SKIPPOOL: Improved Sparse Hierarchical Graph Pooling with Differentiable Exploration

Sarith Imaduwage

Independent Researcher
imaduwagesarith932@gmail.com

Abstract

Hierarchical pooling in conjunction with Graph Neural Networks (GNN) improves performance in graph classification tasks. Hierarchical pooling has to produce Multi-resolution representations while preserving graph-level information. One such hierarchical pooling is **S**tandard-**S**parse-**P**ooling (SSP). SSP assigns an importance score to each node, selects the Top-K nodes, and scales their attributes by their scores to produce the output. We reveal SSPs' tendency to pool Over Representative Regions (ORR) on the graph's signal space leaving some regions unpooled thus proving that SSP is incapable of preserving graph-level information robustly. We propose to overcome this by an improved differentiable exploration over the graph's signal space dubbed as skipping hence the name SKIPPOOL. We tested SKIPPOOL & its variant SKIPPOOL-FULL each against matching pooling methods. Proposed methods achieve new state-of-the-art performance on the majority of benchmark datasets. Moreover, we show that they are more robust at capturing the graph signal space consequently preserving more graph-level information than their counterparts. Proposed methods require a reasonably few parameters and their execution time can be kept low with parallelization.

Code — https://github.com/SarithRavI/SkipPool

1 Introduction

Multi-resolution representation captures the input across various resolutions or scales to achieve maximal invariance at a local level through local pooling (Bronstein, Bruna, and Cohen 2021). Various works have investigated how local pooling on graphs can improve Graph Neural Network (GNN) architectures which are the go-to solution for graph-structured data. Additionally, local pooling with GNNs allows the accumulation of information from the furthest in the graph and computational efficacy due to the smaller pooled graph. GNNs are shown to be effective across node classification (Hamilton, Ying, and Leskovec 2017) & link prediction (Schütt et al. 2017) and in graph-level classification (Xu et al. 2019) & regression (Gilmer et al. 2017; Duvenaud et al. 2015) tasks where the preservation of graph-level information is important. In graph-level tasks some information may be lost while capturing low-resolution representations of the graph, thus effective graph pooling must preserve crucial graph-level information while leveraging the benefits of local pooling. *Hierarchical Graph Pooling* does exactly this by extracting hierarchical structures such as clusters in input nodes' feature space. Hierarchical graph pooling can be categorized into two categories: *Dense* methods that try to overcome the inherent *flat* nature of GNNs through the aggregation of nodes such that hierarchical clusters are formed & *Sparse* methods only select those deemed as necessary to represent hierarchical clusters present. For example,DIFFPOOL (Ying et al. 2018), a dense method calculates a soft assignment matrix for all the nodes over a predetermined number of clusters.

According to the SRC framework (Grattarola et al. 2022) pooling operators are a combination of 3 functions: ***S**elect*, ***R**educe*, and ***C**onnect*. *Select* specifies the scheme that selects a subset of nodes, *Reduce* specifies the transformation of selected nodes to form output nodes, and *Connect* specifies relations among output nodes. The Select function of sparse methods assigns an importance score s_i to each node i, then selects a subset based on the rank determined by scores. We define a standard setup where the scores are a function of node attributes x_i that is $s_i = f(x_i)$, Select nodes with Top-K scores, and Reduce selected node i' as $g(s_{i'})x_{i'}$ where g is an arbitrary function. We abuse the notation s_i also for $g(s_i)$, meaning $s_i := g(s_i)$. We define methods that adhere to this setup as Standard-Sparse-Pooling (SSP). TOPKPOOL (Gao and Ji 2019; Cangea et al. 2018) & SAGPOOL (Lee, Lee, and Kang 2019) belong to SSP. TOPKPOOL calculate scores as $f(x_i) = p^\top x_i$ and SAGPOOL calculate as $f(x_i) = p^\top \mathrm{GNN}(x_i)$ where p is a trainable vector and GNN is any GNN.

Sparse methods like ASAPPOOL (Ranjan, Sanyal, and Talukdar 2020), KMIS (Bacciu, Conte, and Landolfi 2023) & EDGEPOOL (Diehl 2019) add complexities to this standard setup, so a thorough analysis of SSP is needed. The key is to understand the role of scores in Selection and Reduction. The *existing perspective* is to treat scores as gating values: $s_i \simeq 1$ on nodes to be selected, $s_i \simeq 0$ on the ones to be discarded. Thus the ideal choice of *g* is the *Sigmoid* function. But *empirically* s_i can differ from the desired values (i.e. 0 & 1) and still perform well or even outperform (see Appendix A.1). Such misalignment between the empirical

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

results and the perspective from which SSP is interpreted is not studied in the literature. In this work, we address this gap with a new perspective that fits the Select & Reduce functions of SSP with a more general definition of pooling. Our new perspective reveals that SSP can't effectively form hierarchical representations mainly due to its tendency to over-represent certain regions on the signal space. As a result of over-representation certain regions are under-represented discarding a portion of hierarchical information. A balanced representation of the whole signal space is required.

Central to our contributions is a new learnable selection technique that avoids over-representation by reaching nodes that otherwise will be left out. While keeping scoring as same as in SSP, what's novel is that the Top-K ranking is coupled with a learnable selection scheme that learns to select nodes. We dub this as *skipping* on which a new pooling operator is proposed namely SKIPPOOL. SKIPPOOL, skips nodes ranked by Top-K that cause over-representation and only selects nodes that contribute to increasing the quality of hierarchical representations. Skipped nodes are aggregated to the selected nodes which also increases the hierarchical representation. Unlike dense methods where such aggregation is dense SKIPPOOL maintains a low sparsity while the output graph also maintains the sparsity of the input graph. In summary, our contributions are as follows; **(i)** Introducing a new perspective on SSP from which drawbacks of SSP are identified, **(ii)** Improving SSP using novel differentiable exploration i.e. skipping, **(iii)** Test SKIPPOOL against existing works and empirically confirming its efficacy, **(iv)** Confirming that SKIPPOOL preserve significantly more node signal space information.

