

An Empirical Study of Retrieval-Enhanced Graph Neural Networks

Dingmin Wang¹, Shengchao Liu², Hanchen Wang³, Bernardo Cuenca Grau¹,
Linfeng Song⁴, Jian Tang², Le Song⁵, Qi Liu⁶

¹University of Oxford, ²Mila-HEC Montreal, ³University of Cambridge, ⁴Tencent AI Lab

⁵BioMap and MBZUAI, ⁶The University of Hong Kong

Abstract. Graph Neural Networks (GNNs) are effective tools for graph representation learning. Most GNNs rely on a recursive neighborhood aggregation scheme, named message passing, thereby their theoretical expressive power is limited to the first-order Weisfeiler-Lehman test (1-WL). An effective approach to this challenge is to explicitly retrieve some annotated examples used to enhance GNN models. While retrieval-enhanced models have been proved to be effective in many language and vision domains, it remains an open question how effective retrieval-enhanced GNNs are when applied to graph datasets. Motivated by this, we want to explore how the retrieval idea can help augment the useful information learned in the graph neural networks, and we design a retrieval-enhanced scheme called GRAPHRETRIEVAL, which is agnostic to the choice of graph neural network models. In GRAPHRETRIEVAL, for each input graph, similar graphs together with their ground-true labels are retrieved from an existing database. Thus they can act as a potential enhancement to complete various graph property predictive tasks. We conduct comprehensive experiments over 13 datasets, and we observe that GRAPHRETRIEVAL is able to reach substantial improvements over existing GNNs. Moreover, our empirical study also illustrates that retrieval enhancement is a promising remedy for alleviating the long-tailed label distribution problem.

1 Introduction

Graph neural networks (GNNs) are a class of neural architectures for supervised learning which has been adopted in a plethora of applications involving graph-structured data, such as molecular design [9, 25], drug discovery [2], recommendation systems [42, 8, 38, 37], and knowledge graph completion [30, 44].

To improve the representation power and performance of GNN models, existing studies mainly focus on the architecture design, and a growing plethora of new GNN models [20, 40, 4] have been proposed in the past few years. Despite that newly-designed GNN models have brought improvement over a wide range of graph-based predictive tasks, some open challenges remain to be unsolved. One such issue is the example forgetting a.k.a. catastrophic forgetting [33, 45, 24] since the training process does not ensure that the trained model will completely “remember” the training examples.

Motivated by the success of retrieval-enhanced models in language and vision domains [22, 11, 29], in this paper, we aim at exploring

the effects of retrieval-enhancement in the regime of GNNs. Our experimental results show that the performance of GNN models can be enhanced if the training data is also exploited at the testing time for making predictions. This is in contrast to the supervised learning setting, where the training data is typically discarded once training has been completed and the model relies solely on the learned parameters for making predictions with respect to the given input.

In this study, we focus on two common tasks in the realm of graph neural networks: graph classification and graph regression. Given training examples consisting of graph-label pairs (\mathcal{X}_i, l_i) , where nodes in each graph are annotated with numeric feature vectors, we aim to learn a model that predicts label l given an input \mathcal{X} . At training time, GRAPHRETRIEVAL consists of the following steps.

1. *Standard supervised training.* Initially, we train the GNN-based model of interest from examples in the usual way, and obtain a trained model \mathcal{M} as a result.
2. *Indexing.* We apply \mathcal{M} to each of the training examples $\{(\mathcal{X}_1, l_1), \dots, (\mathcal{X}_n, l_n)\}$ and obtain the corresponding representation vectors $h_{\mathcal{X}_1}, \dots, h_{\mathcal{X}_n}$. We then construct a single Maximum Inner Product Search (MIPS) index using FAISS [17], where the key is $h_{\mathcal{X}_i}$ and the value is the corresponding example (\mathcal{X}_i, l_i) .
3. *Self-attention based adapter.* The adapter is parameterized with two learnable matrices \mathbf{W}_1 and \mathbf{W}_2 and a bias vector \mathbf{b} , which can be optimised using the training data after the model \mathcal{M} and the index have been obtained. Note that \mathbf{W}_1 , \mathbf{W}_2 and \mathbf{b} are the only additional parameters required on top of the baseline GNN.

Given an input \mathcal{X} at testing time, we compute its label l according to the following steps.

1. *Standard GNN application.* We apply \mathcal{M} to \mathcal{X} to obtain its representation vector $h_{\mathcal{X}}$ and its corresponding initial label $l_{\mathcal{X}}$ in the usual way.
2. *Graph retrieval.* We rank the representation vectors in the index according to their L2 similarity to $h_{\mathcal{X}}$, and retrieve the top- k vectors $\{h_{\mathcal{X}^1}, \dots, h_{\mathcal{X}^k}\}$ with highest similarity score; we then retrieve the corresponding example graphs and labels $(\mathcal{X}^1, l^1), \dots, (\mathcal{X}^k, l^k)$.
3. *Self-attention.* Based on the learned adapter and \mathcal{M} , we compute the final output label l as a weighted sum of the initially predicted label $l_{\mathcal{X}}$ and the labels l^1, \dots, l^k of the retrieved training examples.

We have implemented our approach and used it to enhance three strong GNN backbone models, GCN [20], GIN [40], and PNA [4].

* Both Bernardo and Qi serve as the corresponding authors of this paper. Shengchao and Hanchen make equal contribution to this paper.

