How Useful Is Graph Pooling for Node-Level Tasks?

Yijun Duan[1](✉), Xin Liu[2], Steven Lynden[2], Akiyoshi Matono[2], and Qiang Ma[1]

[1] Kyoto Institute of Technology, Kyoto, Japan
{yijun,qiang}@kit.ac.jp
[2] AIST, Tokyo, Japan
{xin.liu,steven.lynden,a.matono}@aist.go.jp

Abstract. As an essential component of graph neural networks, graph pooling is indispensable for graph-level tasks such as graph classification and generation. However, certain node-level tasks inherently require graph pooling, particularly *multiple instance learning (MIL) on graphs*, a weakly supervised learning paradigm where only set-level labels are available for training node-level predictors. Existing *embedding-based pooling* aggregates node embeddings to obtain a holistic graph-level representation, neglecting direct inference of node labels. To address this limitation, we propose *instance-based pooling*, which maps node embeddings to node probabilities before generating graph representations. We prove that embedding-based pooling methods can be seamlessly transformed into instance-based ones without losing permutation invariance or expressiveness, while the latter offers better interpretability. Extensive experiments on diverse benchmark datasets validate the effectiveness of our proposed method, providing key insights into the selection of pooling methods for different machine learning tasks on graphs.

Keywords: Graph Pooling · Graph Multiple Instance Learning · Instance-level Learning · Weakly Supervised Learning

1 Introduction

Graph representation learning (GRL) is one of the most effective solutions for machine learning tasks on graphs [21]. It consists of two fundamental components: the *message passing* module, which learns node embeddings, and the *pooling* module, which aggregates node information into subgraph representations. Node-level tasks typically rely solely on message passing, while graph-level tasks require both components.

However, *certain node-level tasks inherently rely on graph pooling.* A key reason is the difficulty of obtaining sufficient node labels due to data annotation constraints. Consider the task of *multiple instance learning (MIL) on graphs.* MIL is a weakly supervised learning paradigm where instances are grouped into sets, called bags, and only bag-level labels are available for training instance-level predictors. Traditional MIL assumes that instances are i.i.d., while given

the versatility of graphs in modeling complex relationships in real-world scenarios, extending MIL to graph data can facilitate various practical tasks and attract significant interest. For example, in drug discovery, the goal is to identify pharmacophore molecules (instance/node labels) that contribute to the properties of complex molecular compounds (bag/graph labels). In such a scenario, obtaining bag labels is often significantly more cost-effective and feasible than acquiring instance labels. Fig. 1 illustrates the distinction between semi-supervised learning and multiple instance learning on graphs.

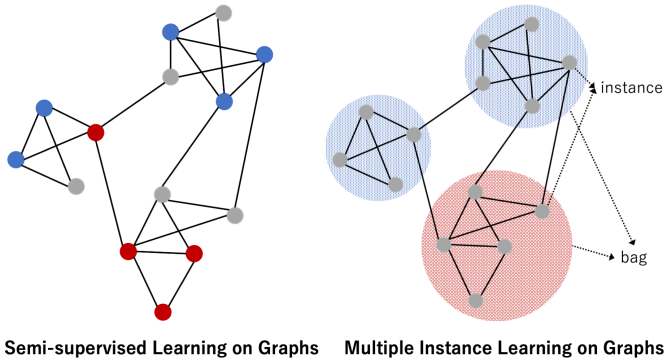

Fig. 1. An illustration of semi-supervised learning and multiple instance learning (our task) on graphs. The colors represent node labels, with gray nodes indicating unlabeled nodes. In multiple instance learning, nodes (called instances) are arranged in sets (called bags), and bag labels are determined by unknown instance labels based on different assumptions. To predict instance labels based on *bag labels*, both the message passing layer and the graph pooling layer are essential.

Can existing graph pooling methods effectively address node-level MIL tasks? Mainstream pooling approaches aggregate instance embeddings into a bag embedding, followed by a classifier computing the bag probability [19]. We refer to this as **embedding-based pooling**. This training paradigm does not involve any inference or fitting of instance labels. To address this limitation, inspired by instance-based MIL methods for non-graph data (e.g., [25, 27]), we propose a novel graph pooling paradigm in this paper: **instance-based pooling**. Instead of directly aggregating embeddings, it first maps instance embeddings to instance probabilities and then constructs bag probabilities based on these instance probabilities. We demonstrate that, under certain conditions, any embedding-based pooling method can be seamlessly transformed into an instance-based pooling method without any loss of permutation-invariance or expressive power. Furthermore, we show that, due to the linearity of instance-based models, SHAP values [20] provide a more precise characterization of each instance's contribution to the model's decision compared to attention scores.

We analyze the performance of embedding-based pooling and instance-based pooling using synthetic bags on four benchmark datasets. Specifically, we evaluate four commonly used attention modules and two feature aggregators. Two strategies for generating bag labels from instance labels are examined. While our primary focus is instance-level classification, we also conduct additional experiments on bag-level tasks to assess the generality of the pooling models. The main experimental findings are summarized as follows: (1) instance-based pooling is more effective in instance classification tasks under the standard MIL assumption; (2) both pooling strategies exhibit comparable performance for instance classification under the collective MIL assumption; (3) for bag classification under the standard MIL assumption, embedding-based pooling is more effective.

In summary, our contributions are as follows:

- We introduce the instance-level graph multiple instance learning task for the first time, providing a deep exploration of the role of graph pooling in node-level tasks.
- We propose a straightforward and general transformation rule that converts embedding-based pooling models into instance-based models.
- We systematically analyze the key theoretical properties of embedding-based and instance-based models, including their permutation invariance, expressiveness and interpretability.
- We conduct extensive experiments on standard graph datasets across various MIL tasks, which offer valuable insights into the selection of pooling methods when handling graph structures at different levels.