2 Background

2.1 Establishing a New Perspective

The core of the new perspective is to interpret SSP as same as one would interpret any other pooling. For example, any pooling is (informally) defined as obtaining a compact representation by assimilating nearby points. As per the new perspective so is SSP. We extend this idea to fit SSP with a more formal definition. Let $x_i \in \mathbb{R}^d$ and $s_i \in \mathbb{R}^+$ be the attributes and score of node i, respectively.

Definition 1 (Bronstein, Bruna, and Cohen 2021) We define pooling as a transformation of a multiset of signals on a domain to another multiset of signals on a coarsened domain. Pooling at scale J is $P_J : \mathcal{X}(\Omega) \rightarrow \tilde{\mathcal{X}}(\tilde{\Omega})$ where $\mathcal{X}(\Omega, \mathcal{C}) := \{x_i : \Omega \rightarrow \mathcal{C}\}$, same for $\tilde{\mathcal{X}}(\tilde{\Omega}, \tilde{\mathcal{C}})$. Ω, $\mathcal{C}$ denote points on the domain and channels of the signal respectively. A function $f : \mathcal{X}(\Omega) \rightarrow \mathcal{Y}$ is said locally stable at scale J if approximation $f \approx f_J \circ P_J$ is attainable with the composition of function $f_J : \tilde{\mathcal{X}}(\tilde{\Omega}) \rightarrow \mathcal{Y}$, and pooling P_J.

See *Def.* 1 of (Xu et al. 2019) for the formal definition of a *multiset*. For any pooled representation $\tilde{x}_j \in \tilde{\mathcal{X}}(\tilde{\Omega})$, the scaling factor $J(\tilde{x}_j)$ signifies the size of the region on domain Ω from which the pooling map to $\tilde{x}_j$. We call this region **Receptive Field** (RF) of $\tilde{x}_j$ denoted by $\mathrm{RF}(\tilde{x}_j) \subseteq \mathcal{X}(\Omega)$. Note, $J(\tilde{x}_j) = |\mathrm{RF}(\tilde{x}_j)|$.

Definition 2 Assume the input node feature space is *countable*. If the condition; The summation of any subset of output features of a GNN layer is injective to that subset, is satisfied then that GNN layer is *maximally expressive*.

This is a direct derivation from the theoretical framework laid in (Xu et al. 2019). The same work introduces the GIN layer which is modeled to satisfy the mentioned condition to achieve maximal expressivity. Whether GIN or similarly powerful GNN satisfies this condition strictly for *any* subset is an empirical concern. But Lemma 5 in (Xu et al. 2019) shows theoretically this is very well possible.

Definition 3 Suppose any output $\tilde{x}_j = s_i x_i \in \tilde{\mathcal{X}}(\tilde{\Omega})$ from the *SSP* denoted by P such that $P(\mathrm{RF}(\tilde{x}_j)) = \tilde{x}_j$, then,

1. $e_s \in \mathbb{R}^+$ such that $s_i = e_s\,|\mathrm{RF}(\tilde{x}_j)|$.
2. $e_d \in \mathbb{R}^+$ such that $\sum_{\forall x_k \in \mathrm{RF}(\tilde{x}_j)} d(x_k, x_i) = e_d$, for a metric $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^+$.
3. If there exists a function $\Phi : \mathcal{X}(\Omega) \rightarrow \mathbb{R}^d$ so that $\Phi(X)$ is unique for each multiset $X \subseteq \mathcal{X}(\Omega)$ then $\mathrm{RF}(\tilde{x}_j) = \Psi(\Phi(\mathrm{RF}(\tilde{x}_j))) = \Psi(s_i x_i + e_r)$ where Ψ is the retraction function of Φ and $e_r \in \mathbb{R}^n$.

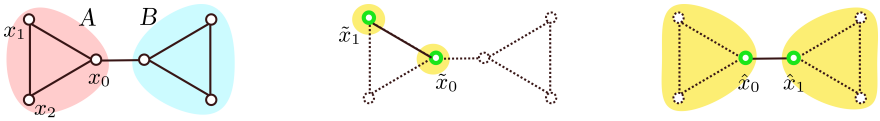
When the input domain Ω is a grid or sequence, pooling maps the input to output at regular intervals on the domain such that $J(\tilde{x}_j)$ is constant for all $\tilde{x}_j$. That is the RF size of each output representation is the same. If the domain is a graph then having a fixed-size RF for all outputs is not straightforward due to irregularities and inconsistencies in graphs. Unless the graph structure is exploited for an equispaced pooling, RFs on graphs must be adaptive and modeled to be learned.

Our new perspective is SSP too should have the notion of RF and since there is no structural exploitation for an equispaced pooling, the learnable scoring in SSP must encompass RF. The relation between s_i and $|\mathrm{RF}(\tilde{x}_j)|$ is defined in *Def.* 3 (1). Informally, a pooled node is an assimilation of points of its RF, thus we additionally need a characterization of RFs using a notion of metric; as in *Def.* 3 (3). We dub e_r as *information loss* as it signifies the difference between output $\tilde{x}_j$ and the vector from which $\mathrm{RF}(\tilde{x}_j)$ can be fully recovered. Suppose SSP pooling requires zero graph-level information loss, then the information loss incurring from mapping an RF to an output node must be zero so from any $\tilde{x}_j$ corresponding $\mathrm{RF}(\tilde{x}_j)$ can be recovered. That is there exists a function Φ as in *Def.* 3 (3) for which $\mathrm{RF}(\tilde{x}_j) = \Phi(s_i x_i)$ and $e_r = 0$.

2.2 Objective of SSP

Here, we discuss how well an SSP under *ideal conditions* can maintain the expressivity of preceding GNN layers. Two conditions need to be met for the ideal condition: Information loss is zero i.e., $e_r = 0$ for all pooled nodes, and SSP has maximum local stability (same notion in *Def.* 1). With the new perspective, SSP can be seen at a high level as a search algorithm.

