CS768 Project Report: Expressive Power of Pooling in Graph Neural Networks

Ekansh Ravi Shankar (22B1032), Geet Singhi (22B1035),
Tanmay Gejapati (22B0969), Ved Danait (22B1818)

May 2025

Contents

1 Problem Statement

In GNNs, hierarchical pooling operators generate local summaries of data by coarsening the graph structure and the vertex features. While there has been a lot of research on analyzing the expressive power of message-passing(MP) Layers in GNNs, a study on how graph pooling affects the expressiveness of a GNN is still lacking. This paper derives sufficient conditions for a pooling operator to fully preserve the expressive power of the MP layers before it. This paper also introduces an experimental setup to verify the expressive power of a GNN equipped with pooling layers, in terms of its capability to perform the WL Graph Isomorphism test.

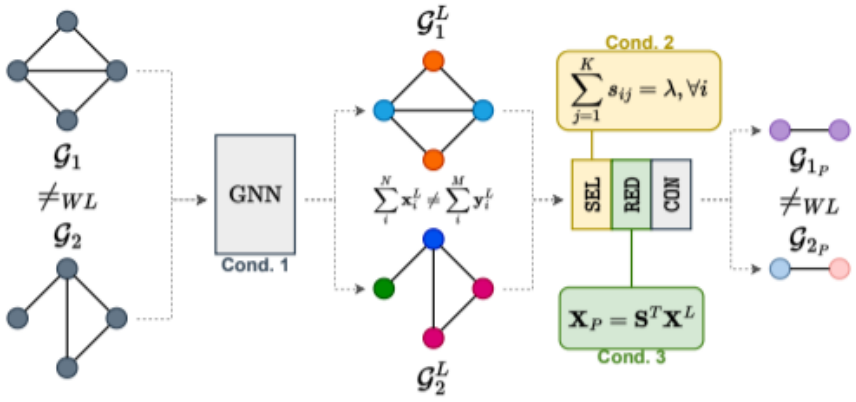


Figure 1: A GNN with expressive MP Layers computes different features for 2 graphs which are WL-distinguishable. A pooling layer generates coarsened graphs that are still WL-distinguishable

2 Literature Review

Early research concentrated on characterizing the ability of GNNs to distinguish non-isomorphic graphs, often referencing the Weisfeiler-Lehman (WL) isomorphism test as a benchmark. It was established that GNNs with well-designed message-passing (MP) layers can be as expressive as the WL test, while higher-order GNNs can match the more powerful k-WL test[4]. Inspired by pooling in convolutional neural networks, recent studies introduced hierarchical pooling operators for GNNs, enabling the learning of abstract, coarser graph representations. These pooling layers, when interleaved with MP layers, have been shown to improve performance in graph and node classification tasks. However, the literature predominantly evaluates pooling operators empirically, by measuring downstream task performance, which is confounded by factors such as model architecture and dataset properties. Some recent works proposed alternative criteria, such as spectral similarity between original and pooled graphs or reconstructability of features[3], but while providing valuable insights, these criteria give results that are, to some extent, contrasting and in disagreement with the traditional evaluation based on the performance of the downstream task. In summary, while the expressive power of MP layers in GNNs is well-studied and closely tied to the WL test, the impact of pooling operators on GNN expressiveness remains insufficiently understood in the literature. This gap motivates the paper's theoretical and empirical investigation into principled criteria for evaluating and comparing pooling operators.

3 Methodology

3.1 The Idea

The paper established the baseline with a GNN with 3 layers followed by a global sum pool reaches nearly 100% accuracy. Then they insert a pooling layer between the second and third GIN Layer which performs and aggressive pooling by using a pooling ratio of 0.1.

3.2 Our contribution

Our contribution is experimentation on 4 different pooling methods:

1. **Graph Multiset Transformer Pooling (GMT)** [1]
   Graph Multiset Pooling addresses the expressiveness limitations of traditional pooling methods like global mean or sum pooling, which often fail to capture structural information. GMT Pool introduces permutation-invariant, learnable multiset functions that aggregate node features while preserving important substructures.

   Given node features $X \in \mathbb{R}^{N \times F}$, GMT Pool applies a scoring function ϕ (typically an MLP) to produce attention weights:

   $$\alpha_i = \phi(X_i), \tag{1}$$

   and aggregates them using a multiset function ρ over the scores and features:

   $$\text{GMT Pool}(X) = \rho\left(\{(\alpha_i, X_i) \mid i = 1, \ldots, N\}\right). \tag{2}$$

   The choice of ρ includes functions like $\max_k$, top-k, or learnable soft variants to retain expressive substructure features.

