1 Question 1

The role of the square mask is key in autoregressive models. It prevents the model from seeing future output tokens by masking them in the attention computation, only previous output tokens will be hence considered [2]. Otherwise the model would have access to future information, thus skewing the results and reducing model's performance.

The positional encoding also plays a key role in transformer architecture, it allows to introduce a notion of position and order in each embedded token. Indeed, contrary to former RNNs, transformer do not process data sequentially, but the position of each token is still a very important information. Here is the formula of the positional encoding which are added to the embedding before going through the network [2] :

$$PE_{(pos,2i)} = sin\left(\frac{pos}{10000^{2i/d}}\right)$$

$$PE_{(pos,2i+1)} = cos\left(\frac{pos}{10000^{2i/d}}\right)$$

With pos, the index of the position in the sequence, i, the index in the embedding vector and d the dimension of the embedding. The main strength of the sinusoidal representation is that it works great for any sequence length and it can also easily interpolate word position it has not yet seen during inference.

2 Question 2

The two tasks have very different objectives. The language modelling task's objective is to predict the next token of a given sequence while the classification task's objective is to predict a score telling whether a review is positive or negative.
Hence the language modelling task allows our model to understand the language structure with data that does not need labels (as in this case we only intend to predict the next tokens of a sequence) and the classification task fine-tunes the weights of our model to make them specifically optimized for our objective. Usually, pre-training is done on a larger dataset as it does not require labels like classification does.

3 Question 3

Lets start with language modelling task, here is the list of the main parts of the models and its corresponding parameters :

- The first part of the model is the embedding part which has a total number of $ntokens * nhid$ parameters because every token needs to be mapped in a vector of the embedding size corresponding to $nhid$. In our case we had $ntokens = 50001$ and $nhid = 200$ which makes **10.000.200 parameters in total for the embedding part**.

- The second part is the multi-head attention. The main parameters are in Key, Value and Query matrices which are multiplied with the entry to get the q,k,v vectors. These matrices' size is $nhid * (nhid/nhead)$ with $nhead$ the number of head in the attention, we also add the $3 * (nhid/nhead)$ biases for each q,k,v vector. As we multiply the amount of parameter with $nhead$, the total number of parameter for multi-head attention is $3 * nhid * (nhid + 1)$ which in our case equals 120600. Moreover, right at the end of the attention layer, there is a liner layer with input and output size $nhid$ which makes $nhid * (nhid + 1) = 40200$ parameters.

- After the attention layer there are two linear layers with two normalization layers (which are still located in the transformer layer). Both linear layers have input and output size of $nhid$ and each normalization layer has $2 * nhid$ parameters (for scale and offset) which makes a total number of parameter of $2 * nhid * (nhid + 1) + 2 * 2 * nhid = 81200$. As we have several transformer layers, we have to multiply the total parameters of the two previous points by $nlayer = 4$ which makes a total of $4 * (120600 + 40200 + 81200) =$**968.000 parameters for the transformer layers**.

- The last part is the classifier which in the language modelling task is a linear layer with input size $nhid$ and output size $ntokens$ (as we want to predict a score for each token). The number of parameters is hence $ntokens * (nhid + 1) =$**10.050.201 for the classification part**

For the language modelling task, the final **number of parameter is 21.018.401** for our case.

As for the classification task, the model is exactly the same except the last linear classification layer. Instead of having $ntokens$ output size, the linear layer only has two outputs to predict the class score, thus only $2 * (nhid + 1) = 402$ parameters. This modification drops the number of parameter to $10.000.200 + 968.000 + 402 =$ **10.968.602 for the classification task**.

4 Question 4

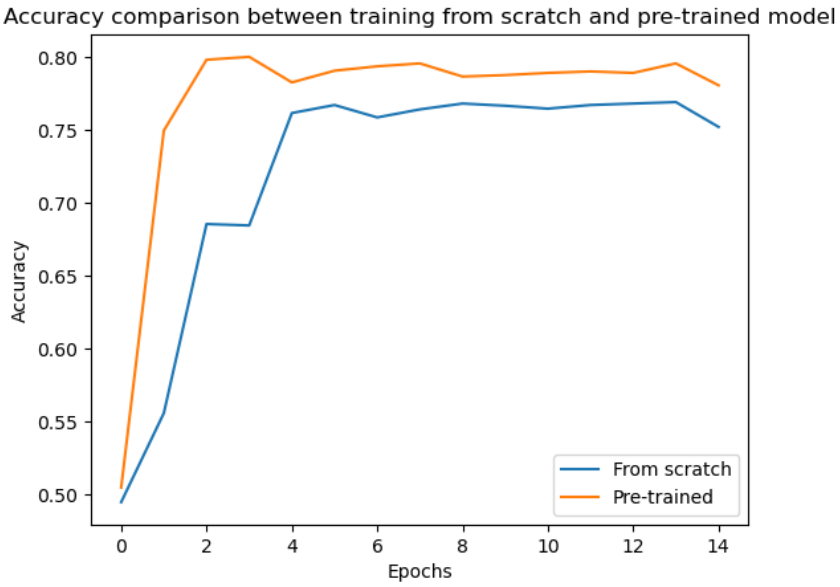

Figure 1: The evolution of accuracy of both models

We can see on the fig. 1 that both the model trained from scratch (we will call it FS) and the pre-trained model (we will call it PT) start with a relatively low accuracy (around 0.5) but manage to learn quickly and within a few epochs, both manages to reach higher accuracy (between 0.75 and 0.80).
However, the PT model converges a lot faster to its optimal performance (at epoch 2) while the FS model takes 4 epochs to converge, moreover, the PT model's accuracy is slighly higher than the FS model's (around 3% better)
We can conclude from these observations that the pre-training seems to indeed have an impact on the training as it converges quicker and has slightly better performances, thus justifying the interest of a pre-training.

5 Question 5

The main limitation of the language modelling objective of the lab is that it is unidirectional which means that the model only reads data from left to right, thus limiting the complete understanding of a context. Another architecture called BERT [1] (Bidirectional Encoder Representations from Transformers) introduced a bidirectional model reading sequences both left-to-right and right-to-left.
They replaced the initial objective of predicting the next token of a sequence by masking random tokens of a sequence and predict them, this architecture is particularly efficient for Question-Answer task as the context is needed in both directions.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.