Proposition 1 Assume the preceding GNN satisfies the condition in *Def.* 2. The output of an SSP under ideal conditions can be viewed as the solution to the following search



Input (a) Likely output (b) Desired output

Figure 1: (a) & (b) show two different output graphs when 2 nodes must be pooled. Yellow blobs around the pooled nodes (green circles) represent the RFs of nodes projected onto the output. In contrast to RFs of (a), RFs of (b) capture both clusters *A*, *B* preserving the hierarchical structure of the graph. Thus (b) is desirable over (a) which we get from SSP. Assume *A* is information-dense, so capturing *A* to the fullest is desirable. Suppose node x_0 is selected and both $d(x_0, x_1)$, $d(x_0, x_2)$ are greater than ϵ in Eq. (1). From Eq. (1), $\Theta(x_0, \mathrm{RF}(\tilde{x}_0)) > \Theta(x_0, \mathrm{RF}(\hat{x}_0))$. This means the chances of pooling $\tilde{x}_0$ are higher than pooling $\hat{x}_0$. If SSP could pool $\hat{x}_0$ and consequently fully capture *A*, it could then reach *B* as in (b). But instead SSP over-represent *A* by pooling redundantly from *A* as in (a).

problem:

$$\{(x_i, \mathrm{r}_k)\} = \underset{\mathbf{s}_{i,k} \in \mathbf{S}}{\text{Top-K}}\ \Theta(x_i, r_k)$$
$$\text{where} \quad \Theta(x, r) = \frac{|r|}{\sum_{\forall x_j \in \mathrm{r}} d(x, x_j) + \epsilon} \tag{1}$$

Proofs are in the Appendix B. $\mathbf{S}$ is a set of tuples with all the possible combinations of $\mathbf{s}_{i,k} = (x_i, \mathrm{r}_k)$ such that x_i is the i^{th} node in $\mathcal{X}(\Omega)$ and r_k is the k^{th} nonempty subset of $\mathcal{X}(\Omega)$. If $|\mathcal{X}(\Omega)| = n$, then $k \leq 2^n - 1$ and $|\mathbf{S}| = n(2^n - 1)$. Eq. (1) means the selection of Top-K tuples of nodes and candidate RFs such that the sizes of those RFs are maximized while the deviation between the picked node and nodes within the corresponding RF is minimal. A small $\epsilon \approx 0$ is added to avoid division by zero. It's easier to understand SSP with a compact description like *Prop.* 1 more than with the existing decoupled Select-Reduce scheme. Much of what was discussed so far is based on $s_i \in \mathbb{R}^+$. In Appendix A.1, we discuss why *Prop.* 1 applies to non-linear activated scores as well.

2.3 Analysis of SSP

In this section, we show a certain characteristic of SSP by unraveling *Prop.* 1. We mainly focus on this characteristic which we deem as a critical drawback of SSP. To that end, we define *Partitions* as following and all the notations are the same as in *Prop.* 1.

Partitions For a given $\mathbf{s}_{i,k} \in \mathbf{S}$, we define all $\mathbf{s}_{i',k'} \in \mathbf{S}$ such that $r_{k'} \subset r_k$ and $\Theta(x_i, r_k) \leq \Theta(x_{i'}, r_{k'})$ as *partitions* of r_k. The logical extreme case is r_k is fully symmetric i.e. all $x_i \in r_k$ are equivalent, where there are $|r_k|$ number of partitions and all partitions $(x_{i'}, r_{k'})$ satisfy $\Theta(x_{i'}, r_{k'}) = 1/\epsilon$. Generally, we can expect a sufficiently large r_k to have two or more partitions.

Over-Representative Regions Suppose a pair (x_k, r_k) that satisfies Eq. (1). Note that, partitions of r_k are also valid solutions to Eq. (1). According to Eq. (1), it's equally likely to output these partitions instead of (or along with) r_k. This makes SSP prone to *over-representation* of some regions at the expense of under-representation of the rest. Regions with higher degrees of symmetry are the most likely for such over-representation. We demonstrate this drawback using the following example:

Example: Suppose Eq. (1) gives Top-K(=3) pairs of outputs $(x_1, r_1), (x_2, r_2), (x_3, r_3)$ such that each r_i represent a distinct region. If r_1 is symmetric and partition into two r_1' & r_1'', then similarly Eq. (1) can output $(x_1', r_1'), (x_1'', r_1''), (x_2, r_2)$. Now we can see region r_3 is left out for partitions of r_1.

Fig. 1 provides another example. (You, Ying, and Leskovec 2019) shows that traditional MP layers, which exploit the graph's structural symmetries, amplify symmetries in output node embeddings. Hence one could expect this to be a common drawback in SSP. We conclude the theoretical analysis of SSP and Appendix A.2 provides strong empirical evidence of unbalanced pooling in SSP due to ORR. Ideal conditions aren't strictly needed to utilize *Prop.* 1 for insights, for example, aforementioned evidence for ORR is observed under a general setup. Therefore insights coming from *Prop.* 1 are likely valid for any general setup.

2.4 Continuous relaxation of Categorical sampling

Sampling from a categorical distribution within the forward pass of a stochastic network poses challenges due to the inability to compute gradients for discrete random variables. By continuous relaxation of a categorical distribution on a simplex one can approximate categorical samples and with the reparameterization trick (Kingma, Salimans, and Welling 2015) gradients of the distribution parameters can be computed. Gumbel-Softmax-Trick (GST) (Maddison, Mnih, and Teh 2017; Jang, Gu, and Poole 2017) is such a continuous relaxation of a categorical sampling method; Gumbel-Max-Trick (GMT) (Maddison, Tarlow, and Minka 2014). GST uses concrete distribution and the distribution is controlled by a temperature τ. A concrete random vector $C = (C_1, ..., C_k)$ i.e. the sampling $C \sim \text{Concrete}(\theta_0, ..., \theta_k)$ can be obtained by the following where θ_i is i^{th} category logit & G_i is independent sampling from the Gumbel$(0, 1)$ distribution for each category;

$$G_i = -\log(-\log u_i),\ u_i \sim \text{Uniform}(0, 1).$$
$$C_i = \frac{\exp\{(\theta_i + G_i)/\tau\}}{\sum_{j=0}^{k} \exp\{(\theta_j + G_j)/\tau\}}.$$

Apart from GST's use as a surrogate for rather hard objectives of producing strictly discrete latent representations, it

can be used as an alternative to soft attention since GST encourages discreteness and leads toward hard attention (Raffel et al. 2017). Note that higher τ (eg: $\tau = 10$) yields a uniform distribution simulating more explorative sampling and lower τ (eg: $\tau = 0.1$) simulates more exploitive sampling requiring some balance between exploration and exploitation. Such equilibrium can be attained by subjecting sampling to a Boltzmann exploration (Sutton 1990), in which temperature is gradually decreased from a higher τ adhering to some annealing scheme.

