Deep Learning Fall 2024

To do these exercises, you will use Python 3 and the following packages:

- PyTorch. This package is a versatile deep learning framework that facilitates building and training of neural networks.

You are not strictly forced to use these packages, but it is highly recommended. Feel free to use other packages you think are necessary.

Additionally, the python package pytest may be helpful to test your implementation. You can install pytest via `pip install pytest`.

We will work with the provided script: **`ex05.py`**

Exercise 1 (Presence)

You will build a Transformer model from scratch. Have a look at the accompanying Python script ex05.py. There you will find different modules that together form a (slightly simplified) Transformer architecture and you will implement the body of the forward function of the each module step by step. You can run tests on your implementation by running `pytest ex05.py` to check whether the shapes are correct (you may need to install `pytest`).

a) Self-Attention. Implement the forward method of the `SelfAttention` module. The forward method should first project the input into queries **Q**, keys **K**, and values **V**. Then, self attention should be applied according to the formula:

$$\text{Attention}\left(\mathbf{Q},\mathbf{K},\mathbf{V}\right) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{d_k}}\right)\mathbf{V}$$

Hint: Use the `transpose` method of PyTorch tensors, but mind the batch dimension.
Hint: The key dimension d_k can be inferred from **K** via the `size` method.

b) Multi-head Self-Attention. Implement the forward method of the `MultiHeadSelfAttention` module. Each head is a `SelfAttention` module and operates on the same input **x**. The results of all heads then need to be concatenated (`torch.cat`) and fed into the output projection (`w_o`).
Hint: You can treat the `ModuleList` object `heads` like a standard Python list, where each element is a `SelfAttention` module.

c) TransformerBlock. Implement the forward method of the `TransformerBlock` module. A transformer block consists of first applying multi-head self-attention and then a fully connected feedforward layer. Apply layer normalization before feeding the input into each of the two submodules. Furthermore, each of these two submodules should be wrapped with a residual connection.

d) TransformerClassifier. Implement the forward method of the `Transformer` module. First, apply the token embeddings to the input. Second, apply the transformer model to the embeddings. Third, aggregate the results over the sequence dimension via mean pooling. Lastly, apply the classifier on this aggregated representation.

Exercise 2 (Presence)

For testing your implementation, there is a dummy task prepared that you can invoke by executing the script via `python3 ex05.py` – the task is: given an array of binary values, determine if the number of ones is even or odd. Play with the hyper-parameters of the model and of the optimization procedure to get a feeling of their effect. What is the smallest configuration with which you can achieve a reasonable test accuracy (higher than 95%)?