Matias Etcheverry
matias.etcheverry9@gmail.com

Lab session # 7
ALTEGRAD 2022

12/14/22

Warning: I didn't include the dataset in the .zip files as it was to big to be uploaded.

1 Question 1

In this question, we assume there is no embedding of the node features, nor `tanh` activation.

The sum operator operates on all the nodes. Thus, this operator already encodes a huge part of the prediction. The second MLP block may then be very simples: its weights are $\mathbf{1}$, while its bias is zero. With this second MLP block, we can find a very simple first MLP block. Indeed, the mapping ϕ may simply be equal to the features (=values) of the nodes, with a normalization factor, without any bias.

For instance, let $x_1 = 6$, then we have $\phi(x_1) = \frac{1}{n}[6, \dots, 6]$ in $\mathbb{R}^n$, with n the size of the hidden dimension. We then apply the sum operator on the columns of ϕ, and the second MLP block. Normalizing by n ensures that we can have a hidden dimension of any size. We finally obtain the right prediction. Moreover, we can find a huge set of valid solutions for the 2 block of MLP by playing with multiplication factors in the wieghts of the MLP blocks.

2 Question 2

Let's use the embedding vector $[0, 1]$ which maps the features of a node $\mathbb{R}^{2\times 1}$ into features in $\mathbb{R}$. We don't add any bias in the embedding. This embedding corresponds to taking the 2^{nd} features of each node:

	$\mathcal{X}_1$		$\mathcal{X}_2$	
	first node	second node	first node	second node
Original values	$[1.2, -0.7]$	$[-0.8, 0.5]$	$[0.2, -0.3]$	$[0.2, 0.1]$
Embedding	-0.7	0.5	-0.3	0.1
`tanh`	≈ -0.60	≈ 0.46	≈ -0.29	≈ 0.10
Sum over all nodes	≈ -0.14		≈ -0.19	

Table 1: Hidden features with respect to the readout function.

In the end the embedding computed on all nodes is different. In fact, any vector as $[\alpha, \beta]$ where $\alpha \neq \beta$ would work as an embedding. If $\alpha = \beta$, then the embedding of each node would be the sum of its features which causes problems in that case.

3 Predicting sum of digits

Here are the test curves I obtained after training the 2 models on 20 epochs:

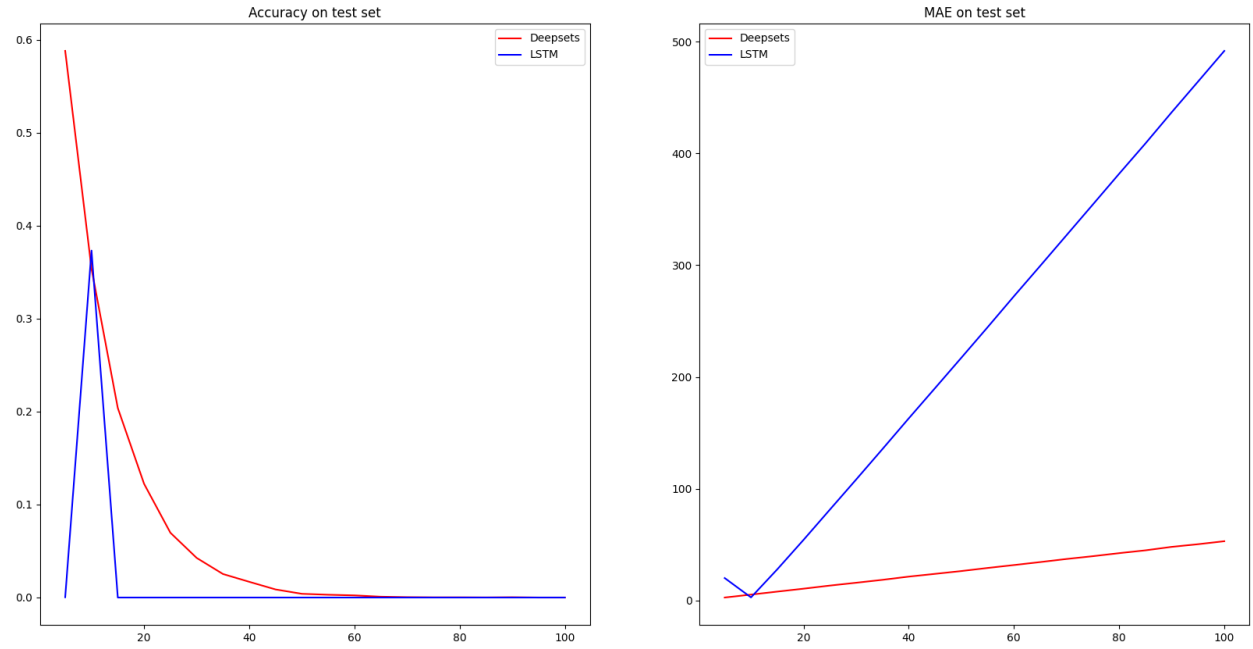

Figure 1: Comparion of LSTM and Deepsets models.

We see that the DeepSets model is performing better, in terms of accuracy and MAE.

4 Question 3

Let the input be the graph and the targets the class it belongs to. The most important part is the embedding of the nodes.
Let's suppose a graph G_1 is made of equation while another graph G_2 is made of irrational numbers. To distinguish them, we need a strong embedding layer. First of all, we tokenize all the nodes of each graph. A tokenized equation will simply be a sequence of digits, and those digits refer to some mathematical operations: a fraction bar , a number 3 or mathematical operator $\sim$.
Then we use an embedding layer to distinguish between equation nodes in G_1 and numbers in G_2.

5 Antibiotic Resistance Classification

table 2 shows the metrics obtained after training for 50 epochs. We may be overfitting a bit.

	Loss	Accuracy
Train set	0.4344	77.49%
Test set	0.5769	74.42%

Table 2: Metrics obtained on resistance classification

6 Question 4

Proteins arise as sequences of amin oacids. for isntance, if we take the 3 amin oacids α, β and γ, then the sequences $\alpha\beta\gamma$ could lead to a different protein than $\beta\alpha\gamma$. In that case, having a invariance on permutation may not be relevant. Moreover, in the case of large sequence of amin oacids, $XXXXXX\alpha\beta\gamma XX$ may have some common properties as $XXX\alpha\beta\gamma XXXXXXX$. Thus, it is still interesting to be equivariant on permutations.

The current model is permutation equivariant until the readout functions which looses the order of the amin oacids as information.
One way to keep the order of the amin acids it to compute features on batch of amin oacids (3 in our example), with a sliding window, using a dense layer. We then apply a readout functions on those features.

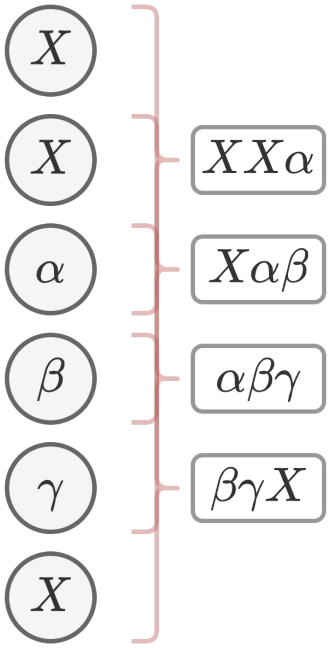

Figure 2: Aggregation function on batch of amin oacids to preserve global invariance on permuations.