3 Methodology

3.1 Overview

The key idea of **SKIPPOOL** is to avoid partitioning by grouping nearby redundant nodes, allowing the selection of far nodes. Let the input graph be $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{X})$, where $\mathcal{V}$ is vertices set, $\mathcal{E}$ is edges set, $\mathcal{X} \subset \mathbb{R}^d$ is node features set. Let $\boldsymbol{A}$ be the adjacency matrix of $\mathcal{G}$ and, $N = |\mathcal{V}|$. SKIPPOOL utilizes a *scoring* method to rank the input nodes and then uses a *grouping* method to assimilate nearby points. Starting from the highest-ranked node, first SKIPPOOL uses the grouping method, then *skips* to the next highest-rank node that is not in the group. We limit the maximum length of a skip named as the *stride* length F. Grouping & skipping continue until the *Termination condition* is met. There can't be more skipping beyond a certain length called the *Termination length* denoted by n. Based on the implementation, either $n = N - F$ or $n = N$.

3.2 Main Components

Scoring Similar to SSP we employ a scoring mechanism to rank nodes. Let π_s permute nodes by descending order of their scores and let sorted node-score pairs be the set; $V = \{(s_0, v_0), ..., (s_{N-1}, v_{N-1})\}$. $\pi_s(i)$ is the original (prior sorting) index of node $v_i \in V$.[1]

Grouping Let the *scope* of a node be the set of nodes on V that fall within the stride length starting from that *source* node. Grouping finds nodes in the scope closer to the source node and discards the distant ones. Let the set $g_i = \{(s_i, v_i), ..., (s_{i+F-1}, v_{i+F-1})\}$ be the scope of node $v_i \in V$ with stride length F. For an arbitrary similarity metric $d(\cdot, \cdot)$, we aim to find a subset $c_i \subset g_i$ such that there is a certain distant $d^*(v_i)$ where $d(v_i, v_j) > d^*(v_i)\ \forall v_j \in c_i$, and $d(v_i, v'_j) \leq d^*(v_i)\ \forall v'_j \in g_i / c_i$. Let $\boldsymbol{X}(i)$ be the feature vector of node v_i. In this work $d(\cdot, \cdot)$ is defined as:

$$d(v_i, v_j) = \exp(-\|\boldsymbol{X}(i) - \boldsymbol{X}(j)\|) \tag{2}$$

Let $\mathfrak{D}_i = \{d(v_i, v_j); \forall v_j \in g_i\}$ be the set of similarity metrics between the source node v_i and each node in g_i. $\mathfrak{D}'_i = \left\{d'_0, ..., d'_{F-1}\right\}$ is the sorted $\mathfrak{D}_i$ by descending order $\pi_g : I \to I$ where $I = \{0, 1, ..., F-1\}$ (see Eq. 3). We get the set c_i by finding a threshold on $\mathfrak{D}_i$ such that it satisfies the condition of $d^*(v_i)$. To that end, we use GST to sample an index over indexes of $\mathfrak{D}'_i$. Let $\mathfrak{C}_i$ be a concrete random vector over indexes of $\mathfrak{D}'_i$. We get $d^*(v_i)$ by Eq. (4).[2]

$$\begin{gathered} \mathfrak{D}'_i = \left\{d'_{\pi(k)}; d'_{\pi(k)} = d\left(v_i, v_{i+k}\right) \in \mathbb{R}^+, \forall k \in I\right\} \\ \pi(k) < \pi\left(k'\right) \Longrightarrow d'_{\pi(k)} \geqslant d'_{\pi(k')} \quad \forall k, k' \in I \end{gathered} \tag{3}$$

$$d^*(v_i) = \mathfrak{C}_i \cdot \mathfrak{D}'_i \tag{4}$$

Modeling Category Logits To model category logits over indexes of $\mathfrak{D}'_i$, we use a simple attention mechanism that aims to encapsulate the relation between v_i and indexes of $\mathfrak{D}'_i$. The attention is calculated between v_i (Query) and each node in g_i (Keys). The attention on index k is: $\alpha_i^k = w^T(\boldsymbol{X}(i) \| \boldsymbol{X}(\bar{g}_i^k))$ where $\bar{g}_i = \pi_g(g_i)$. Note $\bar{g}_i$ is equivalent to the set we get permuting g_i by π_g that sorted $\mathfrak{D}_i$ to $\mathfrak{D}'_i$.[3] $w \in \mathbb{R}^{2d}$ is a learnable vector.

Instead of using α_i directly as category logits which we dub as *local* logits, we model a more *global* relationship. In other words, $\mathfrak{C}_i$ should encapsulate the probability distribution of d^* positioning over index I irrespective of the source node v_i such that the probability of d^* being on $i \in I$ is same for all $v_i \in V$. We justify this choice in Appendix C.1. We define global logits as:

$$\alpha^k = \sum_{m=0}^{n-1} \frac{\exp(\alpha_m^k)}{\sum_{d=0}^{F-1} \exp(\alpha_m^d)} \tag{5}$$

n is the termination length. Having category logits at our disposal, we model $\mathfrak{C}_i \sim \text{Concrete}(\alpha^0, \alpha^1, ..., \alpha^{F-1})$, using GST as follows:

$$\mathfrak{C}_i^k = \frac{\exp\left\{(\alpha^k + G_i^k)/\tau\right\}}{\sum_{d=0}^{F-1} \exp\left\{(\alpha^d + G_i^d)/\tau\right\}} \tag{6}$$

Every G_i^k is independently sampled from $\text{Gumbel}(0, 1)$ for all $i < n$ and $k < F$ combinations.

