A. Efficient Computation for HaarPooling

For the HaarPooling introduced in Definition 1, we can develop a fast computational strategy by virtue of fast adjoint Haar transforms. Let $\mathcal{G}_{0\rightarrow K}$ be a coarse-grained chain of the graph $\mathcal{G}_0$. For convenience, we label the vertices of the level-$j$ graph $\mathcal{G}_j$ by $V_j := \{v_j^{(1)}, \ldots, v_j^{(N_j)}\}$.

An efficient algorithm for HaarPooling

The HaarPooling can be computed efficiently by using the hierarchical structure of the chain, as we introduce as follows. For $j = 1, \ldots, K$, let $c_j^{(j)}$ be the number of children of $v_j^{(j)}$, i.e. the number of vertices of $\mathcal{G}_{j-1}$ which belongs to the cluster $v_k^{(j)}$, for $k = 1, \ldots, N_j$. For $j = 0$, we let $c_0^{(0)} = 1$ for $k = 1, \ldots, N_0$. Now, for $j = 0, \ldots, K$ and $k = 1, \ldots, N_j$, define the weight for the node $v_j^{(j)}$ of layer $j$ by

$$w_j^{(j)} := \frac{1}{\sqrt{c_j^{(j)}}}. \quad (1)$$

Let $W_{0\rightarrow K} := \{w_k^{(j)} | j = 0, \ldots, K, k = 1, \ldots, N_j\}$. Then, for $j = 0, \ldots, K$, the weighted chain $(\mathcal{G}_{j\rightarrow K}, W_{j\rightarrow K})$ becomes a filtration if each parent of the chain $\mathcal{G}_{j\rightarrow K}$ has at least two children.

Let $j = 0, \ldots, K$. For the $j$th HaarPooling layer, let $\{\phi^{(j)}_\ell \}_{\ell=1}^{N_j}$ be the Haar basis for the $j$th layer, which we also call the Haar basis for the filtration $(\mathcal{G}_{j\rightarrow K}, W_{j\rightarrow K})$ of a graph $\mathcal{G}$. For $k = 1, \ldots, N_j$, we let $X(v_k^{(j)}) = X(v_k^{(j)}, \cdot) \in \mathbb{R}^{d_j}$ the feature vector at node $v_k^{(j)}$. We define the weighted sum for feature $X \in \mathbb{R}^{N_j \times d_j}$ for $d_j \geq 1$ by

$$S^{(j)}(X, v_k^{(j)}) := X(v_k^{(j)}), \quad v_k^{(j)} \in \mathcal{G}_j, \quad (2)$$

and recursively, for $i = j + 1, \ldots, K$ and $v_k^{(i)} \in \mathcal{G}_i$,

$$S^{(i)}(X, v_k^{(i)}) := \sum_{v_k^{(i-1)} \in v_k^{(i)}} w_k^{(i-1)} S^{(i-1)}(X, v_k^{(i-1)}). \quad (3)$$

For each vertex $v_k^{(i)}$ of $\mathcal{G}_i$, the $S^{(i)}(X, v_k^{(i)})$ is the weighted sum of the $S^{(i-1)}(X, v_k^{(i-1)})$ at the level $i - 1$ for those vertices $v_k^{(i-1)}$ of $\mathcal{G}_{i-1}$ whose parent is $v_k^{(i)}$.

Theorem 1. For $0 \leq j \leq K - 1$, let $\{\phi^{(j)}_\ell \}_{\ell=1}^{N_j}$ for $i = j + 1, \ldots, K$ be the Haar bases for the filtration $(\mathcal{G}_{j\rightarrow K}, W_{j\rightarrow K})$ at layer $i$. Then, the compressive Haar transform for the $j$th HaarPooling layer can be computed by, for the feature $X \in \mathbb{R}^{N_j \times d_j}$, and $\ell = 1, \ldots, N_j$,

$$(\Phi_j^T X)_\ell = \sum_{k=1}^{N_j} S^{(i)}(X, v_k^{(i)}) w_k^{(i)} \phi^{(i)}_\ell (v_k^{(i)}), \quad (4)$$

where $i$ is the largest possible number in $\{j + 1, \ldots, K\}$ such that $\phi^{(i)}_\ell$ is the $\ell$th member of the orthonormal basis $\{\phi^{(i)}_\ell \}_{\ell=1}^{N_i}$ for $\ell_2(\mathcal{G}_i)$, $v_k^{(i)}$ are the vertices of $\mathcal{G}_i$ and the weights $w_k^{(i)}$ are given by Equation (1).