2 Problem Formulation and Notations

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote the input graph with node set $\mathcal{V}$ and edge set $\mathcal{E}$. $\mathcal{V} = \{v_1, v_2, ..., v_n\}$, where n is the number of nodes. k denotes the number of classes. Nodes (called instances) are arranged in sets (called bags). Only bag labels are available during model training.

Let $\mathcal{X}$ denote a bag consisting of m instances $\{v_1, v_2, ..., v_m\}$. In MIL, bag label $\mathcal{Y}$ is determined by unknown instance labels $\{y_1, y_2, ..., y_m\}$ based on different assumptions. In this paper, the standard assumption [8] is mainly focused. Given a trigger class $t \in \{1, 2, ..., k\}$, we refer to the nodes that belong to t as positive instances, and the other nodes as negative instances. The standard MIL assumption states that positive bags contain at least one positive instance, and negative bags contain only negative instances. Formally, $\mathcal{Y}$ is determined by:

$$\mathcal{Y} = \begin{cases} +1 & if \quad \exists y \in \{y_1, y_2, ..., y_m\}, y = t \\ -1 & if \quad \forall y \in \{y_1, y_2, ..., y_m\}, y \neq t \end{cases} \tag{1}$$

Our target is then to learn an instance-level binary classifier to predict the label of nodes within the bags. Our work is useful in many real-world scenarios where it is desirable to detect key instances that trigger the bag label, such as automating cancer diagnosis and grading.

3 Preliminary: Embedding-Based Pooling

When dealing with set-level representation learning on graphs, a widely used solution is **embedding-based pooling** [19]. In our problem, it computes the bag probability $\theta(\mathcal{X}) \in [0, 1]$ in the following way:

$$\theta(\mathcal{X}) = \texttt{CLS}(\texttt{AGGR}(\{\alpha_i \cdot \texttt{MP}(v_i), v_i \in \mathcal{X}\})) \tag{2}$$

$$\alpha_i = \texttt{ATT}(v_i) \tag{3}$$

Here, $\theta(\mathcal{X})$ consists of 4 functions: `MP`, `CLS`, `ATT`, and `AGGR`. `MP` consists of message-passing layer(s), and works as a node feature extractor. Attention module `ATT` computes the attention weight α_i for node v_i, which can be used to measure the instance contribution in the bag representation and interpret model predictions. The aggregator `AGGR` aggregates the attended instance embeddings to obtain a bag representation. The most common practice is `sum`, with alternative choices including `max` [23]. Finally, the bag embedding is processed by the classifier `CLS`, and a bag label is predicted. The selection of `MP`, `CLS`, `ATT` and `AGGR` jointly determines the implementation of a pooling method and its effectiveness and complexity.

4 Proposed Method: Instance-Based Pooling

One inherent limitation of embedding-based pooling is its inability to train instances in the label space. For node-level MIL tasks, a natural assumption is that an effective pooling method should be capable of estimating instance probabilities during training. To achieve this, we do not require complex designs—simply swapping `CLS` and `AGGR` in embedding-based models suffices. We refer to the new model as **instance-based pooling**, which is formulated as follows:

$$\theta(\mathcal{X}) = \texttt{AGGR}(\{\texttt{CLS}(\alpha_i \cdot \texttt{MP}(v_i)), v_i \in \mathcal{X}\}) \tag{4}$$

$$\alpha_i = \texttt{ATT}(v_i) \tag{5}$$

Different from embedding-based pooling, `CLS` learns the instance probability $\theta(v_i)$ during training, which is then combined by `AGGR` to form the bag probability $\theta(\mathcal{X})$. This allows the learning model to effectively accommodate both bag-level training and instance-level inference [3]. Similarly, the implementations of `MP`, `ATT`, `CLS` and `AGGR` are task-specific. Finally, an illustration of aforementioned two types of pooling paradigms is shown below in Fig. 2.

[3] Another alternative design of instance-based pooling follows the formulation $\theta(\mathcal{X}) = \texttt{AGGR}(\alpha_i \cdot \{\texttt{CLS}(\texttt{MP}(v_i)), v_i \in \mathcal{X}\}), \alpha_i = \texttt{ATT}(v_i)$.

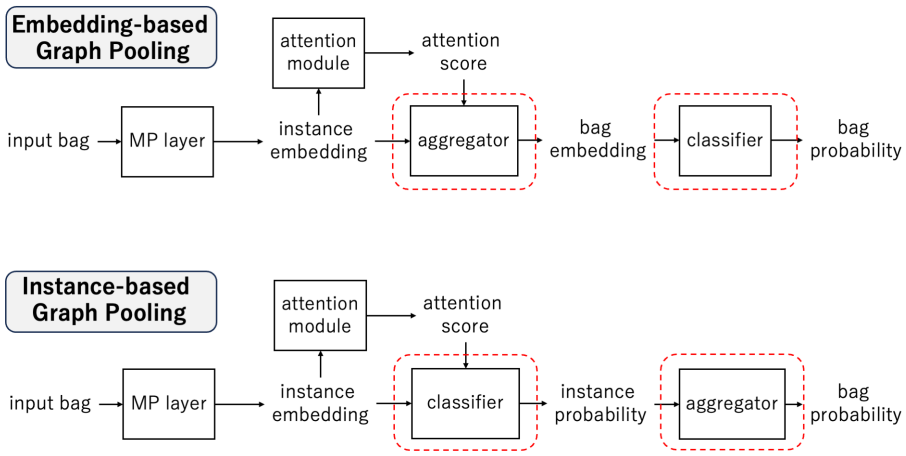

Fig. 2. A comparison between **embedding-based pooling** (existing) and **instance-based pooling** (proposed). Embedding-based pooling: *instance embedding -> bag embedding -> bag probability*; instance-based pooling: *instance embedding -> instance probability -> bag probability.*

5 Key Theoretical Properties of Pooling Models

In this section, we first prove that the transformation from embedding-based pooling to instance-based pooling does not compromise the original model's permutation invariance (Sec. 5.1) or expressive power (Sec. 5.2), which are key properties of MIL models and graph neural networks, respectively. Finally, by investigating the relationship between instance probability and SHAP values (Sec. 5.3), we demonstrate that instance-based models offer superior interpretability.