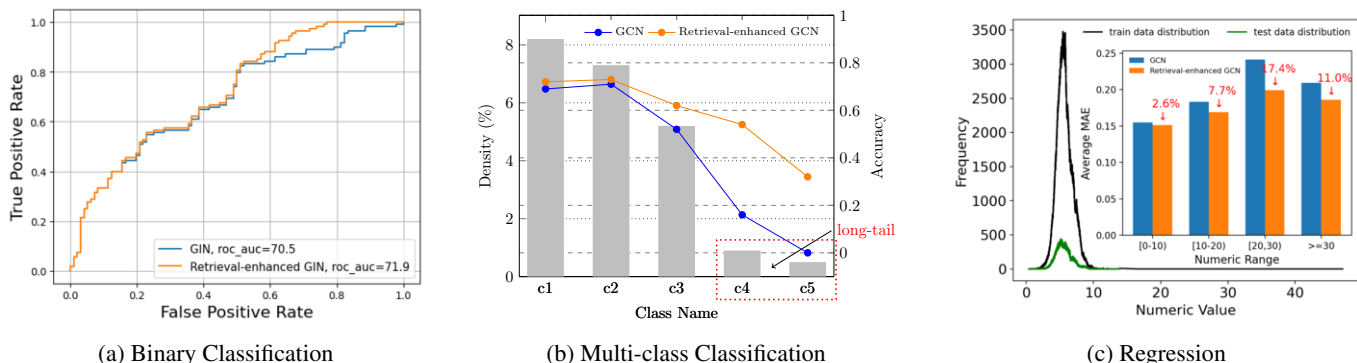


Figure 1: Retrieval-enhanced graph neural networks achieve consistent improvements over existing GNN models. (a) Experimental results of GIN w/ and w/o retrieval-enhancement on an OGB dataset (BBBP); (b) Part of experimental results of GIN w/ and w/o retrieval-enhancement on a reactant-product catalyst dataset with 888 common catalyst types (USPTO). The grey bar represents the percentage of per-class samples in a class-imbalanced training dataset and blue and yellow line charts denote the per-class accuracy in a class-balanced testing dataset (100 samples per class); (c) The black line and the green line represent the label distribution of the training and testing dataset of PCQM4M, respectively, and the bar chart represents the average MAE results of the testing datasets in each numeric range. The numeric ranges [10,20), [20,30) and ≥ 30 are considered as long-tailed numerical ranges, where we see a much larger improvement than that in [0,10).

The effectiveness and feasibility of GRAPHRETRIEVAL are evaluated on 13 benchmarks, including twelve one-instance graph datasets and one multi-instance graph dataset. For example, in Figure 1 (a), retrieval-enhanced GIN achieves a 2% ROC_AUC improvement on the BBBP dataset when compared with GIN without using the retrieved results; besides, we conduct experiments over two datasets with *long-tailed label distribution* [43, 28, 12], wherein many labels are associated with only a few samples. From Figure 1 (b) and (c), we can observe that our proposed approach brings consistent improvements over baselines in both the classification and the regression tasks. In particular, from the dissected experimental results, we can notice that the retrieval enhancement exhibits a significant improvement in the prediction of the long-tailed classes or values. Taking the bar chart in Figure 1(c) as an example, we can see only an average 2.6% MAE improvement in the non-long-tailed numerical range but a more than 17% improvement in the long-tailed number ranges. In summary, our contributions are as follows:

- With negligible computational costs¹, we introduce GRAPHRETRIEVAL, a simple yet effective framework for enhancing graph neural networks. GRAPHRETRIEVAL is a flexible framework that can be applied to various graph prediction tasks, while remaining agnostic to the choice of graph neural network models.
- We extensively evaluate GRAPHRETRIEVAL on thirteen datasets consisting of various graph sizes, ranging from small-scale single-instance molecular datasets to large-scale multi-instance therapeutics-related datasets. Our experimental results demonstrate that GRAPHRETRIEVAL outperforms prior baseline models.
- Furthermore, our comprehensive empirical study of retrieval-enhanced graph neural networks demonstrates that they offer a promising approach to improving performance on datasets with long-tailed label distributions.

¹ On one hand, the computational cost of retrieving similar graphs using FAISS [17] is negligible due to the tool’s well-proven efficiency and scalability. On the other hand, calculating the attention weights for the input graph and retrieved graphs in the adapter module only amounts to a single layer matrix operation. Therefore, the overall computational costs incurred by GRAPHRETRIEVAL are trivial.

2 Related Work

In this section, we briefly discuss recent GNN architectures and retrieval-augmented models in Deep Learning.

2.1 Graph Neural Networks

Graph Neural Networks have become the de-facto standard for Machine Learning tasks over graph-structured data. The fundamental idea behind GNNs is that of *Message Passing*—that is, to iteratively update each node-level representation by aggregating information from its neighborhoods throughout a fixed number of layers [20, 10, 40, 1].

In molecule classification applications, Weave [18] explicitly learns both the atom- and bond-level representation during message passing; D-MPNN [41] emphasizes the information delivered along different directions; more recently, N-Gram Graph [26] and AWARE [5] focus on modeling the walk-level information on molecules. The choice of the aggregation function is another key aspect of GNN design. GAT [35] adds an attention module; GGNN [23] combines information from its neighbors and its own representation with a GRU unit [3]; finally, PNA [4] adopts multiple aggregation functions for combining messages.

2.2 Retrieval-augmented Models

Prior studies in different domains show that models depending only on input features and learned parameters are less powerful than models augmented by external knowledge [22, 11, 36, 21, 39]. The authors of [39] propose the use of a kNN-augmented attention layer for language modeling tasks. In [21], the authors advocate for the use of the entire dataset (rather than using just a single datapoint) for making predictions; in particular, they show that there exist meaningful dependencies between the input and the training dataset which are lost during training. Other related works such as retREALM [11] and KSTER [16] augment the Transformer model by retrieving relevant contexts in a non-parametric way.

In contrast to prior work, however, GRAPHRETRIEVAL focuses on the retrieval of graph-structured data. Retrieving similar graphs can be much more challenging than retrieving similar text, where

pre-trained models like BERT [6] can provide universal representations for measuring similarities. Such pre-trained models are missing for diverse graph-structured data. Furthermore, graph retrieval is also negatively affected by over-smoothing and over-squashing in existing GNNs, which are two unsolved and challenging issues inherent to current GNN models that might affect the learning of graph embeddings, which in turn may negatively impact the quality of retrieval. Given that our framework allows for off-the-shelf use of *flexible* GNN as the base model, so at the current stage, these issues are perpendicular to our proposed retrieval algorithm. However, we believe that our framework could achieve more gains in improving the quality of the predictions if GNNs without over-smoothing and over-squashing issues appear in the future.

3 Background

In this section, we recapitulate the basics of GNNs and formulate the graph classification and regression tasks accordingly.

3.1 Message-passing GNNs

In the context of GNNs, we define a graph \mathcal{G} as a tuple

$$\mathcal{G} := (\mathcal{V}, \mathcal{E}, \{\mathbf{x}_v\}_{v \in \mathcal{V}}, \{\mathbf{x}_e\}_{e \in \mathcal{E}}),$$

where \mathcal{V} is a set of nodes with each node $v \in \mathcal{V}$ annotated with a numeric feature vector \mathbf{x}_v , and \mathcal{E} is a set of (undirected) edges where each edge $e \in \mathcal{E}$ is also annotated with a feature vector \mathbf{x}_e . For instance, in a molecule classification scenario, nodes may represent atoms, edges may represent bonds between molecules, and feature vectors encode information about atoms and bond types.

