TopoPool: An Adaptive Graph Pooling Layer for Extracting Molecular and Protein Substructures

Mattson Thieme
Northwestern University,
AbbVie

Majdi Hassan
Mila Quebec,
AbbVie

Chetan Rupakheti
AbbVie

Kedar Thiagarajan
Northwestern University

Abhishek Pandey
AbbVie

Han Liu
Northwestern University

Abstract

Within molecules and proteins, discrete substructures affect high level properties and behavior in distinct ways. As such, explicitly locating and accounting for these substructures is a central problem when learning molecular or protein representations. Typically represented as graphs, this task falls under the umbrella of graph pooling, or segmentation. Given the highly variable size, number, and topology of these substructures, an ideal pooling algorithm would would adapt on a graph-by-graph basis and use local context to locate optimal pools. However, this poses a challenge where differentiability is concerned, and each of the *learnable* graph pooling methods proposed to date must make strong a priori assumptions in regards to the number or size of the learned pools. As such, demand remains for a graph pooling algorithm that can maintain differentiability while retaining adaptability in the size and number of learned pools. To meet this demand, we introduce the Topographical Pooling Layer (TopoPool): a differentiable, hierarchical graph pooling layer that learns an arbitrary number of varying sized pools without making any a priori assumptions about their number or size. Additionally, it naturally uncovers only connected substructures, increasing the interpretability of the learned pools and obviating the need for exogenous regularizers to enforce connectedness. We evaluate TopoPool on diverse molecular and protein property prediction tasks, where we achieve competitive performance against existing methods. Taken together, TopoPool represents a novel addition to the graph pooling toolbox and is particularly relevant to areas such as drug design, where locating and optimizing discrete, connected molecular substructures is of central importance.

1 Introduction

Graph neural networks (GNNs) have become the de facto models for learning representations of graph-structured data and have revolutionized structural biology (Senior et al., 2020), drug design (Gilmer et al., 2017), and molecular property prediction (Wu et al., 2019). To facilitate graph-level predictions, where node and edge features must be collapsed into a single graph-level feature, a wide variety of learnable graph coarsening methods have been introduced (Gao & Ji, 2021; Grattarola et al., 2021; Ying et al., 2018; Lee et al., 2019; Diehl, 2019) each with their own set of trade-offs. Heuristic pooling methods often fail to capture the complex structural information in the graph, while learnable methods are often expensive and sensitive to the specific choice of parameters. Learnable methods must also contend with maintaining differentiability. To do so, methods to date have had to make strong a priori assumptions about either the number or size of the learned pools, a severe

The NeurIPS 2023 Workshop on New Frontiers of AI for Drug Discovery and Development (AI4D3 2023), New Orleans, LA, USA, 2023.

disadvantage in domains where the optimal pool size is either unknown or variable across examples. The pharmacokinetic properties of molecules, for example, depend on discrete, connected molecular substructures that vary greatly in size, shape, and relative position within the molecule (Gasteiger, 2003). In this setting, an ideal graph pooling algorithm would be able to dynamically adapt to the input graphs and learn both the size and the number of clusters on a graph-by-graph basis.

In this work, introduce the Topographic Pooling layer (TopoPool). TopoPool is the first differentiable graph pooling layer that 1) dynamically adapts to the input graphs and learns the optimal number and size of pools on a graph-by-graph basis. TopoPool also naturally locates only *connected* substructures, obviating the need for exogenous regularizers and loss terms required by other learnable methods to enforce the notion that nearby nodes should be pooled together. To our knowledge, as of the time of writing, no other graph pooling approaches are adaptive in this way. Each either sets an upper limit on the number of clusters (Ying et al., 2018; Jo et al., 2021) samples a pre-defined number/ratio of ranked nodes or edges (Ranjan et al., 2019; Diehl, 2019; Jo et al., 2021; Gao & Ji, 2021; Baek et al., 2021), and requires exogenous regularizers to enforce connectivity in the learned pools. Our contributions are as follows:

- We introduce TopoPool, the first hierarchical graph pooling algorithm that pools entire graphs without making assumptions about the number or size of the learned pools.
- In experiments on real-world molecular datasets, where graph pooling is particularly challenging, we demonstrate competitive performance against the existing approaches.
- We provide an efficient PyTorch implementation of the TopoPool layer that can be easily integrated into existing GNN pipelines.

