Statistical Methods in AI

# How Powerful are Graph Neural Networks?

Team 29
Surabhi Jain (2021121004)
Shruti Kolachana (2020102053)
Maulesh Gandhi (2020112009)
Deepthi Chandak (2020102013)

# Contents

# Objectives

1

To understand the working of GNNs and the WL test

2

To show that GNNs are at most as powerful as the WL test in distinguishing graph structures

3

To develop a simple neural architecture and show that its discriminative power is equal to the power of the WL test.

4

To validate the above results by training different GNN variants

# Graphs

- G = (V, E); V: set of nodes; E: set of edges
- Adjacency matrix {A}: commonly used to depict a graph. If a graph has N nodes, then A has an NxN dimension.
- Two graphs can be visually and structurally different. Still, when we convert them to their adjacency matrix representation, they can have the same adjacency matrix.
- Traditional Graph Analysis techniques have the limitation of requiring previous knowledge of the graph at a particular level of certainty before using the algorithm. Most notably, graph-level categorization is not possible.

# Graph Neural Networks

“Message passing neural network” framework:

GNNs adopt a “graph-in, graph-out” architecture:

- accept a graph as input, with information loaded into its nodes, edges and global context
- progressively transform these embeddings, without changing the connectivity of the input graph.

Simplest GNN:

- learn new embeddings for all graph attributes (nodes, edges, global)
- use a separate multilayer perceptron (MLP) (or another differentiable model) on each component of a graph: we call this a GNN layer.
- for each node vector, apply the MLP and get back a learned node-vector.

# Graph Neural Networks

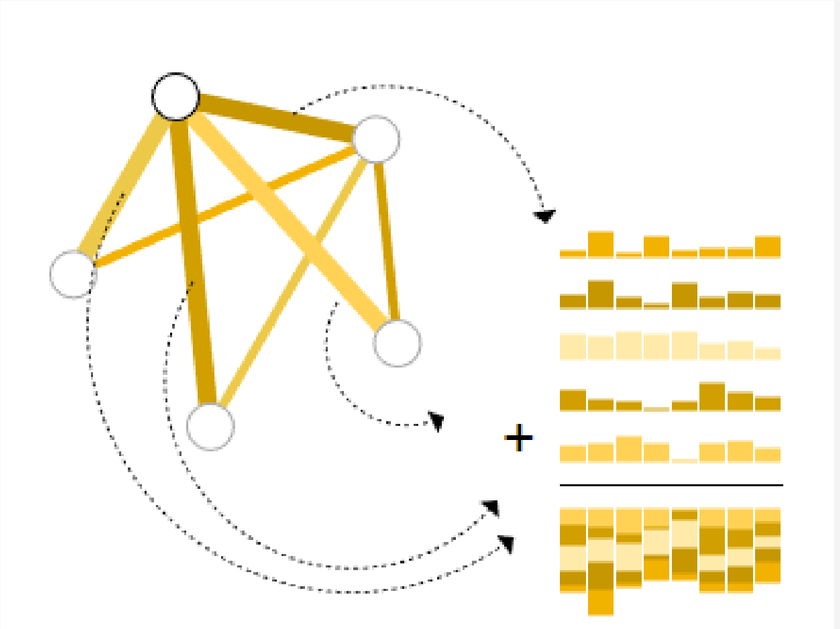
**How to make predictions in any of the tasks ?**

Consider Binary classification:

Node embeddings given-apply linear classifier

Edge embeddings given-apply pooling

1. For each item to be pooled, gather each of their embeddings and concatenate them into a matrix.
2. The gathered embeddings are then aggregated



# Graph Neural Networks

- GNNs use graph structures $G = (V, E)$ & node feature vectors $X_v$ for $v \in V$ to learn the representation vector $h_v$ of a node (or) $h_G$ of a graph via the 'neighbourhood aggregation strategy' (iteratively updates representation of a node by aggregating representations of its neighbours.)
- After $k$-iterations of aggregation, a node's representation captures the structural information within its $k$-hop network neighbourhood.
- $k^{th}$ layer of a GNN is:

$$h_v^{(k)} = COMBINE^{(k)}\left(\left\{h_v^{(k-1)}, a_v^{(k)}\right\}\right)$$
$$a_v^{(k)} = AGGREGATE^{(k)}\left(\left\{h_u^{(k-1)} : u \in N(v)\right\}\right)$$