   GMT Pool provides higher expressiveness than invariant graph pooling (e.g., sum, mean) by leveraging multiset representations. It is also permutation invariant by its construction. And last but not the least it provides flexibility to choose between hard (e.g., top-k) and soft (e.g., attention-based) selection mechanisms.

   It achieves state-of-the-art results on several graph classification benchmarks, particularly those requiring structural awareness.

2. **Learnable Cluster Pooling (LCP)**
   We implement a novel pooling mechanism termed Learnable Cluster Pool (LCP), inspired by clustering-based pooling methods such as DiffPool[5] and MinCutPool[2]. LCP learns a soft cluster assignment matrix through a neural network, enabling end-to-end differentiable pooling of graph nodes into a fixed number of clusters.

   Given node features $\mathbf{X} \in \mathbb{R}^{B \times N \times F}$ and adjacency matrices $\mathbf{A} \in \mathbb{R}^{B \times N \times N}$ for a batch of graphs, LCP computes a soft assignment matrix $\mathbf{S} \in \mathbb{R}^{B \times N \times K}$ using a learnable MLP. This matrix is then used to compute cluster-level features and adjacency:

   $$\mathbf{X}' = \mathbf{S}^\top \mathbf{X} \in \mathbb{R}^{B \times K \times F}, \tag{3}$$

   $$\mathbf{A}' = \mathbf{S}^\top \mathbf{A}\mathbf{S} \in \mathbb{R}^{B \times K \times K}. \tag{4}$$

   To guide the learning of meaningful clusters, we incorporate two auxiliary losses: 1. Entropy loss, which encourages confident (low-entropy) assignments and 2. Diversity loss, which

encourages orthogonal (diverse) cluster assignments by penalizing deviation from identity in $\mathbf{S}^\top\mathbf{S}$.
The total auxiliary loss is given by:

$$\mathcal{L}_{\text{aux}} = \lambda_1 \cdot \mathcal{L}_{\text{entropy}} + \lambda_2 \cdot \mathcal{L}_{\text{diversity}}, \tag{5}$$

with $\lambda_1 = \lambda_2 = 0.1$ in our experiments.

3. **Hard Spectral Pooling via Laplacian Eigenvectors (SpecPool)**
This method implements a classical spectral pooling approach by directly computing the normalized graph Laplacian and projecting node features onto its leading eigenvectors. Given a graph with node features $X \in \mathbb{R}^{n\times d}$ and adjacency matrix $A \in \mathbb{R}^{n\times n}$, the normalized Laplacian $L = I - D^{-1/2}AD^{-1/2}$ is computed for each graph in a batch. Eigen decomposition is performed on L, and the top-k eigenvectors $U_k \in \mathbb{R}^{n\times k}$ are selected based on a pooling ratio. The node features are projected into this spectral space via:

$$X' = U_k^\top X,$$

yielding a reduced node set of size k, interpreted as a compressed representation in the frequency domain. Each pooled graph is assumed to have an identity adjacency (i.e., no edges or fully connected) in spectral space. Unlike soft or learnable pooling methods, this approach is non-parametric and relies on eigendecomposition, which is computationally expensive and non-differentiable. While not end-to-end trainable, it serves as a baseline rooted in classical spectral clustering.

4. **Soft Assignments via Attention (SoftPool)**
This method implements a differentiable graph pooling strategy using soft assignments learned through a linear projection. Given input node features $X \in \mathbb{R}^{B\times N\times d}$ and adjacency matrix $A \in \mathbb{R}^{B\times N\times N}$, a linear layer produces soft cluster assignment scores $S \in \mathbb{R}^{B\times N\times k}$, normalized using a softmax over the cluster dimension. The pooled node features and adjacency are computed via:

$$X' = S^\top X, \quad A' = S^\top AS,$$

resulting in a coarsened graph with k nodes per graph. An optional entropy regularization term is applied to the assignment matrix to encourage confident and sparse clusterings:

$$\mathcal{L}_{\text{entropy}} = -\sum_i S_i \log S_i.$$

This method avoids discontinuities and hard decisions, making it fully differentiable and suitable for end-to-end training. It maintains permutation invariance and efficiently preserves information by allowing all nodes to contribute proportionally to the pooled representation.

3.3 Architecture

The GNN architecture used in all experiments is composed of the following sequence: [**2 GIN layers**] – [**1 pooling layer with pooling ratio 0.1**] – [**1 GIN layer**] – [**global sum pooling**] – [**dense readout**]. Each GIN layer employs an MLP with two hidden layers of 64 units and ELU activation functions. The final readout is implemented as a 3-layer MLP with hidden sizes $[64, 64, 32]$, ELU activations, and a dropout rate of 0.5.
The model is trained using the Adam optimizer with an initial learning rate of 1×10^{-4} and a batch size of 32.