5.1 Permutation-invariance

The standard MIL assumption (please refer to Eq. (1)) suggests that neither the ordering nor the dependency of instances within a bag exists. Therefore, when computing the bag probability $\theta(\mathcal{X})$, for any permutation π, we should have $\theta(\{v_1, v_2, ..., v_m\}) = \theta(\{v_{\pi(1)}, v_{\pi(2)}, ..., v_{\pi(m)}\})$. Such property is referred to as *permutation-invariance*. To verify whether the pooling models possess this property, we first introduce the following Fundamental Theorem of Symmetric Functions with monomials [30, 11]:

Theorem 1. *A function $F(\mathcal{X})$ operating on a countable set $\mathcal{X}$ is invariant to the permutation of instances in $\mathcal{X}$ if and only if it can be decomposed in the form $\rho(\sum_{v_i \in \mathcal{X}} \psi(v_i))$, where ρ and ψ are suitable transformations.*

Theorem 1 introduces a general form of functions that satisfy the property of permutation invariance. Based on it, we make the following corollary:

Corollary 1. *When* `AGGR` *is the summation operater, both embedding-based pooling and instance-based pooling are invariant to permutation.*

Proof. The set function $F(\mathcal{X})$ in Theorem 1 is the bag probability $\theta(\mathcal{X})$. For embedding-based pooling, when `sum` is the aggregator, transformation ψ and ρ corresponds to $\alpha_i \cdot \texttt{MP}(v_i)$ and `CLS`, respectively. For instance-based pooling, ψ is the instance probability $\texttt{CLS}(\alpha_i \cdot \texttt{MP}(v_i))$, and ρ is the identity function. □

5.2 Expressiveness

The expressive power of a GNN refer to its ability to encode and process feature information on the graph as well as graph structure information, e.g., the ability to distinguish non-isomorphic graphs [31]. In graph theory, the Weisfeiler-Leman test (WL test) is a classic heuristic method to judge whether two graphs are isomorphic [16]. We prove below that when certain conditions are satisfied, both embedding-based pooling and instance-based pooling are capable of retaining distinct information in the graph. To begin, we first state the following theorem concerning the expressive power of graph pooling operators [1]:

Theorem 2. *For WL-distinguishable graphs, a graph pooler satisfying the following conditions will produce coarsened graphs that remain WL-distinguishable: (1) the message-passing layers compute different sums of node features; (2) the cluster assignment matrix is right-stochastic up to a constant λ; (3) the features of supernodes are convex combinations of the input node features.*

We assume that the pooling models discussed in Sec. 3 and Sec. 4 satisfy the above properties. Note that in this case, `MP` is implied to be powerful, and `AGGR` is implied to be the summation operator. We prove below that, under additional constraints on `CLS`, both instance-based pooling and embedding-based pooling preserve the expressive power of `MP`:

Corollary 2. *Let $\mathcal{X}^1$ and $\mathcal{X}^2$ denote two WL-distinguishable bags, the pooled bags $\hat{\mathcal{X}}^1$ and $\hat{\mathcal{X}}^2$ generated by an instance-based model remain WL-distinguishable if the following additional conditions hold: (4)* `CLS` *is an additive function; (5)* `CLS` *is an injective function.*

Proof. Suppose that $\mathcal{X}^1/\mathcal{X}^2$ contains m nodes, while $\hat{\mathcal{X}}^1/\hat{\mathcal{X}}^2$ contains l supernodes ($l < m$). Let $\mathcal{U}^1$, $\mathcal{U}^2$, $\hat{\mathcal{U}}^1$ and $\hat{\mathcal{U}}^2$ denote the corresponding node embedding/probability matrix of $\mathcal{X}^1$, $\mathcal{X}^2$, $\hat{\mathcal{X}}^1$ and $\hat{\mathcal{X}}^2$, respectively. Here, $\mathcal{U}^1$ and $\mathcal{U}^2$ are computed by `MP`. Given condition 1, we have:

$$\sum_{i=1}^{m} \mathcal{U}_i^1 \neq \sum_{i=1}^{m} \mathcal{U}_i^2 \tag{6}$$

Here, $\mathcal{U}_i^1$ and $\mathcal{U}_i^2$ denote the i-th row of $\mathcal{U}^1$ and $\mathcal{U}^2$ respectively, which correspond to the vector of the i-th instance in the bag. Suppose that $\hat{\mathcal{X}}^1$ and $\hat{\mathcal{X}}^2$ are not WL-distinguishable. Then, there exists a one-to-one mapping ω from $\{1, 2, ..., l\}$ to $\{1, 2, ..., l\}$ such that:

$$\hat{\mathcal{U}}_j^1 = \hat{\mathcal{U}}_{\omega(j)}^2, \forall j = 1, 2, ..., l \tag{7}$$

