Julien Delavande
juliendelavande@gmail.com

Lab session # LAB7
ALTEGRAD 2024

12/05/24

Question 1

The Optimal Parameters that we expect the DeepSets architecture learn:

- Embedding Layer:
  - Weights: Map integer i to a vector $\mathbf{e}_i = \frac{i}{\text{embedding\_dim}} \times \mathbf{1}_{\text{embedding\_dim}}$.
  - Bias: Zero.
- First Linear Layer (fc1):
  - Weights:** $W_{\text{fc1}} = k \times \mathbf{1}_{\text{hidden\_dim} \times \text{embedding\_dim}}$, with k small (e.g., 0.001).
  - Bias:** Zero.
- Activation Function (Tanh):
  - Operates in the linear region ($\tanh(x) \approx x$).
- Second Linear Layer (fc2):
  - Weights: $W_{\text{fc2}} = \frac{1}{k \times \text{hidden\_dim}} \times \mathbf{1}^{\top}_{\text{hidden\_dim}}$.
  - Bias: Zero.

Calculation:
For a multiset $\{i_1, i_2, \ldots, i_n\}$:

- **Embedding Output:**
$$\mathbf{e}_i = \frac{i}{128} \times \mathbf{1}_{128}$$
- **First Linear Layer Output:**
$$h_i = k \times i \times \mathbf{1}_{64}$$
- **Tanh Activation:**
$$a_i = h_i \quad (\text{as } \tanh(h_i) \approx h_i)$$
- **Aggregation:**
$$s = \sum_{j=1}^{n} a_{i_j} = k \times \left(\sum_{j=1}^{n} i_j \right) \times \mathbf{1}_{64}$$
- **Second Linear Layer Output:**
$$\text{output} = W_{\text{fc2}} s = \sum_{j=1}^{n} i_j$$

Question 2

The DeepSets model computes the embedding of a set $X = \{x_1, x_2, \ldots, x_M\}$ as:

$$f(X) = \rho \left(\sum_{i=1}^{M} \phi(x_i) \right),$$

Let's have:

$$X_1 = \{[1.2, -0.7]^{\top}, [-0.8, 0.5]^{\top}\}$$

$$X_2 = \{[0.2, -0.35]^{\top}, [0.2, 0.1]^{\top}\}$$

So if we take:

$$W_\phi = \begin{bmatrix} 1 & 2 \\ -1 & 0 \end{bmatrix}, \; b_\phi = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

This yields:

$$\sum_{x \in X_1} \phi(x) = [1.0, -1.9]^\top, \quad \sum_{x \in X_2} \phi(x) = [0.6, -0.355]^\top.$$

Thus, X_1 and X_2 are embedded into different vectors.

Question 3

Yes, DeepSets can correspond to a submodule of a graph neural network (GNN) architecture in the context of graph classification. Specifically:

- Node-Level Representation: DeepSets can be applied to aggregate node embeddings within a graph. For a graph G, where nodes have features $\{x_1, x_2, \ldots, x_N\}$, DeepSets can aggregate these features as:

  $$h(G) = \rho\left(\sum_{i=1}^{N} \phi(x_i)\right),$$

  ensuring permutation invariance to the ordering of nodes.
- Permutation Invariance: The aggregation mechanism in DeepSets aligns with the permutation invariance required in graph-level pooling operations, such as summing or averaging node embeddings in GNNs.
- Integration with GNNs: DeepSets can act as the pooling layer in a GNN to summarize node-level embeddings into a graph-level representation, which is then used for classification.

Thus, DeepSets can be seamlessly integrated as a submodule in GNNs for graph-level tasks.

