Gertner Victor
gertnervictor@gmail.com

Lab session # 4
ALTEGRAD 2023

11/14/23

1 Question 1

So we are going to compute the number of parameters of the model manually here we omit the biases and the normalization layers, moreover we don't count the parameters in the language model head:

- **Embedding :** We have an embedding that projects a space of size vocab_size onto a space of size n_{hid} .
- **Learned Positional Embedding :** Here it's a projection that goes from max_sentence_size onto a space of size n_{hid} .
- **Attention Model :** Then in this architecture we have 4 stacks of transformer architecture, all have the same parameters. Hence we are going to show for one layer.
  - **Q,V,K Projections :** We have the classical transformer self attention model architecture, with 3 linear projection from n_{hid} to n_{hid2} and then one outer projection to compute the attention from n_{hid2} to n_{hid3} .
  - **Feed Forward :** We have two feed forward that go from n_{hid3} to n_{hid}.

So we have a general formula for the number of parameters in this case :

$$n_{params} = vocab_{size} * n_{hid} + max_{sentence\_size} * n_{hid} + 4 * (3 * n_{hid} * n_{hid2} + n_{hid2} * n_{hid3} + 2 * n_{hid3} * n_{hid}) \quad (1)$$

In our case, this gives us, having $n_{hid} = n_{hid2} = n_{hid3} = 512$, $vocab_{size} = 32000$, $max_{sentence\_size} = 258$
$n_{params} = 32000 * 512 + 258 * 512 + 4 * (5 * 512 * 512) = 22807552$

2 Tensorboard Pretrain Vs Random :

Validation
We are now looking at validation.

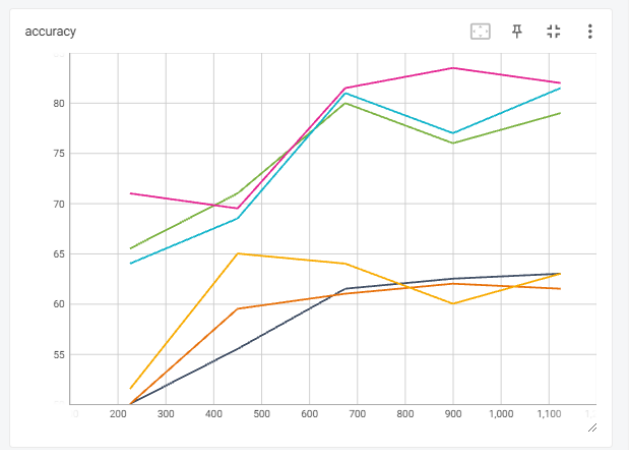

Figure 1: Validation accuracy score for Pretrain Model (Upper curves) and Random Model (Lower curves)

So as we can see, the validation accuracies are way better with a pretrain model than with a a random one. Still, it seems that both are converging pretty clearly.

We are now going to now see the checkpoint with the best validation accuracy for each of those curves.

Curve	Best Step	Value on Valid	Correspond Accuracy on Test
Pretrain Seed 0	1125	81.5	79.9
Pretrain Seed 1	675	80	80
Pretrain Seed 2	900	83.5	83.5
Random Seed 0	900	62	65.4
Random Seed 1	450	65	66.7
Random Seed 2	1125	63	67.4

Table 1: Best validation Score and correspond Test Score

We see that for both models, the value are pretty similar for each seeds it tells us that our model is not dependant on the initialisation and the randomness of adam.

Average Accuracy and Standard deviation

Now let's see the results on Test

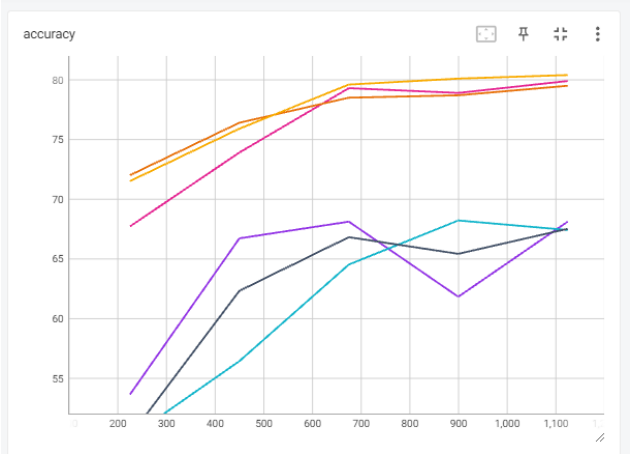

Figure 2: Test accuracy score for Pretrain Model (Upper curves) and Random Model (Lower curves)

