Structural Landmarking and Interaction Modelling: A "SLIM" Network for Graph Classification

Yaokang Zhu[1], Kai Zhang[1*], Jun Wang[1*], Haibin Ling[2], Jie Zhang[3], Hongyuan Zha[4]

[1]School of Computer Science and Technology, East China Normal University, Shanghai, China
[2]Stony Brook University , New York, USA
[3] Institute of Brain-Inspired Intelligence, Fudan University, Shanghai, China
[4] School of Data Science, Shenzhen Institute of Artificial Intelligence and Robotics for Society
The Chinese University of Hong Kong, Shenzhen, China
52184501026@stu.ecnu.edu.cn,{kzhang980, wongjun, haibin.ling, jzhang080}@gmail.com, zhahy@cuhk.edu.cn

Abstract

Graph neural networks are a promising architecture for learning and inference with graph-structured data. Yet, how to generate informative, fixed-dimensional graph-level features for graphs with varying size and topology can still be challenging. Typically, this is achieved through graph-pooling, which summarizes a graph by compressing all its nodes into a single vector after convolutional operations. Is such a "*collapsing-style*" graph-pooling the only choice for graph classification? From complex system's point of view, properties of a complex system arise largely from the interaction among its components. Therefore, we speculate that preserving the interacting relation between parts, instead of pooling them together, could benefit system-level prediction. To verify this, we propose *SLIM*, a graph neural network model for Structural Landmarking and Interaction Modelling. The main idea is to compute a set of end-to-end optimizable sub-structure landmarks, so that any input graph can be projected onto these (spatially) local structural representatives for a faithful, global characterization. By doing this, explicit interaction between component parts of a graph can be leveraged directly in generating useful graph-level representations despite significant topological variations. Encouraging results are observed on benchmark datasets for graph classification, demonstrating the value of interaction modelling in the design of graph neural networks.

Introduction

Complex systems are ubiquitous in natural and scientific disciplines, and how the relation between component parts gives rise to global behaviour of a system is a central research topic in many areas such as system biology (Camacho et al. 2018), neural science (Kriegeskorte 2015), and drug and material discoveries (Stokes et al. 2020; Schmidt et al. 2019). Recently, graph neural networks emerge as a promising architecture for representation learning on graphs – *the structural abstraction of a complex system*. State-of-the-art performances are observed in various graph mining tasks (Bronstein et al. 2017; Defferrard, Bresson, and Vandergheynst 2016; Hamilton, Ying, and Leskovec 2017; Xu et al. 2019; Velickovic et al. 2017; Morris et al. 2019; Wu et al. 2020; Jie et al. 2020; Zhang, Cui, and Zhu 2020). However, due to the non-Euclidean and relational nature, important challenges still exist using graph neural networks for *graph-level* predictive tasks, such as graph classification. For example, in order to generate a fixed-dimensional representation for graphs of arbitrary size, *graph-pooling* is typically adopted to summarize the information of a graph. Though detailed choice varies, majority of the graph-pooling methods end up with "squeezing" all the nodes of a graph into a single vector, blending structural details for feature (dimension) compatibility.

Is such a "collapsing-style" graph pooling the only choice for graph classification? Different perspectives may be taken. From complex system's view, properties of a complex system arise largely from the *interactions* among its components, a universal law underlying a great diversity of real-world systems (Hartwell et al. 1999; Debarsy et al. 2017; Cilliers 1998). For example, properties of molecular drugs depend on functional modules and how they organize with each other (Stokes et al. 2020). Epidemic spreading process in a social network is affected by its community structures (Pastor-Satorras et al. 2015). In these regards, mixing all the component parts of a system into one unit, as implemented through collapsing-style graph pooling, obscures sub-structure identities and their interactions. We speculate this could affect both system-level predictive performance, and model interpretability.

Motivated by this observation, we want to explore whether the explicit modelling of graph parts and their interactions can benefit graph classification, in contrast to pooling the parts together. This is an interesting inductive bias to the design of graph neural networks inspired by complex systems science. There are two big challenges involved.

- *How to manipulate the "parts" of a graph, or sub-structures*[1]? Sub-structures provide a intermediate scale for characterizing high-order information, and have drawn considerable interest in motif discovery (Milo et al. 2002; Alon 2007; Austin R. Benson 2016), graphlets (Shervashidze et al. 2009), graph kernels (Vishwanathan et al. 2010), and GNNs/graph-convolutions (Peng et al. 2020). However, due to the discrete, combinatorial nature, diver-

*Corresponding author.

[1]Sub-structure in this paper means a connected subgraph and will be used interchangably with it.

sified sub-structure instances are typically enumerated in a greedy manner, hence restricted to small motifs (4 or 5 nodes) with limited scope of structural variations. In addition, exact matching of sub-structures may not compensate for their similarity, and so the risk of overfitting may rise (Yanardag and Vishwanathan 2015).

- *How to explicitly model the interaction between component parts of a graph*? Considering that the number of nodes and their topological connections vary from graph to graph, and that a natural (or computationally convenient definition of) correspondence between non-isomorphic graphs hardly exists, it can be highly nontrivial to transform the interacting relation (i.e., adjacency matrix) of a graph into a fixed-dimensional representation. Therefore, despite the effort in exploiting high-order motifs/sub-structures to better contextualize and improve graph convolution, a collapsing-style graph pooling is still the mainstream choice in nowadays GNN architectures in order to bypass complicated topological varabilities.

In this paper, we propose a simple neural network called "Structural Landmarking and Interaction Modelling", or SLIM, to resolve the above two challenges and demonstrate the benefit of modelling the inherent interaction in graphs. As the name suggests, the key idea is to compute a set of structural landmarks to break the "curse of granularity" in handling discrete sub-structures. These spatially-localized, and end-to-end optimizable sub-structure representatives not only preserve intrinsic similarities in profiling/comparing sub-structures, but also serve as a common platform so that any input graph (and even unseen ones) can be projected onto the landmarks so that identities of graph parts and their interactions are captured explicitly as fixed-dimensional graph-level features. In other words, the sub-structure landmarking is the key to solving both of the above two challenges. We show that, by respecting the structural organization of graphs, SLIM is empirically attractive, physically more understandable and offers new insight in graph representation learning.

