Haar Graph Pooling

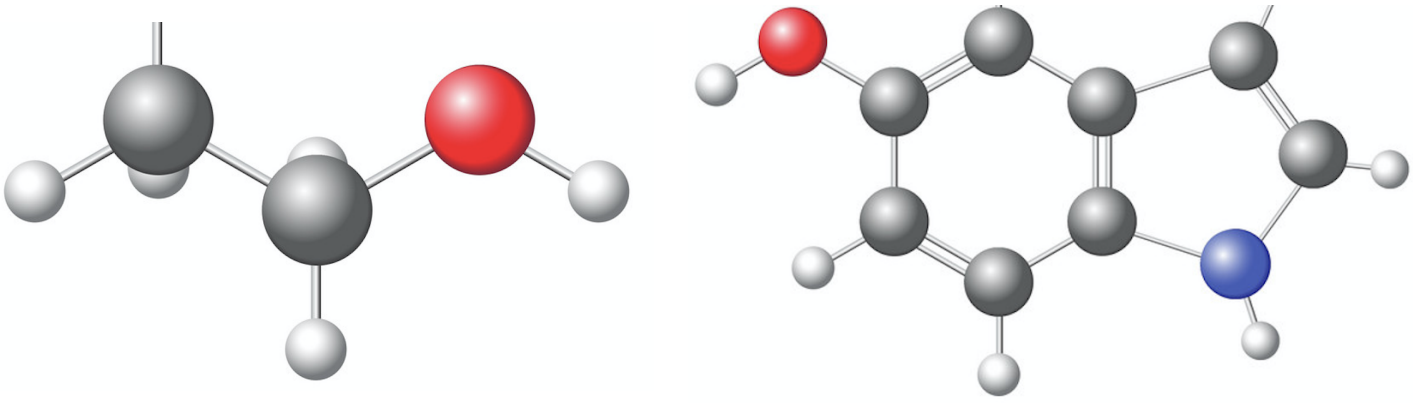
Yu Guang Wang

UNSW/MPI
yuguang.wang@mis.mpg.de

Yanan Fan (UNSW) Ming Li (ZJNU) Zheng Ma (Princeton)
Guido Montúfar (UCLA/MPI) Xiaosheng Zhuang (CityU HK)

