1 Question 1

1.1 Square Subsequent Mask

A mask is utilized to prevent the model from attending to future tokens in a sequence. This is important because the prediction should not depend on future tokens [5]. In order to achieve such a thing, the mask is a matrix that has negative infinity in positions corresponding to future dependencies have a zero value after the application of the softmax function as seen in 1.

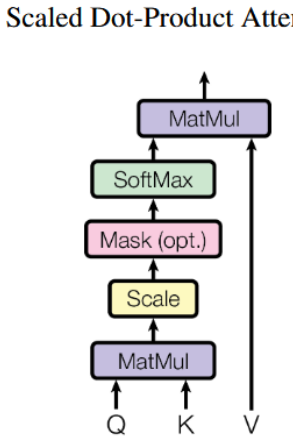

Figure 1: Use of mask in attention computation. Adapted from [5]

1.2 Positional Encoding

The positional encoding can give information to the model about the position of the token in the sequence. Indeed, the model is not aware on how tokens are ordered. As described in [5] and in [4], positional encoding matrix is a combination of sin and cosine functions 2 that, combined with the word embedding matrix will give information about token position for the next layers.

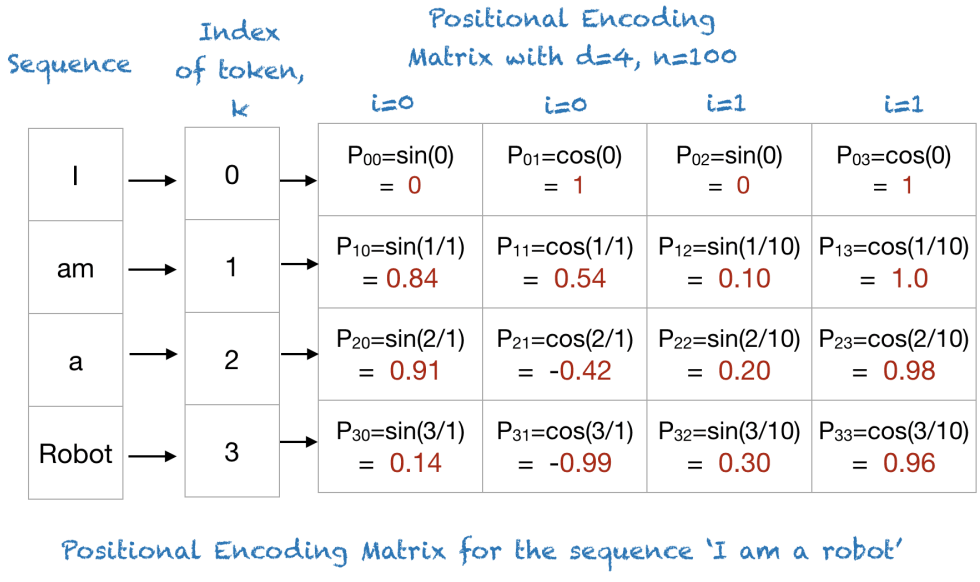

Figure 2: Positional encoding matrix. Adapted from [4]

2 Question 2

2.1 Language modelling

Language modeling and classification tasks are two different tasks that require to switch the classification head. Indeed, **language modelling** predict the next word in a sequence, the output is a probability distribution that represents the likelihood of each word to be the next one in the sequence. It is based on the vocabulary.

2.2 Classification

Classification on the other hand assign a label to an input sequence. More precisely, the goal is to assign a label to an entire sequence of data. That's why the classification head will consist of linear layers that will map the feature representation of the data into the label space.

3 Question 3

To compute the total number of parameters, we have to understand precisely what does each layer does. As we are in a multi-head attention part we are following the scheme of Fig 1 that is done by multiplying nhead times (Fig 3).

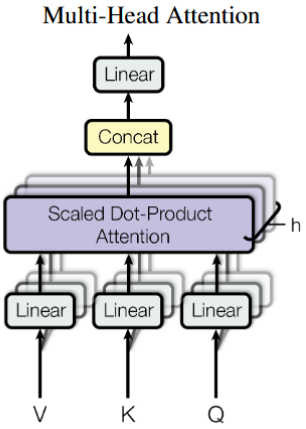

Figure 3: Multi-head attention. Adapted from [5]

Concerning the feed-forward layer as depicted in Fig 4, we have $nparameters ffnn = 2 \cdot ((nhid)^2 + nhid)$.