Our choice to focus on molecular datasets was motivated by the field of medicinal chemistry, where the presence and relevance of pharmacophores (Güner & Bowen, 2014; Schueler, 1961) serves as a prime example of the importance of preserving connected substructures in graph-based molecular representations. Pharmacophores are something like molecular phonemes: canonical ensembles of atoms used to construct larger molecules. The particular structure and position of the pharmacophores within a molecule is critical for determining a molecules' biological activity (Putz et al., 2016). Thus, understanding how these substructures affect pharmacokinetic properties of a molecule is crucial for drug discovery, design, and optimization (Seidel et al., 2020).

2 Preliminaries

We denote a graph G with nodes $V \in \{v_0, \ldots, v_N\}$ and edge set $\mathcal{E}$ as $G(V, \mathcal{E})$. Graph pooling, or coarsening, is the process of mapping $|V| = N$ nodes and onto a new set of nodes $|V'| = M$, where $M \leq N$. To accomplish this, graph pooling operators possess some combination of the following three functions: selection, reduction, and connection (SRC) (Grattarola et al., 2021). Selection is process of grouping the input nodes, reduction is process of aggregating and compressing those groups into new representations, and connection is the process of reconnecting the clustered nodes given the original edge set $\mathcal{E}$. The algorithmic contributions of TopoPool primarily lie in the *Selection* phase, but we also perform reduction and connection to form a fully self-contained graph pooling operator.

Being a differentiable graph pooling layer, TopoPool is designed to be used alongside GNN layers. However, TopoPool is agnostic to the particular GNN aggregation mechanism, and we can abstract this portion away, denoting an arbitrary GNN layer as GNN$(V, \mathcal{E}) : x \in \mathbb{R}^{N \times d} \rightarrow \mathbb{R}^{N \times d'}$, where d' is the dimension of the transformed node representation. As outlined in the following section, it is these transformed node representations, alongside the edge set, that the TopoPool layer ingests.

3 The Topographical Pooling Layer (TopoPool)

The TopoPool layer is a differentiable, hierarchical graph pooling algorithm for locating and aggregating an arbitrary number of discrete, connected substructures in an input graph. It was motivated by the desire to locate and extract pharmacophores from molecular graphs, and inspired by the clean boundaries described in topographic maps. In this section, we describe how the TopoPool layer differentiably coarsens a graph without making any assumptions about the number or size of the clusters.

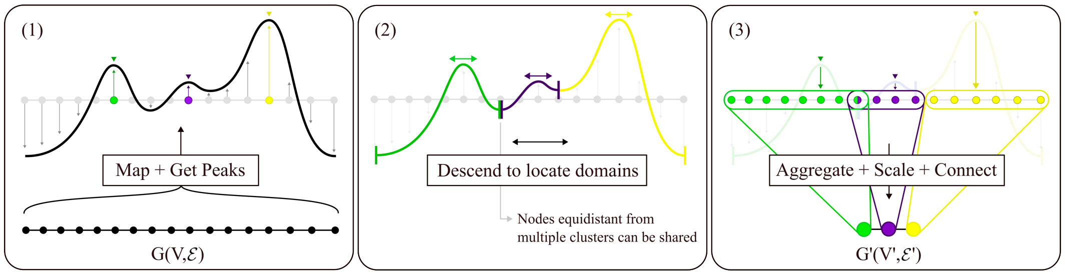
(1)
Map + Get Peaks
G(V,$\mathcal{E}$)
(2)
Descend to locate domains
Nodes equidistant from multiple clusters can be shared
(3)
Aggregate + Scale + Connect
G'(V',$\mathcal{E}$')

Figure 1: A high level overview of the TopoPool algorithm. In panel (1) we generate the scores which, in conjunction with the graph structure, define the 'topography' over the graph. Panel (2) shows how, once we've found the peaks in the topography, the pools are located by descending from the peaks until we reach a trough. This allows the layer to locate pools of variable size that are necessarily connected. Finally, panel (3) shows how the pooled clusters are reduced to the coarsened graph G'. Note that we visualize the algorithm here on a 1D chain graph for clarity only. The TopoPool algorithm can be applied on graphs with arbitrary topologies.