Question 4

In an Erdős–Rényi random graph $G(n, p)$, each of the $N = \binom{n}{2}$ possible edges exists independently with probability p. Thus, for $n = 15$, we have:

$$N = \binom{15}{2} = \frac{15 \times 14}{2} = 105.$$

For $p = 0.2$:

- Expected number of edges:

  $$\mathbb{E}[E] = N \times p = 105 \times 0.2 = 21.$$
- Variance of the number of edges:

  $$\mathrm{Var}(E) = N \times p \times (1-p) = 105 \times 0.2 \times 0.8 = 16.8.$$

For $p = 0.355$:

- Expected number of edges:

  $$\mathbb{E}[E] = N \times p = 105 \times 0.355 = 42.$$
- Variance of the number of edges:

  $$\mathrm{Var}(E) = N \times p \times (1-p) = 105 \times 0.355 \times 0.6 = 25.2.$$

Answer:

- For $p = 0.2$: The expected number of edges is 21, with a variance of 16.8.
- For $p = 0.355$: The expected number of edges is 42, with a variance of 25.2.

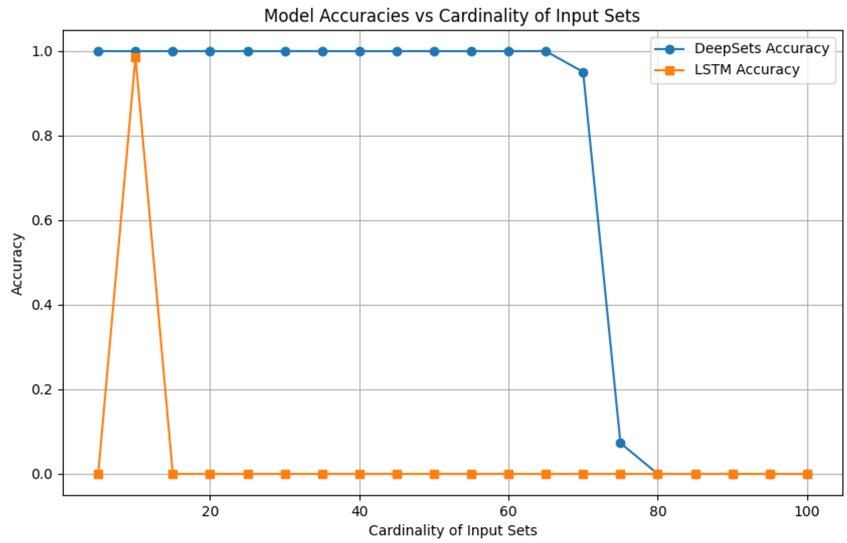


Figure 1: DeepSet to approximate sum: model accuracies vs Cardinality of Input Sets (Part 3 - Task 7)

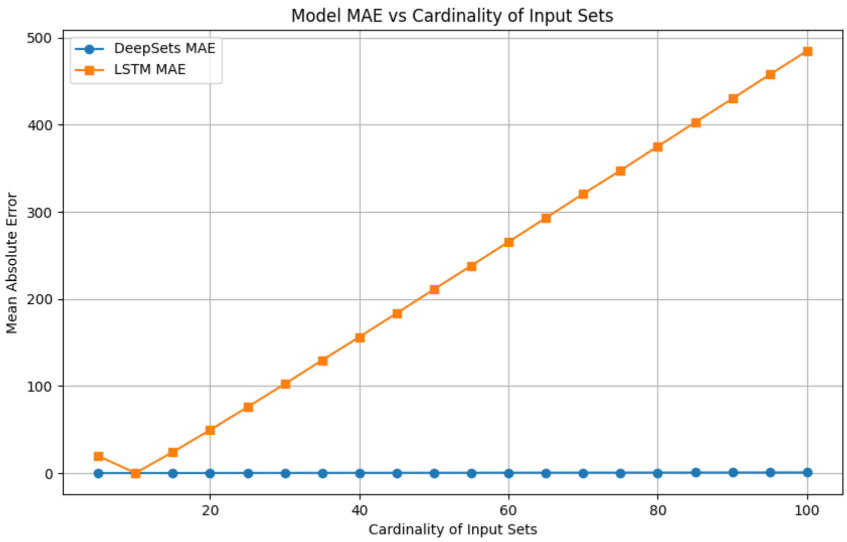
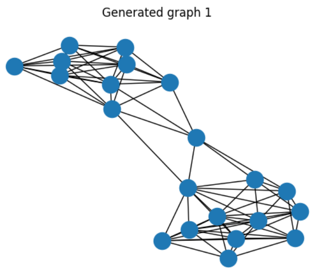
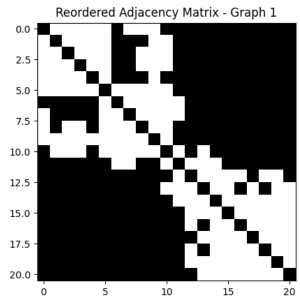