Let's define a new set taking n into account; $V' = \{v_i \in V; 0 \leqslant i < n\}$. We define D,D',$C \in \mathbb{R}^{n \times F}$, such that i^{th} row in each matrix corresponds to $\mathfrak{D}_i$, $\mathfrak{D}'_i$, and $\mathfrak{C}_i$ respectively of the node $v_i \in V'$. We get new $D^* \in \mathbb{R}^n$ by $D^* = \text{SUM}_x(D' \odot C)_x$, where $\odot$ hadamard product & SUM_x is row-wise sum. Note that i^{th} row of D^* is the dot product $D^*[i] = D'[i]^\top \cdot C[i]^\top$ and $D^*[i]$ is $d^*(v_i)$ defined in Eq. (4). We get unnormalized weights $W \in \mathbb{R}^{n \times F}$ using the following:

$$W = \text{ReLU}(K) \quad \text{where} \quad K = D \ominus D^* \quad \in \mathbb{R}^{n \times F} \tag{7}$$

$\ominus$ denotes row-wise subtraction. Recall that $d^*(v_i)$ is a threshold we model to decide nodes within closer proximity to v_i. With the row-wise ReLU in Eq. (7), we add positive weight to all v_j that satisfy $d(v_j, v_i) > d^*(v_i)$ and zero out the rest. By normalizing W in row-dimension we get $\Lambda \in \mathbb{R}^{n \times F}$, such that i^{th} row is $\Lambda[i] = W[i] / \sum_{d=0}^{F-1} W[i]_d$.

Skipping mechanism We use the following simple schema to select rows in Λ, and each selected row called

[1] nodes in a graph have some arbitrary ordering before input to the algorithm, here we are referring to the index according to that ordering.

[2] A set is viewed as a column vector.

[3] Superscript over a column vector denotes an element and Subscript over a row vector denotes an element. $\|$ denotes concatenation.

a *pivot row* corresponds to a pooled node in the new graph. Starting from the first row, $\Lambda[i=0]$, get the *first* index j, where $\Lambda[i]_j = 0$ then move to row $\Lambda[i := i+j]$ and select this row. Continue this selection procedure until the termination condition is met. Let P be a set of indices. At a given moment of the selection procedure, P holds the pivot rows up to that point. That is for each $p \in P$, $\Lambda[p]$ is a pivot row. In SKIPPOOL if $|P| = \lceil rN \rceil$ *or* $i + j > n$ then skipping terminates, in SKIPPOOL-FULL skipping terminates only if $i + j > n$.

Aggregation Here describe how new node features are produced using aggregation. Let $G_i \in \mathbb{R}^{F \times d}$ be a matrix where rows are feature vectors $\boldsymbol{X}(i)$ $\forall v_i \in g_i$ i.e., feature vectors of nodes in the scope of source node v_i, and let s_i be the score value of the source node. We get the new node feature $\boldsymbol{X}'(p)$, for each $p \in P$ as follows:

$$\boldsymbol{X}'(p) = (s_p \Lambda[p] G_p)^\top \quad \in \mathbb{R}^d \tag{8}$$

Edge connectivity Similar to TOPKPOOL & SAGPOOL we define the new adjacency matrix as: $\boldsymbol{A}' = \boldsymbol{A}_{\pi_s(i),\pi_s(j)}$, $\forall i, j \in P$. Subscripts denote the rows & columns of the adjacency matrix.

Temperature Annealing schedule With $\tau \to 0$, $\mathfrak{C}_i$ gives a one-hot-vector, yielding d^* in Eq. (4) strictly an element of $\mathfrak{D}_i$. For $\tau \to$ inf, d^* is a convex combination of elements in $\mathfrak{D}_i$, giving a soft threshold. Incrementally lowering the temperature from high to low, at the initial phase of learning, allows better exploration of a soft threshold via a convex combination of neighboring distances gradually converging to a robust local minimum at a lower temperature. Convergence to strictly an element of $\mathfrak{D}_i$ is not necessary if a soft threshold learned at higher temperature yields better results, which makes such a Temperature Annealing compatible with early stopping in a learning setup. Following (Balin, Abid, and Zou 2019) if T_a and T_b are the highest and lowest temperatures respectively then the temperature at epoch r is modeled as the exponential decay: $T_a(\frac{T_b}{T_a})^{\frac{r}{R}}$, where R is the epochs at which T_b should be reached.

Discussion on Components In SKIPPOOL, one can select the appropriate F for given pooling ratio r as $F = \frac{1}{r} + 1$. This guarantees that rN number of nodes will be pooled. Elements in $W[i]$ do not result from a Positive Definite (PD) Kernel. These elements, however, provide a similarity metric for nodes that are determined as closer. K in Eq. (7) is a Conditional Positive Definite (CPD) kernel (Schölkopf 2000). K depends on D^*, thus each row of W is evaluated on independent CPD kernels. See Appendix C.2 for more details on the proposed CPD kernel.

4 Experiments

As mentioned in (Grattarola et al. 2022; Bianchi and Lachi 2024) some of pooling layers can directly specify the pooling ratio and some can't. SKIPPOOL belongs to the former category while SKIPPOOL-FULL belongs to the latter. Hence we evaluate both against different matching baselines separately, to answer the following questions: **(Q1)** How do both SKIPPOOL and SKIPPOOL-FULL perform against other SSP methods (eg: against TOPKPOOL or SAGPOOL)?, **(Q2)** How do both SKIPPOOL and SKIPPOOL-FULL perform against the broader realm of pooling methods (eg: against DIFFPOOL which is considered to be dense pooling methods)?, **(Q3)** How do SKIPPOOL and SKIPPOOL-FULL overcome the limitations of SSP methods with their skipping, i.e., sampling capabilities?