As we can clearly see, the 3 top curves are the one from the pretrain model, it performs way better and also the convergence is quite clear between the 3 seeds, whereas on the random model, it's not quite clear how it converges. The performance gap is logic, because the weights of the pretrain model are far more precise since the model has already done task using words and building a strong understanding of language.

	Performance	
	Average Accuracy	Standard Deviation
Pretrain Model		
Epoch 1	70.4	1.92
Epoch 2	75.4	1.08
Epoch 3	79.1	0.46
Epoch 4	79.2	0.61
Epoch 5	79.9	0.37
Random Model		
Epoch 1	51.2	1.67
Epoch 2	61.8	4.2
Epoch 3	66.5	1.48
Epoch 4	65.1	2.61
Epoch 5	67.7	0.31

Table 2: Average Accuracy and Standard Deviation for Pretrain and Random Methods

As we can see, it's much more stable on the pretrain model, as the standard deviation is going faster to a small values, whereas for the random model, the standard deviation isn't decreasing, it's fluctuating between low and high values. This is also to take on account not only does the pretrain is more precise, it's also more stable and behave better in training : when we see it converging we are sure it's going to be the best possible accuracy value we can get in this setup.

3 Hugging Face

Here, with my script, I only achieve to show the eval curves on TensorBoard and I can't find why on internet. Thus, I'm only showing here the validation cures (which are the eval curves of HugginFace if I'm not mistaken).

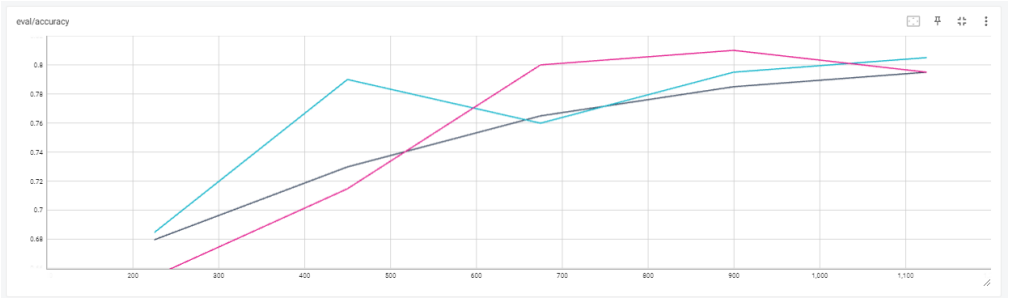

Figure 3: Validation curves on Hugging Face

So as we can see, we have the same global shape than the fairseq one, that is nice, it seems that our hyper parameters are well set. Moreover we see that the validation curves converge clearly.
I cannot do more for this task.

4 Question 2

Description of the parameters of **LoraConfig** :

- **r** :It's the rank of the the decomposition matrix A and B, The rank has to be really small compared to the real rank of the weights matrix in order to have efficiency gains compared to classic fine tuning.
- **lora_alpha** : It's the scaling factor of the learned weights : The matrices are scaled by $\frac{lora\_alpha}{lora\_rank}$, it shows how much impactfull our fine tunning is going to be.
- **target_modules** :It's the part of our model where we are going to use the Lora strategy. Here we see that all the attention layer matrices are concerned.
- **lora_dropout** : It's the dropout rate on the neuron creating the decomposition matrix : percentage of neuron set off during each epochs of training. This permits to avoid over fitting during training since our model is artificially less complex.
- **bias** : It's the parameter specifying the bias type for Lora it can be 'none', 'all', or 'lora_only'.
- **task_type** : It's the type of task that the model will fine tune. Here it's set on Causal_ML which stands for causal language modeling.