When given a graph \mathcal{G} as input, a message-passing GNN [10] with T layers updates each node feature vector throughout by aggregating messages from its neighbors. Formally, for each layer $1 \leq t \leq T$ and each node $v \in \mathcal{V}$, the feature vector $\mathbf{h}_v^{(t)}$ for node v in layer t is computed as follows:

$$\mathbf{m}_v^{(t)} = \sum_{\{u, v\} \in \mathcal{E}} M_{t-1}(\mathbf{h}_v^{(t-1)}, \mathbf{h}_u^{(t-1)}, x_{\{u, v\}}), \quad (1)$$

$$\mathbf{h}_v^{(t)} = U_t(\mathbf{h}_v^{(t-1)}, \mathbf{m}_v^{(t)}), \quad (2)$$

where $\mathbf{h}_v^{(0)} = \mathbf{x}_v$ for each $v \in \mathcal{V}$, and where M_t and U_t represent the GNN's message passing and update functions in the t -th layer, respectively. The concrete description of these functions is given in terms of the learnable parameters of the model. Finally, a ReadOut function is applied to the node feature vectors in the outermost layer $t = T$ to obtain a graph-level representation vector:

$$\mathbf{h}_{\mathcal{G}} = \text{AGG}\left(\left\{\mathbf{h}_v^{(T)} \mid v \in \mathcal{G}\right\}\right), \quad (3)$$

where **AGG** can be any permutation-invariant operation on multi-sets applied to a vector in a component-wise manner, such as average, summation, or attention-based aggregation.

3.2 Graph Property Prediction

Given an input \mathcal{X} , there are usually two tasks of interest:

1. *Single-instance prediction*, where \mathcal{X} is a single graph and the target is to classify \mathcal{X} into one of predefined classes or map \mathcal{X} to a numeric value.
2. *Multi-instance prediction*, where \mathcal{X} usually consists of two or more graphs and the target is also mapping \mathcal{X} to one of predefined classes or a numeric value.

For both *Single-instance prediction* and *Multi-instance prediction*, a basic structure could be formulated as follows:

$$\mathbf{h}_{\mathcal{X}} = \bigoplus_{\mathcal{G} \in \mathcal{X}} \text{GNN}(\mathcal{G}) \quad (4)$$

$$l_{\mathcal{X}} = \Phi(\mathbf{h}_{\mathcal{X}}), \quad (5)$$

where \bigoplus is a concatenation operation, $\mathbf{h}_{\mathcal{X}}$ denotes the representation of the input \mathcal{X} , $l_{\mathcal{X}}$ denotes a list of predicted probability values for predefined classes in the classification or a numeric value in the regression task, GNN represents any kind of graph neural networks, $\text{GNN}(\mathcal{G})$ outputs the graph-level representation for \mathcal{G} , and $\Phi(\cdot)$ represents a set of task-specific layers, e.g., a binary classification task corresponds to a dense layer followed by a sigmoid function.

4 Methodology

In this section, we describe the details of GRAPHRETRIEVAL. In what follows, we fix an arbitrary set of training examples $\{(\mathcal{X}_1, l_1), \dots, (\mathcal{X}_n, l_n)\}$ consisting of pairs (\mathcal{X}_i, l_i) of a graph \mathcal{X}_i and a corresponding prediction l_i (i.e., a class label in the case of classification, or a number in the case of regression). Furthermore, we fix a GNN given by a set of message-passing, update, and read-out functions. To avoid confusion, we use the superscript to denote retrieved examples.

4.1 Initial Training and Indexing

GNN Training The first step in our approach is to train the GNN in a standard supervised way using the given examples. As a result, we obtain a trained GNN model \mathcal{M} with concrete values for each of the parameters of the model; these parameter values will remain fixed in all subsequent steps.

Indexing of Training Examples After training, we apply \mathcal{M} to each of the example graphs $\mathcal{X}_1, \dots, \mathcal{X}_n$ and obtain the corresponding graph-level representation vectors $\mathbf{h}_{\mathcal{X}_1}, \dots, \mathbf{h}_{\mathcal{X}_n}$ as described in Equation (3). We then input key-value pairs $[\mathbf{h}_{\mathcal{X}_i}, (\mathcal{X}_i, l_i)]$ for each $1 \leq i \leq n$ to the FAISS retrieval engine [17] to construct an index from these key-value pairs. Using a FAISS index will significantly facilitate the retrieval of similar graphs at testing time. It is worth emphasizing that the index only needs to be constructed once throughout our entire pipeline.

4.2 Training Self-attention Based Adapter

Given an input \mathcal{X} , we compute the corresponding prediction l according to the following steps: First, we apply the trained GNN \mathcal{M} to \mathcal{X} to obtain, in the usual way, an initial prediction $l_{\mathcal{X}}$ and graph-level representation $\mathbf{h}_{\mathcal{X}}$. The main novelty in our approach, however, lies in the following two steps, where we first retrieve a set of most similar graphs from the index and subsequently exploit self-attention to compute the final prediction l .

Graph Retrieval. In this step, we use the FAISS index to find the top- k vectors $\{\mathbf{h}_{\mathcal{X}^1}, \dots, \mathbf{h}_{\mathcal{X}^k}\}$ that are most similar to $\mathbf{h}_{\mathcal{X}}$, and to subsequently retrieve the corresponding example graphs and labels

$(\mathcal{X}^1, l^1), \dots, (\mathcal{X}^k, l^k)$. The value k is considered a hyper-parameter, and we use L2 distance to compute vector similarity scores.²

Retrieval Dropout. Since the index is built from the training dataset, the most similar graph for each input graph is always itself during the training stage. Inspired by [16] and to reduce overfitting, we also adopt the *retrieval dropout* strategy. Specifically, during the training stage, we will retrieve $k + 1$ graphs, in which the most similar graph will be dropped. In the evaluation stage, *retrieval dropout* is disabled, as evaluation graphs are usually not part of the training data.

Self-attention Based Adapter We exploit self-attention to compute the final prediction l for input graph \mathcal{X} as a weighted sum of the initially predicted label $l_{\mathcal{X}}$ and the top- k example labels l_{i_1}, \dots, l_{i_k} with respective weights w_0, \dots, w_k .