TopoPool takes as input a vector of node representations $h \in \mathbb{R}^{N\times d}$ and the edge set $\mathcal{E}$, and yields a new graph with nodes V' with features $h' \in \mathbb{R}^{K\times d}$ and edges $\mathcal{E}'$, where K is the number of learned clusters.

Select: The first step in the TopoPool algorithm is to generate the topography (Figure 1, (1)). Using a single linear layer $f_\theta(\cdot) \in \mathbb{R}^{d\times 1}$, TopoPool maps the node representations $h \in \mathbb{R}^{N\times d}$ onto a scalar valued graph signal $f_\theta(h) = s \in \mathbb{R}^{N\times 1}$. These scalar values are what define the graph signal that, in conjunction with the network structure, we treat like a topography over the graph. Once s is generated, we locate all the peak nodes with a simple message passing operation, returning `True` in a binary mask for all $s_i : s_i > s_j, \forall j \in \mathcal{N}_i$, where $\mathcal{N}_i$ is the one-hop neighborhood of v_i. We denote the set of all such local peaks as P, where $|P| = K, K \leq N$, and each peak node $v_k \in P$ defines the source of its own unique cluster. Then, from each source node v_k, we *expand and descend*, growing each cluster outward while the neighboring node scores $s_j \leq s_k$ (Figure 1, (2)). We refer to the new clustered nodes by their peak node of origin, k, and accumulate constituent members of cluster k in a set C_k. In this descent process, nodes are assigned to clusters on a first-come, first-served basis. However, if some node is equidistant from multiple peaks, and therefore reached at the same iteration in the descent phase, it is assigned to both clusters, as shown in Figure 1, panel (2). Experimentally, we found that allowing multiple clusters to share nodes does not impact performance. Therefore, for simplicity, we do not penalize the double counting of these shared nodes.

Reduce: Once clusters have been located and every node assigned, members of each cluster k are aggregated (Figure 1, (3)) with a max pool over their features, yielding a representation $h_k = \max_i(h_{i\in C_k})$. Each cluster representation h_k is then scaled by the score s_k on its peak node v_k. This step is important for a few reasons: 1) it allows gradients to flow back to the linear layer $f_\theta(\cdot)$, 2) it improves prediction performance by scaling the influence of each cluster, and 3) it adds a degree of interpretability be clearly showing the relative differences in importance between the learned clusters. To ensure consistency across graphs with varying numbers of clusters, we normalize these peak scores s_k over each graph with a softmax, such that the final representation for the k^{th} pool becomes:

$$h_k = \frac{e^{s_k}}{\sum\limits_{i\in P} e^{s_i}} \cdot \max_i(h_{i\in C_k}), \quad h_k \in \mathbb{R}^{1\times d} \tag{1}$$

And the node representations for the final pooled graph are the concatenation of all the pooled representations: $h' \in \mathbb{R}^{K\times d}$.

Connect: To connect the learned clusters and generate the new edge set $\mathcal{E}'$, we simply draw an undirected edge between two clusters if there existed at least one edge between their constituent nodes in the input edge set $\mathcal{E}$.