Datasets In order to analyze the ability to capture complex hierarchical structures we employ graphs from different domains chosen from 6 benchmark datasets. We use protein datasets: PROTEINS (Borgwardt et al. 2005; Dobson and Doig 2003), D&D (Dobson and Doig 2003; Shervashidze et al. 2011), chemical compound datasets: NCI1, NCI109 (Wale and Karypis 2006; Shervashidze et al. 2011), and FRANKENSTEIN (Orsini, Frasconi, and De Raedt 2015) dataset which contains a set of molecular graphs, and the social network dataset REDDIT-BINARY (Yanardag and Vishwanathan 2015). To answer Q3 we use 6 graphs of varying sizes, namely *Grid*, *Ring* extracted from the PyGSP (Defferrard et al. 2024) library and *Airplane*, *Car*, *Guitar*, *Person* from the ModelNet40 (Wu et al. 2015) dataset.

Model configurations To avoid performance boosts from factors external to pooling layers, we use modestly expressive GCN (Kipf and Welling 2017) layers as our message passing (MP) layers. SKIPPOOL is evaluated on the architecture presented in (Lee, Lee, and Kang 2019), and SKIPPOOL-FULL on the architecture similar presented in (Bacciu, Conte, and Landolfi 2023). Hyperparameter spaces considered for each experiment are in Appendix E. In each experiment, we use *stratified* train/val/test split, and specifications related to splitting/repetitions are in the appendix. Best models are selected using cross-validation on validation data. The number of total epochs is 100k unless specified otherwise, and all the models are trained with early stopping subject to patience criteria. The model training stops (early) if the validation loss doesn't improve for 50 epochs, and the model is restored to the epoch with the least validation loss on which test data is evaluated. In the temperature annealing, T_a, T_b, and R are set to 10,0.1,1k respectively. When the pooling ratio is 0.5 or 0.1 the stride length F for SKIPPOOL is 3, 11 respectively, and for the SKIPPOOL-FULL stride length that pools closest to the required output size is picked. For the scoring of nodes in SKIPPOOL and variants we used the scoring method used in SAGPOOL: $f(x_i) = p^\top \mathrm{GNN}(x_i)$.

Baselines SKIPPOOL is compared against global pooling SET2SET (Vinyals, Bengio, and Kudlur 2015), GLOBAL-ATTENTION (Li et al. 2016), SORTPOOL (Zhang et al. 2018) & hierarchical pooling DIFFPOOL, TOPKPOOL, SAGPOOL, ASAPPOOL all of which can specify a fixed ratio. SKIPPOOL-FULL against methods that can't specify a ratio like BDO, GRACLUS, EDGEPOOL, KMIS variants.

4.1 Results for Graph Classification

Hierarchical Pooling - Fixed Table 1 compares the performance of SKIPPOOL against baselines. The fixed pooling ratio is $r = 0.5$. We note that SKIPPOOL outperforms previous state-of-the-art methods, with an average improvement of 1.2% over ASAPPOOL and 4.5%, 3.5% over TOPKPOOL,SAGPOOL respectively, across the first 4 datasets.

Method	D&D	PROTEINS	NCI1	NCI109	FRANKENSTEIN
SET2SET$_g$	71.60±0.87	72.16±0.43	66.97±0.74	61.04±2.69	61.46±0.47
GLOBAL-ATT.$_g$	71.38±0.78	71.87±0.60	69.00±0.49	67.87±0.40	61.31±0.41
SORTPOOL$_g$	71.87±0.96	73.91±0.72	68.74±1.07	68.59±0.67	63.44±0.65
DIFFPOOL	66.95±2.41	68.20±2.02	62.32±1.90	61.98±1.98	60.60±1.62
TOPKPOOL	75.01±0.86	71.10±0.90	67.02±2.25	66.12±1.6	61.46±0.84
SAGPOOL	76.45±0.97	71.86±0.97	67.45±1.11	67.86±1.41	61.73±0.76
ASAPPOOL	**76.87±0.7**	**74.19±0.79**	**71.48±0.42**	**70.07±0.55**	**66.26±0.47**
SKIPPOOL	**77.18±0.48**	**74.76±0.47**	**73.67±0.51**	**72.23±0.73**	**62.33±0.61**

Table 1: Comparison of SKIPPOOL with previous global and hierarchical pooling. Subscript g denotes global poolings. Average accuracy is reported for 10 random seeds. The best performance among hierarchical poolings is **bolded**, and the next best is in **violet**.

Data Set	BDO	GRACLUS	EDGEPOOL	KMIS$_{avg}$	KMIS$_{max}$	SKIPPOOL-Full
D&D	76.69±1.79	75.17±2.11	74.70±1.57	73.56±1.19	**76.91**±1.06	**76.66**±0.94
REDDIT-B	85.63±1.43	84.05±5.81	85.98±1.57	86.98±1.13	**87.57**±1.96	**87.72**±1.66

Table 2: Comparison of SKIPPOOL-FULL with poolings that can not directly specify a pooling ratio r. KMIS$_{avg}$ & KMIS$_{max}$ are two variants of KMIS.

Even though SKIPPOOL has an average gain of 0.87%,0.5% over TOPKPOOL,SAGPOOL respectively on FRANKENSTEIN, there is a significant performance drop against ASAPPOOL on FRANKENSTEIN. ASAPPOOL gain this performance by becoming a dense pooling at will. We see at the first pooling layer ASAPPOOL outputs a denser graph with a node degree of 2, 3 times the node degree if it was to maintain the sparsity of the original graph, consequently at the last layer, graphs with 2-3 nodes become complete graphs. Nearly 88% of the dataset output graphs with 2-3 nodes at the last layer. Whereas SKIPPOOL, SAGPOOL & TOPKPOOL are *truly* sparse methods as they maintain the sparsity of the original graph.

Hierarchical Pooling - Full We test SKIPPOOL-FULL on 2 relatively large benchmark datasets, DD & REDDIT-BINARY. According to Table 2, SKIPPOOL-FULL achieves comparatively high accuracies against trainable sparse pooling methods KMIS(avg), KMIS(max) and GRACLUS. SKIPPOOL-FULL achieves the state-of-the-art performance on the REDDIT-BINARY dataset and achieves the second highest performance on D&D. Finding the best stride size F is more flexible than finding the best k in KMIS, especially if roughly a particular ratio of the original graph should be pooled. For example, suppose one wants to pool on average 50% of graphs in D&D, then in the case of KMIS, the closest result to this threshold is achievable by setting $k = 1$, which on average pool 32%, however in case of the SKIPPOOL-FULL one can expect somewhat closer ratio to the threshold by setting $F > 3$. For $F = 3$ pooled ratio is expected to be much higher but for $F = 5$ we see the pooled ratio reaches 53%. In general one can pool $\sim r$ using SKIPPOOL-FULL with $F > \frac{1}{r} + \lambda$ for $\lambda > 1$. Results in Table 2 are obtained with $F = 5$ in both datasets where average pooled ratios of 53% & 52% are obtained. The above results give favorable answers to Q1 & Q2.