In the remainder of the paper, we will first introduce GNNs and pooling strategies, and discuss why the collapsing-style graph pooling dominated the literature. Then we discuss the proposed method, and relate it with a number of important topics including dictionary learning, graph isomorphism and graph kernels. Finally, experimental evaluations are reported and conclusions are made.

Related Work: GNNs and Graph Pooling

Graph neural networks (GNNs) (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017; Xu et al. 2019) for the task of graph classification typically involve two key steps, namely graph convolution and graph pooling.

The goal of graph convolution is to pass information among neighboring nodes in the general form of

$$h_v = \texttt{AGGREGATE}\left(\{h_u, u \in \mathcal{N}_v\}\right),$$

where $\mathcal{N}_v$ is the neighbors of v (Hamilton, Ying, and Leskovec 2017; Xu et al. 2019). The convolution aggregates information of the rooted sub-trees growing from each node, and more convolutional layers can capture lager sub-structures/sub-trees and may lead to improved discriminative power (Xu et al. 2019). Multiple layers may be even combined together via `CONCATENATE` function (Hamilton, Ying, and Leskovec 2017; Xu et al. 2019) for improved modelling.

The goal of graph pooling is to generate compact graph-level representations with compatible dimension. This is typically achieved by squeezing all the nodes in a graph $\mathcal{G}$ into a single vector, as

$$h_{\mathcal{G}} = \texttt{READOUT}\left(\{f(h_v), \forall v \in \mathcal{V}\}\right),$$

where $\mathcal{V}$ is the vertex set of $\mathcal{G}$, and f a transform on node features. The dimension of $h_{\mathcal{G}}$ usually equals that of individual node features h_v's, meaning that all the nodes in $\mathcal{V}$ collapse into a single one. Readout functions include max (Cangea et al. 2018), sum (Xu et al. 2019), deep-sets (Zaheer et al. 2017).

Recently, more sophisticated pooling algorithms are developed, which employ hierarchical (multiple) pooling layers to fully exploit the non-flat structure of complex real-world graphs. For example, in hierarchical differential pooling (Ying et al. 2018), a graph is reduced to a smaller size in each layer based on a projection matrix through an auxiliary GNN model. Hierarchical Graph Pooling with Structure Learning (Zhang et al. 2019) combines graph pooling with graph reconstruction. Eigen-Pooling uses sub-graph eigenvectors for progressive graph coarsening (Ma et al. 2019). Self-attention pooling (Lee, Lee, and Kang 2019) evaluates GCN-based self-attention scores for layer-wise node down-sampling. Graph U-Net (Gao and Ji 2019) performs down-sampling and up-sampling in an encoder-decoder setting. Sort-pooling (Zhang et al. 2018) rearranges the nodes into a linear chain, so that standard 1d-convolution and pooling are applied to sorted nodes.

While hierarchical pooling is naturally affected by local and hierarchical topology, the resultant graph-level feature is in the form of a single, aggregated node vector. How hierarchical pooling affects the performance of GNN is still being actively investigated (Mesquita, Souza, and Kaski 2020).

As can be noticed, almost all the current graph-pooling techniques adopt a "collapsing-style" integration, where features of all the nodes in a graph are ultimately reduced to a single vector, despite the choice of the pooling function, the number of pooling levels, and the pooling criteria.

What motivates the collapsing-style graph pooling? Apparently, it's the need of fixed-dimensional graph-level representations: whereas the number of nodes varies freely, the dimension of the node feature remains constant and so is much easier to handle than varying graph sizes. However, a profound reason behind the choice lies in the difficulty in performing profiling and identification of sub-structures. Sub-structures are building blocks of a graph: important relations like interaction or alignment are all defined on top of them, and they provide an intermediate scale with better contextualization than individual nodes. Unfortunately, current practices of sub-structure manipulation mainly focus on exact matching and greedy enumeration (Wernicke 2006; Milo et al. 2002). Such an over-delicate granularity basically requires remembering potentially an exponential amount of unique

sub-structures, making it infeasible to match and compare substructure instances; besides, exact matching also brings the risk of overfitting (Yanardag and Vishwanathan 2015). Therefore, pooling the graph parts (sub-structures) together into a single unit emerges as a popular choice, when lacking convenient global criterion to profile, represent, and compare (or align) highly diversified sub-structure identities.

One potential downside of this convenience (pooling everything together) is that it can mix the identity of graph sub-structures and their interactions. Therefore, although the pooling can still be injective (Xu et al. 2019), structural details of the graph may not be recovered, requiring nontrivial effort to trace the behaviour of subsequent classifiers back to local and meaningful parts of the graph for interpretation purposes. Besides, from complex system's view point, interaction between component parts can also be valuable in system-level predictive tasks. Therefore, we are interested in graph-pooling paradigms that are more topology-preserving and physically understandable (and so potentially more interpretable) by resolving the exact challenge behind collapsing-style pooling, i.e., sub-structure profiling/manipulation.

The Proposed Algorithm

Graphs are structural abstraction of complex systems, and so accurate graph classification should depend on how global properties of a system relate to its structure. Inspired by the fundamental law of complex systems that emphasizes interaction between parts (Debarsy et al. 2017; Cilliers 1998), we introduce "structural landmarking and interaction modelling", or SLIM, in the design of GNNs for graph classification.

Problem Setting. Given n labeled graphs $\{(\mathcal{G}_i, y_i)\}$'s for $i = 1, 2, ..., n$; each graph is defined on the node/edge set $\mathcal{G}_i = (\mathbf{V}_i, \mathbf{E}_i)$ with adjacency matrix $\mathbf{A}_i \in \mathbb{R}^{n_i \times n_i}$, where $n_i = |\mathbf{V}_i|$, and $y_i \in \{\pm 1\}$. The node attribute matrix for each graph $\mathcal{G}_i$ is denoted by $\mathbf{X}_i \in \mathbb{R}^{n_i \times c}$. Our goal is to train an inductive model to predict labels of the testing graphs.

The key idea of SLIM is to define the "parts" of graphs using a feasible granularity, and model their "interactions" explicitly. We illustrate in Figure 1 and summarize as follows.