Let $\mathcal{M} \in \mathbb{R}^{m \times l}$ denote the assignment matrix, where $\mathcal{M}_{ij}$ represents the weight that assigns a node i to a supernode j. Given condition 3, we have:

$$\sum_{i=1}^{m} \texttt{CLS}(\mathcal{U}_i^1 \cdot \mathcal{M}_{ij}^1) = \sum_{i=1}^{m} \texttt{CLS}(\mathcal{U}_i^2 \cdot \mathcal{M}_{i\omega(j)}^2), \forall j = 1, 2, ..., l \tag{8}$$

Give condition 4, we can swap CLS with the summation operator as:

$$\texttt{CLS}(\sum_{i=1}^{m} \mathcal{U}_i^1 \cdot \mathcal{M}_{ij}^1) = \texttt{CLS}(\sum_{i=1}^{m} \mathcal{U}_i^2 \cdot \mathcal{M}_{i\omega(j)}^2), \forall j = 1, 2, ..., l \tag{9}$$

Given condition 5, we have:

$$\sum_{i=1}^{m} \mathcal{U}_i^1 \cdot \mathcal{M}_{ij}^1 = \sum_{i=1}^{m} \mathcal{U}_i^2 \cdot \mathcal{M}_{i\omega(j)}^2, \forall j = 1, 2, ..., l \tag{10}$$

We sum the vectors of all supernodes:

$$\sum_{j=1}^{l} \sum_{i=1}^{m} \mathcal{U}_i^1 \cdot \mathcal{M}_{ij}^1 = \sum_{j=1}^{l} \sum_{i=1}^{m} \mathcal{U}_i^2 \cdot \mathcal{M}_{i\omega(j)}^2 \tag{11}$$

which can be rewritten as:

$$\sum_{i=1}^{m} \mathcal{U}_i^1 \cdot \sum_{j=1}^{l} \mathcal{M}_{ij}^1 = \sum_{i=1}^{m} \mathcal{U}_i^2 \cdot \sum_{j=1}^{l} \mathcal{M}_{i\omega(j)}^2 \tag{12}$$

Since $\sum_{j=1}^{l} \mathcal{M}_{ij}^1 = \sum_{j=1}^{l} \mathcal{M}_{i\omega(j)}^2 = \lambda$, $\forall i = 1, 2, ..., m$ (Cond. 2). We have:

$$\sum_{i=1}^{m} \mathcal{U}_i^1 = \sum_{i=1}^{m} \mathcal{U}_i^2 \tag{13}$$

Notice that the above equation contradicts Eq. (6). Therefore, $\hat{\mathcal{X}}^1$ and $\hat{\mathcal{X}}^2$ remain WL-distinguishable. □

For embedding-based models, note that Eq. (9) can be directly derived from Eq. (7). That means, it is not necessary to assume the additivity of CLS function to generate WL-distinguishable coarsened bags, while the remaining four conditions are required to hold.

5.3 Interpretability

In this study, we adopt the SHAP framework [20] to analyze the interpretability of MIL methods motivated by [12], where SHAP values provide insights into how each input instance contributes to a bag-level prediction. A key distinction from [12] is that our analysis targets at input instances which are not mutually independent, which is a fundamental characteristic of graph-structured data.

Clearly, instance-based pooling is a linear model. Regarding the independence of instances, we consider the following two cases: (1) the interdependence among instances will be sufficiently captured by `MP`, and thus instances can be assumed independent during the pooling phase for computational simplicity; (2) instances remain dependent during pooling. We discuss both cases separately below. In what follows, we denote the SHAP value of $\theta(v_i)$ as $\phi(v_i)$.

Independent Instances For linear models with independent features, the computation of SHAP values relies on the following theorem [20]:

Theorem 3. *Give a linear regression model $f(x) = \sum_{i=1}^{m} \omega_i x_i + b$, where all features $x_1, x_2, ..., x_m$ are independent, we have:*

$$\phi(x_i) = \omega_i(x_i - E[x_i]) \tag{14}$$

which immediately inspires our computation as follows:

Corollary 3. *For instance-based pooling models, $\phi(v_i)$ takes the following simple form, where instance probabilities $\theta(v_1)$, $\theta(v_2)$, ..., $\theta(v_m)$ are independent:*

$$\phi(v_i) = \theta(v_i) - E[\theta(v_i)] \tag{15}$$

Proof. In Eq. (14), x_i, ω_i, and b correspond to $\theta(v_i)$, 1, and 0 in Eq. (4), respectively. Note that `AGGR` is required to be the summation operator. □

Dependent Instances For linear models with dependent features, the computation of SHAP values becomes more complex. First, we state the following theorem regarding the contribution function $\tau(\cdot)$ defined on an instance subset $\mathcal{X}_\mathcal{S}$, where $\mathcal{X}_\mathcal{S}^*$ denote a subset of input instances:

Theorem 4. *For instance-based pooling models, $\tau(\mathcal{X}_\mathcal{S})$ takes the following form, where instance probabilities $\theta(v_1)$, $\theta(v_2)$, ..., $\theta(v_m)$ are dependent:*

$$\tau(\mathcal{X}_\mathcal{S}) = \sum_{v_i \in \mathcal{X}_{\overline{\mathcal{S}}}} E[\theta(v_i) | \mathcal{X}_\mathcal{S} = \mathcal{X}_\mathcal{S}^*] + \sum_{v_i \in \mathcal{X}_\mathcal{S}} \theta(v_i) \tag{16}$$