We give the algorithmic implementation of Theorem 1 in Algorithm 1, which provides a fast algorithm for HaarPooling at each layer.

B. Proofs

Proof for Equation (6) in Section 4. We only need to prove the first formula. The second is obtained by definition. To simplify notation, we let $f = X^T_\mathcal{A}$. By construction of Haar basis, for some layer $j$, the first $N_{j+1}$ basis vectors

$$\phi^{(j)}_\ell (v) = \phi^{(j+1)}_\ell (p)/\sqrt{|Pa_\mathcal{G}(v)|}, \quad \text{for } p = Pa_\mathcal{G}(v).$$
**Algorithm 1** Fast HaarPooling for One Layer

**Input:** Input feature $X_j^{in}$ for the $j$th pooling layer given $j = 0, \ldots, K - 1$ in a GNN with total $K$ HaarPooling layers; the chain $G_j \rightarrow G_k$ associated with the HaarPooling; numbers $N_i$ of nodes for layers $i = j, \ldots, K$.

**Output:** $\Phi^T_j X_j^{in}$ from Definition 1.

**Step 1:** Evaluate the sums for $i = j, \ldots, K$ recursively, using Equations (2) and (3):

$$S^{(i)}(X_j^{in}, v_k^{(i)}) \forall v_k^{(i)} \in V_i.$$

**Step 2:**

for $\ell = 1$ to $N_j+1$

Set $N_K = 0$.

Compute $i$ such that $N_{i+1} + 1 \leq \ell \leq N_i$.

Evaluate $\sum_{k=1}^{N_i} S^{(i)}(X_j^{in}, v_k^{(i)})u_k^{(i)}$ in Equation (4) by the two steps:

(a) Compute the product for all $v_k^{(i)} \in V_i$:

$$T_{\ell}(X_j^{in}, v_k^{(i)}) = S^{(i)}(X_j^{in}, v_k^{(i)})u_k^{(i)}(v_k^{(i)}).$$

(b) Evaluate sum $\sum_{k=1}^{N_i} T_{\ell}(X_j^{in}, v_k^{(i)})$.

end for

Then, the Fourier coefficient of $f$ for the $\ell$th basis vector is the inner product

$$\langle f, \phi^{(j)}_\ell \rangle = \sum_{v \in G_j} f(v)\phi^{(j)}_\ell(v)$$

$$= \sum_{p \in G_j} \sum_{v = Pa_G(v)} f(v)\phi^{(j+1)}_\ell(p)\sqrt{|Pa_G(v)|}$$

$$= \sum_{p \in G_j} \bar{f}(p)\phi^{(j+1)}_\ell(p) = \langle \bar{f}, \phi^{(j+1)}_\ell \rangle$$

where we have let

$$\bar{f}(p) := \frac{1}{\sqrt{|Pa_G(v)|}} \sum_{v = Pa_G(v)} f(v).$$

This then gives

$$\sum_{\ell=1}^{N_j+1} \left| \langle f, \phi^{(j)}_\ell \rangle \right|^2 = \sum_{\ell=1}^{N_j+1} \left| \langle \bar{f}, \phi^{(j+1)}_\ell \rangle \right|^2.$$  \hspace{1cm} (5)