1. Identification and Embedding of Graph Parts. The first step identifies constituent parts of graphs (local sub-structure), and embeds them in a metric space. In Figure 1, rooted subtrees around each node (shaded circles) are sub-structure instances, and embedded as points (colored dots). Treating sub-structures as continuous vectors instead of discrete objects preserves intrinsic similarity, generalizes to unseen examples, and facilitates subsequent landmarking.

2. Finding Landmarks of Graph Parts. The second step is aimed at computing a pre-defined number (K) of landmarks for embedded sub-structure instances. In Figure 1, red hexagons denote landmarks. By associating each substructure with its closest landmark, we can systematically define the "identity" of any sub-structure across different graphs using the K landmarks. This is a code-book that breaks the curse of granularity, and also serves as a common platform for interaction modelling.

3. Projecting Graphs onto the Landmarks. The third step projects graphs onto the common set of sub-structure landmarks. Each graph can then be described as pairwise connections among the K (groups of) structural landmarks. This "projective pooling" can well preserve topological details in the form of fixed dimensional interaction matrices ($\mathbb{R}^{K \times K}$) for subsequent classification.

Sub-structure Identification and Embedding

The goal of sub-structure embedding is to extract substructure instances and embed in a metric space. Ideal substructures should reflect the functional organization under proper scales, which may require complicated procedures and domain knowledge. In this paper, we choose the rooted subtrees (k-hop subgraphs) as in (Xu et al. 2019) because of its simplicity, so that each sub-structure will then be associated exactly (or centered around) with one node.

Here we consider representing each sub-structure by vector form. Since each sub-graph is associated with one node, the n_i sub-graphs extracted from $\mathcal{G}_i$ can be conveniently represented as

$$\mathbf{Z}_i = \mathbf{A}_i^{(k)} \mathbf{X}_i \tag{1}$$

where $\mathbf{A}_i^{(k)}$ is the k-th order adjacency matrix of the ith graph such that $\mathbf{A}_i^k(p, q) = 1$ if node p and q are *within* k-hops away in graph $\mathcal{G}_i$, and 0 otherwise. The j^{th} row of $\mathbf{Z}_i$ then summarizes the counts of the c types of nodes in the k-hop sub-graph centered around the jth node.

We can also refine it to model the node distribution in each of the k layers in the k-hop rooted sub-graph (including the center node), as

$$\mathbf{Z}_i = [\mathbf{X}_i \ \tilde{\mathbf{A}}_i^{(1)} \mathbf{X}_i \ \tilde{\mathbf{A}}_i^{(2)} \mathbf{X}_i \ ... \ \tilde{\mathbf{A}}_i^{(k)} \mathbf{X}_i], \tag{2}$$

where $\tilde{\mathbf{A}}_i^{(k)}$ specifies whether two nodes in $\mathcal{G}_i$ are *exactly* k-hops away, namely $\tilde{\mathbf{A}}_i^{(k)}(p, q) = 1$ if node p and q are *exactly* k-hops away in graph $\mathcal{G}_i$, and 0 otherwise. Considering that $\mathbf{A}_i^{(k)} = \mathbf{I} + \tilde{\mathbf{A}}_i^{(1)} + \tilde{\mathbf{A}}_i^{(2)} ... + \tilde{\mathbf{A}}_i^{(k)}$, (2) can be deemed as a layer-wise decomposition of (1). One may further apply a weighting factor on each layer (that decays with the depth of the layer), or compute a weighted mixture of different layers. We have used (2) in our experiments.

Next we embed sub-structure instances (rows of $\mathbf{Z}_i$'s) in a latent space to promote important relations between these substructures: (1) structural similarity: i.e., two sub-structures look very similar; and (2) topological proximity: i.e., two substructures frequently connect with each other in the training graphs. In these two scenarios, the embedding of the two substructures should be close to each other in the Euclidian space, namely the embedding should reflect (or, be smooth with) both structural similarities and relational interactions. We learn a simple, one-layer transform (with ℓ_2-regularization of model complexity) to promote the smoothness of embedding w.r.t. structural similarity, as

$$f(\mathbf{Z}_i) = \sigma\left(\mathbf{Z}_i \cdot \mathbf{T} + \mathbf{B}\right), \tag{3}$$

where $\mathbf{T}$ is transform matrix, $\mathbf{B}$ is bias matrix (a bias vector repeated n_i times row-wise) and $\sigma(\cdot)$ is the RELU function.

On the other hand, to maintain the smoothness of embedding w.r.t. the topological interaction, we borrow ideas from