Proof. We make the following derivations:

$$\tau(\mathcal{X}_{\mathcal{S}}) = E[\theta(\mathcal{X})|\mathcal{X}_{\mathcal{S}} = \mathcal{X}_{\mathcal{S}}^*] \tag{17}$$

$$= E[\sum_{v_i \in \mathcal{X}_{\overline{\mathcal{S}}}} \theta(v_i) + \sum_{v_i \in \mathcal{X}_{\mathcal{S}}} \theta(v_i)|\mathcal{X}_{\mathcal{S}} = \mathcal{X}_{\mathcal{S}}^*] \tag{18}$$

$$= \sum_{v_i \in \mathcal{X}_{\overline{\mathcal{S}}}} E[\theta(v_i)|\mathcal{X}_{\mathcal{S}} = \mathcal{X}_{\mathcal{S}}^*] + \sum_{v_i \in \mathcal{X}_{\mathcal{S}}} E[\theta(v_i)|\mathcal{X}_{\mathcal{S}} = \mathcal{X}_{\mathcal{S}}^*] \tag{19}$$

$$= \sum_{v_i \in \mathcal{X}_{\overline{\mathcal{S}}}} E[\theta(v_i)|\mathcal{X}_{\mathcal{S}} = \mathcal{X}_{\mathcal{S}}^*] + \sum_{v_i \in \mathcal{X}_{\mathcal{S}}} \theta(v_i) \tag{20}$$

Notice that a key distinction from independent instances is that, when $v_i \in \mathcal{X}_{\overline{\mathcal{S}}}$, we have $p(\theta(v_i)|\mathcal{X}_{\mathcal{S}} = \mathcal{X}_{\mathcal{S}}^*) \neq p(\theta(v_i))$, which leads to $E[\theta(v_i)|\mathcal{X}_{\mathcal{S}} = \mathcal{X}_{\mathcal{S}}^*] \neq E[\theta(v_i)]$. In this case, solving for $\tau(\mathcal{X}_{\mathcal{S}})$ depends on a proper and efficient estimation of $E[\theta(v_i)|\mathcal{X}_{\mathcal{S}} = \mathcal{X}_{\mathcal{S}}^*]$, which is typically more difficult than estimating $E[\theta(v_i)]$. Given $\tau(\mathcal{X}_{\mathcal{S}})$, $\phi(v_i)$ can be then computed based on the definition of the Shapley value [20]. □

6 Experimental Settings

6.1 Datasets

We use four widely-used benchmark datasets consisting of diverse types of networks, including Cora-ML [2], CiteSeer [28], Amazon [24] and Actor [22]. All of the above datasets are available at PyTorch Geometric Library [7]. Below is a brief description of each dataset above.

- Cora-ML is a citation network consisting of 2,995 scientific publications classified into one of 7 classes. Each publication in the dataset is described by a 2879-dim 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The network contains 16,316 citation links in total.
- CiteSeer is another classical citation network containing of 3,327 scientific publications from 6 research areas, such as AI, DB, and HCI. Similarly, each paper is described by a 3703-dim 0/1-valued word vector. This citation network consists of 9,104 links.
- Amazon denotes a Amazon product network, where nodes represent goods and edges represent that two goods are frequently bought together. Given product reviews as 745-dim bag-of-words node features, the classification task is to map goods to their respective product category. This network contains 7,650 nodes, 238,162 edges and 8 classes.
- Actor refers to an actor network, where each node corresponds to an actor, and the edge between two nodes denotes co-occurrence on the same Wikipedia page. The 932-dim node features correspond to some keywords in the Wikipedia pages. The task is to classify the nodes into categories in term of words of actor's Wikipedia. This network contains 7,600 nodes, 30, 019 edges and 5 classes.

6.2 Analyzed Methods and Metrics

We implement and test four representative embedding-based pooling models with their instance-based version. They include: (1) two most straightforward and commonly used methods, `sum`-pool and `mean`-pool; (2) two attention models, `GlobalAttention` [18] and `Set2Set` [26]. To enhance the reproducibility of the experimental results, we rely on the open source implementation of all models provided by PyTorch Geometric Library [7]. For a fair comparison, all models use GraphSAGE [10] as the node feature extractor `MP`, and a 1-layer MLP as the classifier `CLS`. For `sum`-pool, `ATT` is equivalent to the identity function. All the analyzed methods are evaluated by the classification accuracy and AUC.

6.3 Bag Label Generation

We design the following process to synthesize bags and their label, which is consistent with the standard MIL assumption (please refer to Eq. (1)). Let bag $\mathcal{X} = \{v_1, v_2, ..., v_m\}$ with label $\mathcal{Y}$ as defined in Sec. 2, the following steps are conducted: (1) a trigger class t is randomly selected based on the uniform distribution; (2) m nodes are randomly sampled to form a bag $\mathcal{X}$. Then, $\mathcal{Y} = 1$ if $\mathcal{X}$ contains an instance belonging to t, and $\mathcal{Y} = -1$ otherwise. Step 2 is repeated until the number of bags reaches the set value. Note that when t represents a minority class, the classification task becomes more challenging due to the imbalanced class distribution, which is common in real-world scenarios [5].

6.4 Key Experimental Configurations

We set the number and size of bags to 500 and 2, respectively. Then, we randomly split the bags into a 60%/20%/20% split for training, validation, and testing. Note that bag labels are used during training, while node labels are used for hyperparameter tuning and testing. Hyperparameters of `MP` (learning rate, weight decay, number of hidden units, and dropout rate) are tuned on the validation set. All models are trained using the Adam optimizer [14] until convergence, with the maximum training epoch being 500. A server equipped with an NVIDIA RTX 6000 Ada GPU is used to conduct the experiments in this study.

7 Experimental Results

7.1 Performance of Analyzed Methods in Node Prediction

Tab. 1 shows the comparison between representative embedding-based methods and their instance-based version in instance prediction. It can be observed that, out of a total of 32 comparisons, instance-based models outperform in 19 cases, underperform in 10 cases, and perform equally with embedding-based models in 3 cases. Thus, instance-based models exhibit a clear advantage. The experiments indeed suggest that when training node classifiers with set labels, adopting the instance-based pooling is more likely to be a better practice.

