Core Knowledge Learning Framework for Graph

Bowen Zhang[1], Zhichao Huang[2], Guangning Xu[3], Xiaomao Fan[1], Mingyan Xiao[4], Genan Dai[1]*, Hu Huang[5]*

[1]Shenzhen Technology University
[2]Beijing Normal University, Zhuhai
[3]Hong Kong Baptist University
[4]California State Polytechnic University
[5]University of Science and Technology of China
zhang_bo_wen@foxmail.com, iceshzc@gmail.com, xuguangning@hkbu.edu.hk,
fanxiaomao@sztu.edu.cn, mxiao@cpp.edu, daigenan@sztu.edu.cn, huanghu@mail.ustc.edu.cn

Abstract

Graph classification is a pivotal challenge in machine learning, especially within the realm of graph-based data, given its importance in numerous real-world applications such as social network analysis, recommendation systems, and bioinformatics. Despite its significance, graph classification faces several hurdles, including adapting to diverse prediction tasks, training across multiple target domains, and handling small-sample prediction scenarios. Current methods often tackle these challenges individually, leading to fragmented solutions that lack a holistic approach to the overarching problem. In this paper, we propose an algorithm aimed at addressing the aforementioned challenges. By incorporating insights from various types of tasks, our method aims to enhance adaptability, scalability, and generalizability in graph classification. Motivated by the recognition that the underlying subgraph plays a crucial role in GNN prediction, while the remainder is task-irrelevant, we introduce the Core Knowledge Learning (CKL) framework for graph adaptation and scalability learning. CKL comprises several key modules, including the core subgraph knowledge submodule, graph domain adaptation module, and few-shot learning module for downstream tasks. Each module is tailored to tackle specific challenges in graph classification, such as domain shift, label inconsistencies, and data scarcity. By learning the core subgraph of the entire graph, we focus on the most pertinent features for task relevance. Consequently, our method offers benefits such as improved model performance, increased domain adaptability, and enhanced robustness to domain variations. Experimental results demonstrate significant enhancements achieved by our method compared to state-of-the-art approaches. Specifically, our method achieves notable improvements in accuracy and generalization across various datasets and evaluation metrics, underscoring its effectiveness in addressing the challenges of graph classification.

Introduction

Graphs have garnered considerable attention for their ability to represent structured and relational data across diverse fields, as noted in several studies (Xu et al. 2021; Wang et al. 2021a). Graph classification, a fundamental aspect of data analysis, focuses on predicting whole graph properties and has seen substantial research activity in recent years (Wang et al. 2021c; Ma et al. 2020; Yin et al. 2024). This research has practical implications in various applications such as determining the quantum mechanical properties of molecules, including mutagenicity and toxicity (Hao et al. 2020), and identifying the functions of chemical compounds (Kojima et al. 2020). Various graph classification methodologies have been developed, with the majority leveraging Graph Neural Networks (GNNs) to deliver strong performance (Kipf and Welling 2017; Yang et al. 2023, 2024). These methods typically utilize a neighbor-aware message passing mechanism coupled with a readout function to learn discriminative graph representations that effectively reflect the structural topology, thereby facilitating accurate classification.

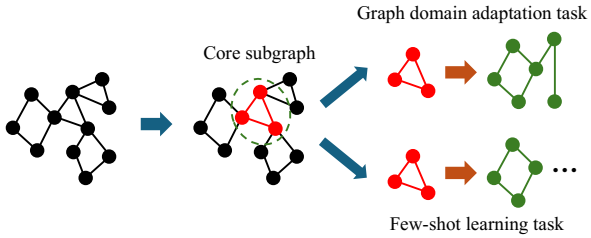


Figure 1: Illustration of the core knowledge learning framework. The framework extracts the core subgraph from the entire graph, which represents the fundamental structure necessary for task-relevant predictions. This core subgraph is then utilized for downstream tasks, including graph domain adaptation and few-shot learning tasks.

Despite its considerable potential, graph classification faces several significant challenges that hinder its broader adoption and effectiveness. These challenges can be broadly categorized into three main areas: (1) *Label Aspect*: Graph classification models are often designed for specific tasks, which limits their ability to transfer knowledge to different prediction tasks. This lack of task-agnostic adaptability reduces the models' versatility and applicability across various domains (Ju et al. 2024). Additionally, differences in

*Genan Dai and Hu Huang are joint corresponding authors.

labeling standards or the quality of annotations across domains can lead to inconsistencies in model predictions, thus affecting overall performance and generalizability. (2) *Domain Shift Aspect*: Graph classification models are generally trained on a single target domain, which diminishes their effectiveness when applied to diverse domains. Adapting these models to various target domains is a significant challenge due to variations in data distribution (Yin et al. 2023b), which can degrade performance. Domain shifts, marked by changes in the data distribution between the source and target domains, amplify this challenge. Effective adaptation mechanisms are essential to maintain model robustness and enhance generalization capabilities. (3) *Data Aspect*: Graph classification struggles with effectively handling small-sample prediction scenarios. The lack of sufficient labeled data in the source domain, combined with data scarcity in the target domain, presents considerable challenges to the adaptation process, potentially leading to poor generalization performance (Liu et al. 2018; Kim et al. 2019; Altae-Tran et al. 2017; Wang et al. 2021b). Additionally, imbalanced data distributions between domains compound these difficulties, calling for strategies to alleviate the effects of data scarcity and ensure fair and accurate model predictions across different domains.

In this paper, we present a new framework meticulously crafted to address the shortcomings of existing graph classification methods. Inspired by (Luo et al. 2020), the framework identifies and separates the essential underlying subgraph that significantly impacts GNN predictions from the task-irrelevant portions of the graph. Our approach leverages fundamental principles of graph-based learning to effectively address these issues. As depicted in Fig. 1, we utilize CKL to efficiently extract the core subgraph, which is then used for downstream applications such as graph domain adaptation and few-shot learning tasks. Specifically, our algorithm employs knowledge from these subgraphs to guide the adaptation process, enhancing our understanding of data distribution and improving the efficacy of domain transfer. Additionally, for the few-shot learning task, we implement a bi-level optimization strategy to determine how the extracted subgraph correlates with the labels, and adapt this understanding to various tasks. We adopt a comprehensive strategy, integrating multiple techniques to systematically tackle each challenge. Methods for extracting core subgraph features focus on capturing critical features that remain stable across different tasks, while task-specific adaptation layers are designed to fine-tune the model's parameters according to the specific requirements of the downstream tasks. This structured approach ensures our model's robustness and adaptability in handling diverse graph-based learning challenges. Our contribution is summarized as follows:

- We introduce a novel framework, named CKL, which learns the core subgraph instead of the whole graph representation learning.
- We employ the learned core subgraph for graph domain adaptation and few-shot learning tasks. By utilizing the core subgraph knowledge, the proposed CKL enhances the robustness and scalability of graph classification.
- We show the effectiveness of our proposed CKL with thorough experimentation, showing significant improvements in performance compared to leading methods across various datasets and evaluation metrics.