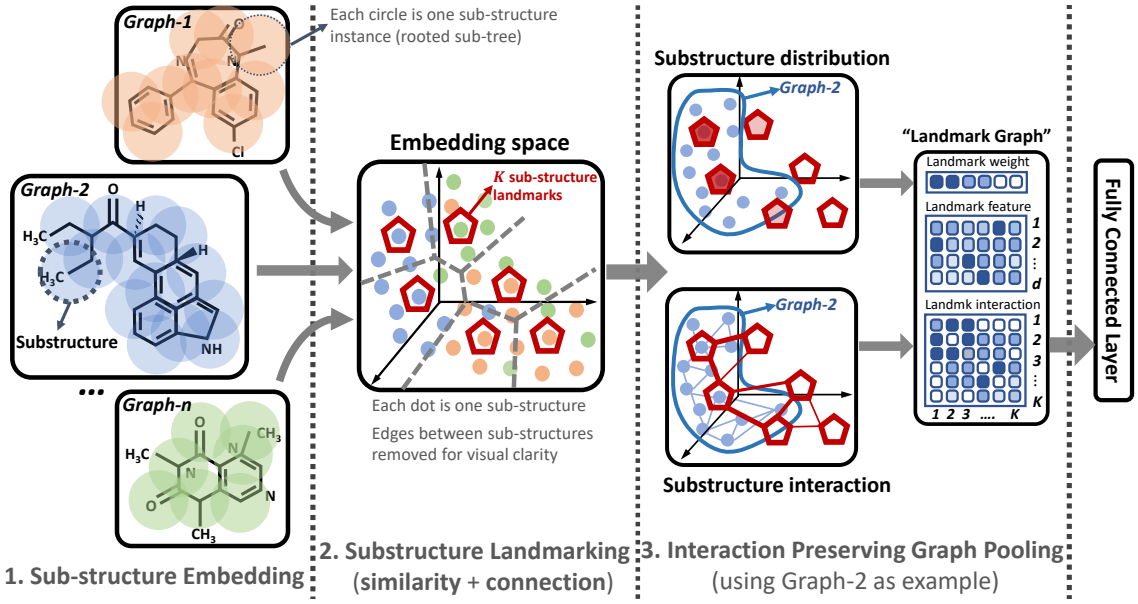


Figure 1: Three main steps of SLIM. (1) Sub-structure embedding: extract local sub-graphs and embed them in a metric space. (2) sub-structure landmarking: compute sub-structure representatives through unsupervised clustering across graphs. (3) Identity-preserving graph pooling: project each graph on the common set of sub-structure landmarks for final prediction.

node2vec (Grover and Leskovec 2016). Let $\mathbf{H}_i = f(\mathbf{Z}_i) \in \mathbb{R}^{n_i \times d'}$ be the embedding of the n_i sub-graph instances from $\mathcal{G}_i$. We maximize the log-likelihood of the co-occurrence of sub-structure instances in each graph as

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n_i} \sum_{l \in \mathcal{N}_j^i} \log \left(\frac{\exp\langle \mathbf{H}_i(j,:), \mathbf{H}_i(l,:) \rangle}{\sum_{l'} \exp\langle \mathbf{H}_i(j,:), \mathbf{H}_i(l',:) \rangle} \right). \quad (4)$$

Here $\mathbf{H}_i(j,:)$ is the j^{th} row of $\mathbf{H}_i$, $\langle , \rangle$ is inner product, and $\mathcal{N}_j^i$ are the neighbors of node i in graph $\mathcal{G}_i$. This is a parametric node2vec loss and can encourage strongly interconnecting sub-structures to be embedded close to each other.

Sub-structure Landmarking

The goal of structural landmarking is to identify a set of informative structural representatives in the latent space of embedded sub-structures. It then serves as a dictionary that breaks the curse of granularity in manipulating a potentially exponential amount of sub-structure instances.

Let $\mathbf{U} = \{\boldsymbol{\mu}_j, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_K\}$ be structural landmarks in the latent space of embedded sub-structures. To fully represent diverse sub-structures, each sub-graph instance should be faithfully approximated by the closest landmark. We use a soft assignment matrix $\mathbf{W}_i \in \mathbb{R}^{n_i \times k}$ for each graph $\mathcal{G}_i$, whose jk^{th} entry is the probability that the j^{th} sub-structure from $\mathcal{G}_i$ belongs to the k^{th} landmark $\boldsymbol{\mu}_k$. Inspired by the deep embedding clustering (Junyuan, Ross, and Ali 2016), $\mathbf{W}_i$ is parameterized by a Student's t-distribution

$$\mathbf{W}_i(j,k) = \frac{(1 + \|\mathbf{H}_i(j,:) - \boldsymbol{\mu}_k\|^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{k'}(1 + \|\mathbf{H}_i(j,:) - \boldsymbol{\mu}_{k'}\|^2/\alpha)^{-\frac{\alpha+1}{2}}}, \quad (5)$$

Here we use $\alpha = 1$, and $\mathbf{H}_i(j,:)$ denotes the j^{th} row (sub-structure) from graph $\mathcal{G}_i$. We then perform the clustering by minimizing the following KL-divergence as in (Junyuan, Ross, and Ali 2016)

$$\min_{\mathbf{U}, \mathbf{H}_i' s} \sum_i \mathrm{KL}\left(\mathbf{W}_i, \widetilde{\mathbf{W}}_i\right)$$
$$s.t. \ \widetilde{\mathbf{W}}_i(j,k) = \frac{\mathbf{W}_i^2(j,k)/\sum_l \mathbf{W}_i(l,k)}{\sum_{k'}\left[\mathbf{W}_i^2(j,k')/\sum_l \mathbf{W}_i(l,k')\right]}. \quad (6)$$

Here, $\widetilde{\mathbf{W}}_i$ is a self-sharpening version of $\mathbf{W}_i$, and minimizing the KL-distance in (6) forces each sub-structure instance to be assigned to only a few landmarks/clusters

The goal of deep clustering (6) is to favor "frequent" sub-structure patterns (or cluster centers) as landmarks $\boldsymbol{\mu}_k$'s; otherwise, if rare (low-density) sub-structures are used as landmarks to generate graph-level feature, they are very likely absent from testing graphs, leading to poor generalization. In other words, deep clustering serves an unsupervised regularization to stabilize the learning process; another benefit is that $\boldsymbol{\mu}_k$'s can be examined by looking into the sub-structures assigned to it, making the interpretation highly practical.

Interaction Preserving Graph Projection

The goal of this step is to project structural details of each graph ($\mathcal{G}_i$'s) onto the K sub-structure landmarks, via indicator matrix $\mathbf{W}_i$'s (5). Regardless of the size/topology, any input graph can then be transformed into a standardized "landmark graph", which always has K nodes corresponding to the K landmarks, with varying landmark weights, landmark features, and landmark interaction.

Landmark weight. The density of the K sub-structure land-

marks in graph $\mathcal{G}_i$ can be computed as

$$\mathbf{p}_i = \mathbf{W}'_i \cdot \mathbf{1}_{n_i \times 1}. \quad (7)$$

Intuitively, it quantifies the probability that the K structural landmarks can be found in graph $\mathcal{G}_i$. Unlike counting discrete sub-graphs in graph kernels, the density $\mathbf{p}_i$ is a soft probability and learned in an end-to-end fashion due to the optimizable structural landmarks $\boldsymbol{\mu}_k$'s.

Landmark feature. The mean of the sub-structure instances mapped to each of the K landmarks in graph $\mathcal{G}_i$ is

$$\mathbf{M}_i = \mathbf{X}'_i \cdot \mathbf{W}_i \cdot \mathbf{P}_i^{-1} \quad (8)$$

with $\mathbf{P}_i = \text{diag}(\mathbf{p}_i)$, and the kth column of $\mathbf{M}_i$ is the mean of structures mapped to the kth landmark.