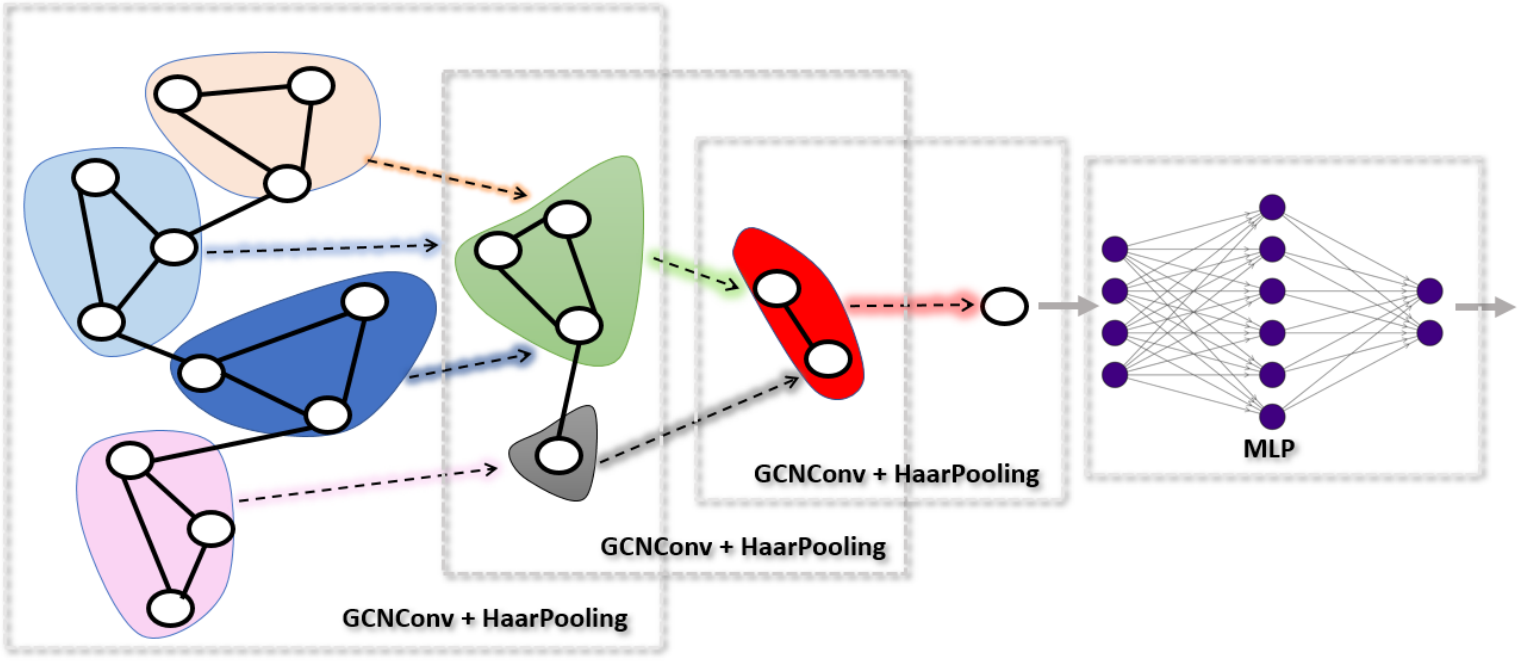
ICML 2020

Graph Classification on Quantum Chemistry



Graph-structured data: two molecules with atoms as nodes and bonds as edges. The number of nodes of each molecule is different and each has its own molecular structure. The input data set in graph classification or regression is a set of pairs of such individual graph and the feature defined on the graph nodes.

Graph Neural Networks



Deep graph neural networks (GNNs) are designed to work with graph-structured inputs. A GNN is typically composed of multiple *graph convolution* layers, *graph pooling* layers, and fully connected layers. Computational flow of a Graph Neural Network consisting of three blocks of GCN graph convolutional and HaarPooling layers, followed by an MLP. In this example, the output feature of the last pooling layer has dimension 4, which is the number of input units of the MLP.

Extracting Structural Information by Graph Convolution

- Spatial-based Graph Convolution A typical example is the widely used GCNConv, proposed by *Kipf & Welling (2017).*

$$X^{\text{out}} = \widehat{A} X^{\text{in}} W.$$

- Here $\widehat{A} = \widetilde{D}^{-1/2}(A + I)\widetilde{D}^{-1/2} \in \mathbb{R}^{N \times N}$ is a normalized version of the adjacency matrix A of the input graph, where I is the identity matrix and $\widetilde{D}$ is the degree matrix for $A + I$.
- Further, $X^{\text{in}} \in \mathbb{R}^{N \times d}$ is the array of d-dimensional features on the N nodes of the graph, and $W \in \mathbb{R}^{d \times m}$ is the filter parameter matrix.

How GNNs handle input graphs with varying number of nodes and connectivity structures?

- One way is to utilize graph pooling. It is a computational strategy to reduce the number of graph nodes while preserve as much as geometric information of the original input graph data; in this way, one has a unified graph-level rather than node-level representation for graph-structured data while the size and topology of an individual graph are changing.

Haar Graph Pooling

HaarPooling provides cascading pooling layers, i.e., for each layer, we define an orthonormal Haar basis and its compressive Haar transform. Each HaarPooling layer pools the graph input from the previous layer to output with a smaller node number and the same feature dimension. In this way, all the HaarPooling layers together synthesize the features of all graph input samples into feature vectors with the same size. We then obtain an output of a fixed dimension, regardless of the size of the input.

