Graph Self-supervised Learning with Augmentation-aware Contrastive Learning

Dong Chen
National University of Defense Technology
Changsha, Hunan, China
dongchen@nudt.edu.cn

Xiang Zhao*
National University of Defense Technology
Changsha, Hunan, China
xiangzhao@nudt.edu.cn

Wei Wang
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, Guangdong, China
The Hong Kong University of Science and Technology
Hong Kong, China
weiwcs@ust.hk

Zhen Tan
National University of Defense Technology
Changsha, Hunan, China
tanzhen08a@nudt.edu.cn

Weidong Xiao
National University of Defense Technology
Changsha, Hunan, China
wdxiao@nudt.edu.cn

ABSTRACT

Graph self-supervised learning aims to mine useful information from unlabeled graph data, and has been successfully applied to pre-train graph representations. Many existing approaches use contrastive learning to learn powerful embeddings by learning contrastively from two augmented graph views. However, none of these graph contrastive methods fully exploits the diversity of different augmentations, and hence is prone to overfitting and limited generalization ability of learned representations. In this paper, we propose a novel Graph Self-supervised Learning method with Augmentation-aware Contrastive Learning. Our method is based on the finding that the pre-trained model after adding augmentation diversity can achieve better generalization ability. To make full use of the information from the diverse augmentation method, this paper constructs new augmentation-aware prediction task which complementary with the contrastive learning task. Similar to how pre-training requires fast adaptation to different downstream tasks, we simulate train-test adaptation on the constructed tasks for further enhancing the learning ability; this strategy can be deemed as a form of meta-learning. Experimental results show that our method outperforms previous methods and learns better representations for a variety of downstream tasks.

CCS CONCEPTS

• **Information systems** → **Web mining**.

*Corresponding author

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583246

KEYWORDS

Graph Neural Networks, Self-supervised Learning, Contrastive Learning

ACM Reference Format:
Dong Chen, Xiang Zhao, Wei Wang, Zhen Tan, and Weidong Xiao. 2023. Graph Self-supervised Learning with Augmentation-aware Contrastive Learning. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3543507.3583246

1 INTRODUCTION

Graphs are ubiquitous in our lives, and their applications are common in real-world systems, such as social network graphs, molecular structure graphs, and traffic road maps [2, 27, 35]. These graph structures often contain rich structural patterns and semantic information, which can support many downstream tasks, such as recommendation systems, molecular properties prediction and traffic forecasts systems.

The past few years have witnessed rapid advances in graph representation learning using graph neural networks (GNNs) [16], which refers to the use of neural networks to extract and discover features and patterns in graph-structured data [5, 15]. Despite great successes of GNNs in addressing graph-based tasks, these successes rely heavily on using massive amounts of carefully labeled data to conduct supervised learning. However, accurate annotation is very time-consuming and labor-intensive. In addition, trained with supervised data alone may result in models that are overfitting and may fail to generalize well [23]. Recently, self-supervised learning becomes an effective technique to address the above two problems, as it allows models to acquire knowledge from unlabeled data. Substantial work has shown that self-supervised learning, on the large language corpus and image datasets, can learn universal representations, which are beneficial for downstream tasks and can avoid training a new model from scratch [10, 22].

In recent years, contrastive learning has become the prevalent form of self-supervised learning, and has achieved many successes.

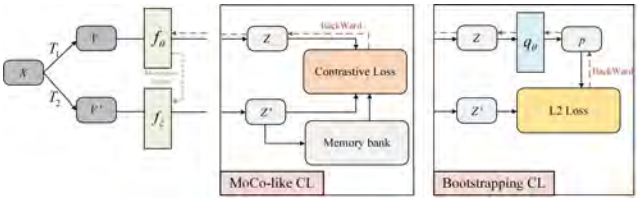


Figure 1: Difference between MoCo-like and Bootstrapping contrastive learning

It aims to build effective representation by pulling the representations of semantically similar (or positive) pairs together and pushing those of the dissimilar (or negative) pairs apart. We can construct a positive pair by creating two augmented versions of the same sample, and a negative pair by pairing two irrelevant samples [1, 31]. Restricted by the requirement of the large batch for obtaining numerous negative samples, MoCo [10] maintains the dictionary as a queue of data samples, allowing it to be large to store more negative samples.

Nowadays, contrastive learning have been successfully adopted to graphs [29]. In the earlier graph contrastive learning methods [9, 30], the positive and negative pairs in the graph are generated by fixed graph augmentation methods, such as node/edge perturbation and node feature masking. One critique for these traditional methods is that they require numerous negative samples for achieving promising results. The emergence of bootstrapping [1] contrastive modes has been proposed to tackle this problem [24]. As shown in Figure 1, a training example X is separately augmented by two augmentation function T_1 and T_2, then two encoder f_θ and f_ξ are applied in augmented example V and V', which ξ is momentum updated by θ [10]. The difference between MoCo-like Contrastive Learning and Bootstrapping Contrastive Learning is Bootstrapping CL adopts bootstrapping strategy [6] to get away from large amounts of negative samples.

The augmentation procedure defined in computer vision is extending the family of augmentations and composing them stochastically to achieve better generalization performance [1]. However, existing graph contrastive learning method use fixed augmentation method. The fixed augmentation in graph CL methods may limit the diversity of the augmentations, resulting in overfitting and poor generalization. Another augmentation-free contrastive learning, such as AFGRL [17] throws away the augmentation procedure and constructs positive samples by complex node transformation, they also didn't need negative samples and avoided the choice of augmentation. But it adds more complexity to constructing a positive sample, which is far more time-consuming and memory-consuming than applying an augmentation operation. We also analyze the time and space complexity in Section 5.

Due to the lack of diversity during the training, the quality of the learned graph representation of the previous contrastive method still has much room for improvement. Inspired by aforementioned problem, we propose a self-supervised learning framework for graphs, called *A*ugmentation-aware *B*ootstrapping via *G*raph *M*eta-*L*earning (ABGML). Precisely, our proposed ABGML is formed by a meta-learning framework, which contains two branches, i.e., the contrastive branch and augmentation-aware prediction branch. We improve augmentation diversity by constructing an augmentation pool. Then, at each iteration, two augmentation methods are randomly sampled in augmentation pool. The contrastive branch has more augmentation diversity, which originated from the newly designed augmentation pool. Specifically, two different augmentations are sampled from this pool first instead of use fixed on the entire training process.

The augmentation-aware prediction branch uses the node representation to predict the type of augmentation applied to the graph; this prediction enables the model to know how the current graph is being perturbed so that each node can learn information about the structure of the entire graph, hence, we named it augmentation-aware prediction task. This task with global-level perspective can be complementary with contrastive learning to better learn the representations.