Landmark Interaction. The adjacency matrix $\mathbf{A}_i$ can be projected onto the K landmarks by

$$\mathbf{C}_i = \mathbf{W}_i \cdot \mathbf{A}_i \cdot \mathbf{W}'_i, \quad (9)$$

which quantifies how the K types of sub-structures interact with each other in graph $\mathcal{G}_i$. We can further normalize it using the density of the K landmarks as $\tilde{\mathbf{C}}_i = \mathbf{P}_i^{-1}\mathbf{C}_i\mathbf{P}_i^{-1}$.

These features can be used together for final classification. For simplicity, we will use the (normalized) sub-structure interaction matrix $\tilde{\mathbf{C}}_i$ as graph-level feature, in order to clearly demonstrate the value of interaction modelling. Namely, $\tilde{\mathbf{C}}_i$ is vectorized and fed to an FC-layer for final classification.

We illustrate the architecture in Figure 2. The objective function includes the classification part (cross-entropy loss), and two regularization terms (parametric node2vec loss $\mathcal{L}_{pn2v}$ (4) reflecting sub-structure interaction inside each graph, and deep clustering loss $\mathcal{L}_{deep_clustering}$(6) reflecting the sub-structure similarity across graphs).

$$\mathcal{L} = \underbrace{-\lambda_1 \cdot \mathcal{L}_{pn2v} + \lambda_2 \cdot \mathcal{L}_{deep-clustering}}_{regularization-terms} + \underbrace{\mathcal{L}_{cross-entropy}}_{classification-loss}$$

Discussions

Dictionary Learning View

The structural landmarks μ_k's can be considered as basis of a "structural dictionary" to define and reconstruct diversified sub-structure instances, and K controls dictionary size. In general, neither too small nor too large dictionary is desirable. On the one hand, too few basis fails to recover basic structures and underfits the data. On the other hand, too many basis may leads to overfitting (Marsousi et al. 2014). In the literature of sparse dictionary learning, researchers have studied the condition under which a faithful sparse recovery can be achieved by the so-called "self-coherence", which measures the redundancy (correlation) of basis vectors. In particular, a higher coherence indicates a higher redundancy among the basis vectors, in which case dictionary learning can become unidentifiable (Donoho and Huo 2001). Similar observations apply to supervised learning scenario (Mehta and Gray 2013).

These insights explain why exact sub-graph matching can overfit (Yanardag and Vishwanathan 2015): it corresponds to a maximal (in fact, exponential) dictionary size due to the combinitorial nature of graph sub-structures, in this case, the redundancy (or coherence) between the landmarks can be quite large and violates the recovery condition, and so the learning process becomes unstable.

In practice, the optimal choice of K can be non-trivial. One may pick an empirical value (e.g. $K = 100$), or use validation set to choose K from pre-determined grid values.

Related Work and Comparisons

Graph Isomorphism. GNNs such as Graph Isomorphism Network (GIN) (Xu et al. 2019) can have great potential in graph isomorphism test by generating injective maps (Xu et al. 2019; Morris et al. 2019). This is achieved by choosing a representation that can capture all the differences between sub-structures (e.g. sum-pooling operator), so that different graphs will have different embeddings (i.e., injective map). However, an ideal classification is not injective, since two graphs with different structures may have the same label. In this regard, SLIM tries to find a tradeoff in handling the similarity/difference among sub-structures. Instead of trying to capture even the slightest difference of sub-structures as in GIN, it first groups sufficiently close sub-structures into the same cluster and treat them as the same entity, with a tunable granularity controlled by the landmark set size K. The biggest advantage of doing this is that it endows convenient, cluster-level identity to the sub-structures, so that a graph can then be projected onto these clusters/landmarks to generate a relational description of the graph with fixed dimension ($\mathbb{R}^{K\times K}$). In comparison, GIN adopted a continuous description of sub-structures (without further quantization into cluster-level identities), and so a fixed-dimensional, graph-level representation can only be obtained by squeezing all the nodes (or rooted-subtrees) into a single one.

Graph Pooling/Coarsening. Our approach has significant differences with existing methods. First, these methods shrink a graph through the pooling layer either by node sampling (Lee, Lee, and Kang 2019; Gao and Ji 2019) or grouping (Ying et al. 2018; Ma et al. 2015), and so the coarsening is local (inside each graph) and the number of clusters is bounded by the the size of the graph. In contrast, we use a global clustering on sub-structures collected from across different graphs to generate landmarks and model sub-structures on a global basis, and so the number of clusters is much larger. Such a global clustering also contributes to algorithm stability. Second, existing methods mostly generate **single-vector** representation whose dimension depends on that of the node features, while ours is a $K \times K$ **interaction feature matrix** quantifying the interaction between the K landmarks. This helps better preserve understandable topological information.

Graphlets and Graph Kernels. Graphlets and Graph kernels both exploit sub-structures, but they need to sample from pre-defined sub-structure candidates. In contrast, we allow sub-structure landmarks that are **end-to-end optimizable** to generate discriminative, graph-level interacting pattern. Second, graph kernels measure similarity between all possible pairs of sub-structures across two graphs; while SLIM models interaction between sub-structures in the same graph.

Experiments

Benchmark data. We have used 8 benchmark data sets, including 5 cheminformatics/bioinformatics datasets MUTAG

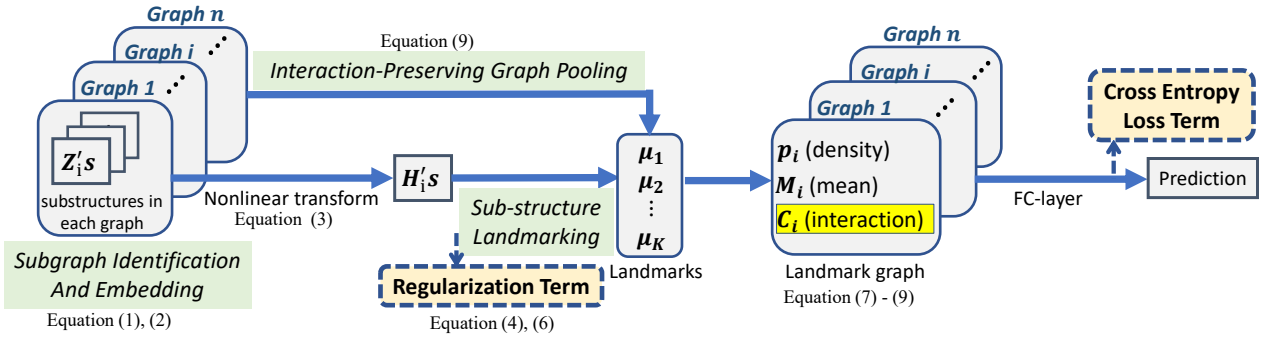


Figure 2: End-to-end training architecture of the SLIM network.