Related Work

Unsupervised Domain Adaptation Unsupervised domain adaptation (UDA) is a specialized area within machine learning focused on developing domain-invariant representations from a labeled source domain to be utilized in an unlabeled target domain. Traditional UDA methods typically involve aligning feature distributions between the source and target domains through techniques such as maximum mean discrepancy (MMD) (Long et al. 2015) or adversarial training (Ganin et al. 2016). Recent progress in UDA has been directed towards creating more efficient and scalable solutions to address domain shifts. A notable trend involves the adoption of deep learning strategies, including deep adversarial domain adaptation (DADA) (Tzeng et al. 2017) and domain-adaptive contrastive learning (DaCo) (Feng et al. 2023). These approaches use neural networks to derive domain-invariant features. Additionally, methods like self-training (Yarowsky 1995) and pseudo-labeling (Lee et al. 2013) have been implemented to exploit unlabeled data in the target domain, enhancing the domain adaptation process.

Graph Domain Adaptation Graph domain adaptation (GDA) applies the concepts of unsupervised domain adaptation to graph-structured data. Its objective is to transfer insights from a labeled source graph to an unlabeled target graph, addressing challenges such as domain shift and label scarcity (Ju et al. 2024). This issue is particularly pertinent in fields like social network analysis, where graphs depict social interactions, and bioinformatics, where graphs represent molecular structures. Existing methods mainly focus on how to transfer information from source graphs to unlabeled target graphs to learn effective node-level (Wu, Pan, and Zhu 2022; Zhu et al. 2021; Dai et al. 2022; Guo et al. 2022) and graph-level (Yin et al. 2023a; Yehudai et al. 2021; Ding et al. 2021; Yang et al. 2020) representation. Despite recent progress, the field continues to face obstacles such as misalignment in category distributions between source and target domains and the absence of scalable, effective algorithms for graph domain adaptation. Tackling these issues necessitates the development of robust, scalable algorithms capable of deriving domain-invariant representations from a limited amount of labeled data.

Few-shot Learning Few-shot Learning (FSL) aims to train a model capable of generalizing to new classes based on only a small number of examples from those classes, often just one or a few. Meta-learning is a key strategy for FSL, enhancing the model's ability to generalize robustly. This technique involves extracting meta-knowledge that is applicable across a range of meta-tasks, enabling the model to adapt to new, unseen meta-tasks after sufficient meta-training. Meta-learning approaches for FSL can be divided into two main types: metric-based and optimization-based methods. Metric-based methods, such as Matching Network (Vinyals et al. 2016) and ProtoNet (Snell, Swersky,

and Zemel 2017), focus on learning a metric to assess similarity between new instances and a few examples by mapping them into a metric space. For example, Matching Network achieves this by encoding the support and query sets separately to calculate similarities, while ProtoNet creates prototypes by averaging the support set representations for each class and classifies queries based on the Euclidean distance to these prototypes. On the other hand, optimization-based methods (Li et al. 2017; Ravi and Larochelle 2016) concentrate on learning how to adjust model parameters efficiently using gradients from a few examples. MAML (Finn, Abbeel, and Levine 2017), for instance, optimizes initial model parameters for quick fine-tuning with minimal examples. Another method employs an LSTM as a meta-learner to update parameters for specific tasks (Ravi and Larochelle 2016). Recently, FSL has been applied to graphs through meta-learning approaches (Ding et al. 2020; Liu et al. 2021), demonstrating success in attributed networks. However, extracting meta-knowledge for satisfactory performance typically requires numerous meta-training tasks with a diverse class set and corresponding nodes. Our framework addresses the challenge of limited diversity in meta-tasks, offering a potential improvement over existing methods.

Preliminary

Graph Neural Networks

Considering the graph $G = (\mathcal{V}, \mathcal{E})$, let $\boldsymbol{h}_v^{(k)}$ represent the embedding vector of node v at layer k. For each node $v \in \mathcal{V}$, we gather the embeddings of its neighbors from layer $k-1$. Subsequently, the embedding $\boldsymbol{h}_v^{(k)}$ is updated iteratively by merging v's previous layer embedding with the embeddings aggregated from its neighbors. The process is as follows:

$$\boldsymbol{h}_v^{(k)} = \mathcal{C}^{(k)}\left(\boldsymbol{h}_v^{(k-1)}, \mathcal{A}^{(k)}\left(\left\{\boldsymbol{h}_u^{(k-1)}\right\}_{u\in\mathcal{N}(v)}\right)\right), \quad (1)$$

where $\mathcal{N}(v)$ represents the neighbors of v. $\mathcal{A}^{(k)}$ and $\mathcal{C}^{(k)}$ represent the aggregation and combination operations at the k-th layer, respectively. At last, we summarize all node representations at the K-th layer with a readout function into the graph-level representation, which can be formulated as:

$$\boldsymbol{z} = F(G) = \mathrm{READOUT}\left(\left\{\boldsymbol{h}_v^{(K)}\right\}_{v\in\mathcal{V}}\right), \quad (2)$$

where $\boldsymbol{z}$ is the graph-level representation of G and $\boldsymbol{\theta_e}$ denotes the parameter of our GNN-based encoder. K denotes the number of the graph convolutional layers. The readout function can be implemented using different ways, such as the summarizing all nodes' representations (Xu et al. 2019) or using a virtual node (Li et al. 2016).

After obtaining the graph representation $\boldsymbol{z}$, we introduce a multi-layer perception (MLP) classifier $H(\cdot)$ to output label distributions for final classification: $\hat{\mathbf{p}} = H(\mathbf{z})$, where $\hat{\mathbf{p}} \in [0,1]^C$ and $\boldsymbol{\theta_c}$ denotes the parameters of the classifier.

Explainer of GNN

Following the methodology in (Luo et al. 2020), we partition the entire graph into two components, denoted as $G_{total} = G_{sub} + G_{rest}$. Here, G_{sub} represents the critical subgraph that significantly influences the predictions of the GNN, and is thus considered the explanatory graph. Conversely, G_{rest} includes the remaining edges that do not impact the GNN's predictions. To identify the essential subgraph G_{sub}, the approaches described in (Luo et al. 2020; Ying et al. 2019) focus on maximizing the mutual information between the labels and G_{sub}:

$$\max_{G_{sub}} MI(Y, G_{sub}) = H(Y) - H(Y|G = G_{sub}), \quad (3)$$

In this scenario, Y represents the prediction made by the GNN when G_{total} is used as input. Mutual information quantifies the likelihood of Y when only a specific segment of the graph, G_{sub}, is processed by the GNN. This concept derives from conventional forward propagation methods used to provide clear explanations of how the model functions. For example, the importance of an edge (i, j) is underscored when its removal leads to a significant change in the GNN's output, suggesting that this edge is critical and should be included in G_{sub}. If an edge's removal does not substantially affect the output, it is considered non-essential for the model's decision-making. Since $H(Y)$, the entropy of Y, is linked to the fixed parameters of the GNN during the explanation phase, the objective is to minimize the conditional entropy $H(Y|G = G_{sub})$.