A note on scalability: The TopoPool layer is inherently scalable as complexity of the *expand* portion algorithm is the same as BFS. The cost for this step is $\mathcal{O}(N + E)$, where N and E are the number of nodes and edges in each graph, respectively. We note however that this is the worst case, where only

Dataset	Task	Graphs	Nodes	Edges	Features	Classes
PPBR	Regression	1,614	28.8	91.7	39	1
Caco-2	Regression	906	29.3	92.4	39	1
MUTAG	Classification	188	17.9	39.6	7	2
PROTEINS	Classification	1,113	39.1	145.6	3	2
ENZYMES	Classification	600	32.6	124.3	3	6

Table 1: Molecular property prediction datasets used in our experiments. The 'Nodes' and 'Edges' columns reflect the average number of nodes and edges per graph.

a single cluster is learned. In most cases, the actual runtime will be $\mathcal{O}(C_{MAX} + E)$, where C_{MAX} is the size of the largest learned cluster, and typically $C_{MAX} << N$. This runtime is on par with benchmark graph pooling methods (Section 4.2) and ours runs in comparable time.

4 Experiments

In our experiments, we aim to answer the following questions:

Q1 - Is TopoPool competitive with existing learnable and heuristic graph pooling operators?

Q2 - Can the TopoPool layer be easily integrated into existing GNN pipelines?

Q3 - Are the learned clusters interpretable and consistent across examples?

Following the example of (Lee et al., 2019; Ying et al., 2018; Diehl, 2019; Ranjan et al., 2019), we benchmark our model on molecular and protein property prediction tasks. High level descriptions of the datasets can be found in Table 1.

High level descriptions of the datasets can be found in Table 1.

Two of our datasets, PPBR and Caco-2, come from the Therapeutic Data Commons (TDC) (Huang et al., 2022), a cross-modality repository of therapeutic data designed to evaluate AI capabilities across the stages of discovery. Molecules in each dataset are stored as SMILES strings and converted to a graph structure before training using the RDKit library. As edges in molecular graphs represent bonds, they are all undirected. Node features in the TDC datasets are a concatenation of 1-hot vectors expressing the atom type, degree, formal charge, chiral tag, and aromaticity; 39 features in all per node. For both datasets, we report the mean absolute error (MAE) on the test set.

Additionally, as the real-world drug discovery process yields increasingly diverse and increasingly novel compounds over time (Feinberg et al., 2020) our objective must be to learn a model that generalizes not only to new molecules, but to molecules composed of unseen substructures. Therefore, to make our evaluations as realistic as possible, we use the canonical scaffold splits (defined in the TDC databases for all tasks) that force the training and testing sets to have maximally dissimilar molecular structures.

PPBR (noa) stands for Plasma Protein Binding Rate, and the PPBR dataset labels drug molecules by the percentage of that drug bound to plasma proteins in the blood. This rate inversely affects the efficiency with which a drug is delivered to a site of action and is a measure of *distribution*, or how a drug moves between the various tissues of the body.

Caco-2 (Wang et al., 2016) labels drug molecules by the rate of the drug passing through Caco-2 cells (a human colon epithelial cancer cell line) and approximates the rate at which the drug permeates through intestinal tissue. Fundamentally, this expresses a measure of *absorption* of a drug molecule.

MUTAG: (Debnath et al., 1991) is a widely used chemical dataset in which each molecule is labeled as either mutagenic or non-mutagenic, representing whether the compound has a mutagenic effect on the germ cells of the bacterium Salmonella typhimurium.

PROTEINS (Borgwardt et al., 2005) is a dataset of protein graphs labeled as either enzymes or non-enzymes. In contrast to the other datasets, nodes in PROTEINS represent amino acids and two nodes are connected (again via undirected edges) if the amino acids are less than 6 Angstroms apart.

ENZYMES: (Borgwardt et al., 2005) Like the proteins dataset, the ENZYMES dataset consists of protein structures. Unlike the other tasks, the ENZYMES dataset is multiclass classification problem. Each protein belongs to one of six enzyme classes based on the chemical reactions they catalyze. Given its complexity and real-world relevance, the ENZYMES dataset is often used to evaluate the performance of graph-based algorithms, particularly GNNs. The multiclass nature of the classification task also makes it a useful dataset for testing the ability of the TopoPool layer to handle more complex tasks.

