Benjamin Lapostolle
benjamin.jean.lapostolle@gmail.com

1 Question 1

Long Short-Term Memory (LSTM) networks are not inherently permutation invariant models. Permutation invariance refers to the property where the model's output remains the same regardless of the order of the input elements. LSTMs, being a type of recurrent neural network (RNN), are designed to process sequential data and are sensitive to the order of the input. For instance, for a natural language processing task, they can be effective to predict words in a sequence given the context of the word. The order of the words will impact the result, hence the model will not be a permutation invariant model. In conclusion, I would not recommend the use of LSTM for this task (results of the comparison between the two models on our dataset: see figure 1).

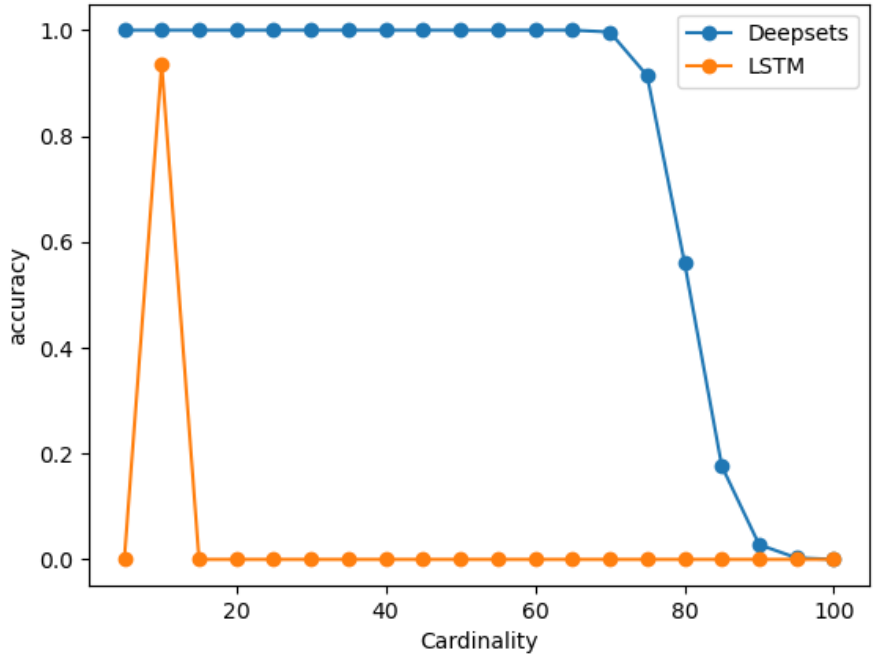

Figure 1: Accuracy on the test set

2 Question 2

The architectural differences between the graph neural network (GNN) The architectural distinctions between Graph Neural Networks (GNNs) and DeepSets primarily manifest in the utilization of the adjacency matrix to compute node embeddings before their summation for graph embedding. Specifically, the adjacency matrix encapsulates information regarding inter-node relations, a feature unnecessary in our scenario where there are no relations among items in our sets. Additionally, in our previous lab, we explored employing the mean function for generating the graph embedding, which would add other differences between the two architectures. In the case of unconnected graphs, the adjacency matrix reduces to the identity matrix.

Consequently, when employing the summation operator in GNNs, the architecture mirrors that of DeepSets. This coherence is logical, as a graph with no connections essentially represents a collection of nodes lacking any distinctive contextual relationships.

3 Question 3

For a stochastic block model with $r = 2$, two edge probability matrices P from which homophilic and heterophilic graphs can be sampled are as follows: Heterophilic Graph:

$$P_{\text{het}} = \begin{bmatrix} 0.05 & 0.8 \\ 0.8 & 0.05 \end{bmatrix}$$

Homophilic Graph:

$$P_{\text{hom}} = \begin{bmatrix} 0.8 & 0.05 \\ 0.05 & 0.8 \end{bmatrix}$$

Within the homophilic graph, the likelihood of edges within the same community is increased as the diagonal elements (representing within-community edge probabilities) if higher than the off-diagonal elements (representing between-community edge probabilities). Conversely, the heterophilic graph exhibits elevated off-diagonal elements, contributing to a higher incidence of edges between distinct communities and a low incidence of edges within blocks.

We are interested in computing the expected number of edges between nodes in different blocks of a stochastic block model with n = 20, containing 4 blocks of 5 nodes each. The diagonal elements of matrix P are set equal to 0.8, while its off-diagonal elements equal to 0.05). First, we compute the number of potential edges between nodes belonging to different blocks, then we can compute the average number of inter-blocks edges by multiplying by the probability that an edge is created. We know that for a graph of n nodes, we have $\binom{n}{2}$ potential edges. Inside one block of size 5, we have $\binom{5}{2}$. Therefore, the number of inter-blocks edges is $N = \binom{n}{2} - 4\binom{5}{2} = 150$, since there are 4 blocks. One can also compute this number by saying that each node in a block is connected to 15 nodes from other blocks, so there are $4 \times 5 \times 15 = 300$ edges, but we need to divide by 2 since we counted twice each edge, hence the result.

In conclusion, the result is: $\mathbb{E}(N_{inter}) = 150 \times 0.05 = 7.5$

4 Question 4

The binary cross-entropy loss is well-suited for predicting discrete labels. Since every operation in our neural network needs to be differentiable, returning labels as output is not feasible. Hence, we employ the binary cross-entropy loss. However, in scenarios where adjacency matrices can assume real values, it is no longer necessary to employ such workaround, and we can straightforwardly utilize standard losses like Mean Squared Error (MSE).

The loss will be then written as follows:

$$L = 1 - \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} (A_{ij} - \hat{A}_{ij})^2 \tag{1}$$