Optimizing Eqn. 3 directly is impractical due to the 2^M potential candidates for G_{sub}, where M represents the total number of edges. To simplify this, we assume the graph follows the Gilbert random graph model (Gilbert 1959), in which the edges of the subgraph are considered independent. Here, $e_{ij} = 1$ indicates that the edge (i, j) is included, and 0 indicates it is not. Under this model, the probability of any graph configuration can be expressed as a product of individual probabilities:

$$P(G) = \Pi_{(i,j)\in\mathcal{V}} P(e_{ij}). \quad (4)$$

Assuming the distribution of edge e_{ij} follows the Bernoulli distribution: $e_{ij} \sim Bern(\theta_{ij})$. Then the Eqn. 4 can be rewrite as:

$$\begin{aligned}\min_{G_{sub}} H(Y|G = G_{sub}) &= \min_{G_{sub}} \mathbb{E}_{G_{sub}}[H(Y|G = G_{sub})] \\ &\approx \min_{\Theta} \mathbb{E}_{G_{sub}\sim q(\Theta)}[H(Y|G = G_{sub})],\end{aligned}$$

where $q(\Theta)$ is the distribution of the core subgraph.

Methodology

The purpose of learning core knowledge is to learn to determine the most essential subset of sample features. Focusing on the graph field, our purpose is to learn the subgraph structure that can represent the entire graph, and use the subgraph to replace the calculation of the entire graph. Motivated by (Ma et al. 2019), where the real-life graphs are with underlying structures, we develop the core knowledge learning (CKL) module to learn the core subgraph of graphs and utilize the knowledge for the downstream tasks learning.

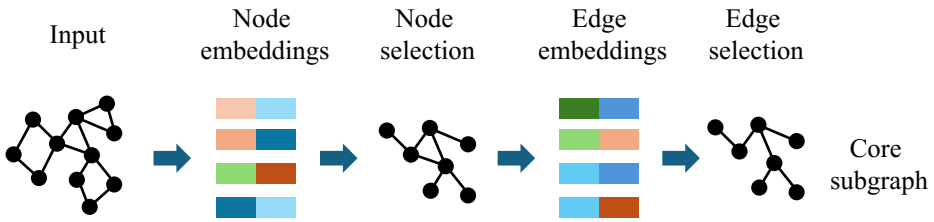


Figure 2: An overview of the proposed CKL. CKL first utilizes the node embeddings for node selection, and then cooperates the edge embeddings for edge selection to obtain the core subgraph.

Core Knowledge Learning

To learn the core knowledge of a graph, we need to determine the important nodes subset from the whole graph. Traditional graph explainer methods (Ma et al. 2019; Luo et al. 2020) direct learn the edges sampling process to determine the explainable subgraph. However, they typically assume the distribution of edges as prior knowledge, which is difficult to satisfy in real scenarios. Thus, we propose the CKL module to learn the probability of edge and node selection. Specifically, given a graph $G = (A, \mathbf{X})$, we first obtain the node embeddings with l-layer GNNs, and then map node v_i, v_j and e_{ij} into the same feature space with multilayer perception (MLP):

$$\mathbf{H} = GNN(A, \mathbf{X}), \quad \mathbf{E}_n = MLP(\mathbf{H}), \quad \mathbf{E}_e = MLP(\mathbf{E}),$$

where E denotes the features of edges.

Node selection. We first calculate the node sampling probability p_v with the Sigmoid function to map the node embeddings into $[0, 1]$:

$$p_v = Sigmoid(\mathbf{E}_{n_v}\mathbf{W}_v), \tag{5}$$

where $\mathbf{W}_v \in \mathbb{R}^{d\times 1}$ and d is the dimension of $\mathbf{E}_{n_v}$. A larger probability of node sampling will lead to a higher probability of node mask with $m_v = 1$, indicating the corresponding node v is important for the core knowledge learning. However, the node sampling process is non-differentiable (Yu, Liang, and He 2023), we relax m_v with Gumbel-softmax (Gal, Hron, and Kendall 2017; Jang, Gu, and Poole 2016):

$$m_v = Sigmoid(\frac{1}{t}\log\frac{p_v}{1-p_v} + \log\frac{u}{1-u}), \tag{6}$$

where t is the temperature parameter and $u \sim Uniform(0, 1)$.

Edge selection. After obtaining the probability of node sampling, we further evaluate the probability of edges corresponding to the sampled nodes. Specifically, we concat the embeddings of edge e_{ij} and the connected nodes n_i, n_j, and calculate the edge mask probability $m_{e_{ij}}$ with:

$$\mathbf{E}_{fusion} = Cat(\mathbf{E}_{n_i}, \mathbf{E}_{n_j}, \mathbf{E}_{e_{ij}}),$$
$$m_{e_{ij}} = Sigmod(\mathbf{E}_{fusion}\mathbf{W}_e),$$

where Cat denotes the concat operation, $\mathbf{W}_e \in \mathbb{R}^{k\times 1}$, k is the dimension of $\mathbf{E}_{fusion}$.

With the node and edge selection process, we mask the whole graph G_{total} to obtain the core graph G_{sub}. Finally,

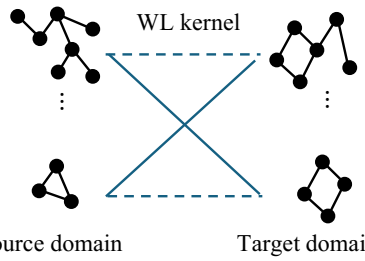


Figure 3: The core subgraph in the graph domain adaptation task. CKL employs a kernel function to assess the similarity between source and target subgraphs, and assigning labels to the target graphs based on the most similar source graph.

we follow (Ying et al. 2019) to modify the conditional entropy with cross-entropy $H(Y, Y_{sub})$, where Y_{sub} is the prediction of the GNN model with G_{sub} as the input:

$$\min_{\Theta} \mathbb{E}_{G_{sub}\sim q(\Theta)}[H(Y|G = G_{sub})]. \tag{7}$$

where $q(\Theta)$ is the distribution of the core subgraph.

