1 Question 1

The square mask is a square matrix of size `(sz, sz)` used in the attention mechanism to mask future positions for each input vector in the self-attention layer of the transformer. The idea is to set all the future positions to `-inf`, in order not to take into account future words before applying the softmax. This leads to an unidirectional self-attention. Positional encoding is a way to account for the order of the words in the input sequence. Before input embeddings are fed into the self-attention layer, we calculate a positional encoding vector for each embedding and then add this vector to the corresponding input embedding.

2 Question 2

We need to replace the classification head in order to switch from the initial task (language modeling) to the new task (sentiment analysis) when we want to process fine-tuning. Fine-tuning is based on a pre-trained model which targets an other task, so the target is different and we reinitialize the parameters on the module. In the language modeling task, we consider the entire sequence and predict the next token. The output is therefore a vector of the size of the vocabulary, to which we apply the softmax function to obtain probabilities for each word and choose the most probable token. For classification, we have the same output except that we only consider the last vector of the sentence (excluding padding) as its representation in order to predict the class of the sentence. We obtain for each sentence a 2-dimensional vector to which we apply softmax function and argmax to predict the class (a scalar belonging to $0, 1$).

3 Question 3

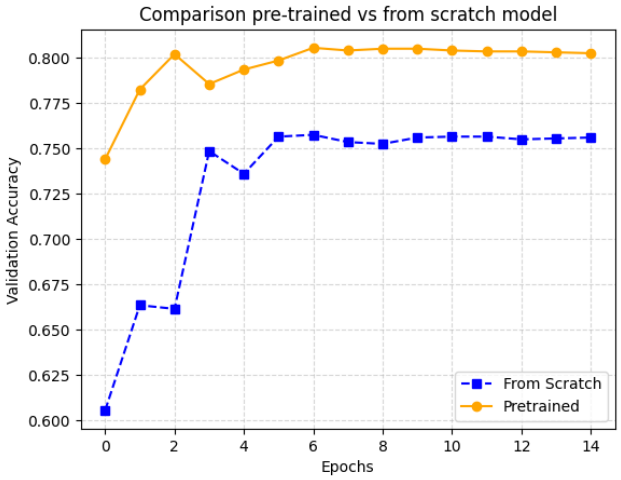
Number of parameters
We have $d_{\text{model}} = n_{\text{hid}} = d_{\text{embedding}}$.

- **Embedding layer:**
$$d_{\text{embedding}} \times n_{\text{tokens}} = n_{\text{hid}} \times n_{\text{tokens}}$$
- **Transformer block (case of one layer):**
  - **For the multi-head attention layer:**
    - Linear projection that transform the input embeddings into queries (Q), keys (K), and values (V) for multi-head attention:
      - $n_{\text{head}} \times 3 \times n_{\text{hid}} \times \frac{n_{\text{hid}}}{n_{\text{head}}} = 3 \times n_{\text{hid}} \times n_{\text{hid}}$ parameters for the weight, where n_{head} is the number of heads, 3 the number of projections (Q + K + V), n_{hid} is the dimension of the embeddings, and $\frac{n_{\text{hid}}}{n_{\text{head}}}$ is the dimension of the Q, K, and V vectors.
      - $3 \times n_{\text{hid}}$ for the bias.
    - Projection to produce the output of the layer:
      - $n_{\text{head}} \times n_{\text{hid}} \times \frac{n_{\text{hid}}}{n_{\text{head}}} = n_{\text{hid}} \times n_{\text{hid}}$ parameters for the weight.
      - n_{hid} for the bias.
  - **Feed-forward network:** We have dim feedforward $= n_{\text{hid}}$. Thus, for each of the feed-forward layers, we have
    - $n_{\text{hid}} \times n_{\text{hid}}$ parameters for the weight,
    - n_{hid} parameters for the bias.

  We multiply the total by the number of transformer decoder layers n_{layers}.
- **Classification head:**
  - Weight matrix of the linear layer in the classification head:
$$n_{\text{hid}} \times N$$
    where $N = n_{\text{tokens}}$ in the case of language modeling task, and $N = n_{\text{classes}}$ in the case of classification task.
  - Bias: N with same possible values.

4 Question 4

We observe a significant performance difference between the model that is ”pretrained” and the one trained ”from scratch.” The former has a faster convergence and achieve higher accuracy.



The pretrained model has already learned some useful representations and features from the pretraining task and a large corpus of text. So it’s starting from a point that is closer to the optimal solution for the new task, compared to the other model. So it requires fewer epochs to adjust its weights and achieve good performance on the sentiment analysis task, leading to faster convergence.
The final accuracy is higher because pretrained model is trained on a large and diverse dataset, which helps them generalize better to unseen data. Plus it may have captured the nuances in the text so gains in terms of world knowledge and relatively long-range dependencies handling (how each word is related to each other), that are important for sentiment analysis.

5 Question 5

BERT [1] uses masked language modeling, that is a task of predicting which words should fill blanks of a sentence. So BERT considers both past and future context when learning to reconstruct a sentence. This means that the model can use all the available surrounding information to understand the meaning of a word and make more accurate predictions through better understanding of the context. Thus a limitation of left-to-right language modeling task we are using is that it only considers past context for prediction, and so it leads to less meaningful representations. Indeed words can depend both on past as well as on future positions.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.