Table 1. Performance of all analyzed methods in terms of instance-level MIL on graphs. "`-emb`" and "`-ins`" represent embedding-based and instance-based models, respectively. In the comparison, results where instance-based models perform better are underlined in blue (positive results), those where embedding-based models perform better are underlined in red (negative results), and equivalent results are underlined in gray.

Datasets	Cora-ML		CiteSeer		Amazon		Actor	
Metrics	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
`sum-emb`	0.963	0.953	0.874	0.877	0.888	0.520	0.602	0.690
`sum-ins`	0.947	0.961	0.901	0.908	0.888	0.693	0.689	0.683
`mean-emb`	0.926	0.765	0.927	0.742	0.909	0.510	0.675	0.536
`mean-ins`	0.942	0.843	0.875	0.828	0.909	0.634	0.634	0.532
`GlobalAttention-emb`	0.935	0.881	0.891	0.882	0.948	0.968	0.684	0.610
`GlobalAttention-ins`	0.871	0.938	0.896	0.835	0.948	0.885	0.652	0.613
`Set2Set-emb`	0.828	0.501	0.868	0.951	0.751	0.515	0.878	0.526
`Set2Set-ins`	0.930	0.746	0.873	0.936	0.838	0.739	0.913	0.589

7.2 Pooling with `max` as the Aggregator

While most MIL models use `sum` as the aggregator, `max` that takes the feature-wise maximum across all instances is still regarded as an effective alternative in certain scenarios [23, 11]. With `max` as the aggregator, the embedding-based pooling takes the following form:

$$\theta(\mathcal{X}) = \texttt{CLS}(\texttt{max}(\{\alpha_i \cdot \texttt{MP}(v_i), v_i \in \mathcal{X}\})) \tag{21}$$

Similarly, the instance-based pooling is re-formulated as:

$$\theta(\mathcal{X}) = \texttt{max}(\{\texttt{CLS}(\alpha_i \cdot \texttt{MP}(v_i)), v_i \in \mathcal{X}\}) \tag{22}$$

We additionally conduct a comparative analysis of embedding-based pooling and instance-based pooling when `max` is used. Consistent with `sum`-pool, GraphSAGE and a 1-layer MLP are selected as `MP` and `CLS`, respectively. `ATT` is implemented as an identity function. The results are summarized in Tab. 2.

Table 2. Performance of pooling methods with `max` as the aggregator. Positive, negative and equivalent results are underlined in blue, red and gray, respectively.

Datasets	Cora-ML		CiteSeer		Amazon		Actor	
Metrics	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
`max-emb`	0.909	0.860	0.874	0.775	0.985	0.788	0.582	0.546
`max-ins`	0.936	0.842	0.905	0.899	0.738	0.905	0.663	0.546

It is encouraging to see from Tab. 2 that with `max` as the aggregator, instance-based pooling still exhibits as a better practice in most test scenarios (5/8). Embedding-based models outperform in only 25% of the comparisons. It is worth

noting that although instance-based pooling using `max` can still be proved to be permutation-invariant as long as it can be arbitrarily approximated by a Hausdorff continuous symmetric function [23], its WL-test-equivalence as well as superior interpretability over embedding-based models no longer hold.

7.3 Pooling under the Collective MIL Assumption

Our previous discussion focuses on the *standard* MIL assumption, which can be relaxed to the *collective* MIL assumption, where bags are not triggered by a single instance, but by an accumulation of multiple instances. More concretely, under the collective assumption, a bag $\mathcal{X} = \{v_1, v_2, ..., v_m\}$ is positive if and only if it contains more than $\gamma \cdot m$ positive instances. The label generation rule is formally presented as follows, where $\gamma \in [0, 1]$, $y_i \in \{0, 1\}$:

$$\mathcal{Y} = \begin{cases} +1 & if \quad \sum_{i=1}^{m} y_i \geq \gamma \cdot m \\ -1 & if \quad \sum_{i=1}^{m} y_i < \gamma \cdot m \end{cases} \tag{23}$$

To synthesize bags, we conduct the following sampling process: (1) a trigger class t is randomly selected as the positive label; (2) m nodes are arbitrarily sampled to form a bag $\mathcal{X}$. Then, $\mathcal{Y} = 1$ if $\mathcal{X}$ contains no less than $\gamma \cdot m$ instance belonging to t, and $\mathcal{Y} = -1$ otherwise. Step 3 is repeated until the number of bags reaches the predefined value. In our experiments, we set $m = 3$ and $\gamma = 0.5$. This means that a bag is labeled as positive only if it contains at least two trigger instances. To prevent the number of positive bags from being too small, we ensure that 20% of the bags include two trigger instances and one non-trigger instance. In this way, at least 20% of the bags are positive. The experimental results show that, under the collective MIL assumption, instance-based and embedding-based models perform comparably: each leads in 13 comparisons, while they tie in 6 cases. Taken together with the results in Tab. 1, Tab. 2, and Tab. 3, we consider the performance of instance-based models on node-level tasks to be reliable.

Table 3. Model performance under the *collective MIL assumption*. Positive, negative and equivalent results are underlined in blue, red and gray, respectively.