The self-attention mapping takes as input a *query vector* and a set of key-value pairs and yields the weights w_0, \dots, w_k . In our setting, the query vector \mathbf{q} is obtained by multiplying a learnable matrix \mathbf{W}_1 to the representation $h_{\mathcal{X}}$ for the input \mathcal{X} :

$$\mathbf{q} = \mathbf{W}_1 \mathbf{h}_{\mathcal{X}}, \quad (6)$$

In turn, to obtain the keys, we pack the $k + 1$ graph representations $\{\mathbf{h}_{\mathcal{X}}, \mathbf{h}_{\mathcal{X}^{(1)}}, \dots, \mathbf{h}_{\mathcal{X}^{(k)}}\}$ as row vectors into a matrix \mathbf{H} , and then multiply \mathbf{H} with a learnable matrix \mathbf{W}_2 to obtain matrix

$$\mathbf{K} = \mathbf{W}_2 \mathbf{H}, \quad (7)$$

Finally, the attention weights $\mathbf{Attn} = (w_0, \dots, w_k)$ are obtained by computing the scaled product of \mathbf{q} with the transpose of \mathbf{K} and adding a $(k + 1)$ learnable bias vector ϕ (refer to Figure 2 for the visual illustration)

$$\mathbf{Attn} = \text{softmax} \left(\frac{\mathbf{q} \mathbf{K}^T}{\sqrt{d}} + \phi \right), \quad (8)$$

where d is the dimension of the query vector \mathbf{q} . Intuitively, the bias vector ϕ allows the model to encode the ranking between the different retrieved graphs; without it, the model cannot distinguish between the different retrieved graphs and the ranking information is lost as a result.

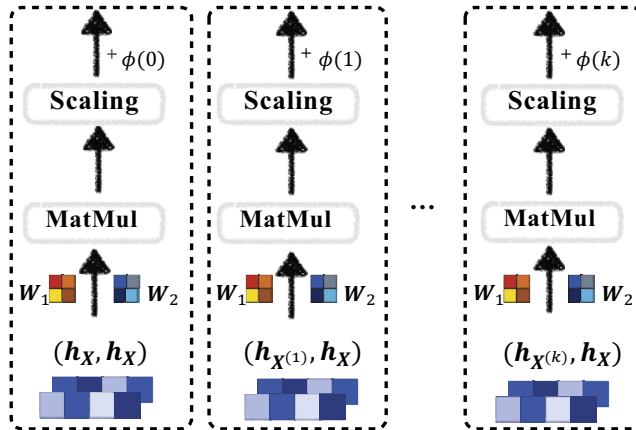


Figure 2: Scheme on Scaled Dot Product Attention.

² Recall that, given d -dimensional vectors v^1 and v^2 , the L2 distance between v^1 and v^2 is given by $\sqrt{\sum_{i=1}^d (v_i^1 - v_i^2)^2}$

Loss Function Given an input \mathcal{X} , the ground-truth label c , $l_{\mathcal{X}}$ and the top- k example labels $\{l^1, \dots, l^k\}$, the loss function for the classification task is written as:

$$\mathcal{L} = -\log \left(\mathbf{Attn}[0] \times l_{\mathcal{X}}[c] + \sum_{l^i=c} \mathbf{Attn}[i] \right) \quad (9)$$

and the loss function for the regression task is written as:

$$\mathcal{L} = \left(\sum \mathbf{Attn} \odot \mathbf{z} - c \right)^2 \quad (10)$$

where $\mathbf{z} = [l_{\mathcal{X}}, l^1, \dots, l^k]$ and \odot is the element-wise product.

For testing. Given an input \mathcal{X} , $l_{\mathcal{X}}$ and the top- k example labels $\{l^1, \dots, l^k\}$, the final predicted result for the classification task is: $\text{argmax}(L_{\mathcal{X}})$, where $L_{\mathcal{X}} \in \mathbb{R}^{k+1}$ is obtained by adjusting the logits in $l_{\mathcal{X}}$ – that is, for each $c \in [1, \dots, k]$,

$$L_{\mathcal{X}}[c] = \mathbf{Attn}[0] \times l_{\mathcal{X}}[c] + \sum_{l^i=c} \mathbf{Attn}[i] \quad (11)$$

and the final predicted result for the regression task is: $\sum \mathbf{Attn} \odot \mathbf{z}$, where $\mathbf{z} = [l_{\mathcal{X}}, l^1, \dots, l^k]$ and \odot is the element-wise (Hadamard) product.

4.3 Two-phase Model Training

Algorithm 1 Pipeline of Two-phase Model Training

Require: Training and validation datasets; randomly initialised GNN and Adapter; training epochs m_1 for GNN and m_2 for the adapter; the number of retrieved graphs k .

- 1: **for** $i = 1$ **to** m_1 **do**
- 2: Train the GNN model over the training dataset. Optimal GNN parameters of a trained model \mathcal{M} are saved by evaluating the validation dataset.
- 3: **end for**
- 4: Construct FAISS index.
- 5: **for** $i = 1$ **to** m_2 **do**
- 6: Train the Adapter over the train set using the index and the fixed model \mathcal{M} , where the *retrieval dropout* strategy is applied in this stage. The optimal parameters of the Adapter are saved by evaluating the validation dataset.
- 7: **end for**
- 8: **return** \mathcal{M} and Adapter defined by $\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}$.

The overall training pipeline is illustrated in Algorithm 1. Lines 1–4 correspond to the initial training of the GNN model and the construction of the index. In Lines 5–7 we train the adapter module and optimise the values in the learnable matrices \mathbf{W}_1 and \mathbf{W}_2 and the bias vector \mathbf{b} described in Section 4.2, where the GNN model \mathcal{M} is fixed. Finally, when learning \mathbf{W}_1 and \mathbf{W}_2 from the training data using our entire pipeline, it is worth noticing that the most similar graph of an example graph \mathcal{X} will be itself, so we will always drop it and consider only the other retrieved graphs.

5 Experiments

In this section, we report empirical evaluations of GRAPHRETRIEVAL for 13 graph property prediction datasets. We show that retrieval-enhanced models yield excellent performance gains when compared with baseline GNN models³.

³ Our codes are available in <https://github.com/wdimmy/GraphRetrieval>.

5.1 Benchmark Dataset

We used a collection of 13 datasets consisting of 8 small-scale molecule datasets [14], 2 computer vision datasets [7], 1 large-scale therapeutics-related dataset (USPTO [27]) and 2 large-scale quantum chemistry datasets (PCQM4M and PCQM4Mv2 [13]). Detailed information on these Benchmark datasets is summarized in Table 1.