4 Experiments

4.1 Dataset

Real-world and synthetic benchmark datasets are unsuitable for evaluating the expressive power of Message Passing (MP) layers when combined with pooling layers, as they are not specifically designed to relate GNN power to the Weisfeiler-Lehman (WL) test. While the EXP dataset was proposed to test GNNs with expressivity beyond the WL test, the paper introduces the **EXPWL1**.

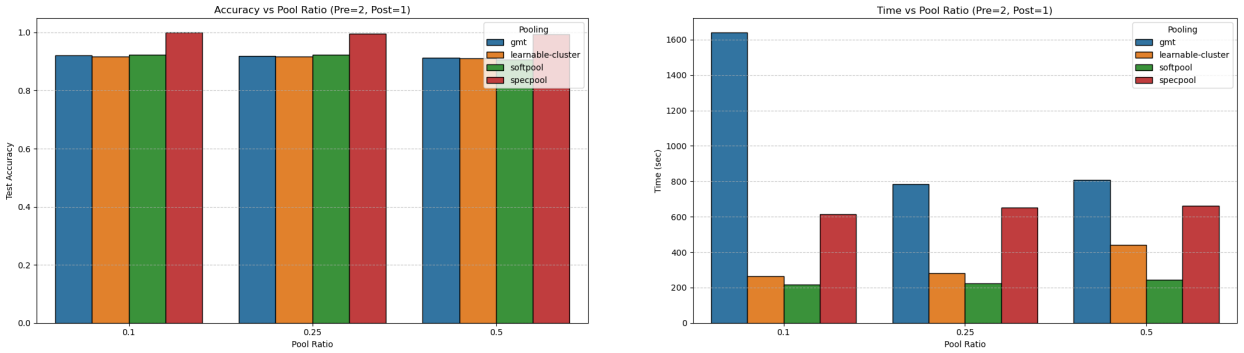
EXPWL1 comprises a collection of graphs $\{G_1, ..., G_N, H_1, ..., H_N\}$ representing propositional formulas that can be satisfiable or unsatisfiable. Each pair (G_i, H_i) consists of two non-isomorphic graphs distinguishable by a WL test, which encode formulas with opposite SAT outcomes. Therefore, any GNN with expressive power equal to the WL test can distinguish them and achieve approximately 100% classification accuracy.

Compared to the original EXP dataset, the paper increased the total number of graphs to 3000 and expanded the average graph size from 55 to 76 nodes. This modification enables the application of aggressive pooling (up to 90% node reduction) while preserving non-trivial graph structures. The **EXPWL1** dataset and reproduction code are publicly available

4.2 Procedure

The study evaluated four pooling operators (specpool, gmt, softpool, learnable-cluster) using a GIN-based GNN on the EXPWL1 dataset. We conducted a partial hyperparameter sweep varying: 1) pool ratios (0.1, 0.25, 0.5), 2) pre-pooling MP layers (2-3), and 3) post-pooling MP layers (1-2). Models were trained for 250 epochs with Adam optimizer (lr=1e-4) using negative log-likelihood loss plus auxiliary regularization. Performance was assessed through a randomized train/val/test splits, with test accuracy and training time recorded for the best validation-loss model. The dataset was partitioned into 80% training, 10% validation, and 10% test sets using stratified sampling.

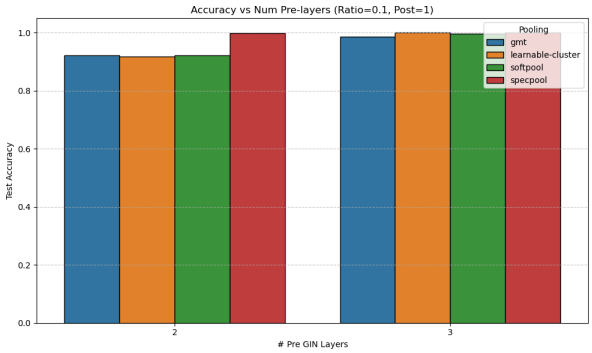
5 Results and Analysis



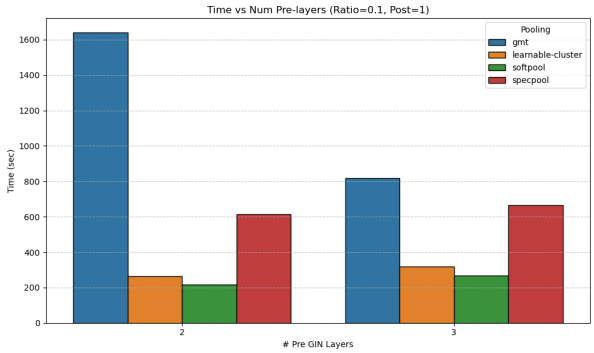
(a) Accuracy vs Pool Ratio

(b) Time vs Pool Ratio

Figure 2: Effect of pool ratio on accuracy and runtime (Pre=2, Post=1)