4.1 Model Configuration

While the TopoPool layer is capable of hierarchical pooling, we demonstrate it here as a single pooling layer following graph convolution layers. We do for simplicity's sake as adding another pooling layer added complexity without materially improving performance. The base model, as well as all training and hyperparameters, are identical in all our experiments, with only the final pooling layer being swapped out. The GNN layers in our model are based on the GCN architecture (Kipf & Welling, 2016), as it is widely used, efficient and performed well in our experiments. However, we note that the TopoPool layer can be applied in conjunction with any GNN aggregation layer, and additionally report unoptimized performance results with GAT (Veličković et al., 2017) and GraphSAGE (Hamilton et al., 2017) as the backbone in Figure 2. Our model architecture is straightforward, with three graph convolution layers, a single graph pooling layer, and three linear readout layers to map the learned representation onto the final prediction. As it is well known that incorporating global information helps in pharmacokinetic prediction models (Feinberg et al., 2020) we concatenate node features in each layer with an average pool over the batch dimension. We found that the addition of LayerNorm (Ba et al., 2016) after each convolutional layer helped to stabilize training by stabilizing the gradients flowing to the linear layer $f_\theta(\cdot)$. After each convolutional layer, we concatenate batch-wise max and average pools over the node features. These concatenated features act like skip connections and are summed together to form the input to the final readout layers.

Training is performed with the Adam Optimizer (Kingma & Ba, 2014), learning rate 5e-3, learning rate decay, and patience of 200 epochs. We use a batch size of 128 for all experiments and randomly shuffle the training and validation sets. Each model configuration was tested over ten independent trials and we report both the average and standard deviation of the performance over these trials in Table 2. All training and model details are available in our GitHub[1].

4.2 Benchmark Methods

Graph Pooling methods can be broadly categorized into global or hierarchical pooling. Global methods produce graph-level features by aggregating node-level, and occasionally edge-level, features across the entire graph, usually via a sum, mean, or max aggregation. On the other hand, hierarchical pooling methods implement some version of the SRC framework (Grattarola et al., 2021), reducing a graph to one with fewer nodes and edges, rather than collapsing it into a single graph-level feature and discarding the edges.

To date, all graph pooling algorithms, including learnable algorithms, depend on some heuristics to specify the size of the clusters, number of clusters, or cluster thresholds (Lee et al., 2019; Gao & Ji, 2021; Ying et al., 2018; Diehl, 2019; Grattarola et al., 2021). Unfortunately, this is antithetical to need to locate clusters of unknown size and shape, such as those corresponding to pharmacophores in a drug molecule. In our experiments, we compare TopoPool against a range of heuristic and learnable graph pooling operators outlined below.

No Pool refers to a simple GNN model in which the final graph-level representation is obtained with a simple max pool over all the learned node representations.

TopK (Gao & Ji, 2021) The authors propose a graph pooling method based on the U-Net architecture (Ronneberger et al., 2015), a popular model in the field of semantic segmentation. The method constructs hierarchical graphs and captures the corresponding nodes' representations at different levels. Downsampling is performed by the TopK pooling layer, which drops nodes based on a learnable projection score and a hyperparameter that specifies the ratio of nodes to retain.

[1]https://github.com/MattsonThieme/TopoPool

Base	Pooling	Dataset				
		ENZYMES(↑)	PROTEINS(↑)	MUTAG(↑)	Caco2(↓)	PPBR(↓)
GCN	Global	0.410(0.094)	0.727(0.055)	0.732(0.095)	0.444(0.083)	8.92(0.33)
	TopK	0.395(0.081)	0.746(0.029)	0.689(0.128)	0.407(0.097)	8.88(0.32)
	SAG	0.405(0.059)	0.727(0.032)	0.705(0.100)	0.412(0.070)	8.85(0.31)
	DiffPool	0.410(0.079)	0.737(0.023)	0.705(0.155)	0.389(0.044)	8.86(0.53)
	EdgePool	0.403(0.091)	**0.762(0.039)**	0.768(0.042)	0.395(0.061)	8.78(0.35)
	ASAPool	0.392(0.052)	0.750(0.037)	0.774(0.094)	0.455(0.139)	8.78(0.61)
	TopoPool	**0.427(0.080)**	0.759(0.024)	**0.779(0.131)**	**0.382(0.046)**	**8.63(0.43)**
GAT	***TopoPool***	*0.433(0.032)*	*0.771(0.037)*	*0.753(0.120)*	*0.450(0.055)*	*8.70(0.34)*
SAGE	***TopoPool***	*0.438(0.051)*	*0.747(0.041)*	*0.811(0.120)*	*0.376(0.031)*	*8.96(0.21)*