$h_u^{(k)}$ is feature vector of node $v$ at the $k^{th}$ iteration/layer, $h_u^{(0)} = X_v$, $N(v)$ is a set of nodes adjacent to $v$, $AGGREGATE^{(k)}(.)$ & $COMBINE^{(k)}(.)$ are crucial and vary for different GNN models like the GraphSAGE and GCN.

- Node classification: node representation $h_v^{(K)}$ of the final iteration is used for prediction.
- Graph classification: $READOUT$ function aggregates node features from the final iteration to obtain the entire graph's representation:

$$h_G = READOUT(\{h_v^K \mid v \in G\})$$

# The Weisfeiler-Lehman Test

- **Graph isomorphism** indicates if two graphs are topologically identical
- WL test is a powerful test known to distinguish a broad class of graphs because of its injective aggregation update that maps different node neighborhoods to different feature vectors.
- The WL test iteratively:

1. ***aggregates*** the labels of nodes and their neighborhoods
2. ***hashes*** the aggregated labels into unique new labels.

- 1-dimensional form of WL test, “naïve vertex refinement”, is analogous to **neighbor aggregation** in GNNs.
- The algorithm decides that two graphs are non-isomorphic if at some iteration the labels of the nodes between the two graphs differ.

# WL test : Example

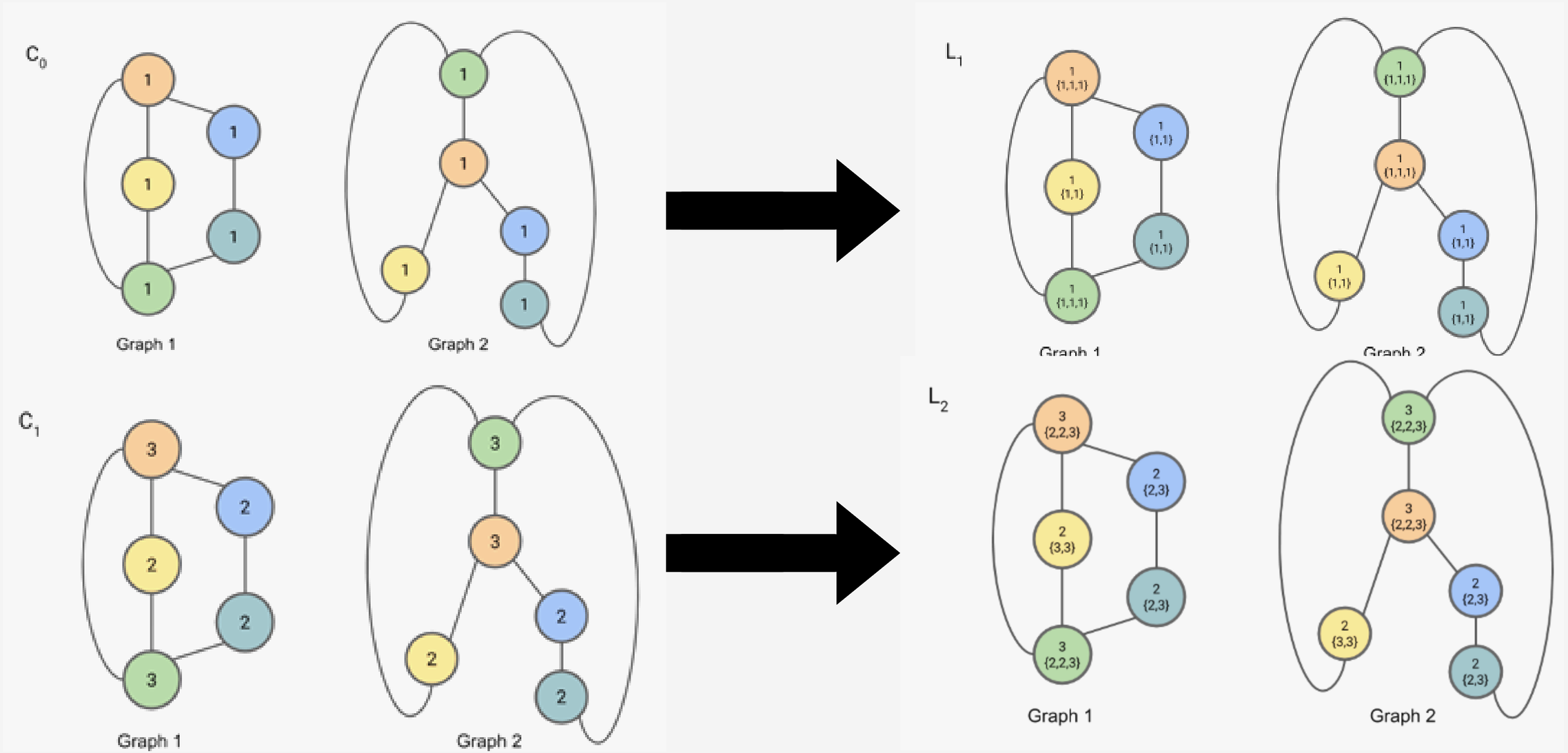
1. To begin, we initialize $C_{0,n} = 1$ for all nodes $n$.