For efficient optimization of Eqn. 7, we simplify the conditional entropy with cross-entropy $H(Y|Y_{sub})$, where Y_{sub} is the output of subgraph G_{sub}. With the simplification, we optimize Eqn. 7 with Monte Carlo approximation:

$$\begin{aligned}
&\min_{\Theta}\mathbb{E}_{G_{sub}\sim q(\Theta)}[H(Y|Y_{sub})] \\
&\approx \min_{\Theta} -\frac{1}{K}\sum_{k=1}^{K}\sum_{c=1}^{C} P(Y=c)\log P(Y_{sub}=c) \\
&= \min_{\Theta} -\frac{1}{K}\sum_{k=1}^{K}\sum_{c=1}^{C} P(Y=c|G=G_{total}) \\
&\quad \log P(Y=c|G=G_{sub}^k),
\end{aligned} \tag{8}$$

where K is the number of sampled subgraphs, C is the number of labels, and G_{sub}^k denotes the k-th sampled subgraph.

Graph Domain Adaptation Learning

Problem setup. Denote a graph as $G = (V, E, \mathbf{X})$ with the node set V, the edge set E, and the node attribute matrix $\mathbf{X} \in \mathbb{R}^{|V|\times F}$ with F denotes the attribute dimension. The labeled source domain is denoted as $\mathcal{D}^s = \{(G_i^s, y_i^s)\}_{i=1}^{N_s}$, where y_i^s denotes the labels of G_i^s. The unlabeled target domain is $\mathcal{D}^t = \{G_j^t\}_{j=1}^{N_t}$, where N^s and N^t denote the number of source graphs and target graphs. Both domains share

the same label space $\mathcal{Y}$, but have different distributions in the graph space. Our objective is to train a model using both labeled source graphs and unlabeled target graphs to achieve superior performance in the target domain.

The extracted core subgraph is the underlying subgraph that makes important contribution to GNN's prediction and remaining is task-irrelevant part. Therefore, in the graph domain adaptation task, we measure the similarity of the source domain core subgraph and target domain core subgraph, ignoring the domain shift. Given two sampled subgraphs from source domain $G_i^S = (V_i^S, E_i^S)$ and target domain $G_j^T(V_j^T, E_j^T)$, graph kernels calculate their similarity by comparing their substructure using a kernel function. In formulation,

$$K\left(G_{sub}^S, G_{sub}^T\right) = \sum_{v_1 \in V_i^S} \sum_{v_2 \in V_j^T} \kappa\left(l_{G_{sub}^S}(v_1), l_{G_{sub}^T}(v_2)\right) \tag{9}$$

where $l_{G_{sub}^S}(v_1)$ represents the local substructure centered at node v_1 and $\kappa(\cdot,\cdot)$ is a pre-defined similarity measurement. We omit $l_G(\cdot)$ and leave $\kappa(u_1, u_2)$ in Eq. 9 for simplicity. In our implementation, we utilize the Weisfeiler-Lehmah (WL) subtree kernel for the comparison of source and target core subgraph.

Weisfeiler-Lehmah (WL) Subtree Kernels. WL subtree kernels compare all subtree patterns with limited depth rooted at every node. Given the maximum depth l, we have:

$$\begin{aligned} K_{\text{subtree}}^{(i)}\left(G_1^S, G_2^T\right) &= \sum_{v_1 \in V_1^S} \sum_{v_2 \in V_2^T} \kappa_{\text{subtree}}^{(i)}\left(u_1, u_2\right) \\ K_{WL}\left(G_1^S, G_2^T\right) &= \sum_{i=0}^{l} K_{\text{subtree}}^{(i)}\left(G_1^S, G_2^T\right) \end{aligned} \tag{10}$$

where $\kappa_{\text{subtree}}^{(i)}(u_1, u_2)$ is derived by counting matched subtree pairs of depth i rooted at node u_1 and u_2, respectively. Considering the number of nodes of core subgraph is limited, we utilize the whole subgraph for the WL kernel calculation.

For each target domain core subgraph, we first calculate the most similar source domain subgraph and assign the same label to the target graph:

$$Y_j^T = Y_i^S, \quad s.t. \max_{i,j} K(G_j^T, G_i^S). \tag{11}$$

In this way, we avoid complex domain alignment operations and only align core subgraphs to achieve effective graph domain transfer learning.

Few shot learning

Problem setup. The target is to learn a predictor from a set of few-shot molecular property prediction tasks $\{\mathcal{T}_\tau\}_{\tau=1}^{N_t}$ and generalize to predict new properties given a few labeled molecules. The τ-th task $\mathcal{T}_\tau$ predicts whether a molecule $x_{\tau,i}$ with index i is active $(y_{\tau,i} = 1)$ or inactive $(y_{\tau,i} = 0)$ on a target property, provided with a small number of K labeled samples per class. This $\mathcal{T}_\tau$ is then formulated as a 2-way K-shot classification task with a support set $\mathcal{S}_\tau = \{(x_{\tau,i}, y_{\tau,i})\}_{i=1}^{2K}$ containing the $2K$ labeled samples and a query set $\mathcal{Q}_\tau = \{(x_{\tau,j}, y_{\tau,j}\}_{j=1}^{N_\tau^q}$ containing N_τ^q unlabeled samples to be classified.

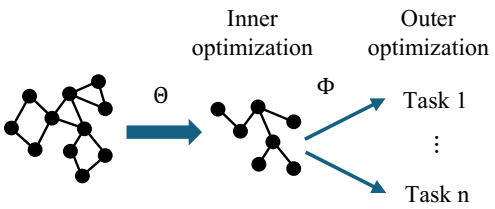


Figure 4: The core subgraph in the few-shot learning task. CKL employs the bi-level method to optimize the core subgraph learning and multi-task prediction.

In order to achieve the goal of few shot learning on multiple tasks, we utilize the learned core subgraph as the input of GNN for prediction, i.e.,

$$\hat{Y} = GNN_\Phi\left(G_{sub}(\Theta)\right), \tag{12}$$

where Φ is the parameters of task relevant embedding function and classifier and Θ denotes the collection of parameters of CKL molecular. The training loss $\mathcal{L}(\mathcal{S}_\tau, f_{\theta,\Phi})$ evaluated on $\mathcal{S}_\tau$ follows:

$$\mathcal{L}(\mathcal{S}_\tau, f_{\Theta,\Phi}) = \sum_{(x_{\tau,i} y_{\tau,i}) \in \mathcal{S}_\tau} -\mathbf{y}_{\tau,i}^\top log(\hat{\mathbf{y}}_{\tau,i}), \tag{13}$$

where $\mathbf{y}_{\tau,i} \in \mathbb{R}^2$ is a one-hot ground-truth. Observing that the Eqn. 13 contains two distinct parameters Θ and Φ, we further use bi-level optimization methods to optimize them simultaneously:

$$\begin{aligned} \min_\Phi F(\Theta^\star) &= \sum \mathcal{L}_{outer}\left(f_{\Theta^\star,\Phi}(A, X, \mathcal{S}_\tau)\right), \\ s.t. \Theta^\star &= \sum \mathcal{L}_{inner}\left(f_{\Theta,\Phi}(A, X, \mathcal{S}_\tau)\right), \end{aligned} \tag{14}$$

where $\mathcal{L}_{inner}$ is the loss function in Eqn. 8 and $\mathcal{L}_{outer}$ is the loss function of Eqn, 13.