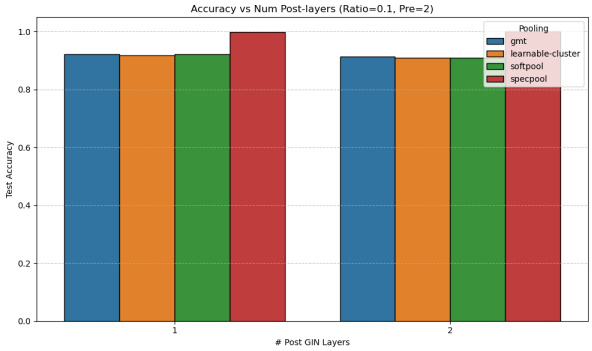


(a) Accuracy vs # Pre GIN Layers

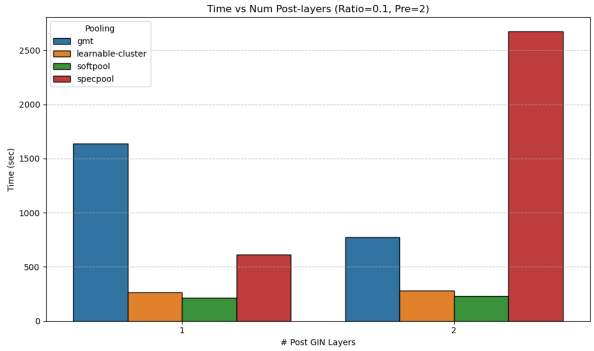


(b) Time vs # Pre GIN Layers

Figure 3: Effect of number of pre-GIN layers (Ratio=0.1, Post=1)



(a) Accuracy vs # Post GIN Layers



(b) Time vs # Post GIN Layers

Figure 4: Effect of number of post-GIN layers (Ratio=0.1, Pre=2)

- We validated the fact that in EXPWL1 when using too many MP layers, at least one node ends up containing enough information to accurately classify the graphs. Thus it makes sense that to ensure that the evaluations are meaningful, we use no more that 2 GIN layers
- All 4 pooling methods turned out to be expressive, with the SpecPool method performing the best
- GMT is usually the slowest, followed by SpecPool, followed by LCP and then finally SoftPool which is the quickest

6 Conclusions

- We were able to verify the expressiveness of the 4 pooling methods, although 3 of them had slightly borderline accuracies, it still matches the theoretical expectations
- We also saw that using more than 3 pre-pooling GIN layers leads to near perfect expressiveness

7 Contributions

Each of us implemented one pooling method from scratch. All of us worked together on ideation, setup for experiments and common implementations.

- **Ekansh:** Learnable Cluster Pooling
- **Geet:** GMT Pooling
- **Tanmay:** SoftPool
- **Ved:** Spectral Pooling

8 Challenges and Future Work

8.1 Issues

Getting some of the pooling layers to work well with the rest of the architecture was tricky and needed some trial and error. Experiments were very slow without GPU so we were restricted in how much we could do.

8.2 Future Work

We had some ideas on adapting different standard datasets for testing WL-expressiveness. This paper currently uses a synthetic dataset, so we wanted to split a standard dataset into non-isomorphic pairs using the WL test to see if the same results could be reproduced on real life datasets.

References

[1] Jinwoo Baek, U Kang, and Seungjae Ryan Hwang. "Accurate learning of graph representations with graph multiset pooling". In: *International Conference on Learning Representations (ICLR)*. 2021.

[2] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. "Spectral Clustering with Graph Neural Networks for Graph Pooling". In: *International Conference on Machine Learning (ICML)*. 2020. URL: https://arxiv.org/abs/1907.00481.

[3] Daniele Grattarola et al. "Understanding Pooling in Graph Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* (2022). DOI: 10.1109/TNNLS.2022.3194237.

[4] C. Morris et al. "Weisfeiler and Leman go neural: Higher-order graph neural networks". In: *AAAI*. 2019.

[5] Rex Ying et al. "Hierarchical Graph Representation Learning with Differentiable Pooling". In: *Advances in Neural Information Processing Systems*. 2018. URL: https://arxiv.org/abs/1806.08804.

A Appendix

Github: https://github.com/VedTheKnight/CS768-Project