2. At iteration $i$ of the algorithm (beginning with $i = 1$), for each node $n$, we set $L_{i,n}$ to be a tuple containing the node's old label $C_{i-1,n}$ and the multiset of compressed node labels $C_{i-1,m}$ from all nodes $m$ neighboring $n$ from the previous iteration $(i - 1)$.

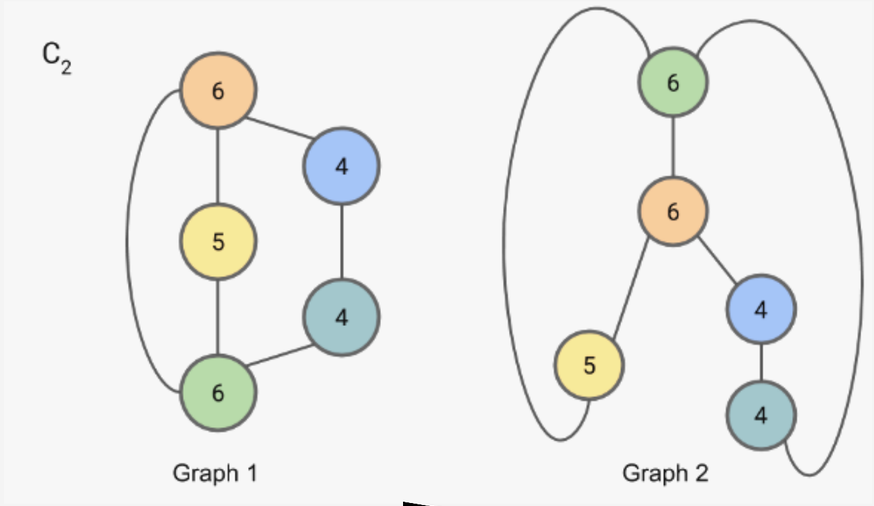
3. We then complete iteration $i$ by setting $C_{i,n}$ to be a new "compressed" label, such as a hash of $L_{i,n}$. Any two nodes with the same labels $L_{i,n}$ must get the same compressed label $C_{i,n}$.