Inner optimization. We first optimize the parameter Φ with a gradient descent based optimizer by fixing Θ,

$$\Theta_t = \Theta_{t-1} - \alpha \nabla_\Theta \mathcal{L}_{inner}(\mathcal{S}_\tau, f_{\Theta,\Phi}), \tag{15}$$

where α is the learning rate.

Outer optimization. Following (Finn, Abbeel, and Levine 2017), we employ the gradient-based meta-learning strategy and initialize Φ with set of meta-training tasks $\{\mathcal{T}_\tau\}_{\tau=1}^{N_t}$, which acts as the anchor of each task $\mathcal{T}_\tau$. Specifically, we fix parameter Θ and optimize Φ as Φ_τ on $\mathcal{S}_\tau$ in each $\mathcal{T}_\tau$ during the outer optimization period. Φ_τ is obtained by taking a few gradient descent updates:

$$\nabla_\Phi \mathcal{L}(\mathcal{S}_\tau, f_{\Theta,\Phi}) = \partial_\Theta \mathcal{L}_{outer} \nabla_\Phi \Theta(\Phi) + \partial_\Phi \mathcal{L}_{outer}. \tag{16}$$

We follow (Yin and Luo 2022) to optimize the parameter Φ with:

$$\Phi = \Phi - \beta \nabla_\Phi \mathcal{L}_{outer}(f_{\Theta^\star,\Phi}(A, X, \mathcal{S}_\tau)). \tag{17}$$

The details of updating Θ and Φ are shown in Algorithm 1. The core knowledge learning is shown in lines 2-3, the graph domain adaptation task is shown in lines 6-9, and the few-shot learning is shown in lines 12-22.

Methods	M0→M1	M1→M0	M0→M2	M2→M0	M0→M3	M3→M0	M1→M2	M2→M1	M1→M3	M3→M1	M2→M3	M3→M2	Avg.
WL subtree	74.9	74.8	67.3	69.9	57.8	57.9	73.7	80.2	60.0	57.9	70.2	73.1	68.1
GCN	73.0±1.7	68.7±1.5	66.8±3.5	69.2±0.9	53.9±3.4	53.4±2.7	69.3±0.8	74.0±1.1	55.1±1.3	42.6±1.9	55.5±3.5	57.9±2.9	61.6
GIN	74.1±1.8	73.4±3.4	65.4±1.5	70.4±2.9	58.9±2.7	61.2±1.1	73.2±3.8	77.7±3.0	63.1±3.7	63.9±2.4	67.4±2.3	73.2±1.9	68.5
GMT	69.0±4.0	67.4±3.8	60.3±4.2	66.5±3.8	54.9±1.6	54.8±3.6	65.6±4.2	70.4±3.2	64.0±2.3	56.8±4.3	64.7±1.5	61.1±3.5	63.0
CIN	68.5±2.1	65.1±2.6	65.4±1.3	63.6±2.8	57.3±3.4	59.0±3.1	59.3±1.5	68.3±1.3	58.1±2.4	71.1±3.1	60.7±1.7	61.7±2.4	63.2
CDAN	74.2±0.3	73.7±0.5	68.8±0.2	71.8±0.4	59.9±2.0	58.6±1.9	70.7±1.4	74.3±0.3	59.2±1.2	69.0±1.6	60.0±1.2	62.7±1.3	66.9
ToAlign	75.5±1.9	67.1±3.8	68.1±1.5	63.3±2.7	55.6±1.2	67.3±4.3	69.4±3.3	77.0±1.2	57.6±1.6	74.9±2.4	59.0±3.3	64.6±3.4	66.6
MetaAlign	74.5±0.9	73.8±0.6	69.4±1.2	72.6±1.3	59.8±1.8	70.7±2.7	72.0±0.5	75.6±0.6	62.4±2.1	72.3±1.9	62.2±1.1	72.0±1.2	69.7
DEAL	76.3±0.2	72.4±0.7	68.8±1.0	72.5±0.7	57.6±0.6	67.6±1.9	77.4±0.6	80.0±0.7	64.9±0.7	72.8±1.4	70.3±0.3	76.2±1.3	71.4
CoCo	77.5±0.4	75.7±1.3	68.3±3.7	74.9±0.5	65.1±2.1	74.0±0.4	76.9±0.6	77.4±3.4	66.4±1.5	71.2±2.7	62.8±4.2	77.1±0.6	72.2
CKL	**78.6±0.8**	**76.8±1.3**	**73.9±1.7**	**75.4±1.1**	**68.2±2.0**	**75.3±0.9**	**78.5±0.7**	**81.3±1.4**	**67.9±1.2**	**75.2±2.1**	**69.4±1.3**	**78.4±1.6**	**74.9**

Table 1: The classification results (in %) on Mutagenicity under edge density domain shift (source→target). M0, M1, M2, and M3 denote the sub-datasets partitioned with edge density. **Bold** results indicate the best performance.

Experiment

Experimental Settings

Datasets. For the graph domain adaptation task, we utilize 9 graph classification datasets for evaluation, i.e., Mutagenicity (M) (Kazius, McGuire, and Bursi 2005), Tox21_AhR, FRANKENSTEIN (F) (Orsini, Frasconi, and De Raedt 2015), and PROTEINS (Dobson and Doig 2003) (including PROTEINS (P) and DD (D)), COX2 (Sutherland, O'brien, and Weaver 2003) (including COX2 (C) and COX2_MD (CM)), BZR (Sutherland, O'brien, and Weaver 2003) (including BZR (B) and BZR_MD (BM)) obtained from TUDataset (Morris et al. 2020). The details statistics are presented in Table 3. Additionally, we follow (Yin et al. 2023a) and partition M, T, F datasets into four sub-datasets based on edge density. For the few-shot learning task, we evaluate the experiments on widely used few-shot molecular property prediction, and the details are introduced in Table 4.