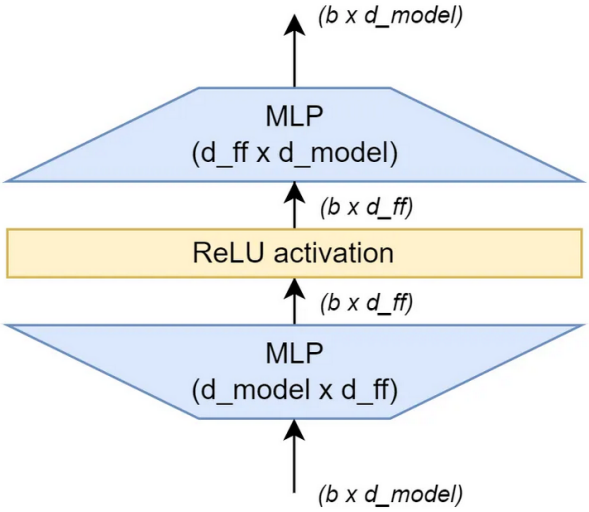

Figure 4: Transformer feed-forward network. Adapted from [3].

Concerning the norm layer as shown in Fig 5, $nparamaters = 2 \cdot nhid$. The two trainable parameters are β and γ that are both of $nhid$ dimension.

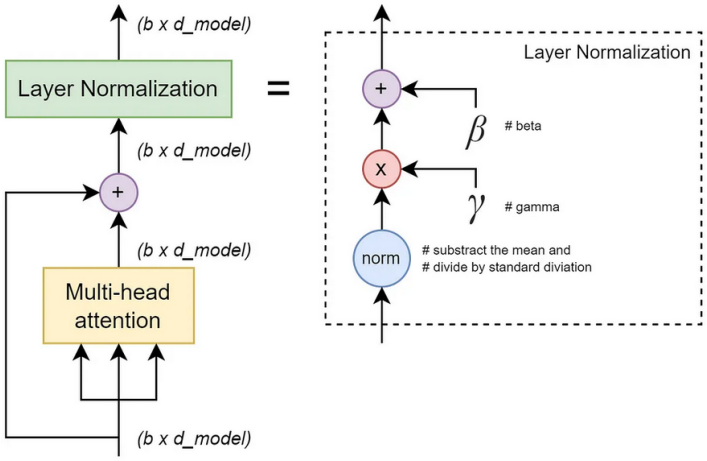

Figure 5: Normalization layer from the encoder of the transformer. Adapted from [3].

3.1 Language Modelling Task

In this task, we have to consider the following components:

- the **embedding layer** that converts token indices into embeddings; so $nparametersemb = ntokens \cdot nhidparameters = 100 \cdot 200 = 20000$ parameters
- the **positional encoding layer** that adds positional information to the embeddings; 0 parameters (as there are non trainable parameters)
- **transformer encoder layers** that process the embeddings to capture contextual information; each layer consists:
  - multi-head attention
    - the multi-head attention mechanism consists of three linear transformation that map the input sequence to the query Q, key K,and value V representations. Each of these transformations has an associated weight matrix of shape (embedding dimension, embedding dimension) and bias vector of length equal to the embedding dimension with n_{head} heads. Then there is the shape of the last linear layer that we need also to consider, the output.
    - therefore for each weights there are $nparam1 = 3 \cdot (embeddingdimension)^2$
    - for each biases there are $nparam2 = 3 \cdot embeddingdimension$
    - $nparametersinput = (nparam1 + nparam2)\}$
    - $nparametersinput = 3 \cdot ((embeddingdimension)^2 + embeddingdimension)$
    - $nparametersinput = 3 \cdot ((n_{hid})^2 + (n_{hid}))$
    - $nparametersinput = 3 \cdot ((200)^2 + 200) = 120600$
    - $nparametersoutput = (200^2 + 200) = 40200$
    - $nparametersmha = 120600 + 40200 = 160800$
  - feed-forward neural network
    - two weight matrices and two biases matrices
    - $nparametersffnn = 2 \cdot ((embeddingdimension)^2 + embeddingdimension)$
    - $nparametersffnn = 2 \cdot ((200)^2 + 200)$
    - $nparametersffnn = 80400$
  - normalization layer
    - both $nhid = 200parameters$ for the self-attention and feed forward
    - $nparameters = 200 \cdot 2 = 400$
  - total number of parameters, we multiply by the number of layers
    - $nparameterstotal = (160800 + 80400 + 400) \cdot 4$
    - $nparameterstotal = 968000$
- **language modelling head**; $nparameters = embeddingdimension \cdot vocabularysize + bias = n_{hid} \cdot n_{tokens} + bias = 200 \cdot 100 + 100 = 20100$

Therefore :

$$\textbf{nparameterslm= 20000 + 968000 + 20100 = 1008100} \quad (1)$$

3.2 Classification Task

The only modification is with the classification head where the number of parameters for this layer is $nparameters = embeddingdimension \cdot numberofclasses + bias = n_{hid} \cdot n_{classes} + bias = 200 \cdot 2 + 2 = 402$

$$\textbf{nparametersclass= 20000 + 968000 + 402 = 988402} \quad (2)$$

These results can be verified thanks to the numel function in pytorch that counts the number of parameters in our model by returning the number of elements in the tensor we give the model as input.