Definition

The HaarPooling for a graph neural network with K pooling layers is defined as

$$X_j^{\text{out}} = \Phi_j^T X_j^{\text{in}}, \quad j = 0, 1, \ldots, K-1,$$

where Φ_j is the $N_j \times N_{j+1}$ compressive Haar basis matrix for the jth layer, $X_j^{\text{in}} \in \mathbb{R}^{N_j \times d_j}$ is the input feature array, and $X_j^{\text{out}} \in \mathbb{R}^{N_{j+1} \times d_j}$ is the output feature array, for some $N_j > N_{j+1}$, $j = 0, 1, \ldots, K-1$, and $N_K = 1$. For each j, the corresponding layer is called the jth HaarPooling layer.

Haar Graph Pooling

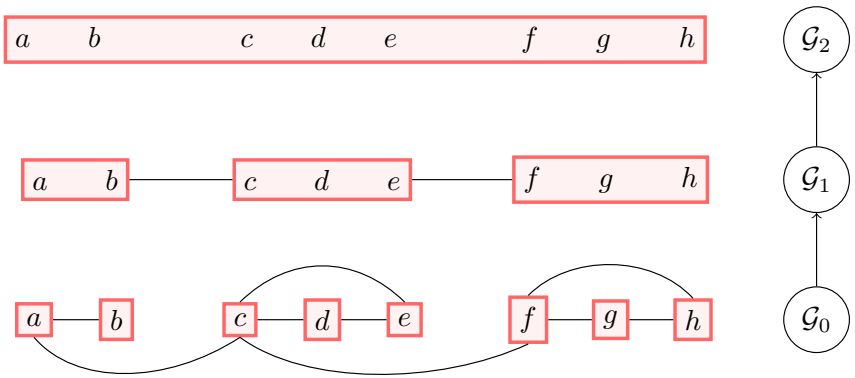
$$X_j^{\text{out}} = \Phi_j^T X_j^{\text{in}}, \quad j = 0, 1, \ldots, K-1.$$

First, the HaarPooling is a hierarchically structured algorithm, and has a global design. The coarse-grained chain determines the hierarchical relation in different HaarPooling layers. The node number of each HaarPooling layer is equal to the number of nodes of the subgraph of the corresponding layer of the chain. As the top-level of the chain can have one node, the HaarPooling finally reduces the number of nodes to one, thus producing a fixed dimensional output in the last HaarPooling layer.

The HaarPooling uses the sparse Haar representation on chain structure. In each HaarPooling layer, the representation then combines the features of input X_j^{in} with the structural information of the graphs of the jth and $(j+1)$th layers of the chain.

By the property of the Haar basis, the HaarPooling only drops the high-frequency information of the input data. The X_j^{out} mirrors the low-frequency information in the Haar wavelet representation of X_j^{in}. Thus, HaarPooling preserves the essential information of the graph input, and the network has small information loss in pooling.