Since $\{\phi^{(j)}_\ell\}_{\ell=1}^{N_j+1}$ forms an orthonormal basis for $L_2(G_{j+1})$,

$$\|\Phi^T_j f\|^2 = \sum_{\ell=1}^{N_j+1} \left| \langle \bar{f}, \phi^{(j+1)}_\ell \rangle \right|^2 = \|\bar{f}\|^2 = \sum_{p \in G_j} |\bar{f}(p)|^2$$

$$= \sum_{p \in G_{j+1}} \left| \frac{1}{\sqrt{|Pa_G(v)|}} \sum_{v = Pa_G(v)} f(v) \right|^2.$$  

This proves the left formula in Equation (6) in Section 4.  \hspace{1cm} $\square$

**Proof of Theorem 1.** By the relation between $\phi^{(j)}_\ell$ and $\phi^{(j)}_\ell$, for $i = j+1, \ldots, K$ and $\ell = 1, \ldots, N_{j+1}$,

$$\langle \Phi^T_j X, \phi^{(j)}_\ell \rangle = \sum_{k=1}^{N_j} X(v_k^{(j)})\phi^{(j)}_\ell(v_k^{(j)})$$

$$= \sum_{k'=1}^{N_{j+1}} \left( \sum_{P_{G}(v_k^{(j)}) = v_k^{(j+1)}} X(v_k^{(j)}) \right)w_{k'}^{(j+1)}\phi^{(j+1)}_\ell(v_k^{(j+1)})$$

$$= \sum_{k'=1}^{N_{j+1}} S^{(j+1)}(X, v_k^{(j+1)})w_{k'}^{(j+1)}\phi^{(j+1)}_\ell(v_k^{(j+1)})$$

$$= \sum_{k''=1}^{N_{j+2}} \left( \sum_{P_{G}(v_k^{(j+1)}) = v_k^{(j+2)}} S^{(j+1)}(X, v_k^{(j+1)})w_{k''}^{(j+2)} \phi^{(j+2)}_\ell(v_k^{(j+2)}) \right)$$

$$= \sum_{k''=1}^{N_{j+2}} S^{(j+2)}(X, v_k^{(j+2)})w_{k''}^{(j+2)}\phi^{(j+2)}_\ell(v_k^{(j+2)})$$

$$\cdots$$

$$= \sum_{k''=1}^{N_i} S^{(i)}(X, v_k^{(i)})w_{k''}^{(i)}\phi^{(i)}_\ell(v_k^{(i)}),$$

where $v_k^{(j+1)}$ is the parent of $v_k^{(j)}$ and $v_k^{(j+2)}$ is the parent of $v_k^{(j+1)}$, and we recursively compute the summation to obtain the last equality, thus completing the proof.  \hspace{1cm} $\square$

**C. Experimental Setting**

The hyperparameters include batch size; learning rate, weight decay rate (these two for optimization); the maximal number of epochs; patience for early stopping. Table 1 shows the choice of hyperparameters for classification benchmark datasets.

**D. Property Comparison of Pooling Methods**

Here we provide a comparison of the properties of HaarPooling with existing pooling methods. The properties in the comparison include time complexity and space complexity, and whether involving the clustering, hierarchical pooling (which is then not a global pooling), spectral-based, node feature or graph structure, and sparse representation. We compare HaarPooling (denoted by HaarPool in the table) to other methods (SortPool, DiffPool, gPool, SAGPool, and EigenPool).

- The SortPool (i.e., SortPooling) is a global pooling which uses node signature (i.e., Weisfeiler-Lehman color of vertex) to sort all vertices by the values of the channels of the input data. Thus, the time complexity
Haar Graph Pooling

Table 1. Hyperparameter setting

<table>
<thead>
<tr>
<th>Data Set</th>
<th>MUTAG</th>
<th>PROTEINS</th>
<th>NCI1</th>
<th>NCI109</th>
<th>MUTAGEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>batch size</td>
<td>60</td>
<td>50</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>max #epochs</td>
<td>30</td>
<td>20</td>
<td>150</td>
<td>150</td>
<td>50</td>
</tr>
<tr>
<td>early stopping</td>
<td>15</td>
<td>20</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>learning rate</td>
<td>0.01</td>
<td>0.001</td>
<td>0.001</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>weight decay</td>
<td>0.0005</td>
<td>0.0005</td>
<td>0.0005</td>
<td>0.0001</td>
<td>0.0005</td>
</tr>
</tbody>
</table>