Besides, for imitate the fine-tuning process when applying the pre-trained model to downstream tasks, we create several sub-tasks with different train/test splits and diverse combinations of augmentation. In each sub-tasks, we perform the above contrastive learning branch and the augmentation-aware prediction branch, which can be deemed as multi-task learning process. Therefore, we integrated the above multi-task learning process with model-agnostic meta-learning (MAML)[3], which can provide good initialization for fast adaptation to a new task, and naturally meets the requirement of self-supervised graph representation. To the best of our knowledge, ABGML first work that learns representations of graphs with multi-task learning and meta-learning framework, which jointly improve the ability of generalization.

Our contributions are summarized as follows:

- We propose an augmentation sample strategy that can bring more diversity to the training process when constructing contrastive pairs in bootstrapping mode.
- We propose an augmentation prediction branch to enhance the global-level perspective training and can be complementary with contrastive learning to better learn the representation.
- We propose a framework to combine meta-learning with graph self-supervised, and such a training pipeline can further improve the generalization ability in downstream tasks.
- Our extensive experiments demonstrate the superiority of our ABGML model in performance on three downstream tasks: node classification, clustering, and similarity search; then detailed ablation studies show the effectiveness of each component.

2 RELATED WORK

2.1 Graph Contrastive Learning

Motivated by the great success of contrastive methods on self-supervised learning, plenty of graph contrastive learning methods have been proposed. Most of them can be roughly divided into three categories, i.e., traditional graph contrastive learning, bootstrapping contrastive learning, and augmentation-free contrastive learning. The traditional graph contrastive method aims to maximize the agreement of node embeddings across two corrupted views of the

[1] "Bootstrap" is used in its idiomatic sense rather than the statistical sense.

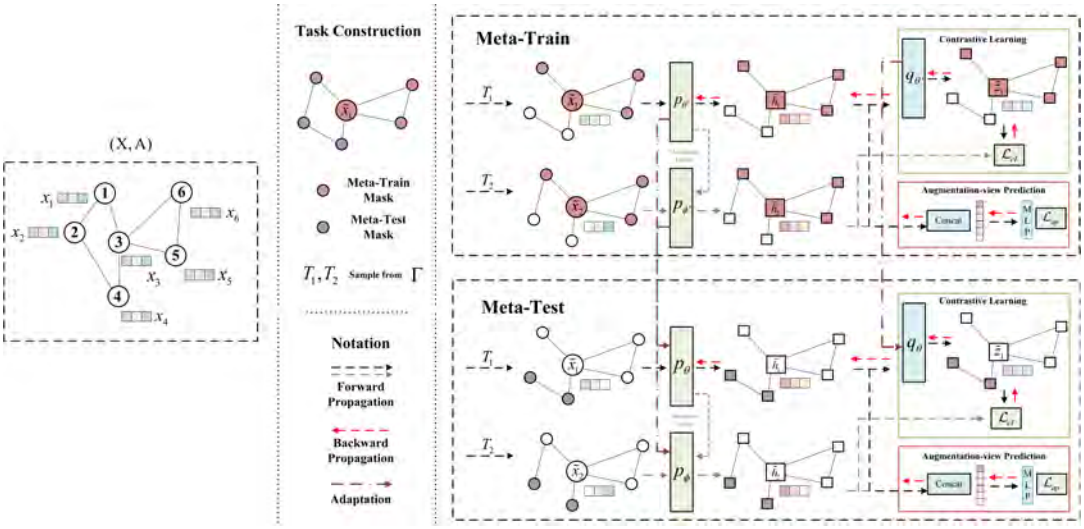


Figure 2: The overall framework of ABGML. The contrastive branch builds representation through the use of two graph encoders, an online encoder p_θ, a target encoder p_ϕ and an online predictor q_θ, where θ and ϕ denote two distinct sets of parameters. θ' and ϕ' is the intermediate adaptation parameter only for meta-train.

graph and minimize the agreement of embeddings across different nodes. It is usually time-costly because it requires numerous negative samples. DGI [26] is a pioneering work highly inspired by Deep InfoMax [11], it leverages corruption to construct negative pairs and maximizing the mutual information between local (node) and global (graph) representation. GRACE [32] and GCA [34] use SimCLR-style [1] in-batch negatives, which first create two augmented views of a graph by randomly perturbing nodes/edges and their features. Then, it learns node representation by pulling together the representation of the same node in the two augmented graphs, while pushing apart the representation of other node. For building large negatives in limited resources, GCC [21] uses MoCo-style negative queues. The queue decouples the dictionary size from the mini-batch size, allowing it to be large.

BYOL [7] is the first work to present the bootstrap method in image contrastive learning, which is to learn representations by predicting previous versions of its outputs, without using negative samples. BGRL [24] introduces the bootstrap method into graph contrastive learning, it adopts BYOL and uses fixed augmentations to learn node representation. The difference between BGRL and BYOL is BGRL does not use a projector network, which is used for dimensionality reduction.

Another contrastive learning form is AFGRL [17], an augmentation-free method, which requires neither augmentation method nor negative samples for learning representations of graphs. It changed the focus of positive sample construction from different augmentation methods to selecting neighbor similar points as positives. However, these additional positive select operations will bring more time and space complexity. This makes it difficult for the model to handle large datasets, which is the purpose of the pre-training model.

Besides, high-quality and informative data augmentation plays a central role in the success of contrastive learning[25]. In the graph domain, different views of a graph provide different contexts for each node. Existing works[33] propose to corrupt the original graph at both structure and attribute levels, i.e., Remove edges (RE) randomly remove a portion of edges in the original graph. Mask features (MF) apart from removing edges, instead randomly mask a fraction of node feature with zeros.

2.2 Meta Learning

In recent years, meta-learning has turned out to be an important framework to address the shortcomings of deep learning systems when applied in few-shot tasks. The main idea behind meta-learning is to design learning algorithms that can leverage prior learning experience to adapt to a new problem quickly and learn a useful algorithm with few samples. Some research has use meta-learning to enhance the learning ability of existing video self-supervised approaches [18]. Most of the existing literature adopts the method called model-agnostic meta-learning (MAML), which can be combined with any learning approach trained with gradient descent. In graph domain, Huang et al.[13] consider the node classification problem where the input graphs, as well as the labels, can be different across tasks. Wang et al.[28] consider the few-shot node classification problem for a setting where the network structure is fixed, but the features of the nodes change with tasks. In this work, we utilize MAML to improve the performance of graph contrastive self-supervised learning. Different from prior efforts, we try to enhance the adaptation between the self-supervised train and test domain.

3 PRELIMINARIES

In this section, we begin by introducing some notations related to graph contrastive learning. Then we describe the basic settings of self-supervised graph representation learning.

3.1 Notations

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the input graph with node set $\mathcal{V} = \{v_i\}_{i=1}^{N}$ and edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. $N = |\mathcal{V}|$ is the number of the node in graph $\mathcal{G}$. We denote the adjacency matrix as $\mathbf{A} \in \{0,1\}^{N \times N}$, where $\mathbf{A}_{ij} = 1$ indicates node i and j are connected and equal 0 vice versa. We further denote the feature matrix as $\mathbf{X} \in \mathbb{R}^{N \times F}$, where F is the dimension of the node features and $\mathbf{X}_{i:}$ indicates the feature vector of node i (the i th row of $\mathbf{X}$).