Chain

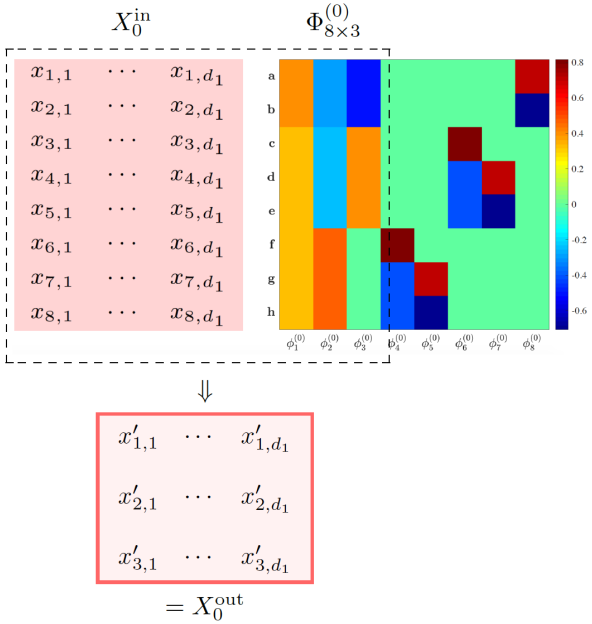


- Based on chain

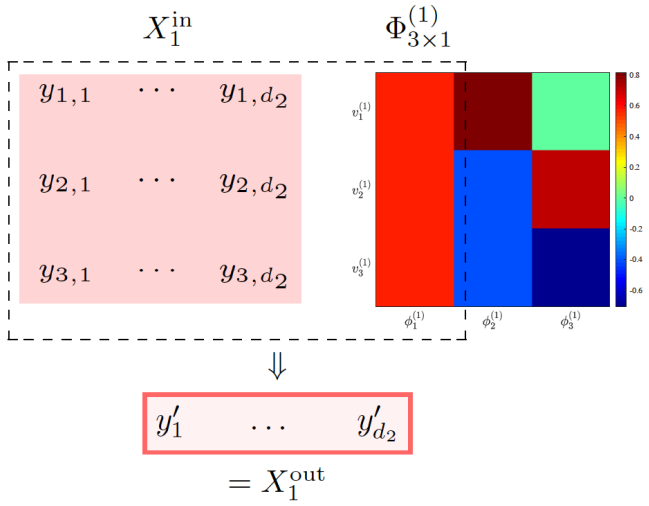
$$\mathcal{G}_{J_0 \to J} = (\mathcal{G}_{J_0}, \ldots, \mathcal{G}_J)$$

- Chain by clustering methods, spectral clustering, k-means, METIS

Computing Strategy of HaarPooling



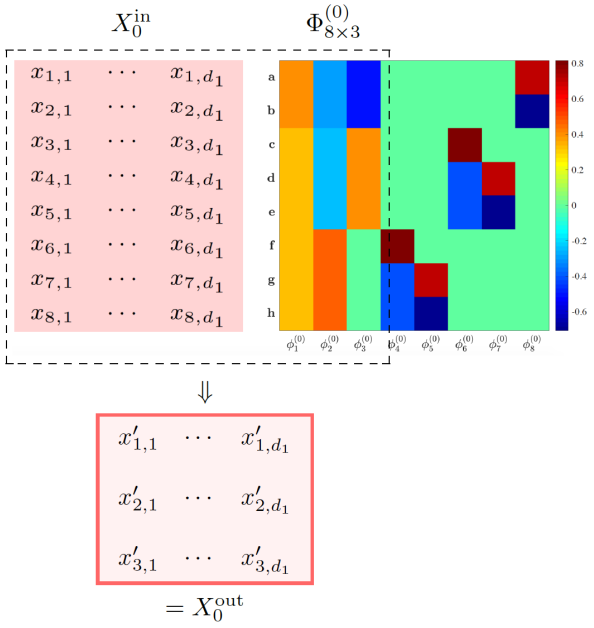
(a) First HaarPooling Layer for $\mathcal{G}_0 \to \mathcal{G}_1$.



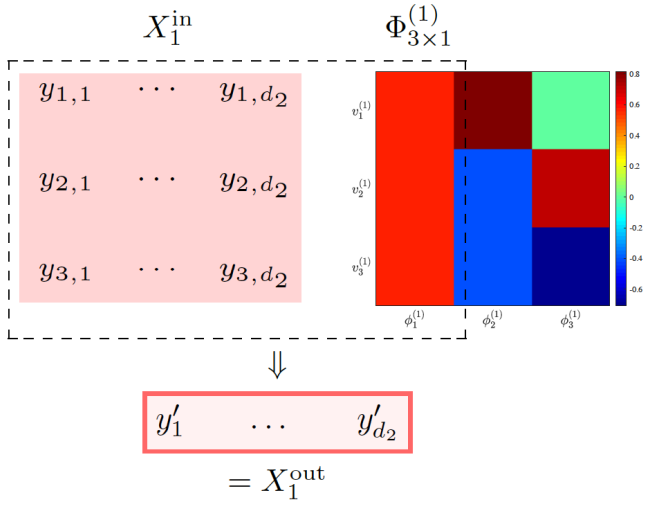
(b) Second HaarPooling Layer for $\mathcal{G}_1 \to \mathcal{G}_2$.

