**Types of Fine Tunning**

1. **PyTorch interface**

Before using the transformers functionality for fine tunning, I looked at the option of fine-tune the models without any wrappers and choosing each step. As this was much slower that the other resources, I didn’t use during the research.  
  
**Advantages:** Some models do require some changes as the fine tunning happens, this is the case of Falcon, this model while training doesn’t use PAD token, in the common cases samples, this is fixed by making the PAD = EOS token. For our case this is enough but not for other specific cases.

This method also allows to change the inputs are present to the model, unfortunately this required much study and investigation for me to fully understand each step and ideal modifications required.

**Disadvantages:** Really slow and more knowledge on the models internal architecture. This also doesn’t allow to use QLoRa method.

1. **QLoRa method with the Transformers library**

As described in the previous note, QLoRa methodology allows for a quicker training of the models by only adjusting some of the parameters and reducing the number of bits. At the same time the transformers library makes the coding easier while fine tunning the models.

The Fine Tuning Library interacts seamlessly with the Hugging Face hosted models, allowing to use the proper Tokenizer and Data Collator (Adjusting padding and truncation), some models require specific modifications like the Falcon or Llama, but not as labour intensive.

To use QLoRa method requires GPU for fine tune usage. This methodology can be switched off on the code, but not having GPU dramatically increases the time for training, as an example: for the model “bert-large-uncased” which it was downloaded with only half of the parameters (180million) to do an epoch of 3,700 samples on GPU took only 4 minutes while on CPU it takes 6 hours!

**Advantages:** it reduces the amount of time required to train the model, using the library makes it easier to understand code and adapt for different cases, it suits HF models. Once fine tuned you the model only saves the adapters (parameters) used for the training, which generally are less than 10mb

**Disadvantages:** It requires GPU, you can only train the models for the task they have been allocated, it doesn’t work in some models during inference. To load the models and use for inference I have to follow a different methodology than the one show on the sample usages.

**Notes:** The samples codes mention that once you reload the model, it adjust the weights and can be use for inference. This did not work for the samples I created, to solve this I had to save the adapters in the hub, and also the full weights, with this the I reload the model with the same configuration and load the weights. This worked perfectly.

**Tasks:** For this research, I only tested the task of classification for fine tunning, I did this on different datasets and different models. I also left a sample code usage to fine tune a text2text model, in this code the fine tunning is for summarization but changing the prompt will allow you to change the task, a specific dataset will be required too.

1. **Auto Train (Inference Endpoint)**

As with the model deployment, Hugging Face also offers hosted cloud storage space to perform model fine tunning. This template is call Auto Train and hosted by Docker.

For this research I didn’t use this method, I only look at the usage and the types of memory available. Similar to the inference endpoint, 15G are free without any GPU.

**Advantages:** It can be done with your computer turned off, it doesn't require code

**Disadvantages:** Less freedom on the type of tasks and modifications, it requires to pay for GPU or more memory, you need to understand the methodology to load the data and train.

**Notes:**

<https://huggingface.co/autotrain>

**Fine Tuning Evaluation**

After fine tuning the classification models, I evaluated them on three separate tasks (different datasets) 1 general, 2 medical:

* IMDB Dataset
* PubMed
* Medical\_Documents

All the datasets are available on the folder Datasets, the results are available on Test\_Results folder