: The graph embedding h^n_{Layer} of the last layer is directly the final graph embedding h_G. (ii) **w/o TAS**: The whole node-selection process TAS is discarded. (iii) **TAS - Type-aware Top-K**: Since we want to analyze the impact of selecting node types by type, all calculated node scores are mixed to select Top-K inside TAS for this ablation. (iv) **RA - Type-aware Readout**: Only output the mixed mean readout h_{Mixed} in the RA module.

As shown in Table 3, we observe that the scores with all modules included are consistently the highest. Unplugging any of these modules leads to a noticeable drop in performance. Among all the ablations, the absence of cross-layer fusion results in the most significant performance degradation with 39% on MUTAG, 22.9% on ENZYEMS, and 3.9% on PROTEINS, underscoring the importance of integrating a global view of graphs into the final embedding. On MUTAG, the removal of **RA - Type-aware readout fusion** causes the second-largest degradation with a significant 6.2% followed by **w/o TAS** (-4.8%) and **TAS - Type-aware Top-K** (-3.1%). Each module has significant influence and importance. On PROTEINS, removing the entire **TAS** leads to the second-largest drop, which is a 3.2% decline, emphasizing the crucial role of graph pooling and node score attention. However, when the **TAS - Type-aware Top-K** is switched to **mixed Top-K**, the smallest performance decline occurs with 1.6% and 0.1% degradation on PROTEINS and ENZYMES, respectively. This indicates that there may still be room for improvement in our node selection process to make it more substantial. On ENZYMES, which is a challenging multi-class dataset with typically unstable and relatively low accuracies among baseline models, the removal of **RA - Type-aware Readout** causes the second-largest degradation with a significant 21.1%, suggesting that readout fusion across different node types may help stabilize the training process.

Module	MUTAG		ENZYMES		PROTEINS	
	AUROC	Acc	AUROC	Acc	AUROC	Acc
HINPool	**100.0±0.0** (-)	**97.7±4.9** (-)	**92.8±1.1** (-)	**79.3±4.3** (-)	**89.8±1.5** (-)	**81.2±1.9** (-)
- w/o Cross Layer Fusion	60.8±10.0 (-39%)	60.0±14.9 (-39%)	71.5±4.5 (-23%)	29.3±6.5 (-63%)	86.2±0.6 (- 4%)	80.0±2.8 (-2%)
- w/o TAS	95.2±6.8 (-5%)	90.0±2.4 (-8%)	91.8±1.4 (-1%)	75.0±4.8 (-5%)	86.9±1.3 (-3%)	77.6±1.7 (-4%)
- w/o TAS - Type-aware Top-K	96.9±3.7 (-3%)	83.3±6.8 (-15%)	92.7±2.3 (-0%)	76.67±5.8 (-3%)	88.4±1.3 (-2%)	79.4±2.0 (-2%)
- w/o RA - Type-aware Readout	93.8±2.1 (-6%)	88.8±3.9 (-9%)	73.2±10.1 (-21%)	38.0±16.6 (-52%)	87.3±1.0 (-3%)	79.4±0.9 (-2%)

Table 3: Ablation studies for several modules of HINPool conducted on MUTAG, ENZYMES, and PROTEINS.

Pooling Ratio Within Layers

We compare the performance of different pooling ratios, specifically the ratio of selected nodes in the TAS module and the number of layers of THeGP. The results are presented in Table 4. When tuning pooling ratios, we begin with three layers of THeGP using a ratio of 0.9, observing how the performance varies when adjusting the ratios across all layers. We find that reducing the pooling ratio to 0.8, 0.7, and 0.6 results in a decline in performance, suggesting that dropping too many nodes may inadvertently discard those representative nodes. As the purpose of pooling is to extract and purify information, we suggest that this process can proceed through each layer at a constant ratio, and extraction can be performed at an appropriate rate.

We then adjust the number of THeGP layers while maintaining the ratio of 0.9 to assess the impact of varying the number of layers. As shown in Table 4, the performance comes to a peak with three layers, outperforming configurations with two or four layers, which both show an approximately 2% drop in AUROC. This indicates that underfitting may occur with fewer layers and over-smoothing with more layers. Finally, we tested a configuration that declines the pooling ratio along with layers, using pooling ratios of [0.9, 0.8, 0.7], as explored in other studies (Amouzad et al. 2024), but this did not lead to better results.