Baselines. For the graph domain adaptation task, we compare our CKL with different baselines: WL subtree (Shervashidze et al. 2011) GCN (Kipf and Welling 2017), GIN (Xu et al. 2019), GMT (Baek, Kang, and Hwang 2021), CIN (Bodnar et al. 2021), CDAN (Long et al. 2018), ToAlign (Wei et al. 2021b), MetaAlign (Wei et al. 2021a), DEAL (Yin et al. 2022) and CoCo (Yin et al. 2023a). For the few-shot learning, we compare CKL with Siamese (Koch et al. 2015), ProtoNet (Snell, Swersky, and Zemel 2017), MAML (Finn, Abbeel, and Levine 2017), TPN (Liu et al. 2018), EGNN (Kim et al. 2019), IterRefLSTM (Altae-Tran et al. 2017) and RAP (Wang et al. 2021b).

Implementation Details. In our CKL, we employ GIN (Xu et al. 2019) as the backbone of feature extraction. For the graph domain adaptation task, we utilize one of the sub-datasets as source data and the remaining as the target data for performance comparison. We set the hidden size to 128 and the learning rate to 0.001 as default. We report the classification accuracy in the experiments. For the few-shot learning task, we use RDKit (Landrum et al. 2013) to obtain the molecular graphs, node and edge features. We use the GIN (Xu et al. 2019) as the backbone for feature extraction. We calculate the mean and standard deviations of ROC-AUC scores on each task by running ten times experiments.

Performance on Different Domains

Tables 5 to 7 present the comparative results of CKL alongside other benchmark methods. Analyzing these results, we observe several key insights:

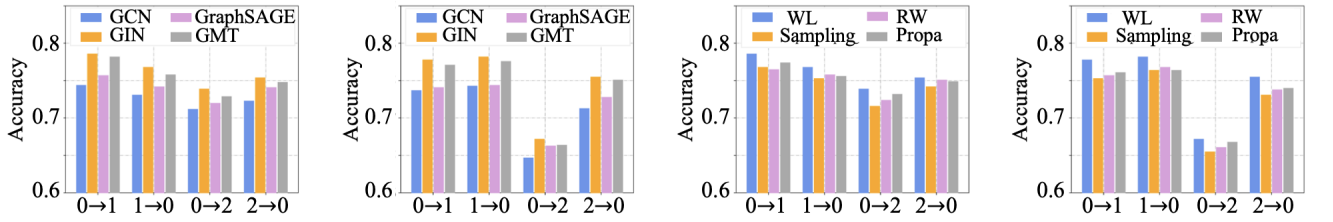
- Superiority of Domain Adaptation Methods in Graphs: Domain adaptation strategies tailored for graphs consistently outperform traditional kernel and GNN-based methods. This suggests that conventional graph methodologies may struggle with adaptability across varying domains due to their limited expressive power. Therefore, the development of domain-invariant techniques is critical for the advancement of Graph Domain Adaptation (GDA). These domain-invariant methods prove essential not only in maintaining performance across diverse datasets but also in facilitating the integration of graphs from disparate sources without loss of fidelity.
- Robust Performance of GDA Techniques: Methods implemented in GDA demonstrate robust performance, notably surpassing traditional domain adaptation strategies. The success of these methods can be attributed to their ability to handle the inherent complexities in graph data. Achieving high-quality graph representations is a complex task, exacerbated by the structural and feature diversity within the graphs. This complexity renders traditional domain adaptation strategies less effective, thus highlighting the specialized nature and effectiveness of graph-specific adaptation methods.
- Efficiency of CKL: The proposed CKL method outshines other competing methods, showcasing its efficiency in core knowledge learning. This efficiency is largely due to CKL's focus on critical components essential for accurate predictions. Specifically, CKL excels by concentrating on learning and enhancing the most relevant parts of the graph for the task at hand, while effectively ignoring or minimizing attention to the less relevant subgraphs. This approach not only boosts performance but also enhances the model's ability to generalize across different tasks by reducing the noise associated with irrelevant data.

Performance on Few-shot Learning

Table 2 details the performance comparisons between CKL and a range of baseline methods in graph-based molecular

Methods	Tox21		SIDER		MUV		ToxCast	
	10-shot	1-shot	10-shot	1-shot	10-shot	1-shot	10-shot	1-shot
Siamese	80.4±0.4	65.0±1.6	71.1±4.3	51.4±3.3	60.0±5.1	50.0±0.2	-	-
ProtoNet	75.0±0.3	65.6±1.7	64.5±0.9	57.5±2.3	65.9±4.1	58.3±3.2	63.7±1.3	56.4±1.5
MAML	80.2±0.2	75.7±0.5	70.4±0.8	67.8±1.1	63.9±2.3	60.5±3.1	66.8±0.9	66.0±5.0
TPN	76.1±0.2	60.2±1.2	67.8±1.0	62.9±1.4	65.2±5.8	50.0±0.5	62.7±1.5	50.0±0.1
EGNN	81.2±0.2	79.4±0.2	72.9±0.7	70.8±1.0	65.2±2.1	62.2±1.8	63.7±1.6	61.0±1.9
IterRefLSTM	81.1±0.2	81.0±0.1	69.6±0.3	71.7±0.1	49.6±5.1	48.5±3.1	-	-
PAR	82.1±0.1	80.5±0.1	74.7±0.3	**71.9±0.5**	66.5±2.1	64.1±1.2	69.7±1.6	67.3±2.9
CKL	**82.3±0.3**	**81.4±0.4**	**75.3±0.5**	71.7±0.7	**67.1±2.3**	**64.4±2.2**	**70.2±1.5**	**68.1±1.6**

Table 2: ROC-AUC scores on benchmark molecular property prediction datasets. Bold results (according to the pairwise t-test with 95% confidence) indicate the best performance.



(a) Accuracy on Mutagenicity (b) Accuracy on FRANKENSTEIN (c) Accuracy on Mutagenicity (d) Accuracy on FRANKENSTEIN

Figure 5: The performance with different GNNs and kernels on different datasets. (a), (b) are the performance of different GNNs, (c), (d) are the performance of different graph kernels. WL denotes WL subtree, RW denotes Random Walk, and Propa denotes Propagation.

encoding tasks. For this analysis, we have omitted results pertaining to Siamese and IterRefLSTM, as their implementation details and the outcomes of their evaluations on the ToxCast dataset are unavailable. The table clearly illustrates that CKL consistently achieves superior performance over other methods that also employ graph-based molecular encoders designed from the ground up. In particular, CKL not only surpasses all compared baselines but does so with a notable margin; it demonstrates an average performance improvement of 1.62% over the highest-performing baseline, EGNN. The performance reveals that methods incorporating few-shot learning techniques to decode relation graphs, such as GNN, TPN, and EGNN, deliver enhanced results when contrasted with more traditional learning frameworks like ProtoNet and MAML. This observation underscores the efficacy of few-shot learning in complex graph analysis tasks, where the ability to rapidly adapt to new, limited data without extensive retraining provides a significant advantage. These few-shot learning methods leverage sophisticated algorithms to capture and utilize the intricate relationships and structural nuances present within the graph data, thereby yielding more accurate and reliable predictions.