Datasets	Cora-ML		CiteSeer		Amazon		Actor	
Metrics	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
`sum-emb`	0.959	0.584	0.830	0.624	0.941	0.553	0.645	0.500
`sum-ins`	0.948	0.621	0.961	0.622	0.781	0.576	0.645	0.516
`mean-emb`	0.970	0.583	0.882	0.600	0.862	0.523	0.671	0.546
`mean-ins`	0.867	0.616	0.857	0.616	0.862	0.512	0.671	0.553
`GlobalAttention-emb`	0.910	0.606	0.844	0.588	0.828	0.515	0.675	0.565
`GlobalAttention-ins`	0.847	0.592	0.873	0.616	0.828	0.507	0.699	0.548
`Set2Set-emb`	0.885	0.592	0.816	0.621	0.803	0.511	0.745	0.503
`Set2Set-ins`	0.874	0.594	0.882	0.609	0.803	0.503	0.755	0.503

7.4 Performance of Analyzed Methods in Bag Prediction

A pooling model that generalizes across different graph structures would bring significant benefits to graph learning tasks. In this section, we examine whether instance-based models can retain their performance advantage in bag-level tasks. To achieve this, we test the performance of all methods on bag-level MIL tasks (see Tab. 4). Following the synthetic method described in Sec. 6.3, we construct 1,500 bags, with 300, 200, and 1,000 bags used for training, validation, and testing, respectively. Following [6], the bag size is increased to be 10.

For bag-level classification tasks, embedding-based models deliver superior performance in the majority of comparisons (18 out of 32). Specifically, their advantage is more pronounced in terms of accuracy. This observation contrasts with the results presented in the preceding sections, and implies that instance-based pooling and embedding-based pooling excel at capturing local and global features, respectively. Therefore, for tasks such as graph classification, our experiments suggest continuing to use classical embedding-based models, unless there are additional requirements for model interpretability. Developing a pooling model that is applicable to both graph-level and node-level learning tasks would be an interesting and challenging direction for future work.

Table 4. Performance of all analyzed methods on the bag prediction task. Positive, negative and equivalent results are underlined in blue, red and gray, respectively.

Datasets	Cora-ML		CiteSeer		Amazon		Actor	
Metrics	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
`sum-emb`	0.736	0.772	0.878	0.702	0.961	0.924	0.676	0.501
`sum-ins`	0.682	0.669	0.878	0.635	0.681	0.893	0.676	0.535
`mean-emb`	0.809	0.717	0.901	0.670	0.896	0.819	0.851	0.499
`mean-ins`	0.806	0.677	0.895	0.681	0.623	0.860	0.851	0.514
`GlobalAttention-emb`	0.834	0.818	0.787	0.860	0.644	0.500	0.917	0.499
`GlobalAttention-ins`	0.788	0.763	0.737	0.783	0.644	0.500	0.917	0.501
`Set2Set-emb`	0.859	0.895	0.845	0.782	0.642	0.502	0.945	0.488
`Set2Set-ins`	0.791	0.850	0.837	0.689	0.642	0.500	0.945	0.496

8 Discussions

In this section, we provide additional discussions as follows:

- For node-level tasks, if training with bag labels can achieve expected performance, it would substantially reduce the effort for node label annotations. Moreover, since the number of bag labels is typically much smaller than that of node labels, this also leads to more efficient model training.
- This study has certain experimental limitations. For example, we have not evaluated the models on a wider range of benchmark datasets or with more

embedding-based models. In addition, we did not examine how model performance varies with bag size. For MIL tasks based on the standard assumption, classification typically becomes more difficult as the bag size increases, especially in the case of synthetic data. Finally, compared to other paradigms for describing the graph pooling process (e.g., the SRC model [9]), the generalizability of embedding-based models is limited.
– Large Language Models (LLMs) can assist MIL on graphs in several key ways. First, LLMs can generate high-quality synthetic labels or bag-level descriptions to augment training data under weak supervision. Second, LLMs may facilitate the interpretation of MIL decisions by generating natural language explanations for instance-level contributions. Finally, their ability to generalize across modalities allows for flexible integration of multimodal metadata, aiding cross-domain graph representation learning.

9 Related Works

This work explores a novel task in Multiple Instance Learning (MIL), a type of weakly supervised learning problem that originated in the 1990s [13, 4]. Instead of receiving a set of instances that are individually labeled, the learning model receives a set of labeled bags, each containing many instances. Due to the diverse characteristics of tasks, MIL can be classified into four categories based on the following attributes: the composition of bags, the types of data distribution, the ambiguity of instance labels, and the task to be performed, each presenting different challenges [3]. MIL typically involves two levels of tasks: bag-level and instance-level, and we tackle the latter. MIL algorithms are often only effective for one of them [11, 3]. Given the prevalence of weakly supervised inexact data in real-world scenarios, MIL finds numerous applications in diverse domains (e.g., [17]). Since MIL is a long-established field that covers a diverse range of topics, we refer the readers to surveys (e.g., [3]) for more information.

On the other hand, this work is closely related to graph pooling from the methodological perspective. Graph pooling models condense a graph into a smaller-sized graph or a single vector, thus working as an essential component for graph-level tasks that require holistic graph-level representations, such as graph classification and graph generation. Generally, designs of graph pooling could be roughly divided into flat pooling (e.g., [18, 26]) and hierarchical pooling, while the latter can be further categorized into node cluster pooling (e.g., [29]) and node drop pooling (e.g., [15]). Existing pooling methods have been demonstrated to be successful in capturing high-order information on a wide range of applications; however, designing effective pooling operators for node-level tasks are still key challenges [19]. Furthermore, graph pooling faces critical challenges including interpretability, robustness, efficiency, expressiveness and so on. Similarly, we refer the readers to [19, 9] for a more complete and detailed introduction on graph pooling techniques. Our work addresses the lack of designs of effective pooling model for node-level tasks, and provides a comprehensive discussion of the key properties of different pooling paradigms.

