Ben KABONGO B.
ben.kabongo_buzangu@ens-paris-saclay.fr

1 Question 1

- **Normalization layers** : Note that all normalization layers have weights and biases that take in and output 512-dimensional data. There are 512 parameters for weights, 512 parameters for biases. A total of 1,024 parameters.
- **Model input** :
  - **Tokens embedding** : The vocabulary size is 32,000. The size of the model embeddings is 512. The number of parameters in the model embedding module is therefore 32,000 x 512 = 16,384,000.
  - **Position embedding** : The position embedding module manages 258 different input positions and projects them into a 512-dimensional space. It comprises 258 x 512 = 132,096 parameters.
  - **Normalization layer**

  When we put together the token embedding, position embedding and normalization layer modules, we have a first total of 16,384,000 + 132,096 + 1,024 = **16,517,120 parameters**.
- **Encoder module** : An encoder module comprises an attention module, two full-connected layers and a normalization layer.
  - **Attention module** : An attention module includes weights and biases for elements K, Q, V and for the final projection. For these four elements, input and output are in dimension 512.
    For just one of these elements, the total number of parameters and biases is 512 x 512 + 512 = 262,656.
    This gives a total of 4 x 262,656 = 1,050,624 parameters.
    Adding the 1024 parameters of the normalization layer, an attention module has a total of 1,051,648 parameters.
  - **Full-connected** : The input and output of the full-connected layers is 512. The total number of weights and biases is therefore 512 x 512 + 512 = 262,656.
    There are two full-connected layers per encoder, making a total of 2 x 262,656 = 525,312 parameters.
  - **Normalization layer**

  The total number of parameters for an encoder module is therefore 1,050,624 + 525,312 + 1,024 = 1,577,984 parameters.
  The model includes 4 layers of encoders. The total number of encoder parameters is 4 x 1,577,984 = **6,331,936 parameters**.

The total number of model parameters is therefore 16,517,120 + 6,307,840 = **22,829,056 parameters**.

2 Task 3

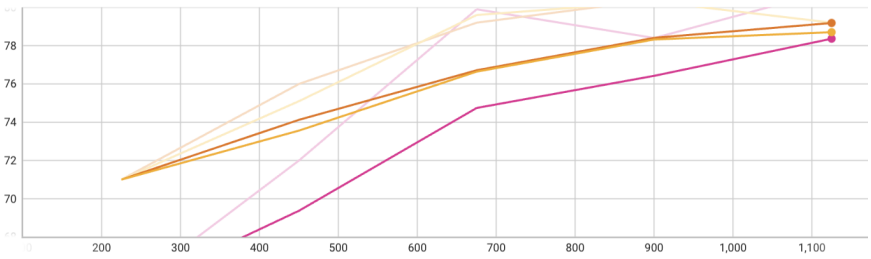

Figure 1: Accuracy of Roberta's finetuning with Fairseq

The average accuracy of the model is **80.10** with a standard deviation of **0.69**.

3 Task 4

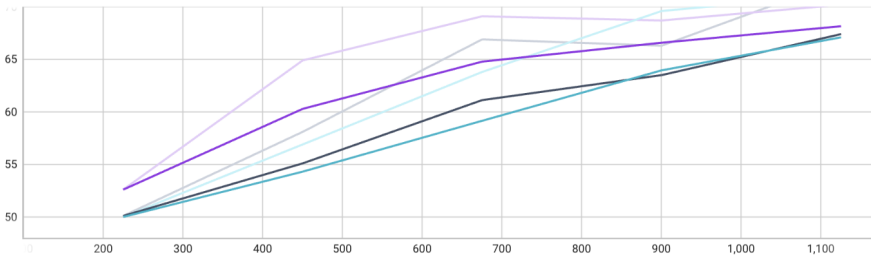

Figure 2: Accuracy of random model's finetuning with Fairseq

The mean accuracy of the random model is **71.39** with a standard deviation of **0.82**.
The random model has a smaller mean accuracy and a larger variance than the pre-trained model.

4 Question 2

The **LoraConfig** [1] class serves as the configuration container for a **LoraModel**.
It includes the following parameters:

- **r** (Type: *int*): Represents the Lora attention dimension.
- **target_modules** (Type: *Union[List[str],str]*): Specifies the names of the modules to which Lora should be applied.
- **lora_alpha** (Type: *int*): Sets the alpha parameter for Lora scaling.
- **lora_dropout** (Type: *float*): Determines the dropout probability for Lora layers.
- **fan_in_fan_out** (Type: *bool*): If set to True, indicates that the layer to be replaced stores weights in the format (fan_in, fan_out).
- **bias** (Type: *str*): Specifies the bias type for Lora, which can be *none*, *all*, or *lora_only*. If *all* or *lora_only* is chosen, the corresponding biases will be updated during training.
- **modules_to_save** (Type: *List[str]*): Lists modules, other than LoRA layers, to be set as trainable and saved in the final checkpoint.
- **layers_to_transform** (Type: *Union[List[int],int]*): Indicates the layer indexes to transform. If specified, LoRA transformations will be applied to the specified layer indexes. If a single integer is passed, the transformations will be applied to the layer at that index.
- **layers_pattern** (Type: *str*): Specifies the layer pattern name, used only if *layers_to_transform* is not None and if the layer pattern is not in the common layers pattern.
- **rank_pattern** (Type: *dict*): Maps from layer names or regexp expressions to ranks different from the default rank specified by *r*.
- **alpha_pattern** (Type: *dict*): Maps from layer names or regexp expressions to alphas different from the default alpha specified by *lora_alpha*.

[1] `https://github.com/huggingface/peft/blob/main/src/peft/tuners/lora/config.py`