ALG.	MUTAG	PTC	NCI1	Protein	D&D	IMDB-B	IMDB-M	COLLAB
GK	81.38±1.74	55.65±0.46	62.49±0.27	71.39±0.31	74.38±0.69	65.87±0.98	43.89±0.38	72.84±0.28
PK	76.00±2.69	59.50±2.44	82.54±0.47	73.68±0.68	78.25±0.51	—	—	—
WLGK	84.11±1.91	57.97±2.49	**84.46±0.45**	74.68±0.49	78.34±0.62	73.40±4.63	49.33±4.75	79.02±1.77
PC-SAN	92.63±4.21	60.00±4.82	78.59±1.89	75.89±2.76	77.12±2.41	71.00±2.29	45.23±2.84	72.60±2.15
DGCNN	85.83±1.66	58.59±2.47	74.46±0.47	75.54±0.94	79.37±1.03	70.03±0.86	47.83±0.85	73.76±0.49
DiffPool	90.52±3.98	—	76.53±2.23	75.82±3.56	78.95±2.40	73.58±3.24	52.13±2.71	79.70±1.84
GNTK	90.12±8.58	67.92±6.98	75.20±1.53	75.61±4.24	79.42±2.18	75.93±3.61	52.82±4.65	**83.60±1.22**
SAG	73.53±9.68	69.67±3.12	74.18±1.29	71.86±0.97	76.91±2.12	72.61±2.23	51.80±2.08	79.88±1.02
GIN	90.03±8.82	64.60±7.00	79.84±4.57	75.28±2.65	77.58±2.94	75.15±5.08	52.33±2.84	80.21±1.92
StrPool	82.21±3.13	71.46±2.21	71.31±1.14	76.89±1.67	**79.72±1.98**	73.77±2.01	50.17±1.28	79.14±0.88
SLIM	**93.28±3.36**	**72.41±6.92**	**80.53±2.01**	**77.47±4.34**	79.61±2.66	**77.23±2.12**	**53.38±4.02**	78.22±2.02

Table 1: Classification on benchmark data-sets (cheminformatics, bioinformatics & social networks).

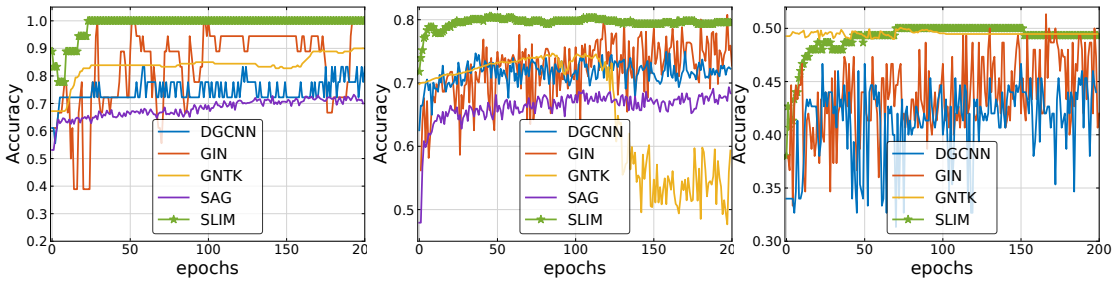
(chemical compound), PROTEINS (protein molecules), NCI1 (chemical compounds for cancer cell lines), PTC (chemical compounds for toxicology prediction) and D&D (enzyme classification), and 3 social network datasets IMDB-B, IMDB-M (movie collaboration), and COLLAB (scientific collaboration network).

Competing methods. We considered following competing methods: (1) Graph neural tangent kernel (GNTK) (Du et al. 2019); (2) Graph Isomorphism Network (GIN) (Xu et al. 2019); (3) End-to-end graph classification (DGCNN) (Zhang et al. 2018); (4) Hierarchical differential pooling (DiffPool) (Ying et al. 2018); (5) Self-attention Pooling (SAG) (Lee, Lee, and Kang 2019); (6) Convolutional network for graphs (PATCHY-SAN) (Niepert, Ahmed, and Kutzkov 2016); (7) Graphlet kernel (GK) (Shervashidze et al. 2009); (8) Weisfeiler-Lehman Graph Kernels (WLGK) (Shervashidze et al. 2011); (9) Propagation kernel (PK) (Neumann et al. 2016);(10) Structured graph pooling (Str-Pool) (Yuan and Ji 2020).

Experimental setting. Our codes are written in Pytorch and run on a server with dualcore CPU @2.10GHz and Nvidia GTX1080Ti graphics card. All the datasets and their 10 even splits are downloaded from (https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets). For each dataset we have 10 rounds, and in each rounds we use 1 split for testing, 90% of the rest 9 splits for training, and 10% of the rest 9 splits for model selection. The validation set will be "recycled" after model selection, namely, combined with the training set together to re-train the model for final testing. Results from the 10 rounds will be averaged and reported. The sub-structure instances are chosen as 3-hop rooted subgraphs around each node. The landmark set size K is chosen from $\{50, 100, 150, 200, 300\}$.The $\mathbb{R}^{K\times K}$ interaction feature (9) is re-shaped into a vector and then fed into a fully connected layer for class label prediction. No drop-out or batch-normalization is used considering the size of the benchmark data. Hyper-parameters include: (1) the number of hidden units in the nonlinear transform (3) is chosen from $\{d, d/2, 2d, 8, 16, 32\}$ where d is the input dimension of the encoder; (2) the optimizer is chosen from SGD and Adagrad, with learning rate $\{1e-2, 5e-2, 1e-3, 5e-3, 1e-4\}$; (3) The regularization parameters λ_1 and λ_2 for the unsupervised loss terms are selected from $\{0.01, 0.1, 1, 10, 100\}$.