Table 2: Performance on the held out test set, best is shown in bold and second best is underlined. For ENZYMES, PROTEINS and MUTAG, we report the classification accuracy on the test set, and for Caco2 and PPBR we report the MAE. Reported values reflect the average and standard deviation in the performance over 10 independent training runs. Performance results with GAT and GraphSAGE as the GNN backbone are shown in italics as hyperparameters for these model configurations were not optimized for the given endpoints. We present them to show that TopoPool can be used with arbitrary GNN layers, as well as to demonstrate the reasonable performance TopoPool can achieve even before hyperparameter optimization.

SAG (Lee et al., 2019), similar to TopK, the SAG pooling layer identifies the most informative nodes by assigning an importance score to each node in a graph using self-attention, then selecting the nodes with the highest scores to form a coarsened graph. Here again, the value of k is a hyperparameter that must be specified a priori.

DiffPool (Ying et al., 2018) is a differentiable graph coarsening method that performs a soft cluster assignment of the nodes to a predetermined number of clusters. While DiffPool is capable of using only a subset of those predetermined number of nodes, the upper limit must still be specified a priori. Additionally, as there are no architectural properties enforcing the notion that nearby nodes be clustered together, they also implement a supplementary link-prediction objective to encourage this behavior.

EdgePool (Diehl, 2019) introduces a sparse, learnable pooling method for graphs based on edge contraction. They first compute a score for each edge, then sort all edges by their score, successively choosing the edges with the highest scores whose two nodes have not yet been part of a contracted edge. While effective, the authors note the disadvantage that each step roughly halves the number of nodes in the graph, and that this cannot be modified by the user.

ASAPool (Ranjan et al., 2019) introduces a novel GNN-based attention network to generate importance scores for each node in a given graph. Similar to DiffPool, it then learns a soft cluster assignment for nodes at each layer and uses these assignments to locate and pool subgraphs. However, while this addresses some issues in the graph pooling literature, it still samples substructures using a variant of top-k selection, where the pooling ratio k must specified a priori.

5 Results and Discussion

Table 2 contains performance results for the benchmark methods and TopoPool on our datasets. Values for GAT/SAGE + TopoPool are shown in italics as these were not optimized for the given configuration. We present them simply to demonstrate the capacity to implement TopoPool with any GNN layer, as well as the ability for TopoPool to achieve reasonable performance even before hyperparameter optimization.

To answer the questions set forth in Section 4:

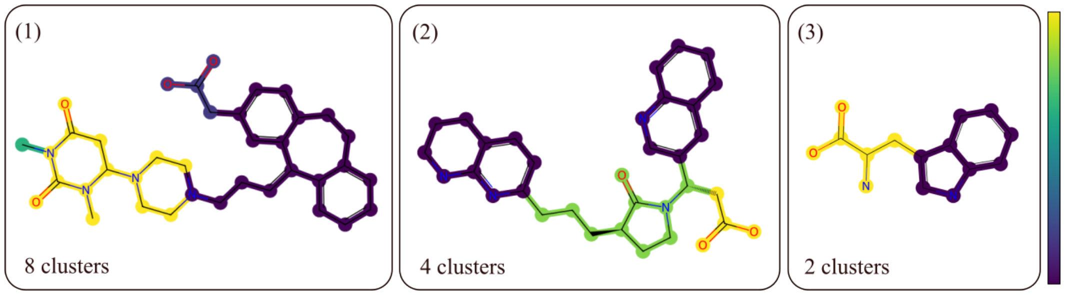

Figure 2: Example clusters learned on the Caco2 Permeability endpoint. Each cluster is colored according to the score s_k on its peak node v_k, which reflect the relative importance of each cluster to the given endpoint. We note the clearly delineated clusters that cleanly subsume discrete molecular structures, as well as the consistency of the pools across molecules.