Table 1: Statistics of datasets and their evaluation metrics. MC and BC denote multi-class and binary classification, respectively. For PCQM4M and PCQM4Mv2, their testing datasets are not publicly available. We report the best result on the development sets.

| Dataset | Train | Valid | Test | Task | Metric |
|----------|-----------|---------|---------|------------|----------|
| BBBP | 1631 | 204 | 204 | BC | ROC_AUC |
| Tox21 | 6264 | 783 | 784 | BC | ROC_AUC |
| ToxCast | 6860 | 858 | 858 | BC | ROC_AUC |
| SIDER | 1141 | 143 | 143 | BC | ROC_AUC |
| ClinTox | 1181 | 148 | 148 | BC | ROC_AUC |
| MUV | 74469 | 9309 | 9309 | BC | ROC_AUC |
| HIV | 32901 | 4113 | 4113 | BC | ROC_AUC |
| BACE | 1210 | 151 | 152 | BC | ROC_AUC |
| PCQM4M | 3,045,360 | 380,670 | – | Regression | MAE |
| PCQM4Mv2 | 3,378,606 | 73,545 | – | Regression | MAE |
| MNIST | 55000 | 5000 | 10000 | MC | Accuracy |
| CIFAR10 | 45000 | 5000 | 10000 | MC | Accuracy |
| USPTO | 505,259 | 72,180 | 144,360 | MC | Accuracy |

Table 2: Experimental results of different methods on the four different testing datasets. The number in the bracket of *Majority-voting* method represents the retrieved number.

| | CIFAR10 | MNIST | HIV | USPTO |
|----------------------|-------------|-------------|-------------|-------------|
| GCN | 64.7 | 96.0 | 78.6 | 33.5 |
| Retrieval | 49.1 | 85.3 | 54.1 | 24.3 |
| Majority-voting (5) | 51.7 | 87.1 | 57.3 | 25.7 |
| Majority-voting (10) | 51.9 | 88.9 | 58.1 | 26.9 |
| Majority-voting (20) | 48.7 | 84.2 | 56.3 | 25.2 |
| GIN | 65.6 | 96.8 | 77.0 | 38.6 |
| Retrieval | 51.1 | 86.2 | 55.8 | 26.5 |
| Majority-voting (5) | 51.1 | 87.9 | 56.9 | 27.7 |
| Majority-voting (10) | 52.9 | 89.4 | 58.9 | 28.2 |
| Majority-voting (20) | 52.7 | 88.6 | 57.3 | 26.3 |
| PNA | 68.7 | 97.2 | 77.0 | 35.2 |
| Retrieval | 51.8 | 86.2 | 56.1 | 26.8 |
| Majority-voting (5) | 52.1 | 87.9 | 57.6 | 27.4 |
| Majority-voting (10) | 52.9 | 88.9 | 58.2 | 27.4 |
| Majority-voting (20) | 52.2 | 87.4 | 57.8 | 26.4 |

5.2 Baseline Models

We adopt three widely-used GNN baselines: Graph Convolutional Networks (GCN) [20], Graph Isomorphism Networks (GIN) [40], and Principal Neighbourhood Aggregation (PNA) [4]. For these baselines, we take the default values of the hyperparameters specified in the publicly-released GitHub repositories. Besides, we adopt two purely retrieval-based approaches:

- *Retrieval*, which retrieves the most similar graph from the FAISS index whose label is as our predicted answer. Note that when eval-

uating using the ROC-AUC metric, we instead use the corresponding similarity score returned by FAISS.

- *Majority-voting*, which retrieves some number of the most similar graphs from the FAISS index whose labels will be ranked based on their frequency, and then we choose the most frequent label as our predicted answer. Note that when evaluating using the ROC-AUC metric, we need to use the similarity score instead of the predicted label. Unlike the *Retrieval* strategy, in the *Majority-voting* strategy, the most frequent label may correspond to multiple instances, leading to multiple similarity scores. In such cases, we select the highest similarity score among them.

Table 3: Experimental results of Self-attention VS Averaging.

| | | CIFAR10 | MNIST | HIV | USPTO |
|-----|----------------|-------------|-------------|-------------|-------------|
| GCN | Self-attention | 67.6 | 97.2 | 80.8 | 39.2 |
| | Averaging | 57.7 | 92.1 | 60.7 | 30.6 |
| GIN | Self-attention | 68.2 | 97.9 | 79.5 | 42.8 |
| | Averaging | 58.6 | 93.2 | 59.3 | 33.1 |
| PNA | Self-attention | 72.3 | 98.2 | 78.0 | 41.6 |
| | Averaging | 58.7 | 92.9 | 58.2 | 32.8 |

5.3 Training Details

GRAPHRETRIEVAL is an agnostic to the choice of graph neural networks, and can be applied to enhance a wide range of GNN models. Training is conducted as described in Algorithm 1, where $m_1 = 300$, $m_2 = 200$ and $k = 3^4$. We use the Adam optimizer [19] with an initial learning rate 0.01 and decay the learning rate by 0.5 every 50 epochs for all the GNN baselines. Batch normalization [15] is applied on every hidden layer. Some hyperparameters for the three GNN baseline models are shown in Phase 2 of the training (Lines 5–7) is repeated 5 times with different initialisation seeds for \mathbf{W}_1 , \mathbf{W}_2 , and \mathbf{b} ; we report the mean and standard deviation of the values obtained in these runs.

5.4 Exploratory Experiments

Predict with only graph retrieval. We compare *Retrieval* and *Majority-voting* methods against well trained graph neural network models on four datasets: CIFAR10, MNIST, PCQM4M and PCQM4Mv2. In particular, for the Majority-voting method, we try three different retrieved numbers to see their performance on four different datasets. The experimental results are shown in Table 2.

From Table 2, we can see that both *Retrieval* and *Majority-voting* method greatly underperform the three well-trained models. For the four datasets with different baseline GNN models, *Retrieval* methods have about 10% \sim 15% accuracy loss when compared with the well-trained GNN model. For the Majority-voting method, we can notice that as we increase the number of retrieved graphs from 10 to 20, the accuracy decreases on all four datasets. One straightforward reason might be that as the number of retrieved results increases, the number of noisy labels will increase accordingly.

⁴ Different numbers of retrieved examples might have different effects for our GRAPHRETRIEVAL framework, and further improvements might be achieved by choosing the number of retrieved examples in a more fine-grained way (e.g., differ across different datasets). In this paper, we choose $k = 3$ according to our preliminary experiments and we leave further exploration of the effects of the number of retrieved examples for future work.