Flexibility of CKL

For the GDA experiments, we use GIN as the backbone to extract the core subgraph feature. To show the flexibility of the proposed CKL, we replace the GIN with different GNN methods. In our implementation, we utilize GCN, GraphSage and GMT instead of GIN to show the flexibility of CKL. Additionally, we replace the WL subtree kernel with Graph Sampling, Random Walk and Propagation.

Figure 5 illustrates the comparative performance of several GNNs and graph kernels over four representative datasets. We have noted similar performance trends across additional datasets as well. The data indicate that among the various GNNs and graph kernels evaluated, GIN and the WL subtree kernel consistently stand out as the top performers in the majority of cases. The superior performance of both GIN and the WL subtree kernel is likely due to their exceptional capabilities in capturing complex graph structures and providing powerful node and graph-level representations. This consistent outperformance validates our selection of GIN and the WL subtree kernel as the primary methods for enhancing task performance. The choice is further justified by their ability to effectively handle the complexities of diverse datasets, making them highly suitable for robust graph analysis and domain adaptation tasks.

Conclusion

In this paper, we introduce a novel approach named CKL that focuses on learning the core subgraph knowledge necessary for downstream tasks. Recognizing the essential role of the underlying subgraph in GNN predictions, while considering the rest as task-irrelevant, we have developed a framework designed for GDA and scalability learning. CKL includes several key components: the core subgraph knowledge submodule, the GDA module, and the few-shot learning module, each aimed at addressing specific challenges in graph classification such as domain shifts, label inconsistencies, and data scarcity. Our experiments show that CKL significantly outperforms existing state-of-the-art methods, demonstrating notable advancements in performance.

Acknowledgments

This research is supported by National Nature science Foundation of China (No.62306184), Natural Science Foundation of Top Talent of SZTU (grant no. GDRC202320), the Research Promotion Project of Key Construction Discipline in Guangdong Province (2022ZDJS112) and Shenzhen Science and Technology 'Program No. RCBS20231211090548077."

References

Altae-Tran, H.; Ramsundar, B.; Pappu, A. S.; and Pande, V. 2017. Low data drug discovery with one-shot learning. *ACS central science*, 3(4): 283–293.

Baek, J.; Kang, M.; and Hwang, S. J. 2021. Accurate Learning of Graph Representations with Graph Multiset Pooling. In *ICLR*.

Bodnar, C.; Frasca, F.; Otter, N.; Wang, Y. G.; Liò, P.; Montufar, G. F.; and Bronstein, M. 2021. Weisfeiler and lehman go cellular: Cw networks. In *NeurIPS*, 2625–2640.

Dai, Q.; Wu, X.-M.; Xiao, J.; Shen, X.; and Wang, D. 2022. Graph transfer learning via adversarial domain adaptation with graph convolution. *TKDE*, 35(5): 4908–4922.

Ding, K.; Wang, J.; Li, J.; Shu, K.; Liu, C.; and Liu, H. 2020. Graph prototypical networks for few-shot learning on attributed networks. In *CIKM*, 295–304.

Ding, M.; Kong, K.; Chen, J.; Kirchenbauer, J.; Goldblum, M.; Wipf, D.; Huang, F.; and Goldstein, T. 2021. A closer look at distribution shifts and out-of-distribution generalization on graphs. In *NeurIPSW*.

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

Feng, K.; Li, C.; Zhang, X.; and Zhou, J. 2023. Towards Open Temporal Graph Neural Networks. *arXiv preprint arXiv:2303.15015*.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 1126–1135.

Gal, Y.; Hron, J.; and Kendall, A. 2017. Concrete dropout. In *NeurIPS*, volume 30.

Ganin, Y.; Ustinova, E.; Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; Marchand, M.; and Lempitsky, V. 2016. Domain-adversarial training of neural networks. *JMLR*, 17(1): 2096–2030.

Gilbert, E. N. 1959. Random graphs. *The Annals of Mathematical Statistics*, 30(4): 1141–1144.

Guo, G.; Wang, C.; Yan, B.; Lou, Y.; Feng, H.; Zhu, J.; Chen, J.; He, F.; and Yu, P. 2022. Learning Adaptive Node Embeddings across Graphs. *TKDE*.

Hao, Z.; Lu, C.; Huang, Z.; Wang, H.; Hu, Z.; Liu, Q.; Chen, E.; and Lee, C. 2020. ASGN: An active semi-supervised graph neural network for molecular property prediction. In *Proceedings of the International ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 731–752.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Ju, W.; Yi, S.; Wang, Y.; Xiao, Z.; Mao, Z.; Li, H.; Gu, Y.; Qin, Y.; Yin, N.; Wang, S.; et al. 2024. A survey of graph neural networks in real world: Imbalance, noise, privacy and ood challenges. *arXiv preprint arXiv:2403.04468*.

Kazius, J.; McGuire, R.; and Bursi, R. 2005. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of medicinal chemistry*, 48(1): 312–320.

Kim, J.; Kim, T.; Kim, S.; and Yoo, C. D. 2019. Edge-labeling graph neural network for few-shot learning. In *CVPR*, 11–20.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations*.

Koch, G.; Zemel, R.; Salakhutdinov, R.; et al. 2015. Siamese neural networks for one-shot image recognition. In *ICMLW*, volume 2. Lille.

Kojima, R.; Ishida, S.; Ohta, M.; Iwata, H.; Honma, T.; and Okuno, Y. 2020. kGCN: a graph-based deep learning framework for chemical structures. *Journal of Cheminformatics*, 12: 1–10.

Landrum, G.; et al. 2013. RDKit: A software suite for cheminformatics, computational chemistry, and predictive modeling. *Greg Landrum*, 8(31.10): 5281.

Lee, D.-H.; et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICMLW*, 896.

Li, Y.; Zemel, R.; Brockschmidt, M.; and Tarlow, D. 2016. Gated Graph Sequence Neural Networks. In *ICLR*.

Li, Z.; Zhou, F.; Chen, F.; and Li, H. 2017. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*.

Liu, Y.; Lee, J.; Park, M.; Kim, S.; Yang, E.; Hwang, S. J.; and Yang, Y. 2018. Learning to propagate labels: Transductive propagation network for few-shot learning. *arXiv preprint arXiv:1805.10002*.