A1 - Given the performance in Table 2, TopoPool is very competitive with the existing learnable and heuristic graph pooling algorithms, achieving superior performance on all but one dataset.

A2 - TopoPool can be easily integrated into existing GNN pipelines, and we report good performance using a GAT and GraphSAGE backbone even before hyperparameter optimization in these configurations.

A3 - As we discuss below and show in Figure 2, the learned clusters are highly interpretable while also being consistent across examples.

Performance: On all but one dataset, TopoPool achieves superior performance to every benchmark method, and on PROTEINS it achieves only a slightly lower accuracy than EdgePool. Additionally, standard deviations in the performance are in line with all the other methods. Also worth noting is that methods like TopK, which retain only the K highest ranked nodes, often lead to even poorer performance than global pooling, implying that there is utility in retaining information from every node.

Training dynamics: Like all learnable graph pooling methods, the exact pools extracted will depend on the particular initialization. However, we do observe that the *relative* ranking of the substructures, for example, the carboxyl group (the trigonal planar functional group colored yellow in (2) and (3)) vs. the aromatic rings (the carbon rings colored purple in all three molecules) is preserved across most of the independent trials.

In regards to training time, our implementation uses advanced indexing in PyTorch for all but the *expand and descend* operation, resulting in training times that are comparable with those of DiffPool and EdgePool. The number of clusters learned is also consistent across independent trials, with the model gradually converging on a similar number as training progresses.

Cluster Interpretability: We can see that clusters are clearly delineated, subsume discrete molecular structures, and similar structures are scored similarly across molecules. We note that the two aromatic rings on either side of the molecule in (2), while scored and colored similarly, are distinct clusters. Also of note is that the relative importance of the carboxyl group is conserved across (most, not all) molecules in the Caco2 Permeability dataset. The relative differences in the scores of similar structures across examples is not necessarily of concern, as these scores are relative *within* a given molecule, and the structural context for each molecule will be different.

Consistency across examples: As we can see in Figure 2, the aromatic rings are, for the most part, scored lower than the substructures containing peripheral oxygens. This is consistent across examples in the test set for this endpoint, and we observe a similar behaviors across Caco2 and PPBR (the two molecular datasets).

Areas for improvement: Learning a variable number of nodes of variable size presents challenging normalization problems, particularly as it relates to how the node representations within a given cluster are aggregated. We tried a number of methods with varying success across datasets. We arrived at the MAX aggregation, as noted in Equation (1), as it achieved good average results across our datasets. However, we note that this is a design choice that can and should be adjusted based on the unique demands of each dataset. Additionally, when considering molecules, there are additional symmetries that the algorithm may be able to exploit. As it stands, the TopoPool algorithm originates a unique cluster from each local maxima in the node scores s. However, due to common symmetries in many molecules, this often results in structures with bilateral symmetry being split into two symmetrical pools. It may improve performance to, for example, merge symmetric, adjacent pools into a single pool. Finally, the learned scores are sensitive to initialization. However, the learned clusters appear to be less so. As our model has a readout layer that maps the clustered representations onto a single value, the scores on each learned cluster have to adapt alongside the weights in the readout layers and will remain sensitive to initialization. This is a limitation of the method and a problem worthy of further study.

6 Conclusion

Here, we presented TopoPool, the first learnable, hierarchical graph pooling layer capable of coarsening an entire graph without making assumptions about the size or number of clusters. It also innately uncovers connected substructures, improving interpretability while obviating the need for exogenous forcing functions or regularizers to ensure connectedness within the pools. We demonstrated its efficacy on real-world molecular and protein property prediction datasets, where it outperformed existing graph pooling algorithms on all but one. We additionally showed that even un-optimized implementations can achieve decent performance. Finally, we provide an efficient, open source implementation of the algorithm built with PyTorch Geometric (Fey & Lenssen, 2019). Given its differentiability, novelty and efficacy, we believe that TopoPool represents a useful addition to the GNN toolkit and a step towards improving molecular and protein representations.