- In the first layer, the input X_1^{in} of size $8 \times d_1$ is transformed by the compressive Haar basis matrix $\Phi_{8\times 3}^{(0)}$ which consists of the first three column vectors of the full Haar basis $\Phi_{8\times 8}^{(0)}$ in (a), and the output is a $3 \times d_1$ matrix X_1^{out}.
- In the second layer, the input X_2^{in} of size $3 \times d_2$ (usually X_1^{out} followed by convolution) is transformed by the compressive Haar matrix $\Phi_{3\times 1}^{(1)}$, which is the first column vector of the full Haar basis matrix $\Phi_{3\times 3}^{(1)}$ in (b).

Computing Strategy of HaarPooling (Continued)



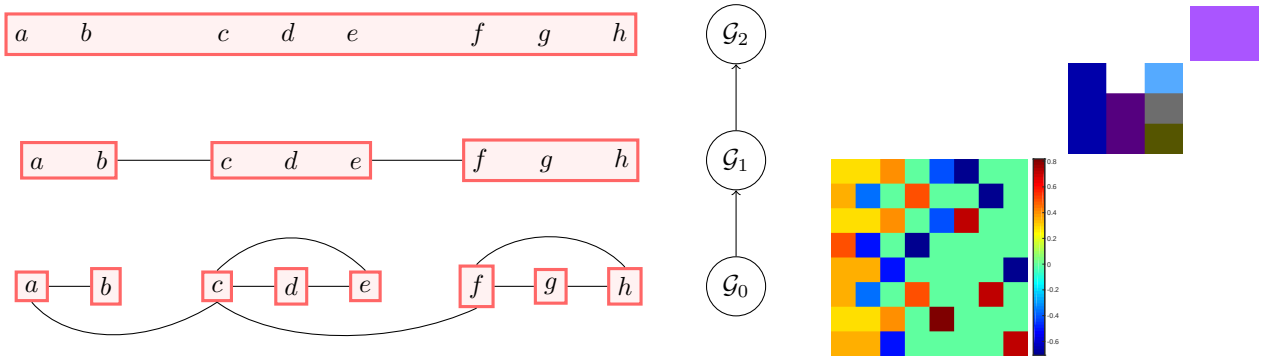
(a) First HaarPooling Layer for $\mathcal{G}_0 \to \mathcal{G}_1$.



(b) Second HaarPooling Layer for $\mathcal{G}_1 \to \mathcal{G}_2$.

- By the construction of the Haar basis in relation to the chain, each of the first three column vectors $\phi_1^{(0)}, \phi_2^{(0)}$ and $\phi_3^{(0)}$ of $\Phi_{8\times 3}^{(0)}$ has only up to three different values. This bound is precisely the number of nodes of $\mathcal{G}_1$.
- This example shows that the HaarPooling amalgamates the node feature by adding the same weight to the nodes that are in the same cluster of the coarser layer, and in this way, pools the feature using the graph clustering information.

Construction of Haar Basis



Gavish et al. (2010), Chui et al. (2015).

- Haar basis is constructed from top to bottom, $\ell \le N^{(1)}$

$$\phi_1^{(2)}(u^{(2)}) = 1, \quad \phi_1^{(1)}(u^{(1)}) = \mathbf{1}(u^{(1)})/\sqrt{N_1}$$

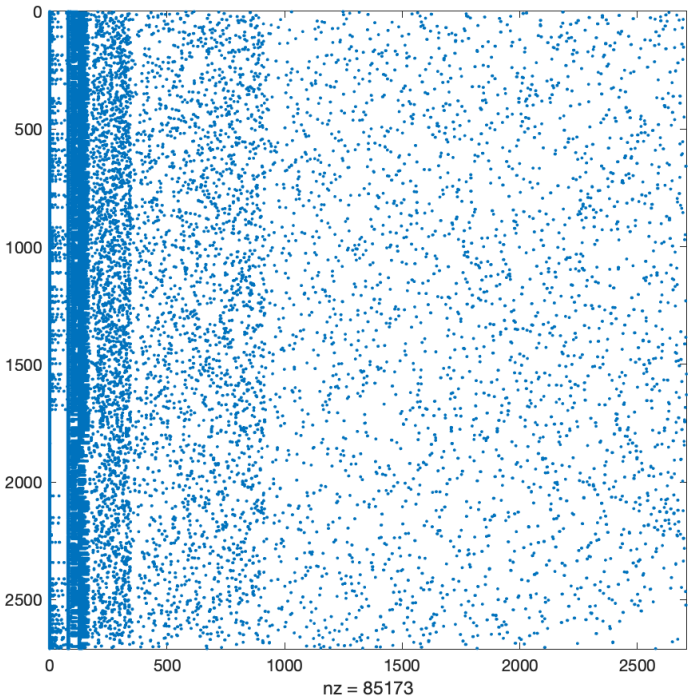
$$\phi_\ell^{(1)}(u^{(1)}) = \sqrt{\frac{N^{(1)}-\ell+1}{N^{(1)}-\ell+2}}\left(\chi_{\ell-1}^{(1)}(u^{(1)}) - \frac{\sum_{j=\ell}^{N^{(1)}}\chi_j^{(1)}(u^{(1)})}{N^{(1)}-\ell+1}\right).$$