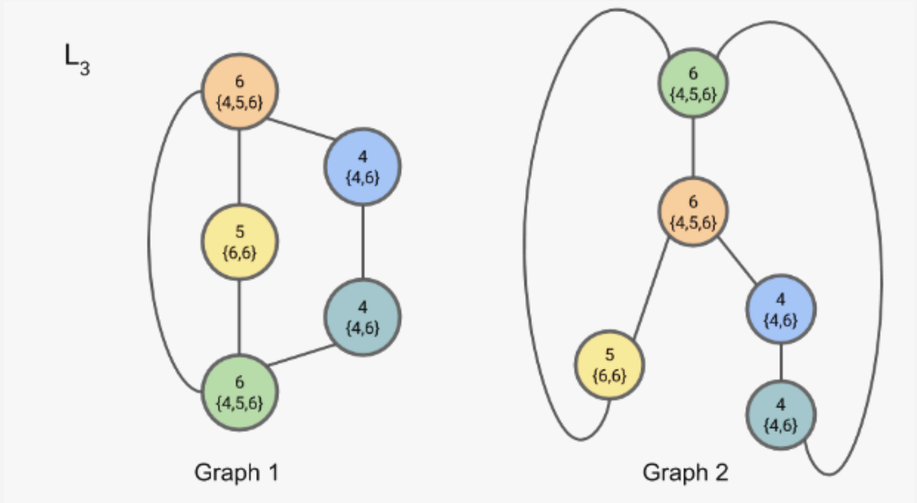
4. Partition the nodes in the graph by their compressed label. Repeat 2 + 3 for up to $N$ (the number of nodes) iterations, or until there is no change in the partition of nodes by compressed label from one iteration to the next.