References

Document: Experimental in vitro DMPK and physicochemical data on a set of publicly disclosed compounds. `https://www.ebi.ac.uk/chembl/document_report_card/CHEMBL3301361/`. Accessed: 2023-5-16.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. July 2016.

Jinheon Baek, Minki Kang, and Sun Ju Hwang. Accurate learning of graph representations with graph multiset pooling, June 2021.

Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, S V N Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21 Suppl 1: i47–56, June 2005.

A K Debnath, R L Lopez de Compadre, G Debnath, A J Shusterman, and C Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34(2):786–797, February 1991.

Frederik Diehl. Edge contraction pooling for graph neural networks. *arXiv.org*, 2019.

Evan N Feinberg, Elizabeth Joshi, Vijay S Pande, and Alan C Cheng. Improvement in ADMET prediction with multitask deep featurization. *Journal of medicinal chemistry*, 63(16):8835–8848, August 2020.

Matthias Fey and J E Lenssen. Fast graph representation learning with PyTorch geometric. *ArXiv*, 2019.

Hongyang Gao and Shuiwang Ji. Graph U-Nets. *IEEE transactions on pattern analysis and machine intelligence*, PP, May 2021.

J Gasteiger. *Chemoinformatics*. Wiley-VCH, Weinheim, 2003.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. April 2017.

Daniele Grattarola, Daniele Zambon, Filippo Maria Bianchi, and Cesare Alippi. Understanding pooling in graph neural networks. October 2021.

Osman Güner and J Philip Bowen. Setting the record straight: the origin of the pharmacophore concept. 54, 2014.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. June 2017.

Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Artificial intelligence foundation for therapeutic science. *Nature chemical biology*, 18(10):1033–1036, October 2022.

Jaehyeong Jo, Jinheon Baek, Seul Lee, Dongki Kim, Minki Kang, and Sung Ju Hwang. Edge representation learning with hypergraphs. June 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. December 2014.

Thomas N Kipf and Max Welling. Semi-Supervised classification with graph convolutional networks. September 2016.

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-Attention graph pooling. April 2019.

Mihai V Putz, Corina Duda-Seiman, Daniel Duda-Seiman, Ana-Maria Putz, Iulia Alexandrescu, Maria Mernea, and Speranta Avram. Chemical structure-biological activity models for pharmacophores' 3d-interactions. *International journal of molecular sciences*, 17(7):1087, 2016.

Ekagra Ranjan, Soumya Sanyal, and Partha Pratim Talukdar. ASAP: Adaptive structure aware pooling for learning hierarchical graph representations. November 2019.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. May 2015.

Fred Warren Schueler. Chemobiodynamics and drug design. *Academic Medicine*, 36(3):285–286, 1961.

Thomas Seidel, Oliver Wieder, Arthur Garon, and Thierry Langer. Applications of the pharmacophore concept in natural product inspired drug design. *Molecular Informatics*, 39(11):2000059, 2020.

Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander W R Nelson, Alex Bridgland, Hugo Penedones, Stig Petersen, Karen Simonyan, Steve Crossan, Pushmeet Kohli, David T Jones, David Silver, Koray Kavukcuoglu, and Demis Hassabis. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, January 2020.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. October 2017.

Ning-Ning Wang, Jie Dong, Yin-Hua Deng, Min-Feng Zhu, Ming Wen, Zhi-Jiang Yao, Ai-Ping Lu, Jian-Bing Wang, and Dong-Sheng Cao. ADME properties evaluation in drug discovery: Prediction of caco-2 cell permeability using a combination of NSGA-II and boosting. *Journal of chemical information and modeling*, 56(4):763–773, April 2016.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. January 2019.

Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 4805–4815, Red Hook, NY, USA, December 2018. Curran Associates Inc.