- Extend to layer $\mathcal{G}^{(0)}$: for $k = 2, \ldots, k_\ell$, $k_\ell = |u_\ell^{(1)}|$, we let

$$\phi_{\ell,1}(v) := \frac{\phi_\ell^{(1)}(v^{(1)})}{\sqrt{|v^{(1)}|}}, \quad \phi_{\ell,k} = \sqrt{\frac{k_\ell-k+1}{k_\ell-k+2}}\left(\chi_{\ell,k-1} - \frac{\sum_{j=k}^{k_\ell}\chi_{\ell,j}}{k_\ell-k+1}\right)$$

where $\chi_{\ell,j}$ for $j = 1, \ldots, k_\ell$, $\chi_{\ell,j}$ is the indicator function on $\{v_{\ell,j}\}$.

Sparsity of Haar Basis Matrix



- Haar Basis for Cora
- Citation network Cora: 2708 nodes, 5429 edges
- Chain by METIS
- Sparsity: 98.84%

HaarPool for Benchmark Graph Classification

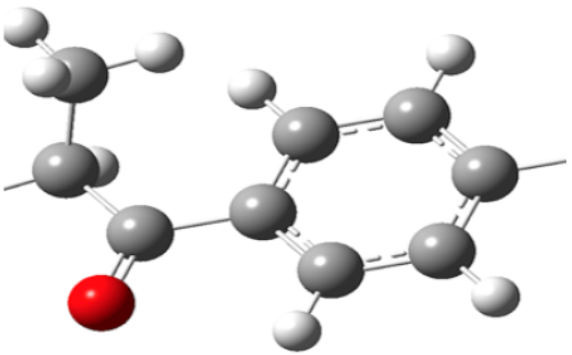
Table 2 reports the classification test accuracy. GNNs with HaarPooling have excellent performance on all datasets. In 4 out of 5 datasets, it achieves top accuracy. It shows that HaarPooling, with an appropriate graph convolution, can achieve top performance on a variety of graph classification tasks, and in some cases, improve state of the art by a few percentage points.

Table 2. Performance comparison for graph classification tasks (test accuracy in percent, showing the standard deviation over ten repetitions of the experiment).

Method	MUTAG	PROTEINS	NCI1	NCI109	MUTAGEN
CSM	85.4	–	–	–	–
GIN	89.4	76.2	**82.7**	–	–
SortPool	85.8	75.5	74.4	72.3*	78.8*
DiffPool	–	76.3	76.0*	74.1*	80.6*
gPool	–	77.7	–	–	–
SAGPool	–	72.1	74.2	74.1	–
EigenPool	–	76.6	77.0	74.9	79.5
HaarPool (ours)	**90.0±3.6**	**80.4±1.8**	78.6±0.5	**75.6±1.2**	**80.9±1.5**

'*' indicates records retrieved from EigenPool (Ma et al., 2019a), '–' means that there are no public records for the method on the dataset, and bold font is used to highlight the best performance in the list.