4 Question 4

What we observe Fig 6 is that there is a performance gap between the pretrained training and the from scratch training. Indeed the pretrained model achieve an accuracy aroung 80 % where as the training from scratch only achieve an accuracy around 75 %. The pretrained model has a faster convergence for the good solution, it requires less epochs. In addition, the pretrained model is more stable than the from scratch one. Furthermore, the first epochs of the of the from scract model has a very low accuracy (around 55 %), this is easily explained by the fact that weights are randomly initialized at the beginning of the training.

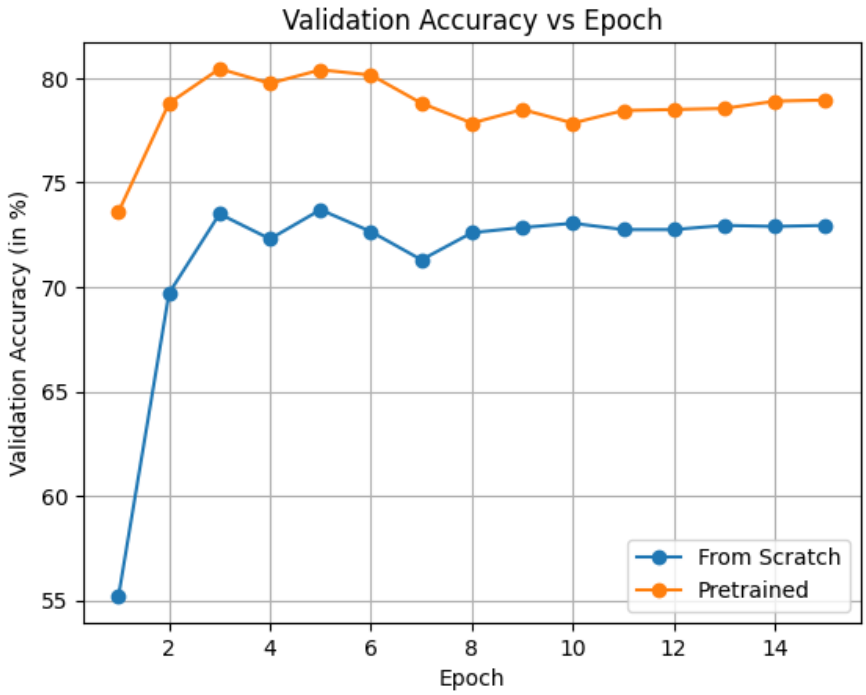

Figure 6: Accuracy score pretrained model vs training from scratch.

5 Question 5

First of all, the language modeling objective used in our current model is a unidirectional approach from left to right. What it does, is that it can predict the next word of the sequence based on the previous ones. One downside of it is that this does not take into account the future context. Is there a mean to take it into account in our training process ?

Rapidly, BERT (or Deep Bidirectional Transformers) can capture the context from both left and right side of a token. As described in [1], the "masked language model (MLM) pre-training objective" consists to mask some of the tokens of the input (randomly) and then to predict the original vocabulary only based on its context. As it is used in BERT is because MLM enables to fuse the left and right context at the same time instead of

keeping only the left to right context. In [1], it is proven that this pre-training has way better results than the classical Left to Right model, cf Fig 7.

Tasks	Dev Set MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$\text{BERT}_{\text{BASE}}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

Figure 7: Accuracy results compared to their pre-training. BERT base & No NSP are with MLM pretraining, LTR is left-to-right pretraig. Adapted from [1].

From an input sequence, some tokens are masked with the MASK token. The model is then trained using cross-entropy loss. This pretraining is used in the derivative of BERT that are presented such as BARThez or CamenBERT [2].

References

[1] Kenton Lee Kristina Toutanova Jacob Devlin, Ming-Wei Chang. Bert: Pre-training of deep bidirectional transformers for language understanding. In *arXiv prepring at arXiv:1810.04805*, 2018.

[2] Antoine J-P Tixier Moussa Kamal Eddine and Michalis Vazirgiannis. Barthez: a skilled pretrained french sequence-to-sequence model. In *arXiv preprint arXiv:2010.12321*, 2020.

[3] Dmytro Nikolaiev. How to estimate the number of parameters in transformer models. In *Medium / Towards Data Science*, 2023.

[4] Mehreen Saeed. A gentle introduction to positional encoding in transformer models. In *Machine Learning Mastery*, 2023.

[5] 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.