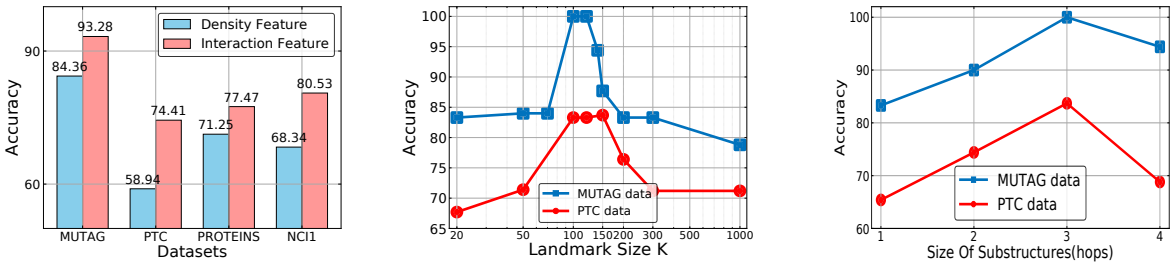
Classification performance. We report the classification performance in Table **??**. As can be seen in Table **??**, GNN-based approaches are more competitive than graph kernels, except that the WL-graph kernel performs the best on the NCI1 dataset. For social networks, the SLIM network gains a competitive score on IMDB-B and IMDB-M, but is inferior on COLLAB. We speculate that social networks do not have node features so our method becomes less advantageous.

Algorithm stability. In Figure 3, we plot the evolution of the testing accuracy v.s. training epochs. Our approach converges faster, and in a more stable manner. This signifies small variance of the training process, and makes it practically easy to determine when to stop training. We speculate



(a) MUTAG data. (b) NCI data. (c) IMDB-M data.

Figure 3: SLIM has a stable performance based on the accuracy-vs-epoch curve.



(a) Interaction vs density (counting). (b) Impact of landmark set size K. (c) Impact of sub-structure size.

Figure 4: The performance of SLIM w.r.t. the choice of hyper-parameters and graph level feature.

that such stability arises from the modelling of sub-structure distribution globally (across graphs).

Different Parameters and Graph-level Features. In Fig. 4 we illustrate performance of SLIM wrt choice of different parameters and graph level features on MUTAG dataset.

In Figure 4(a), we compare performance of SLIM when using the weights (or density) of the landmark $\mathbf{p}_i$ (7), or the interaction matrix $\mathbf{C}_i$ (9), as graph-level features. The interaction feature consistently generates better accuracy than distribution-based features, validating the importance of modeling the interacting relation in graph classification tasks.

In Figure 4(b), we examine the accuracy of SLIM versus landmark set size K. The accuracy is inferior when K is either too small (underfitting) or too large (overfitting). The best performance is obtained for a median K value.

In Figure 4(c), we examine different sub-structure scales by varying size (hops) of rooted sub-trees. The 3-hop subgraphs seem a good choice for all data sets, which could be consistent with meaningful sub-structure scales. Choosing sub-structure size is similar to choosing the number of convolutional layers (size of receptive field) in GNNs. The subtle difference is that in GNN, the convolutional layers are cascaded layerwise, while in our model, different layers of the k-hop sub-graph are modelled concurrently.

Conclusion

GNNs are state-of-the-art model for learning on graphs, but the design can still benefit from concepts of complex systems, in particular its core idea on the origin of system complexity: the interaction between components. We proposed the SLIM network to verify the importance of interaction modelling, and obtained encouraging results in graph classification.

We have a number of interesting future directions under investigation. First, we will explore new ways of obtaining sub-structure instances, and in particular irregular shaped sub-structures and how to incorporate domain knowledge in extracting sub-structure instances. Second, we will study how to select the number of landmarks more adaptively, and combine it with a feature selection module. Third, we will pursue a theoretic delineation on how the design of structural landmarks affects the generalization capacity in graph classification. Finally, we will collaborate with domain experts on interpretation of the model in cheminformatics applications.

Acknowledgements

Kai Zhang is sponsored by Shanghai Pujiang Project. Jie Zhang is supported in part by the National Science Foundation of China No.61672236, National Science Foundation of China No.61573107, Shanghai National Science Foundation 17ZR1444200, Shanghai Municipal Science and Technology Major Project No.2018SHZDZX01.

References

Alon, U. 2007. Network motifs: theory and experimental approaches. *Nature Reviews Genetics*, 8: 450–461.

Austin R. Benson, J. L., David F. Gleich. 2016. Higher-order organization of complex networks. *Science*, 353: 163 – 166.

Bronstein, M. M.; Bruna, J.; LeCun, Y.; Szlam, A.; and Vandergheynst, P. 2017. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 18–42.

Camacho, D.; Collins, K.; Powers, R.; Costello, J.; and Collins, J. 2018. Next-Generation Machine Learning for Biological Networks. *Cell*, 173(7): 1581–1592.

Cangea, C.; Velicković, P.; Jovanović, N.; P., T. K.; and Lió. 2018. Towards sparse hierarchical grap classifiers. In *preprint arXiv:1811.01287*.

Cilliers, P. 1998. *Complexity and Postmodernism: Understanding Complex Systems*. Psychology Press.

Debarsy, N.; Cordier, S.; Ertur, C.; Nemo, F.; Nourrit-Lucas, D.; Poisson, G.; and Vrain, C. 2017. *Understanding Interactions in Complex Systems, Toward a Science of Interaction*. Cambridge Scholars Publishing.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems 29*, 3844–3852. Curran Associates, Inc.

Donoho, D. L.; and Huo, X. 2001. Uncertainty principles and ideal atomic decomposition. *IEEE Trans. Inf. Theory*, 47: 2845–2862.