Quantum Chemistry Graph Regression



- QM7 is a collection of $7,165$ molecules, train/test $= 4/1$.
- Each molecule contains ≤ 23 atoms (including C, O, N, S), atoms are connected by bonds, molecular structure varies (e.g. double/triple bonds, cycles, carboxy, cyanide ...).
- Molecule is a graph, atoms are nodes, bonds are edges and Coulomb energy as weights, then Coulomb energy matrix is adjacency matrix.
- Task: to predict atomization energy of molecule given the molecular structure.

HaarPool for QM7

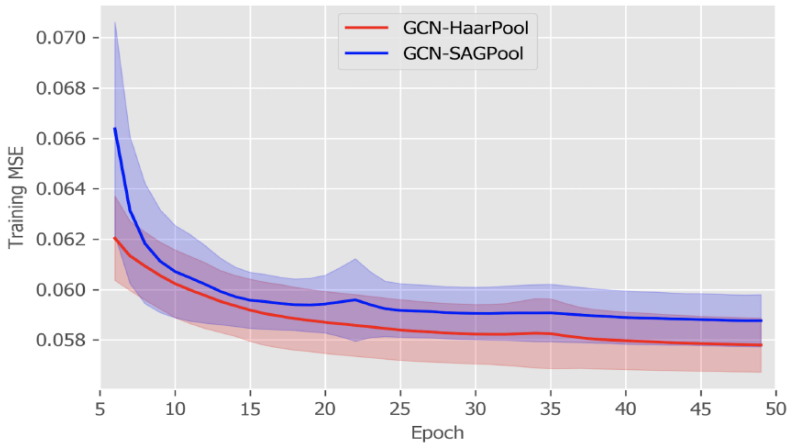
Table 5 shows the results for GCN-HaarPool and GCN-SAGPool, together with the public results of the other methods from *Wu et al. (2018)*. . Compared to the GCN-SAGPool, the GCN-HaarPool has a lower average test MAE and a smaller SD and ranks the top in the table.

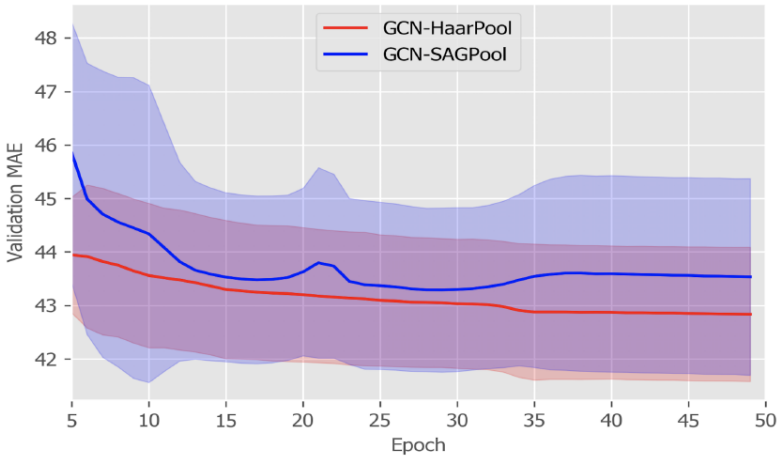
Table 5. Test mean absolute error (MAE) comparison on QM7, with the standard deviation over ten repetitions of the experiments.

Method	Test MAE
RF	122.7 ± 4.2
Multitask	123.7 ± 15.6
KRR	110.3 ± 4.7
GC	77.9 ± 2.1
GCN-SAGPool	43.3 ± 1.6
GCN-HaarPool (ours)	$\mathbf{42.9 \pm 1.2}$

Loss MSE and Validation MAE

We present the mean and SD of the training MSE loss (for normalized input) and the validation MAE (which is in the original label domain) versus the epoch. It illustrates that the learning and generalization capabilities of the GCN-HaarPool are better than those of the GCN-SAGPool; in this aspect, HaarPooling provides a more efficient graph pooling for GNN in this graph regression task.





Computational Complexity

In Table 2, the HaarPool is the only pooling method which has time complexity proportional to the number of nodes and thus has a faster implementation.