Table 2. Property comparison for pooling methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Time Complexity</th>
<th>Space Complexity</th>
<th>Clustering-based</th>
<th>Spectral-based</th>
<th>Hierarchical Pooling</th>
<th>Use Node Feature</th>
<th>Use Graph Structure</th>
<th>Sparse Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SortPool</td>
<td>$\mathcal{O}(</td>
<td>V</td>
<td>^2)$</td>
<td>$\mathcal{O}(</td>
<td>V</td>
<td>)$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>DiffPool</td>
<td>$\mathcal{O}(</td>
<td>V</td>
<td>^2)$</td>
<td>$\mathcal{O}(</td>
<td>k</td>
<td></td>
<td>V</td>
<td>^2)$</td>
</tr>
<tr>
<td>gPool</td>
<td>$\mathcal{O}(</td>
<td>V</td>
<td>^2)$</td>
<td>$\mathcal{O}(</td>
<td>V</td>
<td>+</td>
<td>E</td>
<td>)$</td>
</tr>
<tr>
<td>SAGPool</td>
<td>$\mathcal{O}(</td>
<td>E</td>
<td>)$</td>
<td>$\mathcal{O}(</td>
<td>V</td>
<td>+</td>
<td>E</td>
<td>)$</td>
</tr>
<tr>
<td>EigenPool</td>
<td>$\mathcal{O}(</td>
<td>V</td>
<td>^2)$</td>
<td>$\mathcal{O}(</td>
<td>V</td>
<td>^2)$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>HaarPool</td>
<td>$\mathcal{O}(</td>
<td>V</td>
<td>)$</td>
<td>$\mathcal{O}(</td>
<td>V</td>
<td>^2\epsilon)$</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

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

(worst case) of SortPool is $\mathcal{O}(|V|^2)$ and space complexity is $\mathcal{O}(|V|)$. Other pooling methods mentioned here are all hierarchical pooling.

- DiffPool and gPool both use the node feature and have time complexity $\mathcal{O}(|V|^2)$. The DiffPool learns the assignment matrices in an end-to-end manner and has space complexity $\mathcal{O}(k|V|^2)$ for pooling ratio $k$. The gPool projects all nodes to a learnable vector to generate scores for nodes, and then sorts the nodes by the projection scores; the space complexity is $\mathcal{O}(|V| + |E|)$.

- SAGPool uses the graph convolution to calculate the attention scores of nodes and then selects top-ranked nodes for pooling. The time complexity of SAGPool is $\mathcal{O}(|E|)$, and the space complexity is $\mathcal{O}(|V| + |E|)$ due to the sparsity of the pooling matrix.

- EigenPool, which considers both the node feature and graph structure, uses the eigendecomposition of subgraphs (from clustering) of the input graph, and pools the input data by the Fourier transforms of the assembled basis matrix. Due to the computational cost of eigendecomposition, the time complexity of EigenPool is $\mathcal{O}(|V|^2)$, and the space complexity is $\mathcal{O}(|V|^2)$. 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.

- HaarPool which uses the sparse representation of data by compressive Haar basis has linear time complexity $\mathcal{O}(|V|)$ (up to a $\log |V|$ term), and the space complexity is $\mathcal{O}(|V|^2\epsilon)$, where $\epsilon$ is the sparsity of compressive Haar transform matrix and is usually very small. HaarPooling can be even faster in practice, as the cost of the compressive Haar transform is dependent on the sparsity of the Haar basis matrix. The sparsity of the compressive Haar basis matrix is mainly reliant on the chain/tree for the graph. From our construction, the compressive Haar basis matrix is always highly sparse. Thus, the computational cost does not skyrocket as the size of the graph increases.