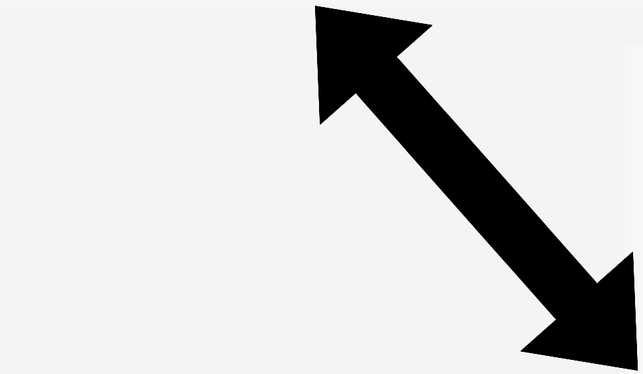
# WL test : Example

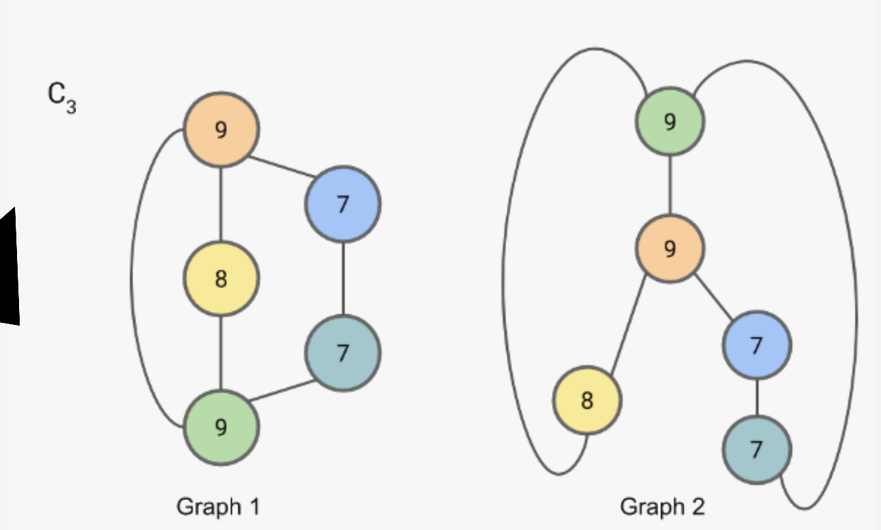


# WL test : Example



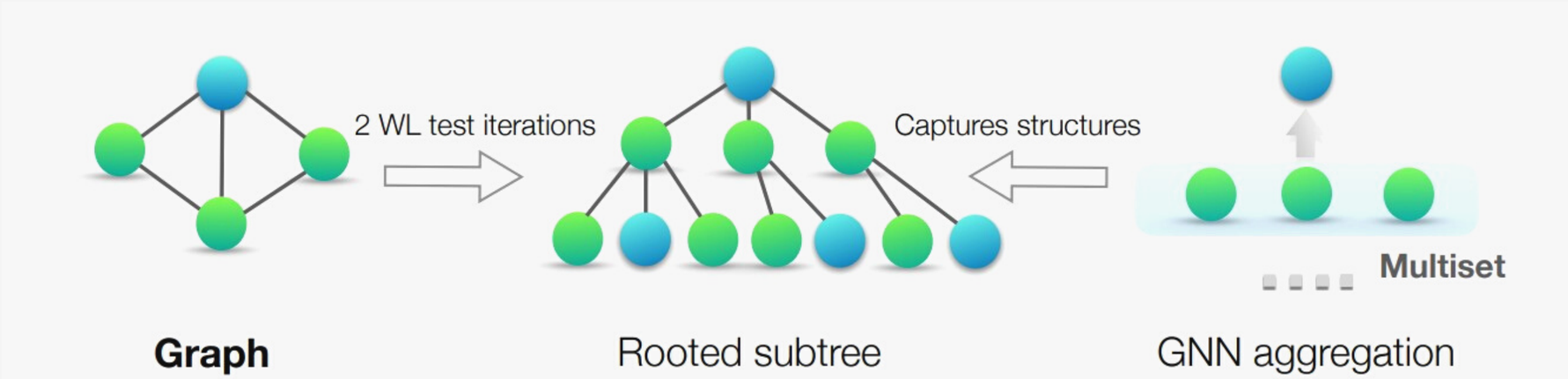





# Basic Framework

- A **multiset** is a generalized concept of a set that *allows multiple instances* for its elements.
- Hence, to have strong representational power, a GNN must be able to aggregate different multisets into different representations.
- **The more discriminative the multiset function is, the more powerful the representational power of the underlying GNN.**
- **Key: GNN can have as large discriminative power as the WL test if the GNN's aggregation scheme is highly expressive and can model injective functions.**



# Conditions for a maximally powerful GNN

**Theorem 3.** *Let $\mathcal{A} : \mathcal{G} \to \mathbb{R}^d$ be a GNN. With a sufficient number of GNN layers, $\mathcal{A}$ maps any graphs $G_1$ and $G_2$ that the Weisfeiler-Lehman test of isomorphism decides as non-isomorphic, to different embeddings if the following conditions hold:*

*a) $\mathcal{A}$ aggregates and updates node features iteratively with*

$$h_v^{(k)} = \phi\left(h_v^{(k-1)}, f\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right)\right),$$

*where the functions $f$, which operates on multisets, and $\phi$ are injective.*

*b) $\mathcal{A}$'s graph-level readout, which operates on the multiset of node features $\left\{h_v^{(k)}\right\}$, is injective.*