10 Conclusions

In this work, we introduce the node-level MIL task for the first time and investigate the role of graph pooling in this context. We propose a general approach to transform classical embedding-based pooling into an instance-based style, which is intrinsically more suitable for node-level tasks. Furthermore, we analyze key theoretical properties of both frameworks, including permutation invariance, expressiveness, and interpretability. Experimental evaluations on benchmark datasets, simulating various application scenarios, demonstrate the effectiveness of instance-based models. This work pioneers the extension of graph pooling beyond graph-level tasks to node-level tasks, providing both theoretical and empirical insights. As future work, we will explore the design of pooling mechanisms with stronger generalization capabilities and higher efficiency.

11 Acknowledgments

This paper is based on results obtained from the project, "Research and Development Project of the Enhanced infrastructures for Post-5G Information and Communication Systems" (JPNP20017), commissioned by the New Energy and Industrial Technology Development Organization (NEDO), and was supported by JSPS KAKENHI Grant Numbers JP25K21275, JP25K03231, JP23H03451.

References

1. Bianchi, F.M., Lachi, V.: The expressive power of pooling in graph neural networks. Advances in neural information processing systems **36**, 71603–71618 (2023)
2. Bojchevski, A., Günnemann, S.: Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. arXiv preprint arXiv:1707.03815 (2017)
3. Carbonneau, M.A., Cheplygina, V., Granger, E., Gagnon, G.: Multiple instance learning: A survey of problem characteristics and applications. Pattern Recognition **77**, 329–353 (2018)
4. Dietterich, T.G., Lathrop, R.H., Lozano-Pérez, T.: Solving the multiple instance problem with axis-parallel rectangles. Artificial intelligence **89**(1-2), 31–71 (1997)
5. Duan, Y., Liu, X., Jatowt, A., Yu, H.t., Lynden, S., Kim, K.S., Matono, A.: Dual cost-sensitive graph convolutional network. In: 2022 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2022)
6. Duan, Y., Liu, X., Jatowt, A., Yu, H.T., Lynden, S.J., Matono, A.: Inexact graph representation learning. In: IJCNN (2024)
7. Fey, M., Lenssen, J.E.: Fast graph representation learning with pytorch geometric. arXiv preprint arXiv:1903.02428 (2019)
8. Foulds, J., Frank, E.: A review of multi-instance learning assumptions. The knowledge engineering review **25**(1), 1–25 (2010)
9. Grattarola, D., Zambon, D., Bianchi, F.M., Alippi, C.: Understanding pooling in graph neural networks. IEEE transactions on neural networks and learning systems **35**(2), 2708–2718 (2022)
10. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in neural information processing systems **30** (2017)

11. Ilse, M., Tomczak, J., Welling, M.: Attention-based deep multiple instance learning. In: International conference on machine learning. pp. 2127–2136. PMLR (2018)
12. Javed, S.A., Juyal, D., Padigela, H., Taylor-Weiner, A., Yu, L., Prakash, A.: Additive mil: Intrinsically interpretable multiple instance learning for pathology. Advances in Neural Information Processing Systems **35**, 20689–20702 (2022)
13. Keeler, J., Rumelhart, D., Leow, W.: Integrated segmentation and recognition of hand-printed numerals. Advances in neural information processing systems **3** (1990)
14. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
15. Lee, J., Lee, I., Kang, J.: Self-attention graph pooling. In: International conference on machine learning. pp. 3734–3743. pmlr (2019)
16. Leman, A., Weisfeiler, B.: A reduction of a graph to a canonical form and an algebra arising during this reduction. Nauchno-Technicheskaya Informatsiya **2**(9), 12–16 (1968)
17. Li, W., Duan, L., Xu, D., Tsang, I.W.H.: Text-based image retrieval using progressive multi-instance learning. In: 2011 international conference on computer vision. pp. 2049–2055. IEEE (2011)
18. Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R.: Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493 (2015)
19. Liu, C., Zhan, Y., Wu, J., Li, C., Du, B., Hu, W., Liu, T., Tao, D.: Graph pooling for graph neural networks: Progress, challenges, and opportunities. arXiv preprint arXiv:2204.07321 (2022)
20. Lundberg, S.: A unified approach to interpreting model predictions. arXiv preprint arXiv:1705.07874 (2017)
21. MA, Y.T.: DEEP LEARNING ON GRAPHS. Cambridge University Press (2021)
22. Pei, H., Wei, B., Chang, K.C.C., Lei, Y., Yang, B.: Geom-gcn: Geometric graph convolutional networks. arXiv preprint arXiv:2002.05287 (2020)
23. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)
24. Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of graph neural network evaluation. arXiv preprint arXiv:1811.05868 (2018)
25. Sirinukunwattana, K., Raza, S.E.A., Tsang, Y.W., Snead, D.R., Cree, I.A., Rajpoot, N.M.: Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images. IEEE transactions on medical imaging **35**(5), 1196–1206 (2016)
26. Vinyals, O., Bengio, S., Kudlur, M.: Order matters: Sequence to sequence for sets. arXiv preprint arXiv:1511.06391 (2015)
27. Wang, X., Yan, Y., Tang, P., Bai, X., Liu, W.: Revisiting multiple instance neural networks. Pattern recognition **74**, 15–24 (2018)
28. Yang, Z., Cohen, W., Salakhudinov, R.: Revisiting semi-supervised learning with graph embeddings. In: International conference on machine learning. pp. 40–48. PMLR (2016)
29. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. NeurIPS **31** (2018)
30. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R.R., Smola, A.J.: Deep sets. Advances in neural information processing systems **30** (2017)
31. Zhang, B., Fan, C., Liu, S., Huang, K., Zhao, X., Huang, J., Liu, Z.: The expressive power of graph neural networks: A survey. IEEE Transactions on Knowledge and Data Engineering (2024)