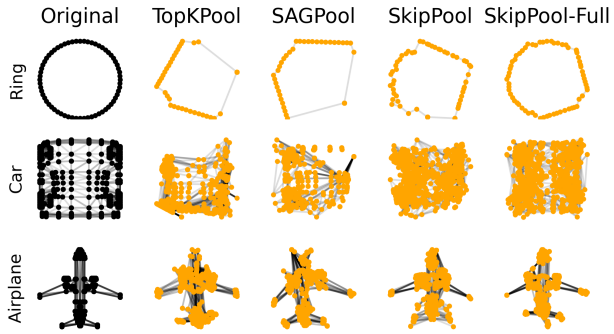


Figure 2: Nodes attributes (node locations) reconstruction. It's clear (strongly in Ring & Car) that SSP discards a portion of the graph due to ORR. Skipping can avoid ORR, resulting in more balanced pooling.

4.2 Preserving Node Attributes

We investigate SKIPPOOL and SKIPPOOL-FULL's ability to preserve node information to answer Q3. Following (Grattarola et al. 2022), we conduct a graph reconstruction task using a *Graph Autoencoder* (GAE) (Kipf and Welling 2016) to test the ability of different pooling methods to reconstruct the original graph from the pooled graph. If the pooling preserves more node information then the reconstruction loss (MSE) must be low. (Grattarola et al. 2022) showed that SSP methods discard important node information by cutting off entire portions of the original graph, leading to higher MSE. This confirms the drawbacks of the SSP we discussed. See Fig. 2. Lower MSE indicates better preservation of node information, resulting from more uniform balanced pooling over the nodes' signal space. Details related to GAE are in Appendix E.3.

MSE losses on standalone point clouds are shown in Table 3. We use baseline values γ to indicate if MSE $> \gamma$

	Grid2D	Ring	Airplane	Car	Guitar	Person	Rank
Ref. γ	78.12	48.15	0.97	0.28	0.91	0.13	
GRACLUS	1.09±0.00	6.00±0.00	0.09±0.00	1.02±0.00	0.10±0.00	0.01±0.00	-
DIFFPOOL	0.10±0.05	0.18±0.03	0.94±0.22	1.43±1.27	1.01±0.25	0.77±0.41	3.67
TOPKPOOL	188±39.2	1322±41.3	0.96±0.28	2.29±0.23	0.56±0.51	0.55±0.12	4.83
SAGPOOL	166±32.7	1485±301	2.68±0.81	2.04±0.29	0.60±0.44	0.62±0.33	5.33
ASAPPOOL	214±29.9	1346±42.5	3.97±1.21	3.39±1.27	0.61±0.43	1.60±0.31	6.5
SKIPPOOL	52.6±18.9	203.6±56.9	0.83±0.35	1.28±0.13	0.51±0.12	0.49±0.07	**3.17**
KMIS$_{avg}$	47.1±24.7	9.42±1.67	5.10±3.04	0.52±0.07	0.39±0.02	0.08±0.06	**2.83**
SKIPPOOL-FULL	39.0±3.40	33.8±17.1	0.31±0.12	0.24±0.03	0.25±0.07	0.27±0.16	**1.67**

Table 3: MSE values on scale of 10^{-4} in the GAE experiment. The Rank column indicates the average ranking of parameterized methods across all the graphs. Entries are Red where the MSE $> \gamma$.

then reconstructed points cannot be matched with original point clouds. We report MSE values of baseline methods and γ values from (Grattarola et al. 2022). SKIPPOOL outperforms its sparse counterparts i.e SAGPOOL, TOPKPOOL & ASAPPOOL in 5 out of 6 point clouds. SKIPPOOL surpass baseline values γ in 3 point clouds whereas SAGPOOL surpass γ only in Guitar cloud. Knowing that SKIPPOOL runs on top of SAGPOOL, we identify that the skipping technique employed in SKIPPOOL can reach regions in the original graph that SAGPOOL leave out. For example, there is an average 85% drop in MSE on Ring cloud for SKIPPOOL compared to its sparse counterparts. We see superior performance from SKIPPOOL-FULL as it outperforms all the baselines including the dense method DIFFPOOL. Further SKIPPOOL-FULL surpass γ in $5/6$ point clouds ranking no.1 among parameterized baselines. Performance of SKIPPOOL-FULL further confirms the efficacy of the skipping mechanism employed: when the restriction of obtaining an exact number of nodes is relaxed the skipping mechanism can adaptively reach regions that baselines tend to leave out.

4.3 Memory and Running Time

If the input node feature dimension is d, then SKIPPOOL & SKIPPOOL-FULL require $3d$ number of parameters wherein SAGPOOL & TOPKPOOL need d parameters and ASAPPOOL needs $2d^2 + 3d$ parameters. SKIPPOOL requires substantially lesser parameters compared to ASAPPOOL even when ASAPPOOL adheres to sparsity. Note that the increase in parameters against SAGPOOL is not substantial. Further,SKIPPOOL & SKIPPOOL-FULL are *CPU-bound* algorithms where the usage of CPU for sequential operations is much higher than overall GPU usage. Given all these sequential operations are on matrix indices, on which the gradient is not calculated, we can parallelize these operations. By resorting to multi-processing we can avoid SKIPPOOL & SKIPPOOL-FULL from incurring large execution times. Table 4 compares the execution time per epoch over different sizes of architectures confirming that the increase in time is not substantial. Sometimes we see SKIPPOOL is faster than ASAPPOOL mainly due to ASAPPOOL outputting denser graphs which incurs computation load on later graph convolutions, whereas sparse graphs from SKIPPOOL don't incur that much computation load.