Liu, Z.; Fang, Y.; Liu, C.; and Hoi, S. C. 2021. Relative and absolute location embedding for few-shot node classification on graph. In *AAAI*, volume 35, 4267–4275.

Long, M.; Cao, Y.; Wang, J.; and Jordan, M. 2015. Learning transferable features with deep adaptation networks. In *ICML*, 97–105.

Long, M.; Cao, Z.; Wang, J.; and Jordan, M. I. 2018. Conditional adversarial domain adaptation. In *NeurIPS*.

Luo, D.; Cheng, W.; Xu, D.; Yu, W.; Zong, B.; Chen, H.; and Zhang, X. 2020. Parameterized explainer for graph neural network. In *NeurIPS*, volume 33, 19620–19631.

Ma, J.; Tang, W.; Zhu, J.; and Mei, Q. 2019. A flexible generative framework for graph-based semi-supervised learning. In *NeurIPS*, volume 32.

Ma, N.; Bu, J.; Yang, J.; Zhang, Z.; Yao, C.; Yu, Z.; Zhou, S.; and Yan, X. 2020. Adaptive-step graph meta-learner for few-shot graph classification. In *CIKM*, 1055–1064.

Morris, C.; Kriege, N. M.; Bause, F.; Kersting, K.; Mutzel, P.; and Neumann, M. 2020. TUDataset: A collection of benchmark datasets for learning with graphs. In *Proceedings of the International Conference on Machine Learning Workshop*.

Orsini, F.; Frasconi, P.; and De Raedt, L. 2015. Graph invariant kernels. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 3756–3762.

Ravi, S.; and Larochelle, H. 2016. Optimization as a model for few-shot learning. In *ICLR*.

Shervashidze, N.; Schweitzer, P.; Van Leeuwen, E. J.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-lehman graph kernels. *JMLR*, 12(9).

Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. *NeurIPS*, 30.

Sutherland, J. J.; O'brien, L. A.; and Weaver, D. F. 2003. Spline-fitting with a genetic algorithm: A method for developing classification structure- activity relationships. *Journal of Chemical Information and Computer Sciences*.

Tzeng, E.; Hoffman, J.; Saenko, K.; and Darrell, T. 2017. Adversarial discriminative domain adaptation. In *CVPR*, 7167–7176.

Vinyals, O.; Blundell, C.; Lillicrap, T.; Wierstra, D.; et al. 2016. Matching networks for one shot learning. In *NeurIPS*, volume 29.

Wang, M.; Wang, S.; Yang, H.; Zhang, Z.; Chen, X.; and Qi, G. 2021a. Is Visual Context Really Helpful for Knowledge Graph? A Representation Learning Perspective. In *ACMMM*, 2735–2743.

Wang, Y.; Abuduweili, A.; Yao, Q.; and Dou, D. 2021b. Property-aware relation networks for few-shot molecular property prediction. *NeurIPS*, 34: 17441–17454.

Wang, Y.; Wang, W.; Liang, Y.; Cai, Y.; and Hooi, B. 2021c. Curgraph: Curriculum learning for graph classification. In *WWW*, 1238–1248.

Wei, G.; Lan, C.; Zeng, W.; and Chen, Z. 2021a. Metaalign: Coordinating domain alignment and classification for unsupervised domain adaptation. In *CVPR*, 16643–16653.

Wei, G.; Lan, C.; Zeng, W.; Zhang, Z.; and Chen, Z. 2021b. ToAlign: Task-Oriented Alignment for Unsupervised Domain Adaptation. In *NeurIPS*, 13834–13846.

Wu, M.; Pan, S.; and Zhu, X. 2022. Attraction and repulsion: Unsupervised domain adaptive graph contrastive learning network. *TETCI*, 6(5): 1079–1091.

Xu, F.; Wang, M.; Zhang, W.; Cheng, Y.; and Chu, W. 2021. Discrimination-Aware Mechanism for Fine-grained Representation Learning. In *CVPR*, 813–822.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How powerful are graph neural networks? In *Proceedings of the International Conference on Learning Representations*.

Yang, L.; Hu, W.; Xu, J.; Shi, R.; He, D.; Wang, C.; Cao, X.; Wang, Z.; Niu, B.; and Guo, Y. 2024. GAUSS: GrAph-customized Universal Self-Supervised Learning. In *WWW*, 582–593.

Yang, L.; Shi, R.; Zhang, Q.; Niu, B.; Wang, Z.; Cao, X.; and Wang, C. 2023. Self-supervised Graph Neural Networks via Low-Rank Decomposition. In *NeurIPS*.

Yang, X.; Deng, C.; Liu, T.; and Tao, D. 2020. Heterogeneous graph attention network for unsupervised multiple-target domain adaptation. *TPAMI*, 44(4): 1992–2003.

Yarowsky, D. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, 189–196.

Yehudai, G.; Fetaya, E.; Meirom, E.; Chechik, G.; and Maron, H. 2021. From local structures to size generalization in graph neural networks. In *ICML*, 11975–11986.

Yin, N.; and Luo, Z. 2022. Generic structure extraction with bi-level optimization for graph structure learning. *Entropy*, 24(9): 1228.

Yin, N.; Shen, L.; Li, B.; Wang, M.; Luo, X.; Chen, C.; Luo, Z.; and Hua, X.-S. 2022. DEAL: An Unsupervised Domain Adaptive Framework for Graph-level Classification. In *ACMMM*, 3470–3479.

Yin, N.; Shen, L.; Wang, M.; Lan, L.; Ma, Z.; Chen, C.; Hua, X.-S.; and Luo, X. 2023a. CoCo: A Coupled Contrastive Framework for Unsupervised Domain Adaptive Graph Classification. In *ICML*, 40040–40053.

Yin, N.; Wan, M.; Shen, L.; Patel, H. L.; Li, B.; Gu, B.; and Xiong, H. 2024. Continuous Spiking Graph Neural Networks. *arXiv preprint arXiv:2404.01897*.

Yin, N.; Wang, M.; Chen, Z.; Shen, L.; Xiong, H.; Gu, B.; and Luo, X. 2023b. DREAM: Dual Structured Exploration with Mixup for Open-set Graph Domain Adaption. In *ICLR*.

Ying, Z.; Bourgeois, D.; You, J.; Zitnik, M.; and Leskovec, J. 2019. Gnnexplainer: Generating explanations for graph neural networks. In *NeurIPS*, volume 32.

Yu, J.; Liang, J.; and He, R. 2023. Mind the label shift of augmentation-based graph ood generalization. In *CVPR*, 11620–11630.

Zhu, Q.; Yang, C.; Xu, Y.; Wang, H.; Zhang, C.; and Han, J. 2021. Transfer learning of graph neural networks with ego-graph information maximization. *NeurIPS*, 34: 1766–1779.