3.2 Self-supervised Graph Representation Learning

In the setting of self-supervised graph representation learning, there is no given class information of nodes during pre-training. Given the graph $\mathcal{G}$ along with $\mathbf{X}$ and $\mathbf{A}$, we aim to learn an encoder $p(\cdot)$ that receiving the graph features and structure as input, and produces node embeddings in low dimensionality $\mathbf{H} = p(\mathbf{X}, \mathbf{A}) \in \mathbb{R}^{N \times F'}$, where $F' \ll F$. These representations can be used in downstream tasks, such as node classification, clustering, and similarity search.

4 METHODOLOGY

In this section, we present our framework ABGML in detail. The overall architecture is illustrated in Figure 2. First, we describe the full pipeline and make an overview of ABGML. Next, we introduce two branches in our framework, which are the contrastive branch and the augmentation-aware prediction branch. Last, we explain the adaptation procedure in the MAML setting, which simulates the process of train-test adaptation during the pre-training.

4.1 Overview

To enhance the augmentation diversity, we explicitly propose an augmentation sampling strategy to randomly augment graphs. Besides, we add augmentation-aware prediction as an auxiliary task for capturing graph-level semantic features which use node features to predict the disturbed state of the whole graph. At last, we leverage the meta-learning framework on the above two branches to improve the generalization ability on downstream tasks.

4.2 Contrastive branch

In order to learn expressive nodes' representation from an input network, we first build the network encoder module. The network encoder module is composed of multiple GNN layers that encode each node to a low-dimensional latent representation. The GNN model involves two key computations for each node v at every layer. The first is **Aggregation** operation, which aggregates messages from v's local neighbors $\mathcal{N}_v$ in an iterative manner. The second is **Transform** operation, which updates v's representation from its representation in the previous layer and the aggregated messages. Formally, a generic GNN layer computes the node representations using two key functions:

$$\mathbf{h}_{\mathcal{N}_i}^{l} = \text{AGGREGATE}^{\,l}\left(\left\{\mathbf{h}_j^{l-1} \mid \forall j \in \mathcal{N}_i \cup v_i\right\}\right), \tag{1}$$

$$\mathbf{h}_i^{l} = \text{TRANSFORM}^{\,l}\left(\mathbf{h}_i^{l-1}, \mathbf{h}_{\mathcal{N}_i}^{l}\right), \tag{2}$$

where h_i^l is the latent representation of node v_i at the l-th layer and $\mathcal{N}_i$ is the set of first-order neighboring nodes of node v_i. For simplicity, we abstract the composition of the two operation AGGREGATE$(\cdot)$ and TRANSFORM$(\cdot)$ with L GNN layer as one parameterized function p_θ with parameters θ. It is worth noting that the network encoder is compatible with arbitrary GNN-based architecture, and here we employ the GCN [16] in our implementation.

In the pre-training, ABGML first sample graph augmentation function T_1 and T_2 from augmentation pool Γ. The augmentation pool is composed of a set of different augmentations functions. In this paper, we only use Edge masking and Feature masking as the base augmentation; we construct new augmentation method based the combination of two base methods with different probability. Then two alternate views of $\mathcal{G}$: $\mathcal{G}_1 = (\tilde{\mathbf{X}}_1, \tilde{\mathbf{A}}_1)$ and $\mathcal{G}_2 = (\tilde{\mathbf{X}}_2, \tilde{\mathbf{A}}_2)$ are produced by applying the above sampled augmentation functions T_1 and T_2 respectively. The online encoder generates online representation from the first augmented graph, $\tilde{\mathbf{H}}_1 = p_\theta(\tilde{\mathbf{X}}_1, \tilde{\mathbf{A}}_1)$; In the same way, the target encoder produces target representation from the second augmented graph, $\tilde{\mathbf{H}}_2 = p_\phi(\tilde{\mathbf{X}}_2, \tilde{\mathbf{A}}_2)$. Then, the online representation is fed into a node-level predictor q_θ that outputs estimates of the target representation, $\tilde{\mathbf{Z}}_1 = q_\theta(\tilde{\mathbf{H}}_1)$.

The online parameters θ (not ϕ) are updated to make the predicted target representation $\tilde{\mathbf{Z}}_1$ closer to the true target representations $\tilde{\mathbf{H}}_2$ for each node, which through the following loss:

$$\mathcal{L}_{cl} = -\frac{2}{N}\sum_{i=0}^{N-1}\frac{\widetilde{\mathbf{Z}}_{(1,i)}\widetilde{\mathbf{H}}_{(2,i)}^{\top}}{\left\|\widetilde{\mathbf{Z}}_{(1,i)}\right\|\left\|\widetilde{\mathbf{H}}_{(2,i)}\right\|}. \tag{3}$$

In the actual training process, we symmetrize this loss, by also predicting the target representation of the first view using the online representation of the second view.

The target network provides the regression targets to train the online network, and its parameters ϕ are an exponential moving average of the online parameters θ. More precisely, given a target decay rate $\tau \in [0,1]$, after each training step, we perform the following update (ϕ does not updates through back propagation):

$$\phi \leftarrow \tau\phi + (1-\tau)\theta. \tag{4}$$

4.3 Augmentation-aware prediction branch

Since the above augmentation method is randomly selected, different augmentation pairs bring more diversity to the contrastive branch. But with the contrastive branch only, the model only captures local semantic feature since each node only contrasting with the same node from other view. To capture global-level semantic features, we construct an augmentation-aware prediction branch. As shown in Figure 2 and Eq 5, by concatenating two features of augmented node representation, the multi-class classification tasks is meant for predicting current augmentation states. We use an example to illustrate the label construction process of this branch. Suppose that we have only two augmentation method T_1 and T_2, we can obtain four different augmentation pairs ($T_1T_1, T_1T_2, T_2T_1, T_2T_2$) by randomly permutate. Since the prediction is order-agnostic, we treat labels T_1T_2 and T_2T_1 as the same label. Thus, two augmentation methods are able to obtain three types of labels. In our approach, we concatenate features from two augmentation views:

$$\tilde{h} = \text{Concat}(\tilde{h}_1, \tilde{h}_2), \tag{5}$$

where $\tilde{h}_1, \tilde{h}_2$ are the representation generated by p_θ and p_ϕ.

The multi-class classification loss can be complementary with contrastive loss to better learn the representations through multi-task learning process:

$$\mathcal{L}_{ap} = -\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log \hat{y}_{ij}, \tag{6}$$

where the N is the number of the nodes in a batch, M is the number of classes. y_{ij} equal 1 if the true label of i th concatenated feature is j. $\hat{y}_{ij}$ is the jth position of output after input $\tilde{h}$.