Pooling Ratios	AUROC	Acc
0.9 0.9	87.9±1.4	80.5±1.3
0.9 0.9 0.9	**89.8±1.5**	**81.2±1.9**
0.9 0.9 0.9 0.9	87.7±0.7	80.0±1.1
0.9 0.9 0.9	**89.8±1.5**	**81.2±1.9**
0.8 0.8 0.8	88.3±1.3	79.1±2.1
0.7 0.7 0.7	88.6±0.7	78.7±1.2
0.6 0.6 0.6	88.5±1.2	79.6±1.9
0.9 0.9 0.9	**89.8±1.5**	**81.2±1.9**
0.9 0.8 0.7	88.4±1.0	78.5±2.7

Table 4: Pooling Ratio within Layers on PROTEINS.

Fusion Function Discussion

We experiment with various type-aware readout fusion functions on the PROTEINS dataset. The results indicate that concatenation yields the best performance. A detailed discussion of the fusion function is provided in Appendix C.

Complexity of HINPool

HINPool comprises 3 to 5 layers of the THeGP module. Each THeGP includes an HGNN, TAS, and RA, as shown in Figure 1. The HGNN we use is similar to SimpleHGN (Lv et al. 2021), with a time complexity of $O(ne)$, where n is the node count, and e is the number of involved neighbors in message passing. The TAS has a time complexity of $O(n \log n)$ due to the Top-K selection process, while the RA operates in $O(n)$, involving embedding averaging and weighted summation. In conclusion, the overall complexity is reduced to $O(n(\log n + e))$, making it scalable for real-world applications. We also performed a runtime test on the PROTEINS testing set, which consists of 111 graphs in total, and the average runtime for HINPool was 0.31 seconds over five trials.

Limitations and Future Work

While HINPool consistently improves performance across datasets, the gains on molecular graphs are relatively modest. This may be because traditional models already capture key patterns through local structures and domain-specific features. Thus, explicitly modeling heterogeneity offers limited additional value. In contrast, HINPool shows substantial gains on protein-related graphs, where complex relational structures and type diversity are central to the task. Extending HINPool to such scenarios offers a promising direction for future work.

5 Conclusion

In this study, we introduced HINPool, a general and effective framework for graph-level property classification on HINs. By systematically constructing HINs from typed nodes and edges and incorporating type-aware pooling mechanisms—including a Type-Aware Selector, Readout Aggregator, and cross-layer fusion—HINPool effectively captures both structural and semantic diversity. Experiments across four benchmark datasets demonstrate consistent performance gains over state-of-the-art pooling methods, and ablation studies confirm the significance of each proposed component. Overall, our results underscore the importance of treating graph-level classification tasks through the lens of heterogeneity. HINPool provides a unified and generalizable approach for integrating HGNNs and type-aware pooling, laying a foundation for future advances in graph-level learning on heterogeneous data.

Acknowledgments

This work was supported in part by the National Science and Technology Council and AviviD.ai under Grant 114-2622-E-002-018, and National Taiwan University and Academia Sinica under Grant NTU-AS-114L104302. Chih-Yu Wang was supported by the National Science and Technology Council under Grant 111-2628-E-001-002-MY3, 114-2221-E-001-017-MY2, and the Academia Sinica under Research Grant AS-KPQ-112-NETZ-10-A.

References

Amouzad, A.; Dehghanian, Z.; Saravani, S.; Amirmazlaghani, M.; and Roshanfekr, B. 2024. Graph isomorphism U-Net. *Expert Systems with Applications*, 236: 121280.

Borgwardt, K. M.; Ong, C. S.; Schönauer, S.; Vishwanathan, S.; Smola, A. J.; and Kriegel, H.-P. 2005. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1): i47–i56.

Cen, Y.; Zou, X.; Zhang, J.; Yang, H.; Zhou, J.; and Tang, J. 2019. Representation learning for attributed multiplex heterogeneous network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 1358–1368.

Chang, Y.; Chen, C.; Hu, W.; Zheng, Z.; Zhou, X.; and Chen, S. 2022. Megnn: Meta-path extracted graph neural network for heterogeneous graph representation learning. *Knowledge-Based Systems*, 235: 107611.

Debnath, A. K.; Lopez de Compadre, R. L.; Debnath, G.; Shusterman, A. J.; and Hansch, C. 1991. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34(2): 786–797.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Dobson, P. D.; and Doig, A. J. 2003. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4): 771–783.