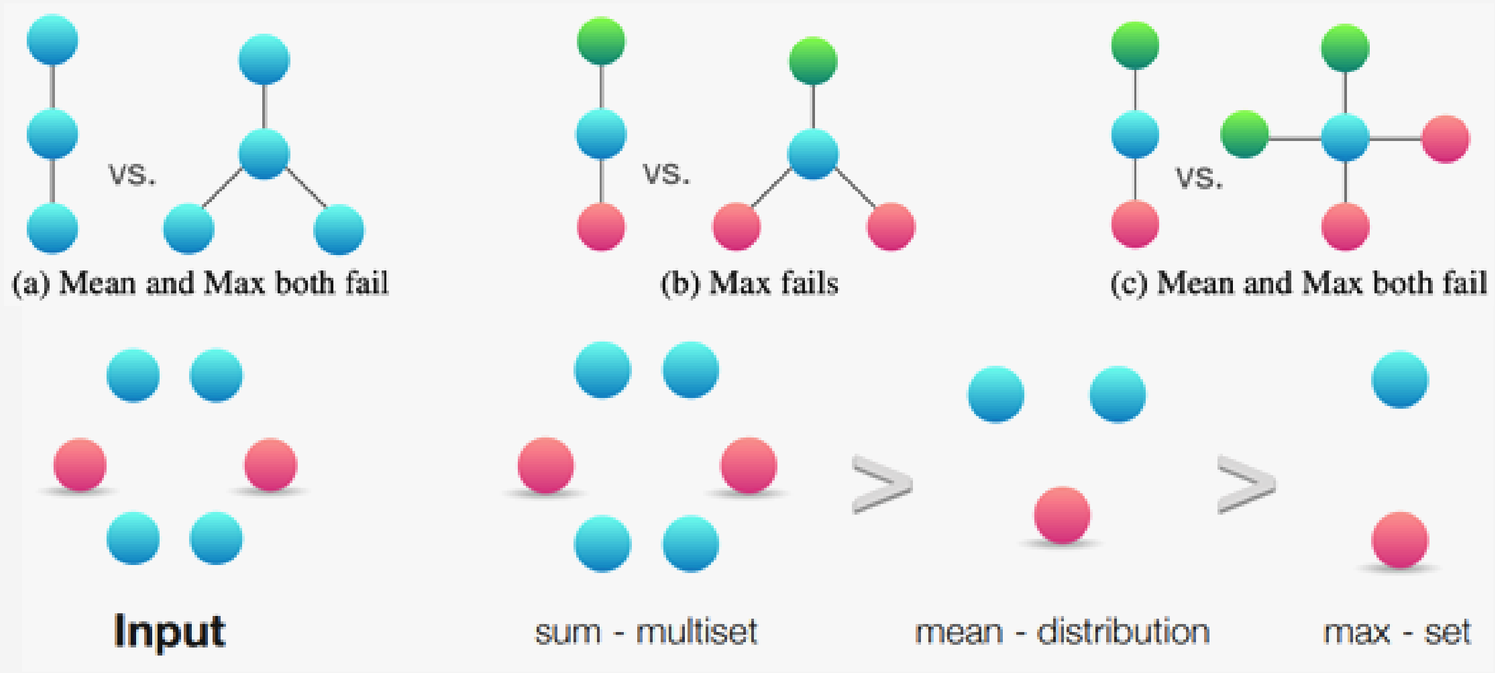
# Pooling Techniques

## Mean-pooling

Mean-pooling captures the distribution (proportions) of elements in a multiset, but not the exact multiset

## Max-pooling

Max-pooling captures neither the exact structure nor the distribution.
It considers multiple nodes with the same feature as only one node (i.e., treats a multiset as a set).



# Graph Isomorphism Network

- To model injective multiset functions for the neighbor aggregation, we develop a theory of “deep multisets”, i.e., parameterizing universal multiset functions with neural networks.
- Sum aggregators can represent injective, in fact, universal functions over multisets.
- Therefore, GIN updates node representations as:

$$h_v^{(k)} = \mathrm{MLP}^{(k)}\left(\left(1+\epsilon^{(k)}\right)\cdot h_v^{(k-1)} + \sum_{u\in\mathcal{N}(v)} h_u^{(k-1)}\right)$$

- Node embeddings learned by GIN can be directly used for tasks like node classification and link prediction.

# Graph Isomorphism Network

- For graph classification tasks, a sufficient number of iterations is key to achieving good discriminative power. Yet, features from earlier iterations may sometimes generalize better. To consider all structural information, we use information from all depths/iterations of the model.

$$h_G = \mathrm{CONCAT}\Big(\mathrm{READOUT}\Big(\Big\{h_v^{(k)} | v \in G\Big\}\Big) \mid k = 0, 1, \ldots, K\Big)$$

- GIN replaces READOUT with summing all node features from the same iterations and provably generalizes the WL test.