4.4 Meta-learning Optimization

The aim of meta-learning perfectly meets the target of the graph pre-training, which meant to improve the generalization ability of the model. Given the contrastive learning (CL) and augmentation-aware prediction (AP) tasks, we fuse the two losses, i.e., contrastive loss and cross-entropy loss, for final optimization. See from Figure 2, during the pre-training process, we employ a two-stage process including meta-train and meta-test for imitating the train-test adaptation. The procedure of the two-stage basically the same, however, the parameter of meta-train is update temporarily only use for meta-test adaptation. To train the model's ability to adapt between different tasks, we randomly construct task at each training step, the details of task construction are list as follows:

4.4.1 Task Construction. A task involves a graph $\mathcal{G}$ with N nodes, consisting of a support set $\mathcal{S}_\mathcal{G}$ and a query set $\mathcal{Q}_\mathcal{G}$. Consider a set of augmentation pool $\Gamma = \{T^1, T^2, ..., T^{n^{pool}}\}$, n_{pool} is the size of augmentation pool. Each tasks sample two augmentation method in Γ, which is T_1 and T_2. Then, two alternate views of $\mathcal{G}$ is produced by applying two different augmentation method. We randomly sample α percent node to compose the support set and the remaining $1-\alpha$ node as the query set. As illustrated in Figure 2, the task $\mathcal{T}_\mathcal{G}$ contains two child tasks (CL and AP), i.e., $\mathcal{T}_\mathcal{G} = (\mathcal{T}_\mathcal{G}^{cl}, \mathcal{T}_\mathcal{G}^{ap})$. We use $c \in \{cl, ap\}$ denote the child tasks, which is defined as:

$$\begin{aligned} \mathcal{T}_\mathcal{G}^c = &\left(\mathcal{S}_\mathcal{G}^c = \{(n_s) \sim p_\mathcal{V}\}, \mathcal{Q}_\mathcal{G}^c = \left\{(n_q) \sim p_\mathcal{V}\right\}\right), \\ \text{s.t.} \quad &\mathcal{S}_\mathcal{G}^c \cap \mathcal{Q}_\mathcal{G}^c = \emptyset,\ T_1, T_2 \sim \Gamma \\ &|\mathcal{S}_\mathcal{G}^c| = \alpha N,\ |\mathcal{Q}_\mathcal{G}^c| = (1-\alpha)N \end{aligned} \tag{7}$$

where the support $\mathcal{S}_\mathcal{G}^c$ and query $\mathcal{Q}_\mathcal{G}^c$ contain nodes n_s and n_q separately, which are randomly sampled from the node distribution $p_\mathcal{V}$ of the graph; and they are mutually exclusive, $|*|$ is represent the number of nodes in $*$.

4.4.2 Meta Training. Given the two training tasks and support set, the above Figure 2 is show the meta-training process. As motivated, to enhance the generalization ability of the pre-training model, it is necessary to optimize the model's ability to quickly adaptation during pre-training itself. We combine CL loss and AP loss from the two branches are combined together to get the final meta loss, which is defined as:

$$\mathcal{L}_{meta} = \beta * \mathcal{L}_{cl} + (1-\beta) * \mathcal{L}_{ap}, \tag{8}$$

where β is a hyper-parameter that is used to control the importance of CL and AP loss. During the meta-train stage, the model parameters are trained by optimization $\mathcal{L}_{meta}$ with respect to θ across all learning tasks. More concretely, the update procedure of parameter in meta-train is defined as follows:

$$\theta' = \theta - \eta\frac{\partial \mathcal{L}_{meta}(\theta, \mathcal{S}_\mathcal{G}^c)}{\partial \theta}, \tag{9}$$

where η is the learning rate of the meta training process. In this process, the actual parameters of the model are not updated, but change the temporarily parameters θ' we created for next stage adaptation.

4.4.3 Meta Test. To mimic the testing process with the fine-tuned model, the model is initialized by temporarily parameters θ', then optimize the adapted parameters θ on the query set $\mathcal{Q}_\mathcal{G}^c$ over all training task. That is, the transferable prior θ will be optimized through the backpropagation of the query loss given by

$$\theta = \theta - \gamma\frac{\partial \mathcal{L}_{meta}(\theta', \mathcal{Q}_\mathcal{G}^c)}{\partial \theta}. \tag{10}$$

We set the adaptation learning rate γ as the same as the learning rate η for simplicity. Through the meta-test process, the training object of the model is to achieve better generalization ability, which well fits the requirements of the pre-train task.

5 COMPLEXITY ANALYSIS

In this section, we provide a brief description of the time and space complexities of the ABGML update step, and illustrate its advantages compared to augmentation-free contrastive methods such as AFGRL. The same analysis applies to traditional contrastive learning methods such as GRACE and bootstrapping method such as BGRL.

Consider a graph with N nodes and M edges, and simple one layer encoders p_θ that compute embeddings in time and space $O(dN)$, d is the average degree of node. ABGML performs two encoder computations per update step, a node-level prediction step and an augmentation-aware prediction step. Since the data used by Meta-train and Meta-test in the meta-learning framework are divided by the whole graph, each node also performs only one update. Both methods backpropagate the learning signal twice (once for each augmentation), and we assume the backward pass to be approximately as costly as a forward pass. The cost for computation of the augmentations is $O(N+M)$.

The total time and space complexity per update step for ABGML is $6C_{encoder}O(dN)+4C_{prediction}O(N)+4C_{augware}O(N)+2O(N+M)$. Since AFGRL need to update the distence between each node for performing k-NN and K-mean clustering, the time and space complexity for AFGRL is $6C_{encoder}O(dN) + 4C_{prediction}O(N) + O(N^2)$, compared to $6C_{encoder}O(dN)+4C_{prediction}O(N)+2O(N+M)$ for BGRL and $4C_{encoder}O(dN)+4C_{prediction}O(N)+O(N^2)$ for GRACE; where C are constants depending on architecture of the different components. Since augmentationa-ware prediction uses only one layer MLP, the ABGML is faster than AFGRL and GRACE on a sufficiently large dataset, which have quadratic complexity.

6 EXPERIMENTS SETUP

To evaluate the effectiveness of our method ABGML, we conduct three downstream tasks (Node classification, Node clustering, and Similarity search) on five widely used datasets, including WikiCS, Amazon-Computers, Amazon-Photo, Coauthor-CS, and Coauthor-Physics. Then, to measure the performance of the model on a much larger dataset, we further test the performance on the ogbn-arxiv dataset [12]. Finally, we also show the sensitivity analysis on different hyperparameters, especially on the size of augmentation pools.

For a comprehensive comparison, the datasets are collected from real-world networks from different domains; their detailed statistics are summarized in Appendix. For WikiCS dataset, which provides 20 canonical train/valid/test splits, we directly used the given splits. For Amazon and Coauthor datasets, we randomly split nodes 20 times into train/valid/test (10/10/80) as these datasets do not provide standard splits. For ogbn-arxiv dataset, we report results on both validation and test sets, as is convention for this task since the dataset is split based on a chronological ordering.