Table 4: Test ROC-AUC (%) performance on 8 molecular prediction Benchmark datasets.

| | BBBP | Tox21 | ToxCast | SIDER | ClinTox | MUV | HIV | BACE | Average |
|------------------------|------------------|------------------|------------------|-------------------|------------------|------------------|------------------|------------------|-------------|
| GCN | 70.0(\pm 0.3) | 73.9(\pm 1.8) | 64.0(\pm 2.3) | 58.6(\pm 2.1) | 92.2(\pm 2.7) | 75.9(\pm 2.0) | 78.6(\pm 0.3) | 81.9(\pm 0.9) | 74.4 |
| GIN | 70.5(\pm 0.7) | 74.9(\pm 0.2) | 64.5(\pm 1.0) | 58.7(\pm 2.1) | 87.0(\pm 3.0) | 76.8(\pm 1.8) | 77.0(\pm 0.2) | 74.8(\pm 0.7) | 73.0 |
| PNA | 70.0(\pm 1.9) | 73.0(\pm 0.3) | 62.0(\pm 1.9) | 58.0(\pm 0.2) | 87.0(\pm 2.3) | 71.0(\pm 3.5) | 77.0(\pm 0.6) | 72.0(\pm 0.5) | 71.2 |
| Retrieval-enhanced GCN | 71.0(\pm 1.2) | 75.9(\pm 1.0) | 65.4(\pm 0.2) | 60.8(\pm 0.3) | 93.2(\pm 2.5) | 77.7(\pm 0.8) | 80.8(\pm 1.0) | 84.1(\pm 1.8) | 76.1 |
| Retrieval-enhanced GIN | 71.9(\pm 2.5) | 77.3(\pm 0.2) | 67.4(\pm 1.6) | 61.5(\pm 1.03) | 89.2(\pm 2.6) | 78.4(\pm 1.7) | 79.5(\pm 0.9) | 81.2(\pm 0.7) | 75.8 |
| Retrieval-enhanced PNA | 71.5(\pm 1.4) | 75.4(\pm 1.8) | 63.1(\pm 0.3) | 59.8(\pm 0.4) | 89.0(\pm 2.0) | 72.9(\pm 1.0) | 78.0(\pm 0.9) | 79.9(\pm 1.9) | 73.7 |

Self-attention vs. Averaging. One intuition about using the self-attention is that we hope the *Self-attention Adapter* module can learn to weigh retrieved examples differently such that the more “important” retrieved results correspond to the higher weights. Here, to demonstrate the necessity and the effectiveness of using the self-attention mechanism in our framework, we compare a self-attention setting against an averaging setting. To be more specific, each label will be assigned an equal weight without distinguishing their degree of importance to our prediction based on Equation 8.

From Table 3, we can observe that incorporating the self-attention mechanism into our framework significantly outperforms using the averaging scheme. For example, for the USPTO dataset, using the self-attention mechanism achieves about 22% accuracy improvement over the averaging scheme. One possible reason is that the retrieved results are not always useful, so the model should not assign the same weight to the retrieved labels and the label obtained by the well-trained model in computing the final prediction. Therefore, adopting self-attention to model the relevance between the target graph and the retrieved graphs is considered to be an effective way to make use of the retrieved results while avoiding sabotaging the performance of the well-trained model.

Table 5: MAE Performance on PCQM4M and PCQM4Mv2.

| | PCQM4M | PCQM4Mv2 |
|------------------------|--------------|--------------|
| GCN | 0.171 | 0.139 |
| GIN | 0.157 | 0.123 |
| Retrieval-enhanced GCN | 0.160 | 0.132 |
| Retrieval-enhanced GIN | 0.142 | 0.115 |

Retrieval Dropout or Not? We investigated the effect of retrieval dropout in our proposed GRAPHRETRIEVAL framework. Our ablation studies on various datasets demonstrate that incorporating retrieval dropout consistently leads to improved performance compared to not using the strategy. In particular, we observed that the gain is less significant in the two image datasets compared to the molecular dataset (e.g., only a 0.05% decrease in accuracy for MNIST but a 0.4% performance drop in HIV datasets). One possible explanation is that the difference between training and inference in the image datasets is not as pronounced as in the molecular datasets. Overall, our experimental results suggest that this training strategy enhances the generalization ability of GRAPHRETRIEVAL.

5.5 Main Results

We can observe that the use of GRAPHRETRIEVAL significantly enhances the performance of each of the baselines, which demonstrates the effectiveness of our approach. We next discuss each of these results in further detail.

Table 6: Classification accuracy on CIFAR10, MNIST and USPTO.

| - | CIFAR10 | MNIST | USPTO |
|------------------------|-------------|-------------|-------------|
| GCN | 64.7 | 96.0 | 33.5 |
| GIN | 65.6 | 96.8 | 38.6 |
| PNA | 68.7 | 97.2 | 35.2 |
| Retrieval-enhanced GCN | 67.6 | 97.2 | 39.2 |
| Retrieval-enhanced GIN | 68.2 | 97.9 | 42.8 |
| Retrieval-enhanced PNA | 72.3 | 98.2 | 41.6 |

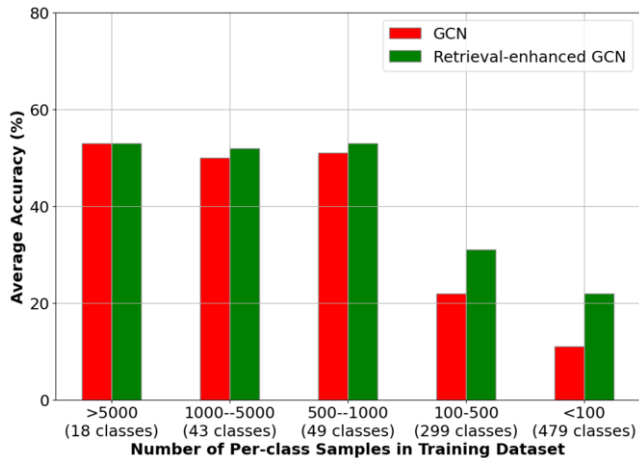


Figure 3: Dissected experiments on USPTO dataset. We classify the training datasets into five groups according to the number of associated samples of each class in the training dataset.

Molecule Datasets. These datasets are highly imbalanced since the label assigned to most samples is 0; The task is to predict the target molecular properties as accurately as possible, where the molecular properties are cast as binary labels, e.g, whether a molecule inhibits HIV virus replication or not. Our results are summarised in Table 4. First, we observe that GRAPHRETRIEVAL improves the classification performance of each of the baselines for all eight molecule datasets; furthermore, the improvements are consistent across all three baselines, which demonstrates that our approach is effective even on highly imbalanced datasets. Second, since these datasets use scaffold spitting, the substructures of the evaluation graphs are rarely reflected in the training graphs; it is therefore encouraging to observe that our approach is helpful even in such a challenging setting.