# Experimentation

- We evaluated and compared the training and test performance of GIN, GNNs with max pooling and mean pooling on the following three datasets:

1. **PROTEINS:** graphs-1113, classes-2, avg nodes-39.06, edges-72.82
2. **MUTAG:** graphs-118, classes-2, avg nodes-17.93, edges-19.97
3. **NCI1:** graphs-4110, classes-2, avg nodes-29.87, edges-32.30

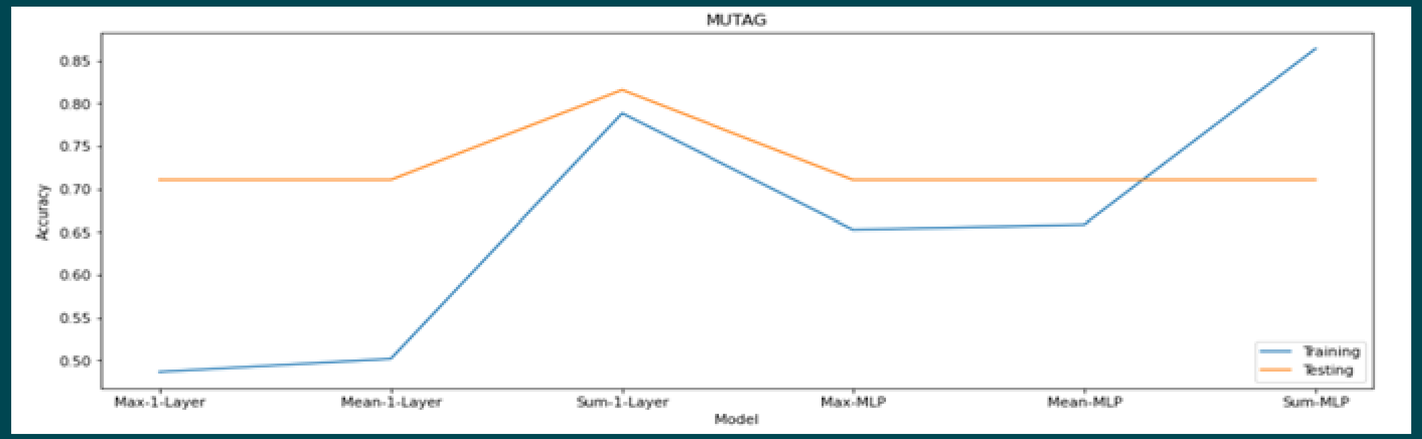
- Training set performance allows us to compare different GNN models based on their representational power.
- Test set performance quantifies generalization ability.

# Training Accuracies

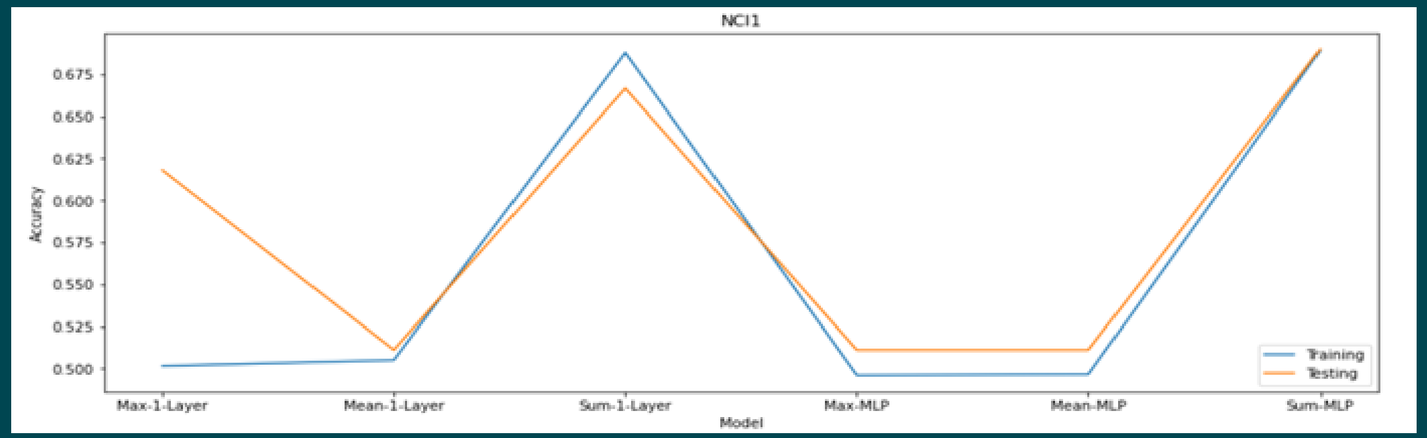
| Dataset | Max-1-Layer | Max-MLP | Mean-1-Layer | Mean-MLP | Sum-1-Layer | Sum-MLP (GIN) |
|---|---|---|---|---|---|---|
| MUTAG | 48.65% | 65.27% | 50.18% | 65.86% | 78.84% | 86.40% |
| NCI1 | 50.15% | 49.6% | 50.5% | 49.65% | 68.78% | 68.89% |
| PROTEINS | 59.89% | 62.61% | 54.02% | 60.22% | 73.05% | 72.91% |