6.1 Method Compared

We primarily compare ABGML against AFGRL [17], BGRL [24] and GCA [34], which are the current state-of-art self-supervised methods on graphs. We also report performances of other previously published results of another representative method, such as DeepWalk [20], DGI [26], GMI [19]and MVGRL [8], as done in [17].

6.2 Implementation details

We use GCN model as the base encoder $f(\theta)$ as done in [24], the formal definition of encoder architecture is:

$$\mathbf{H}^{(l)} = \mathrm{GCN}^{(l)}(\mathbf{X}, \mathbf{A}) = \sigma\left(\hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{X}\mathbf{W}^{(l)}\right), \qquad (11)$$

where $\mathbf{H}^{(l)}$ is represent node embedding matrix of the nodes, l is represent the current layer of GCN, $l \in [1, ..., L]$. $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with self-loops, $\hat{\mathbf{D}} = \sum_i \hat{\mathbf{A}}_i$ is the degree matrix, $\sigma(*)$ is a nolinear activation function, we use ReLU in this paper, and $\mathbf{W}^{(l)}$ is the trainable weight matrix for the layer l.

We use a combination of random node feature masking and edge masking with different masking probabilities p_f and p_e for a fair comparison with [24].

In this work, to add more augmentation diversity, we create an augmentation pool for sampling in meta-training. We set size of augmentation pool $n_{pool} = 4$ across the experiment. We perform parameter search on several hyperparameters, such as masking probabilities p_f and p_e, learning rate η, node embedding dimension D, number of layer of GCN l, support set proportion α, contrastive loss importance β. The best configuration of these parameters are shown in Appendix. We set $\tau = 0.99$ which same as other method. More detail is shown in the supplementary.

6.3 Evaluation protocol

We evaluate ABGML on five public benchmarks (WikiCS, Amazon Photo & Computer, Coauthor CS & Physics) with three node-level tasks, such as node classification, node clustering and node similarity search. To validate the effectiveness on a much larger dataset, we also conduct a node classification experiment on ogbn-arxiv. For all these experiments, we use the fixed learned embeddings to train and test a simple linear classifier. We report the performance when the model has the best performance on validation data. For node clustering and similarity search, we follow the AFGRL setting, which reports the best performance report in the training process.

7 EXPERIMENTAL ANALYSIS

We present an extensive empirical study of performance. All these experiments is conducted by *linear classification* on *frozen* features, following a common protocol [17].

7.1 Overall Performance

Table 1 shows the node classification performance of various methods on five challenging medium-scale datasets specifically proposed for rigorous evaluation of the semi-supervised node classification method. X, A, Y correspond to node features, the adjacency matrix, and labels respectively. The highest performance of unsupervised models is highlighted in **boldface**. The second highest performance is highlighted with underline. OOM indicates Out-Of-Memory on a 24GB RTX 3090 GPU. All experiments are over 20 random dataset split and model initialization.

We have the following observations: Overall, our proposed model ABGML shows strong performance across all five datasets compared with other methods. The performance empirically verifies the superiority of our augmentation-aware contrastive learning framework.

As shown in Table 2 and 3, we also evaluate ABGML on node clustering and similarity search for testing generalization performance. Note that the best hyperparameters for node classification task we adopted. It is worth noting that traditional contrastive methods perform worse than their counterparts on various tasks and our method generally outperforms other methods on almost all datasets. Since the primary aim of the pre-training model is achieving higher generalization performance under different downstream tasks, the experiments of the Tables 1, 2 and 3 are combined to illustrate that our method surpasses the previous state-of-art method AFGRL on different downstream tasks.

In summary, the superior performance of ABGML compared to existing state-of-the-art methods verifies the effectiveness of our proposed method.

7.2 Ablation studies

In this section, we conduct ablation studies on the above five datasets to comprehensive verify the benefit of each component of ABGML. The results are presented in Table 4, The Meta, CL, Aug. Aware and Aug. Pool are represent the Meta-Learning framework, Contrastive Learning Branch, Augmentation-Aware prediction Branch, and Augmentation pool. w/o. means without in Table 4. We have concluded several findings that summarized some results and analyze the phenomenon behind the numbers:

Firstly, the performance of the model has been degraded to varying degrees when each component is removed as shown in first four rows in table. Especially when contrastive learning is removed, the effect is the most obvious, which also shows that contrastive learning is one of the most important components in our model. Secondly, as shown in row 2, 5 and 6, when separately remove

Table 1: Performance on node classification in terms of accuracy in percentage with stadard deviation.

	Training Data	WikiCS	Amazon-Computers	Amazon-Photo	Coauthor.CS	Couthor.Physics
Sup.GCN	X, A, Y	77.19±0.12	86.51±0.54	92.42±0.22	93.03±0.31	95.65±0.16
Raw feat.	X	71.98±0.00	73.81±0.00	78.53±0.00	90.37±0.00	93.58±0.00
node2vec	A	71.79±0.05	84.39±0.08	89.67±0.12	85.08±0.03	91.19±0.04
DeepWalk	A	74.35±0.06	85.68±0.06	89.44±0.11	84.61±0.22	91.77±0.15
DW + feats.	X, A	77.21±0.03	86.28±0.07	90.05±0.08	87.70±0.04	94.90±0.09
DGI	X, A	75.35±0.14	83.95±0.47	91.61±0.22	92.15±0.63	94.51±0.52
GMI	X, A	74.85±0.08	82.21±0.31	90.68±0.17	OOM	OOM
MVGRL	X, A	77.52±0.08	87.52±0.11	91.74±0.07	92.11±0.12	95.33±0.03
GRACE	X, A	77.97±0.63	86.50±0.33	92.46±0.18	92.17±0.04	OOM
GCA	X, A	77.94±0.67	87.32±0.50	92.39±0.33	92.84±0.15	OOM
BGRL	X, A	76.86±0.74	89.69±0.37	93.07±0.38	92.59±0.14	95.48±0.08
AFGRL	X, A	77.62±0.49	89.88±0.33	93.22±0.28	93.27±0.17	95.69±0.10
ABGML	X, A	**78.70±0.56**	**90.17±0.30**	**93.46±0.36**	**93.56±0.19**	**96.05±0.10**

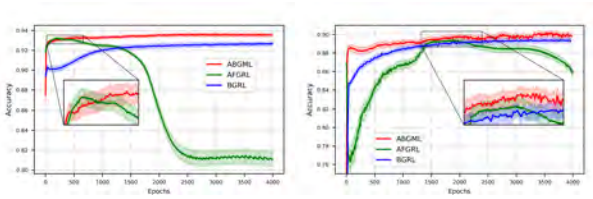
Table 2: Performance on node clustering in terms of NMI and homogeneity