Fu, X.; Zhang, J.; Meng, Z.; and King, I. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of the web conference 2020*, 2331–2341.

Gao, H.; and Ji, S. 2019. Graph u-nets. In *international conference on machine learning*, 2083–2092. PMLR.

Hong, M.-Y.; Chang, S.-Y.; Hsu, H.-W.; Huang, Y.-H.; Wang, C.-Y.; and Lin, C. 2023. TreeXGNN: can gradient-boosted decision trees help boost heterogeneous graph neural networks? In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. IEEE.

Hong, M.-Y.; Huang, Y.-H.; Lin, S.-E.; Teng, Y.-C.; Wang, C.-Y.; and Lin, C. 2024. SynHING: Synthetic Heterogeneous Information Network Generation for Graph Learning and Explanation. arXiv:2401.04133.

Hu, Z.; Dong, Y.; Wang, K.; and Sun, Y. 2020. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*, 2704–2710.

Ji, Y.; Wan, G.; Zhan, Y.; and Du, B. 2023. Metapath-fused heterogeneous graph network for molecular property prediction. *Information Sciences*, 629: 155–168.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

Lee, J.; Lee, I.; and Kang, J. 2019. Self-attention graph pooling. In *International conference on machine learning*, 3734–3743. pmlr.

Lv, Q.; Ding, M.; Liu, Q.; Chen, Y.; Feng, W.; He, S.; Zhou, C.; Jiang, J.; Dong, Y.; and Tang, J. 2021. 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 & data mining*, 1150–1160.

Prechelt, L. 2002. Early stopping-but when? In *Neural Networks: Tricks of the trade*, 55–69. Springer.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, 234–241. Springer.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*, 593–607. Springer.

Shchur, O.; and Günnemann, S. 2019. Overlapping Community Detection with Graph Neural Networks. arXiv:1909.12201.

Stokes, J. M.; Yang, K.; Swanson, K.; Jin, W.; Cubillos-Ruiz, A.; Donghia, N. M.; MacNair, C. R.; French, S.; Carfrae, L. A.; Bloom-Ackermann, Z.; et al. 2020. A deep learning approach to antibiotic discovery. *Cell*, 180(4): 688–702.

Sun, Y.; and Han, J. 2012. *Mining heterogeneous information networks: principles and methodologies*. Morgan & Claypool Publishers.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Wale, N.; Watson, I. A.; and Karypis, G. 2008. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14: 347–375.

Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; and Yu, P. S. 2019. Heterogeneous graph attention network. In *The world wide web conference*, 2022–2032.

Wu, C.; Wu, F.; Huang, Y.; and Xie, X. 2021. User-as-Graph: User Modeling with Heterogeneous Graph Pooling for News Recommendation. In *IJCAI*, 1624–1630.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Yang, S.-w.; Chi, P.-H.; Chuang, Y.-S.; Lai, C.-I. J.; Lakhotia, K.; Lin, Y. Y.; Liu, A. T.; Shi, J.; Chang, X.; Lin, G.-T.; et al. 2021. Superb: Speech processing universal performance benchmark. *arXiv preprint arXiv:2105.01051*.

Yang, X.; Yan, M.; Pan, S.; Ye, X.; and Fan, D. 2023. Simple and efficient heterogeneous graph neural network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 10816–10824.

Ying, Z.; You, J.; Morris, C.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31.

Yun, S.; Jeong, M.; Kim, R.; Kang, J.; and Kim, H. J. 2019. Graph transformer networks. *Advances in neural information processing systems*, 32.

Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Zhang, W.; Fang, Y.; Liu, Z.; Wu, M.; and Zhang, X. 2020. mg2vec: Learning relationship-preserving heterogeneous graph representations via metagraph embedding. *IEEE Transactions on Knowledge and Data Engineering*, 34(3): 1317–1329.

Zhang, Z.; Bu, J.; Ester, M.; Zhang, J.; Li, Z.; Yao, C.; Dai, H.; Yu, Z.; and Wang, C. 2021. Hierarchical multi-view graph pooling with structure learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(1): 545–559.

Zhang, Z.; Bu, J.; Ester, M.; Zhang, J.; Yao, C.; Yu, Z.; and Wang, C. 2019. Hierarchical graph pooling with structure learning. *arXiv preprint arXiv:1911.05954*.