MNIST, CIFAR10 and USPTO. Our results on these datasets are summarised in Table 6. First, in contrast to the molecule datasets, where graphs are highly diverse and exhibit heterogeneous topologies, the two computer vision datasets (MNIST and CIFAR10) rely on grid-structured graphs with a fixed topology [4]. From Table 6, we can observe that these results are consistent with those obtained

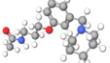
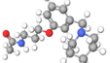
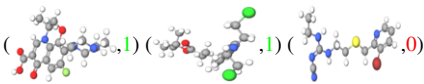
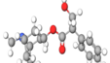
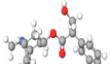
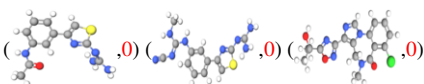
| Model | Input | Predicted Probability | Retrieval |
|------------------------|---|-----------------------|--|
| PNA |  (label=1) | $P_{label=1} = 0.32$ | No Retrieval |
| Retrieval-enhanced PNA |  (label=1) | $P_{label=1} = 0.81$ |  |
| PNA |  (label=1) | $P_{label=1} = 0.79$ | No Retrieval |
| Retrieval-enhanced PNA |  (label=1) | $P_{label=1} = 0.74$ |  |

Table 7: Two cherry-picked examples showing how GRAPHRETRIEVAL “absorbs” useful information from retrieved graphs and “ignores” noise introduced by irrelevant graphs, respectively. In the first example, it assigns a much higher probability (0.81) to the correct label, “1”, compared to PNA. In the second example when retrieved graphs are considered to be noisy information, both retrieval-enhanced PNA and PNA assign a similar probability (0.74 and 0.79, respectively) to the correct label.

for the molecule datasets, and hence our approach is also effective for computer vision problems. Besides, we experiment with another large-scale multi-classification dataset (USPTO). From Table 6, we can notice that the performance improvement on USPTO is much larger than the other two image datasets. One possible reason is that different from MNIST and CIFAR10, USPTO has more classes (888) and the distribution of instances for each class is long-tailed.

To gain more insights into how the retrieval-enhancement help improve the overall performance of baseline models, we trace the model’s performance in each class. We first classify 888 classes into five groups according to the number of samples appearing in the training datasets. Then, we report the accuracy of each class in the testing dataset. To avoid the class imbalance issue, we take a subset of the original testing, such that each class has the same number of samples needed to be predicted. From Figure 3, we observe that the improvement achieved by the retrieval enhancement in 100 – 500 and < 100 groups is much larger than the other three groups, showing that the retrieval enhancement is a promising remedy for increasing the accuracy of those “long-tailed” classes associated with only a few samples in the training dataset.

PCQM4M and PCQM4Mv2. These are two large-scale datasets consisting of millions of graphs. The task is graph regression: predicting the HOMO-LUMO energy gap in electronvolt given 2D molecular graphs. From Table 5, we found that GRAPHRETRIEVAL brings significant performance improvements on both datasets, which suggests that our approach is also beneficial for regression tasks. In particular, as shown in Figure 1(c), when the training dataset shows a long-tailed value distribution, we can see that GRAPHRETRIEVAL can significantly increase the performance for predicting those long-tailed when compared with baseline models. These results are consistent with the results in the classification task (USPTO) wherein many labels are associated with only a few samples, indirectly showing that our proposed approach exhibits substantial generality for improving the model’s performance in different tasks.

Overall, our results have demonstrated that GRAPHRETRIEVAL can bring substantial gains to GNN models. Such results are very promising because it first validates the feasibility of retrieval-enhanced approaches in the world of graph neural networks. Besides, another encouraging point is that the improvement does not come at the significant cost of increasing the model size as other models.

5.6 Case Study

To conclude this section, we cherry-pick two examples of Retrieval-enhanced PNA and PNA predicting the molecular property. In the first example, “1” is the correct label of the given input, and Retrieval-enhanced PNA gives the label=1 a much high probability compared to the PNA model. Since Retrieval-enhanced PNA manages to retrieve some similar graphs with possibly related labels, the probability solely calculated based on the input graph will be rectified. In the second example, the labels of the three retrieved graphs are all different from our ground-truth label, so they are considered to be noisy information. However, both Retrieval-enhanced PNA and PNA output a similar probability for the correct label, which denotes that Retrieval-enhanced PNA is resistant to noisy information. One possible reason is that our proposed GRAPHRETRIEVAL scheme will assign lower weights to those noisy labels by the self-attention mechanism. Overall, the case study shows that retrieved graphs can facilitate graph property prediction and the retrieval-enhanced approach is applied to graph neural networks.

6 Conclusion and Future Work

We have proposed GRAPHRETRIEVAL, a model-agnostic approach for improving the performance of GNN baselines on graph classification and regression tasks. Our empirical results clearly show that GRAPHRETRIEVAL consistently improves the performance of existing GNN architectures on a very diverse range of benchmarks. We believe that our results open the door for the further development of sophisticated methods for explicitly exploiting the training data when making predictions on graphs.

Our research opens a number of interesting avenues for future work. First, it would be interesting to extend our approach to tackle node-level classification tasks. Furthermore, our approach is rather general and flexible, and we believe that it can be applied to advanced methods based on 3D geometric modeling [31, 32] as well as to a broad range of additional tasks [10, 34].

Acknowledgments

This work was supported in whole or in part by the EPSRC projects OASIS (EP/S032347/1), ConCuR (EP/V050869/1) and UK FIRES (EP/S019111/1), the SIRIUS Centre for Scalable Data Access, and Samsung Research UK.