		GRACE	GCA	BGRL	AFGRL	ABGML
WikiCS	NMI	0.4282	0.3373	0.3969	0.4132	**0.4598**
	Hom.	0.4423	0.3525	0.4156	0.4307	**0.4779**
Com.	NMI	0.4793	0.5278	0.5364	0.5520	**0.5659**
	Hom.	0.5222	0.5816	0.5869	0.6040	**0.6239**
Photo	NMI	0.6513	0.6443	0.6841	0.6563	**0.6938**
	Hom.	0.6657	0.6575	0.7004	0.6743	**0.7124**
Co.CS	NMI	0.7562	0.7620	0.7732	0.7859	**0.8033**
	Hom.	0.7909	0.7965	0.8041	0.8161	**0.8342**
Co.Phy.	NMI	OOM	OOM	0.5568	**0.7289**	0.7229
	Hom.	OOM	OOM	0.6018	**0.7354**	0.7329

Table 3: Performance on similarity search in terms of Sim@5 and Sim@10

		GRACE	GCA	BGRL	AFGRL	ABGML
WikiCS	Sim@5	77.54	77.86	77.39	78.11	**78.51**
	Sim@10	76.45	76.73	76.17	76.60	**77.38**
Com.	Sim@5	87.38	88.26	89.47	89.66	**90.18**
	Sim@10	86.43	87.42	88.55	88.90	**89.20**
Photo	Sim@5	91.55	91.12	92.45	92.36	**92.60**
	Sim@10	91.06	90.52	91.95	91.73	**92.12**
Co.CS	Sim@5	91.04	91.26	91.12	91.80	**92.36**
	Sim@10	90.59	91.00	90.86	91.42	**92.10**
Co.Phy.	Sim@5	OOM	OOM	95.04	95.25	**95.65**
	Sim@10	OOM	OOM	94.64	94.86	**95.39**

the CL and Aug. Aware component under normal setting (without Meta train and Meta test), the performance has been degraded to varying degrees; which further proves the effectiveness of CL and Aug. Aware component under normal setting. Thirdly, when only the augmentation-aware prediction task is applied (in row 5), its downstream performance is better than the traditional contrastive method (GRACE, GCA), which demonstrates the effect of our Aug. Aware prediction task. Finally, as shown in the last two lines of Table 4, the performance degradation when Aug. Pool removed.



(a) (b)

Figure 3: Node classification accuracy in Coauthor-CS dataset and Amazon-Computer dataset, the result is obtained by linear probing in pre-trained embeddings

The result proved that increasing the diversity of augmentation will boost the performance of pre-train model.

7.3 Experiment on ogbn-arxiv

Then, we conduct a node classification task on a much larger dataset, ogbn-arxiv. Same as BGRL, considering the difficulty of this task, we expand our model to use 3 GCN layers. Other results are taken from previously published work [24]. In addition, we report results from [12] for node2vec and supervised-learning baseline. We report results on both validation and test sets, as is a convention for this task since the dataset is split based on chronological ordering.

The results are summarized in Table 5, showing that ABGML is surpassing another self-supervised learning method. In this case, GRACE and AFGRL run out-of-memory on this dataset. These results suggest that the performance of contrastive methods such as AFGRL and GRACE may suffer due to their quadratic time and space complexity when scaling up.

7.4 Convergence Analysis

In Figure 3, we plot three models' (ABGML, AFGRL and BGRL) downstream node classification accuracy curve throughout training for Coauthor-CS and Amazon-Computer dataset. The hyperparameter configuration of these model are select the optimal parameters

Table 4: Ablation studies on ABGML

	WikiCS	Amazon-Computers	Amazon-Photo	Coauthor.CS	Couthor.Physics
ABGML	**78.70±0.56**	**90.17±0.30**	**93.46±0.36**	**93.56±0.19**	**96.05±0.10**
- w/o. Meta	78.44±0.65	90.12±0.32	93.35±0.37	93.53±0.19	95.99±0.10
- w/o. CL	78.11±0.54	89.60±0.27	92.84±0.43	93.30±0.19	95.89±0.15
- w/o. Aug. Aware	78.39±0.58	90.07±0.37	93.34±0.35	93.34±0.21	95.86±0.12
- w/o. Meta & CL	77.86±0.55	89.45±0.26	92.76±0.41	93.31±0.18	95.74±0.12
- w/o. Meta & Aug. Aware	78.38±0.49	90.10±0.39	93.35±0.34	93.33±0.18	95.81±0.12
- w/o. Meta & Aug. Aware & Aug. Pool	76.86±0.74	89.69±0.37	93.07±0.38	92.59±0.14	95.48±0.08

Table 5: Classification accuracy on the ogbn-arXiv task along with standard deviations.

	Validation	Test
MLP	57.65±0.12	55.50±0.23
node2vec	71.29±0.13	70.07±0.13
Random-Init	69.90±0.11	68.94±0.15
DGI	71.26±0.11	70.34±0.16
GRACE	OOM	OOM
BGRL	71.46±0.15	70.01±0.20
AFGRL	OOM	OOM
ABGML	72.01±0.11	70.78±0.18
Supervised GCN	**73.00±0.17**	**71.74±0.29**

reported on [17] and [24]. As we see, the downstream node classification accuracy of AFGRL drop quickly when iterating more epochs. This may be the fact that AFGRL selects similar surrounding nodes as positive examples for contrastive learning, and after a long training period, the embedding of the nodes start to converge to the same, which makes the learned representations indistinguishable after fine-tune on downstream tasks. Then, compared to BGRL, we can see that the convergence speed is greatly improved, and ABGML achieves the previous BGRL performance with only a few epochs, which is attributed to our meta-learning process.

7.5 Hyperparameter Analysis

Further, as shown in Figure 4, we conduct hyperparameter sensitivity analysis on Coauthor-CS dataset with the learning rate η, support proportion α, task proportion β, edge mask rate p_e, feature mask rate p_f and the size of augmentation pool n_{pool}.

The green dash line represents the performance of the former state-of-the-art method AFGRL. We found that $\eta = 7e-4$ gives the best performance, and a small learning rate usually achieves better performance. It's better to use a small learning rate because high learning rates increase the risk of losing previous knowledge in pre-training. The support proportion also achieves better performance on a low value. This may be the fact that the ratio of training set larger than test set at fine-tune, which is consistent with pre-training. The performance increases as the proportion of contrastive learning increases. This also confirms that the contrastive learning module plays a major role in pre-training, which consistent with the ablation study. When varying the mask rate p_e and p_f, the performance of the model increases with the mask

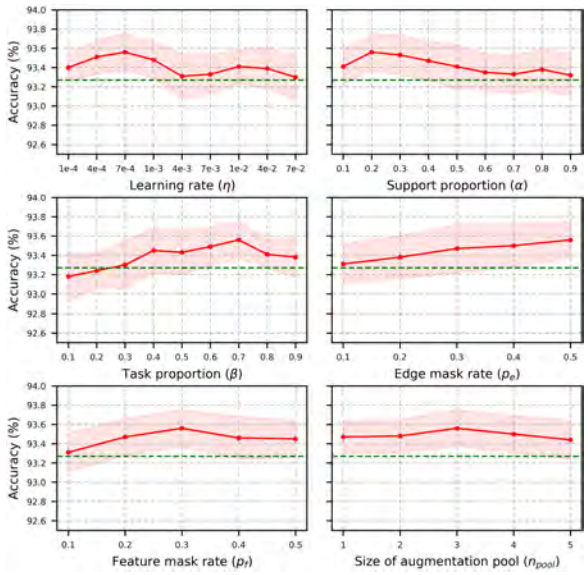


Figure 4: Hyperparameters Analysis on learning rate η, support proportion α, task proportion β, edge mask rate p_e, feature mask rate p_f and size of augmentation pool n_{pool}.

rate, which suggests that more complex augmentation is beneficial for model generalization performance. Then, performance of the model first increases up to a peak and then decreases, suggesting that more ways of augmentation can degrade the model. Last, from the result of the Figure 4, these numbers verifies that our ABGML can be easily trained compared with former methods, i.e., stable over hyperparameters, while outperforming them in most cases.