Du, S. S.; Hou, K.; Salakhutdinov, R. R.; Poczos, B.; Wang, R.; and Xu, K. 2019. Graph Neural Tangent Kernel: Fusing Graph Neural Networks with Graph Kernels. In *Advances in Neural Information Processing Systems 32*, 5723–5733.

Gao, H.; and Ji, S. 2019. Graph u-net. In *Proceedings of the 36th International Conference on Machine Learning*.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable Feature Learning for Networks. In *the 22nd ACM SIGKDD International Conference*.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*.

Hartwell, L.; Hopfield, J.; Leibler, S.; and Murray, A. 1999. From molecular to modular cell biology. *Nature*, 2: 47–52.

Jie, Z.; Cui, G.; Zhang, Z.; Yang, C.; Liu, Z.; and Sun, M. 2020. Graph Neural Networks: A Review of Methods and Applications. In *AI Open*, 1:57–81.

Junyuan, X.; Ross, G.; and Ali, F. 2016. Unsupervised deep embedding for clustering analysis. In *International Conference on Learning Representations*.

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

Kriegeskorte, N. 2015. Deep Neural Networks: A New Framework for Modeling Biological Vision and Brain Information Processing. *Annual Review of Vision Science*, 1: 417–446.

Lee, J.; Lee, I.; and Kang, J. 2019. Self-Attention Graph Pooling. In *International Conference of Machine Learning*, 3734–3743.

Ma, Y.; Wang, S.; Aggarwal, C. C.; and Tang, J. 2015. Graph Convolutional Networks with EigenPooling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*.

Ma, Y.; Wang, S.; Aggarwal, C. C.; and Tang, J. 2019. Graph Convolutional Networks with EigenPooling. In *the 25th ACM SIGKDD International Conference*.

Marsousi, M.; Abhari, K.; Babyn, P. S.; and Alirezaie, J. 2014. An Adaptive Approach to Learn Overcomplete Dictionaries With Efficient Numbers of Elements. *IEEE Transactions on Signal Processing*, 62(12): 3272 – 3283.

Mehta, N. A.; and Gray, A. G. 2013. Sparsity-based generalization bounds for predictive sparse coding. In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, 36 – 44.

Mesquita, D.; Souza, A. H.; and Kaski, S. 2020. Rethinking pooling in graph neural networks. In *Advances in Neural Information Processing Systems 33*.

Milo, R.; Shen-Orr, S.; Itzkovitz, S.; Kashtan, N.; Chklovskii, D.; and Alon, U. 2002. Network Motifs: Simple Building Blocks of Complex Networks. In *Science*, 824–827.

Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, J. E.; Rattan, G.; and Grohe, M. 2019. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *The Thirty-third AAAI Conference on Artificial Intelligence*.

Neumann, M.; Garnett, R.; Bauckhage, C.; and Kersting, K. 2016. Propagation kernels: efficient graph kernels from propagated information. *Machine Learning*, 102(2): 209–245.

Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning convolutional neural networks for graphs. In *In Proceedings of The 33rd International Conference on Machine Learning*.

Pastor-Satorras, R.; Castellano, C.; Mieghem, P. V.; and Vespignani, A. 2015. Epidemic processes in complex networks. *Rev. Mod. Phys.*, 87.

Peng, H.; Li, J.; Gong, Q.; Ning, Y.; and He, L. 2020. Motif-Matching Based Subgraph-Level Attentional Convolutional Network for Graph Classification. In *Proceedings of the AAAI Conference on Artificial Intelligence, 2020, 34(4):5387-5394.*

Schmidt, J.; Marques, M. R. G.; Botti, S.; and Marques, M. A. L. 2019. Recent advances and applications of machine learning in solid-state materials science. *NPJ Computational Material*, 5: 1581–1592.

Shervashidze, N.; Schweitzer, P.; van Leeuwen, E. J.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-Lehman Graph Kernels. *Journal of Machine Learning Research*, 12(77): 2539–2561.

Shervashidze, N.; Vishwanathan, S.; Petri, T.; Mehlhorn, K.; and Borgwardt, K. 2009. Efficient graphlet kernels for large graph comparison. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, volume 5, 488–495.

Stokes, J. M.; Kevin Yang, K. S.; Jin, W.; Cubillos-Ruiz, A.; Donghia, N. M.; MacNair, C. R.; French, S.; Carfrae, L. A.; Bloom-Ackermann, Z.; Tran, V. M.; Chiappino-Pepe, A.; Badran, A. H.; Andrews, I. W.; Chory, E. J.; Church, G. M.; Brown, E. D.; Jaakkola, T. S.; Barzilay, R.; and Collins, J. J. 2020. A Deep Learning Approach to Antibiotic Discovery. *Cell*, 180(4): 688–702.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph Attention Networks. In *International Conference on Learning Representations*.

Vishwanathan, S.; Schraudolph, N. N.; Kondor, R.; and Borgwardt, K. M. 2010. Graph Kernels. *Journal of Machine Learning Research*, 11(40): 1201–1242.

Wernicke, S. 2006. Efficient Detection of Network Motifs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4): 347–359.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2020. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*.

Yanardag, P.; and Vishwanathan, S. V. N. 2015. Deep Graph Kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1365–1374.

Ying, R.; You, J.; Morris, C.; Ren, X.; Hamilton, W. L.; and Leskovec, J. 2018. Hierarchical Graph Representation Learning with Differentiable Pooling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 4805–4815.

Yuan, H.; and Ji, S. 2020. StructPool: Structured Graph Pooling via Conditional Random Fields. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30*.

Zaheer, M.; Kottur, S.; Ravanbakhsh, S.; Poczos, B.; Salakhutdinov, R. R.; and Smola, A. J. 2017. Deep Sets. In *Advances in Neural Information Processing Systems 30*, 3391–3401. Curran Associates, Inc.

Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An End-to-End Deep Learning Architecture for Graph Classification. In *The Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhang, Z.; Bu, J.; Ester, M.; Zhang, J.; Yao, C.; Yu, Z.; and Wang, C. 2019. Hierarchical Graph Pooling with Structure Learning. In *preprint arXiv:1911.05954*.

Zhang, Z.; Cui, P.; and Zhu, W. 2020. Deep Learning on Graphs: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, PP(99): 1–1.