References

- [1] Muhammet Balcilar, Pierre Héroux, Benoit Gauzere, Pascal Vasseur, Sébastien Adam, and Paul Honeine, 'Breaking the limits of message passing graph neural networks', in *ICML*, (2021).
- [2] Mark Cheung and José MF Moura, 'Graph neural networks for covid-19 drug discovery', in *Big Data*, pp. 5646–5648. IEEE, (2020).
- [3] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio, 'On the properties of neural machine translation: Encoder-decoder approaches', in *CoRR*, (2014).
- [4] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Lio, and Petar Velickovic, 'Principal neighbourhood aggregation for graph nets', in *NeurIPS*, (2020).
- [5] Mehmet F Demirel, Shengchao Liu, Siddhant Garg, and Yingyu Liang, 'An analysis of attentive walk-aggregating graph neural networks', in *CoRR*, (2021).
- [6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, 'Bert: Pre-training of deep bidirectional transformers for language understanding', *CoRR*, (2018).
- [7] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson, 'Benchmarking graph neural networks', *arXiv preprint arXiv:2003.00982*, (2020).
- [8] Chen Gao, Xiang Wang, Xiangnan He, and Yong Li, 'Graph neural networks for recommender system', in *WSDM*, pp. 1623–1625, (2022).
- [9] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl, 'Neural message passing for quantum chemistry', in *ICML*, (2017).
- [10] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl, 'Neural message passing for quantum chemistry', in *ICML*, (2017).
- [11] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang, 'Retrieval augmented language model pre-training', in *ICML*, (2020).
- [12] Youngkyu Hong, Seungju Han, Kwanghee Choi, Seokjun Seo, Beomsu Kim, and Buru Chang, 'Disentangling label distribution for long-tailed visual recognition', in *CVPR*, pp. 6626–6636, (2021).
- [13] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec, 'Ogb-lsc: A large-scale challenge for machine learning on graphs', *arXiv preprint arXiv:2103.09430*, (2021).
- [14] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec, 'Open graph benchmark: Datasets for machine learning on graphs', *NeurIPS*, 22118–22133, (2020).
- [15] Sergey Ioffe and Christian Szegedy, 'Batch normalization: Accelerating deep network training by reducing internal covariate shift', in *ICML*, (2015).
- [16] Qingnan Jiang, Mingxuan Wang, Jun Cao, Shanbo Cheng, Shujian Huang, and Lei Li, 'Learning kernel-smoothed machine translation with retrieved examples', in *EMNLP*, pp. 7280–7290, (2021).
- [17] Jeff Johnson, Matthijs Douze, and Herve Jegou, 'Billion-scale similarity search with gpus', *IEEE Transactions on Big Data*, (2019).
- [18] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley, 'Molecular graph convolutions: moving beyond fingerprints', in *Journal of Computer-aided Molecular Design*, pp. 595–608, (2016).
- [19] Diederik P. Kingma and Jimmy Ba, 'Adam: A method for stochastic optimization', in *ICLR*, (2015).
- [20] Thomas N Kipf and Max Welling, 'Semi-supervised classification with graph convolutional networks', in *ICLR*, (2017).
- [21] Janik Kossen, Neil Band, Clare Lyle, Aidan N Gomez, Thomas Rainforth, and Yarin Gal, 'Self-attention between datapoints: Going beyond individual input-output pairs in deep learning', in *NeurIPS*, (2021).
- [22] Mike Lewis, Marjan Ghazvininejad, Gargi Ghosh, Armen Aghajanyan, Sida Wang, and Luke Zettlemoyer, 'Pre-training via paraphrasing', in *NeurIPS*, (2020).
- [23] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel, 'Gated graph sequence neural networks', in *CoRR*, (2015).
- [24] Huihui Liu, Yiding Yang, and Xinchao Wang, 'Overcoming catastrophic forgetting in graph neural networks', in *AAAI*, pp. 8653–8661, (2021).
- [25] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L. Gaunt, 'Constrained graph variational autoencoders for molecule design', in *NeurIPS*, (2018).
- [26] Shengchao Liu, Mehmet F Demirel, and Yingyu Liang, 'N-gram graph: Simple unsupervised representation for graphs, with applications to molecules', in *NeurIPS*, (2019).
- [27] Daniel Mark Lowe, *Extraction of chemical structures and reactions from the literature*, Ph.D. dissertation, University of Cambridge, 2012.
- [28] Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar, 'Long-tail learning via logit adjustment', in *ICLR*, (2020).
- [29] Jun Rao, Tao Qian, Shuhan Qi, Yulin Wu, Qing Liao, and Xuan Wang, 'Student can also be a good teacher: Extracting knowledge from vision-and-language model for cross-modal retrieval', in *CIKM*, pp. 3383–3387, (2021).
- [30] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling, 'Modeling relational data with graph convolutional networks', in *CoRR*, (2017).
- [31] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Saucedo Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller, 'SchNet: A continuous-filter convolutional neural network for modeling quantum interactions', in *NeurIPS*, (2017).
- [32] Kristof Schütt, Oliver Unke, and Michael Gastegger, 'Equivariant message passing for the prediction of tensorial properties and molecular spectra', in *ICML*, (2021).
- [33] Mariya Toneva, Alessandro Sordani, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon, 'An empirical study of example forgetting during deep neural network learning', in *NeurIPS*, (2019).
- [34] Raphael JL Townshend, Martin Vögele, Patricia Suriana, Alexander Derry, Alexander Powers, Yianni Laloudakis, Sidhika Balachandrar, Bowen Jing, Brandon Anderson, Stephan Eismann, et al., 'Atom3d: Tasks on molecules in three dimensions', in *NeurIPS*, (2021).
- [35] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio, 'Graph attention networks', in *CoRR*, (2017).
- [36] Dingmin Wang, Ziyao Chen, Wanwei He, Li Zhong, Yunzhe Tao, and Min Yang, 'A template-guided hybrid pointer network for knowledge-based task-oriented dialogue systems', in *Proceedings of the 1st Workshop on Document-grounded Dialogue and Conversational Question Answering (DialDoc 2021)*, pp. 18–28, (2021).
- [37] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang, 'Knowledge-aware graph neural networks with label smoothness regularization for recommender systems', in *KDD*, pp. 968–977, (2019).
- [38] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui, 'Graph neural networks in recommender systems: a survey', *ACM Computing Surveys*, **55**(5), 1–37, (2022).
- [39] Yuhui Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy, 'Memorizing transformers', in *ICLR*, (2022).
- [40] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka, 'How powerful are graph neural networks?', in *ICLR*, (2018).
- [41] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, et al., 'Analyzing learned molecular representations for property prediction', in *Journal of Chemical Information and Modeling*, (2019).
- [42] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec, 'Graph convolutional neural networks for web-scale recommender systems', in *KDD*, (2018).
- [43] Xiao Zhang, Zhiyuan Fang, Yandong Wen, Zhifeng Li, and Yu Qiao, 'Range loss for deep face recognition with long-tailed training data', in *ICCV*, pp. 5409–5418, (2017).
- [44] Zhao Zhang, Fuzhen Zhuang, Hengshu Zhu, Zhiping Shi, Hui Xiong, and Qing He, 'Relational graph neural network with hierarchical attention for knowledge graph completion', in *AAAI*, pp. 9612–9619, (2020).
- [45] Fan Zhou and Chengtai Cao, 'Overcoming catastrophic forgetting in graph neural networks with experience replay', in *AAAI*, pp. 4714–4722, (2021).