8 CONCLUSION

In this work, we propose ABGML, which combines augmentation-aware prediction task and bootstrapping contrastive learning task in a meta-learning framework. ABGML uses random augmentation function instead fixed augmentation to increase the diversity of the pre-training process. Then we incorporate the augmentation-aware prediction task to facilitate contrastive learning to capture global semantics. Finally, the meta-learning framework further improves the generalization ability of ABGML. In terms of time and space efficiency, our method is superior to AFGRL on large datasets. Through experiments on multiple graphs on various downstream tasks, we empirically show that ABGML is superior to the state-of-the-art methods.

ACKNOWLEDGMENTS

This work was supported by NSFC under grants Nos. 62272469, 71971212 and U19B2024. W. Wang was partly supported by HKUST(GZ) G0101000028 and GZU-HKUST GZU22EG04.

REFERENCES

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 1597–1607.

[2] Austin Derrow-Pinion, Jennifer She, David Wong, Oliver Lange, Todd Hester, Luis Perez, Marc Nunkesser, Seongjae Lee, Xueying Guo, Brett Wiltshire, Peter W. Battaglia, Vishal Gupta, Ang Li, Zhongwen Xu, Alvaro Sanchez-Gonzalez, Yujia Li, and Petar Velickovic. 2021. ETA Prediction with Graph Neural Networks in Google Maps. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, Gianluca Demartini, Guido Zuccon, J. Shane Culpepper, Zi Huang, and Hanghang Tong (Eds.). ACM, 3767–3776. https://doi.org/10.1145/3459637.3481916

[3] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *ICML (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, 1126–1135.

[4] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS (JMLR Proceedings, Vol. 9)*. JMLR.org, 249–256.

[5] Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. In *Proceedings. 2005 IEEE international joint conference on neural networks*, Vol. 2. 729–734.

[6] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. 2020. Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[7] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. 2020. Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning. In *NeurIPS*.

[8] Kaveh Hassani and Amir Hosein Khas Ahmadi. 2020. Contrastive Multi-View Representation Learning on Graphs. In *ICML (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 4116–4126.

[9] Kaveh Hassani and Amir Hosein Khasahmadi. 2020. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*. PMLR, 4116–4126.

[10] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9729–9738.

[11] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. 2019. Learning deep representations by mutual information estimation and maximization. In *ICLR*. OpenReview.net.

[12] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *NeurIPS*.

[13] Kexin Huang and Marinka Zitnik. 2020. Graph Meta Learning via Local Subgraphs. In *NeurIPS*.

[14] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR (Poster)*.

[15] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[16] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR (Poster)*. OpenReview.net.

[17] Namkyeong Lee, Junseok Lee, and Chanyoung Park. 2021. Augmentation-free self-supervised learning on graphs. *arXiv preprint arXiv:2112.02472* (2021).

[18] Yuanze Lin, Xun Guo, and Yan Lu. 2021. Self-Supervised Video Representation Learning with Meta-Contrastive Network. In *ICCV*. IEEE, 8219–8229.

[19] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. 2020. Graph Representation Learning via Graphical Mutual Information Maximization. In *WWW*. ACM / IW3C2, 259–270.

[20] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *KDD*. ACM, 701–710.

[21] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In *KDD*. ACM, 1150–1160.

[22] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences* 63, 10 (2020), 1872–1897.

[23] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2019. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903* (2019).

[24] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, and Michal Valko. 2022. Large-Scale Representation Learning on Graphs via Bootstrapping. In *International Conference on Learning Representations*.

[25] Puja Trivedi, Ekdeep Singh Lubana, Yujun Yan, Yaoqing Yang, and Danai Koutra. 2022. Augmentations in Graph Contrastive Learning: Current Methodological Flaws & Towards Better Practices. In *WWW*. ACM, 1538–1549.

[26] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep Graph Infomax. In *ICLR (Poster)*. OpenReview.net.

[27] Hao Wang, Tong Xu, Qi Liu, Defu Lian, Enhong Chen, Dongfang Du, Han Wu, and Wen Su. 2019. MCNE: an end-to-end framework for learning multiple conditional network representations of social network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1064–1072.

[28] Ning Wang, Minnan Luo, Kaize Ding, Lingling Zhang, Jundong Li, and Qinghua Zheng. 2020. Graph few-shot learning with attribute matching. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1545–1554.

[29] Yaochen Xie, Zhao Xu, Zhengyang Wang, and Shuiwang Ji. 2021. Self-Supervised Learning of Graph Neural Networks: A Unified Review. *CoRR* abs/2102.10757 (2021).

[30] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems* 33 (2020), 5812–5823.

[31] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131* (2020).

[32] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep Graph Contrastive Representation Learning. *CoRR* abs/2006.04131 (2020).

[33] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep Graph Contrastive Representation Learning. *CoRR* abs/2006.04131 (2020).

[34] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph Contrastive Learning with Adaptive Augmentation. In *WWW*. ACM / IW3C2, 2069–2080.

[35] Marinka Zitnik and Jure Leskovec. 2017. Predicting multicellular function through multi-layer tissue networks. *Bioinform.* 33, 14 (2017), i190–i198. https://doi.org/10.1093/bioinformatics/btx252

A DATASETS

We evaluated the performance of ABGML on node-level tasks, i.e., node classification, node clustering, and node similarity search. We conduct experiments on six widely used datasets, including Wiki-CS, Amazon-Computers, Amazon-Photo, Coauthor-CS, Coauthor-Physics, and obgn-arxiv. The detailed statistics are summarized in Table 1.

- **WikiCS** is a reference network constructed based on Wikipedia. Nodes correspond to articles about computer science, and edges are hyperlinks between articles. The nodes are labeled with ten classes, each class representing a branch of

Table 6: Statistics of datasets used in experiments.