# Testing Accuracies

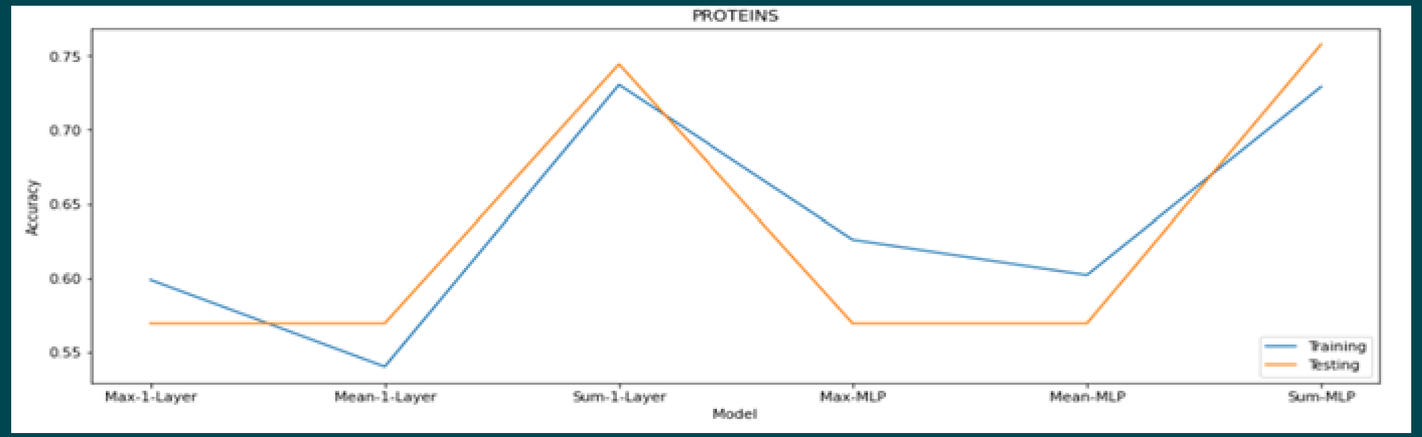
| Dataset | Max-1-Layer | Max-MLP | Mean-1-Layer | Mean-MLP | Sum-1-Layer | Sum-MLP (GIN) |
|---|---|---|---|---|---|---|
| MUTAG | 71.05% | 71.05% | 71.05% | 71.05% | 81.58% | 71.05% |
| NCI1 | 61.80% | 51.09% | 51.09% | 51.09% | 66.66% | 68.98% |
| PROTEINS | 56.95% | 56.95% | 56.95% | 56.95% | 74.44% | 75.78% |




# MUTAG




# NCI1




# PROTEINS

## Datasets

https://chrsmrrs.github.io/datasets/docs/datasets/
MUTAG, NCI1 and PROTEINS

---

## GitHub Repo

https://github.com/MauleshGandhi/Graph_Neural_Networks_SMAI2022_G29.git

---

## References


- https://davidbieber.com/post/2019-05-10-weisfeiler-lehman-isomorphism-test/
- https://arxiv.org/abs/1810.00826


---

## Contributions

Surabhi Jain & Shruti Kolachana : GNN & GIN models
Maulesh Gandhi & Deepthi Chandak : Datasets & training-testing