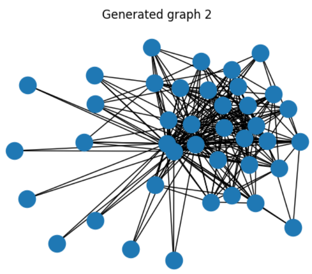
Figure 2: DeepSet to approximate sum: model MAE vs Cardinality of Input Sets (Part 3 - Task 7)



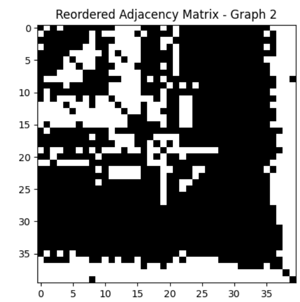
(a) Generated Graph 1



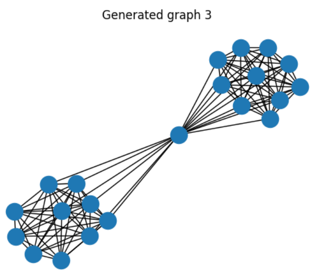
(b) Adjacency Matrix 1



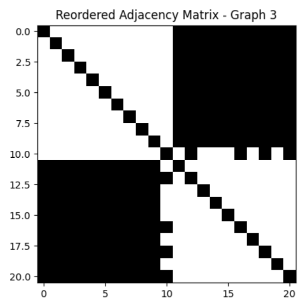
(c) Generated Graph 2



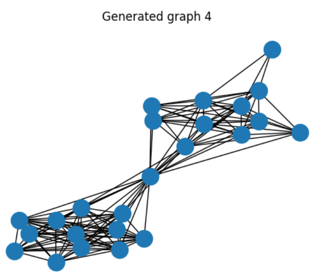
(d) Adjacency Matrix 2



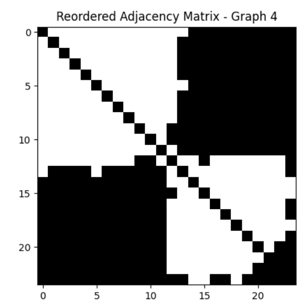
(e) Generated Graph 3



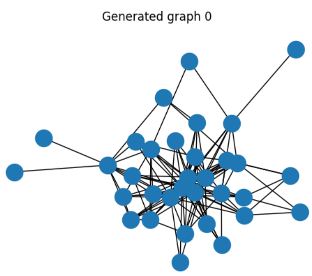
(f) Adjacency Matrix 3



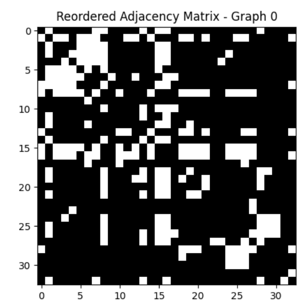
(g) Generated Graph 4



(h) Adjacency Matrix 4



(i) Generated Graph 5



(j) Adjacency Matrix 5

Figure 3: Graph Generation: Visualization of Generated Graphs and Corresponding Adjacency Matrices (Part 4 - Task 11)