Dataset	#Nodes	#Edges	#Features	#Classes
Wiki-Cs	11,701	216,123	300	10
Amazon-Computers	13,752	245,861	767	10
Amazon-Photo	7,650	119,081	745	8
Coauthor-CS	18,333	81,894	6,805	15
Coauthor-Physics	34,493	247,962	6,805	5
ogbn-arXiv	169,343	1,166,243	128	40

the field. Node features are an average of GloVE embeddings of all words in the article. We directly use 20 canonical train/valid/test splits in the dataset.

- **Amazon Computers & Amazon Photos** are two networks constructed from the Amazon co-purchase graph, in which nodes represent goods and edges represent pairs of goods frequently bought together. Goods are classified into 10 and 8 (Computers and Photos) classes based on the good category, and node features are a representation of the product's review. We randomly split the nodes into train/val/test with 1:1:8 proportion, and repeat 20 times.
- **Coauthor CS & Coauthor Physics** are two academic networks from Microsoft Academic Graph, with nodes representing authors and edges representing pairs of authors who have co-author relationships. Authors are classified into 15 and 5 (CS and Physics) classes based on the research field. Each node has a sparse bag-of-words feature based on the paper keywords of the author. We randomly split the nodes into train/val/test with 1:1:8 proportion, and repeat 20 times.
- **ogbn-arXiv** is a citation network, where nodes represent computer science papers on arXiv and edges also indicate coauthorship relationships. In our experiment, we symmetrize this graph as same as other methods' experiments for a fair comparison. Papers are classified into 40 classes based on arXiv subject area. The node features are computed as the average word embeddings of all words in the paper. The dataset is split based on a chronological ordering as a convention setting.

B THE TRAINING PROCESS OF THE ABGML

The overall algorithm is summarized in Algorithm 1.

Algorithm 1: Pre-training process of ABGML

Input: Augmentation pool: Γ; Split proportion of support set: α; Importance factor β; Learning rate η; Pre-trained model $f(\theta)$ parameterized by θ

Output: Model parameters θ

Initialize θ randomly;
while *not converged* **do**
  Randomly sample two augmentation function T_1 and T_2;
  Creating support and query set $\mathcal{S}_{\mathcal{G}}$ and $\mathcal{Q}_{\mathcal{G}}$ using α;
  Creating temporarily parameters $\theta' = \theta$;
  // Meta Train Step:
  Feed sampled support dataset $\mathcal{S}_{\mathcal{G}}$ into network;
  Evaluate $\mathcal{L}_{cl}$ by Eq.3;
  Evaluate $\mathcal{L}_{ap}$ by Eq.6;
  Compute $\mathcal{L}_{meta} = \beta * \mathcal{L}_{cl} + (1 - \beta) * \mathcal{L}_{ap}$;
  Update θ' by Eq.9;
  // Meta Test Step:
  Feed sampled support dataset $\mathcal{Q}_{\mathcal{G}}$ into network;
  Evaluate $\mathcal{L}_{cl}$ by Eq.3;
  Evaluate $\mathcal{L}_{ap}$ by Eq.6;
  Compute overall loss $\mathcal{L}_{meta} = \beta * \mathcal{L}_{cl} + (1 - \beta) * \mathcal{L}_{ap}$;
  Update θ by Eq.10;
end

C COMPARED METHODS

In this section, we explain methods that are compared with ABGML in the experiments.

- **AFGRL** is a agumentation-free method which without relying on manual augmentation techniques and negative samples.
- **BGRL** is a bootstrapping contrastive method which use fixed augmentations and alleviates the need for contrasting with negative examples.
- **GCA** is a variant of GRACE that has the same learning objective but trades off more expressive but expensive graph augmentations for better performance.
- **GRACE** is a traditional contrastive method inspired by SimCLR, which creates two augmented views of a graph by randomly pertubing nodes/edges and their features.
- **DGI** aims to learn node representations by maximizing the mutual information between the node and global summary vector of the graph.
- **GMI** is advanced version of DGI that learns node representations by leveraging more fine-grained information.
- **MVGRL** constructs views of a graph with diffusion kernel and subgraph sampling.

D IMPLEMENTATION DETAILS

As described in above section, we use GCN [16] encoders. The base encoder of ABGML is a GCN model followed by batch normalization and nonlinearity. Following BGRL [24], the predictor q_θ and the augmentation-aware predictor of ABGML is defined as a multi-layer perceptron (MLP) with batch normalization.

For GCA, BGRL, and AFGRL, we adopt the best hyperparameter specifications that are reported in their original paper. For GRACE, since the original paper [32] did not evaluate on the datasets used in our experiments, we follow the result that are reported in the AFGRL paper.

The augmentation pool is composed by a series of augmentation, each augmentation is a combination of random node feature masking and edge masking with different masking probabilities p_f and p_e. We set the size of augmentation pool $n_{pool} = 4$ across the whole experiment, the four expansion methods are: (p_{f1}, p_{e1}), (p_{f2}, p_{e2}), $(p_{f1}, 0)$, and $(0, p_{e1})$. For the fifth augmentation in the ablation experiment is $(p_{f1} + 0.1, p_{e2} + 0.1)$. For the probability adjustment in ablation experiments, we adjusted the corresponding two value, such as p_f for p_{f1} and p_{f2}.

We use Glorot initialization [4] the AdamW optimizer [14] with a base learning rate η and weight decay set to $1e-5$. The learning rate is annealed using a cosine schedule over the course of learning of n_{total} total steps with an initial warmup period of n_{warmup} steps. Hence, the learning rate at step i is computed as:

$$\eta_i \triangleq \begin{cases} \frac{i \times \eta_{\text{base}}}{n_{\text{warmup}}} & \text{if } i \leq n_{\text{warmup}} \\ \eta_{\text{base}} \times \left(1 + \cos \frac{(i - n_{\text{warmup}}) \times \pi}{n_{\text{total}} - n_{\text{warmup}}}\right) \times 0.5, & \\ & \text{if } n_{\text{warmup}} \leq i \leq n_{\text{total}} . \end{cases} \tag{12}$$

We fix n_{total} to be 4,000 total epochs and n_{warmup} to 400 warmup epochs.

The target network parameters ϕ are initialized randomly from the same distribution of the online parameters θ but with a different random seed. The decay parameter τ is also updated using a cosine schedule starting from an initial value of $\tau_{\text{base}} = 0.99$ and is computed as

$$\tau_i \triangleq 1 - \frac{(1 - \tau_{\text{base}})}{2} \times \left(\cos\left(\frac{i \times \pi}{n_{\text{total}}}\right) + 1\right) \tag{13}$$

These annealing schedules for both η and τ follow the procedure used by [7].

E EXPERIMENT ENVIRONMENT

All experiments are conducted on a Linux server with one GPU (GeForce RTX 3090) and CPU (Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz), and its operating system is Ubuntu 18.04.6 LTS. We implement the proposed ABGML with deep learning library PyTorch and PyTorch Geometric. The Python, PyTorch and Pytorch Geometric versions are 3.9.12, 1.11.0 and 2.0.4, respectively.