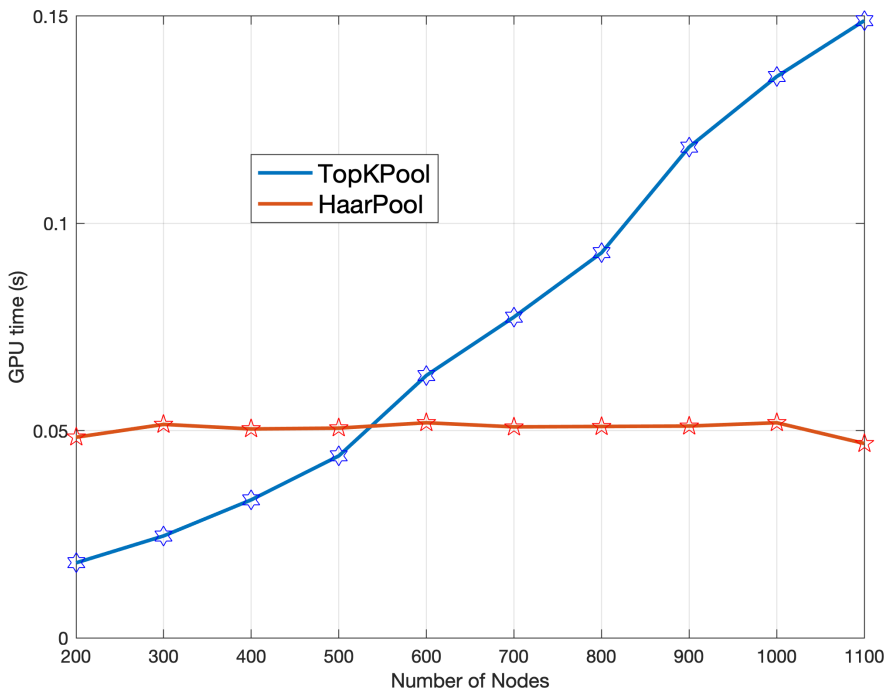
Table 2. Property comparison for pooling methods

Method	Time Complexity	Space Complexity	Clustering-based	Spectral-based	Hierarchical Pooling	Use Node Feature	Use Graph Structure	Sparse Representation
SortPool	$\mathcal{O}(\|V\|^2)$	$\mathcal{O}(\|V\|)$				✓		
DiffPool	$\mathcal{O}(\|V\|^2)$	$\mathcal{O}(k\|V\|^2)$			✓	✓		
gPool	$\mathcal{O}(\|V\|^2)$	$\mathcal{O}(\|V\|+\|E\|)$			✓	✓		
SAGPool	$\mathcal{O}(\|E\|)$	$\mathcal{O}(\|V\|+\|E\|)$			✓	✓	✓	
EigenPool	$\mathcal{O}(\|V\|^2)$	$\mathcal{O}(\|V\|^2)$	✓	✓	✓	✓	✓	
HaarPool	$\mathcal{O}(\|V\|)$	$\mathcal{O}(\|V\|^2\epsilon)$	✓	✓	✓	✓	✓	✓

'$|V|$' is the number of vertices of the input graph; '$|E|$' is the number of edges of the input graph; 'ϵ' in HaarPooling is the sparsity of the compressive Haar transform matrix; 'k' in the DiffPool is the pooling ratio.

GPU time comparison

For empirical comparison, we computed the GPU time for HaarPool and TopKPool on a sequence of datasets of random graphs. For each run, we fix the number of edges of the graphs. For different runs, the number of the edges ranges from 4000 to 121000. The sparsity of the adjacency matrix of the random graph is set to 10%. The following table shows the average GPU time (in seconds) for pooling a minibatch of 50 graphs. For both pooling methods, we use the same network architecture and one pooling layer, and same network hyperparameters, and run under the same GPU computing environment. The table shows that the cost of HaarPool does not change much as the edge number increases, while the cost of TopKPool increases rapidly. When the edge number is at most 25000, TopKPool runs slightly faster than HaarPool, but when the number exceeds 25000, the GPU time of TopKPool is longer.



Paper and codes

Thank you!

Paper https://arxiv.org/abs/1909.11580

Codes https://github.com/YuGuangWang/HaarPool