Data Set	**Method**	$\mathbf{l = 1}$	$\mathbf{l = 2}$
DD	ASAP	1.23	2.45
	SKIP	1.97 (1.6x)	3.42 (1.4x)
	SKIP-FULL	2.11 (1.7x)	3.80 (1.6x)
RED.-B	ASAP	4.05	OOM
	SKIP	3.31 (0.8x)	5.8
	SKIP-FULL	3.62 (0.9x)	6.55

Table 4: Average runtime s/epoch. l: number of pooling & MP layers, OOM: Out of Memory. Within brackets mention: SkipPool's runtime as a fraction of ASAP's runtime.

5 Conclusion

Standard Sparse Pooling (SSP) is simple yet fails to give good hierarchical representations. SSP can avoid Over-Representative Regions (ORR) by enforcing the gathering of partitions together to form larger clusters. A proper node selection instead of Top-k that is robust at separating redundancies from useful nodes can serve the purpose. KMIS (Bacciu, Conte, and Landolfi 2023) works in this direction but with a lack of awareness of partitions, it resorts to a completely heuristic approach that searches for adequately spaced nodes in the domain (graph) rather than searching for adequately separated features in feature space, thus it is composed of many non-differentiable discrete operations.

We introduced a learnable selection mechanism on top of SSP that can avoid ORR by exploring beyond the capacity of SSP. Proposed methods show improvement in graph classification tasks due to their ability to preserve more graph-level information. The notion of ORR is revealed by shifting the perspective on importance scores as gating values to fit with a more generic definition of pooling. Our results indicate that mitigating ORR alone can improve SSP by a significant margin. Therefore, we argue that adding complexities to SSP, as in heavily parameterized approaches, is unnecessary. The focus should be on improving SSP's exploration to achieve better performance.

References

Bacciu, D.; Conte, A.; and Landolfi, F. 2023. Generalizing downsampling from regular data to graphs. In *Pro-*

ceedings of the AAAI Conference on Artificial Intelligence, 6718–6727.

Balin, M. F.; Abid, A.; and Zou, J. Y. 2019. Concrete Autoencoders: Differentiable Feature Selection and Reconstruction. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97, 444–453.

Bianchi, F. M.; and Lachi, V. 2024. The expressive power of pooling in graph neural networks. *Advances in Neural Information Processing Systems*, 36.

Borgwardt, K. M.; Ong, C. S.; Schönauer, S.; Vishwanathan, S. V. N.; Smola, A. J.; and Kriegel, H.-P. 2005. Protein function prediction via graph kernels. *Bioinformatics*, 21: 47–56.

Bronstein, M. M.; Bruna, J.; and Cohen. 2021. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*.

Cangea, C.; Veličković; Kipf, T.; and Liò, P. 2018. Towards sparse hierarchical graph classifiers. *arXiv preprint arXiv:1811.01287*.

Defferrard, M.; Martin, L.; Pena, R.; and Perraudin, N. 2024. PyGSP: Graph Signal Processing in Python.

Diehl, F. 2019. Edge contraction pooling for graph neural networks. *arXiv preprint arXiv:1905.10990*.

Dobson, P. D.; and Doig, A. J. 2003. Distinguishing Enzyme Structures from Non-enzymes Without Alignments. *Journal of Molecular Biology*, 330(4): 771–783.

Duvenaud, D. K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; and Adams, R. P. 2015. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28.

Gao, H.; and Ji, S. 2019. Graph u-nets. In *international conference on machine learning*, 2083–2092. PMLR.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*, 1263–1272. PMLR.

Grattarola, D.; Zambon, D.; Bianchi, F. M.; and Alippi, C. 2022. Understanding pooling in graph neural networks. *IEEE transactions on neural networks and learning systems*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*.

Kingma, D. P.; Salimans, T.; and Welling, M. 2015. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28.

Kipf, T. N.; and Welling, M. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

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

Lee, J.; Lee, I.; and Kang, J. 2019. Self-attention graph pooling. In *International conference on machine learning*, 3734–3743. PMLR.

Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. S. 2016. Gated Graph Sequence Neural Networks. In *4th International Conference on Learning Representations, ICLR 2016*.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *5th International Conference on Learning Representations, ICLR 2017*. OpenReview.net.

Maddison, C. J.; Tarlow, D.; and Minka, T. 2014. A* sampling. *Advances in neural information processing systems*, 27.

Orsini, F.; Frasconi, P.; and De Raedt, L. 2015. Graph invariant kernels. In *Proceedings of the 24th International Conference on Artificial Intelligence*, 3756–3762.

Raffel, C.; Luong, M.-T.; Liu, P. J.; Weiss, R. J.; and Eck, D. 2017. Online and linear-time attention by enforcing monotonic alignments. In *International conference on machine learning*, 2837–2846. PMLR.

Ranjan, E.; Sanyal, S.; and Talukdar, P. 2020. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *Proceedings of the AAAI conference on artificial intelligence*, 5470–5477.

Schölkopf, B. 2000. The Kernel Trick for Distances. In *Advances in Neural Information Processing Systems*.

Schütt, K.; Kindermans, P.-J.; Sauceda Felix, H. E.; Chmiela, S.; Tkatchenko, A.; and Müller, K.-R. 2017. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in neural information processing systems*, 30.

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.

Sutton, R. S. 1990. *Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming*, 216–224. Elsevier.

Vinyals, O.; Bengio, S.; and Kudlur, M. 2015. Order matters: Sequence to sequence for sets. *International Conference on Learning Representations (ICLR)*.

Wale, N.; and Karypis, G. 2006. Comparison of Descriptor Spaces for Chemical Compound Retrieval and Classification. In *Sixth International Conference on Data Mining (ICDM'06)*, 678–689.

Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; and Xiao, J. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1912–1920.

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. 2015. Deep Graph Kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1365–1374.

Ying, Z.; You, J.; Morris, C.; Ren; and Leskovec, J. 2018. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31.

You, J.; Ying, R.; and Leskovec, J. 2019. Position-aware graph neural networks. In *International conference on machine learning*, 7134–7143. PMLR.

Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An end-to-end deep learning architecture for graph classification. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press.