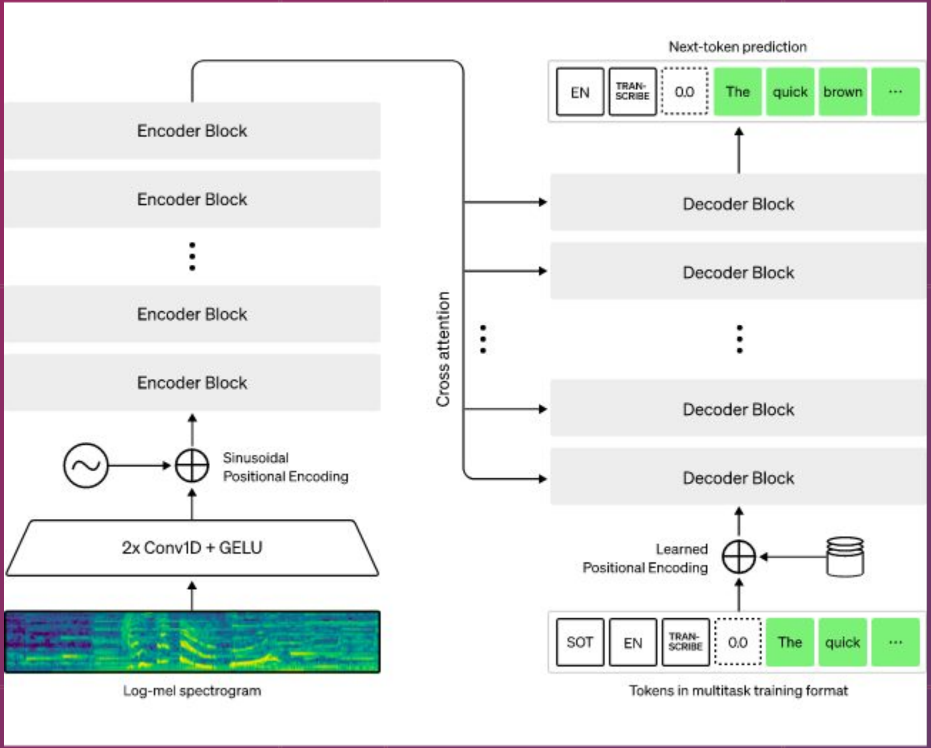
Lab 2
ID2223 / HT2025



Parameter Efficient Fine-Tuning (PEFT) of a Large Language Model on a GPU

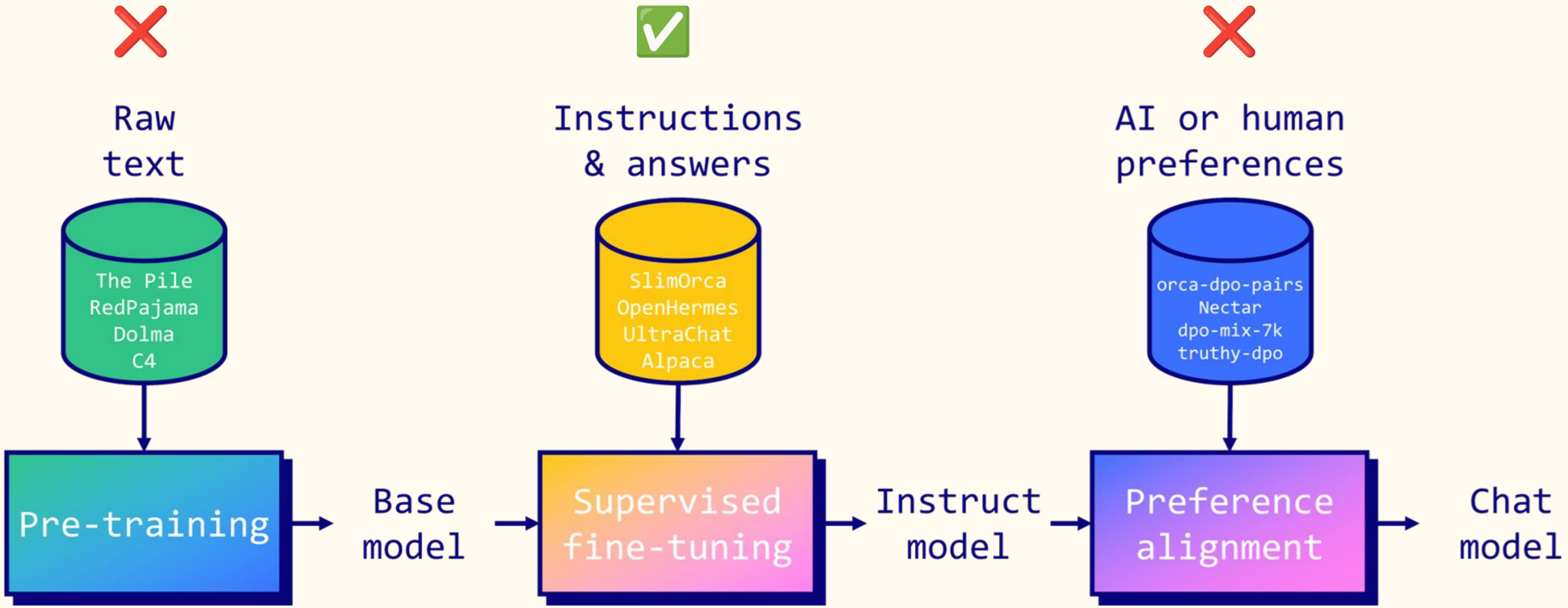
Course Material: Jim Dowling

Source Code for Lab 2

- Store your Source Code on Github

- Use Conda or any virtual environment to manage your python dependencies on your laptop. See more info on how to manage your Python environment here.

Fine-Tune a Large Language Model



Open-source instruction datasets

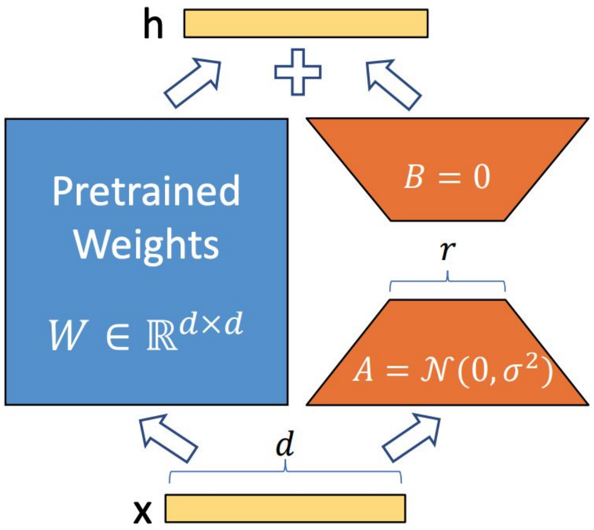
System
You are a helpful assistant, who always provide explanation. Think like you are answering to a five year old.

User
Remove the spaces from the following sentence: It prevents users to suspect that there are some hidden products installed on theirs device.

Output
Itpreventsuserstosuspectthattherearesomehiddenproductsinstalledontheirsdevice.

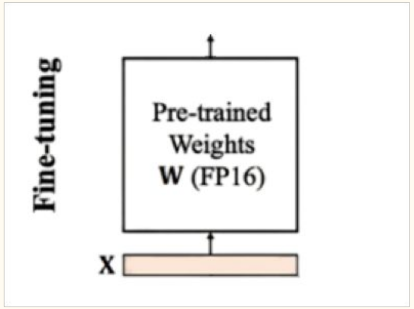
Parameter Efficient Fine Tuning with LoRA

- LoRA (Low-Rank Adaptation) is a technique for PEFT of LLMs by injecting trainable low-rank matrices into the model's layers, significantly reducing the number of parameters to update and the computational cost.
- Fine-tuning can suffer from two problems: model collapse and catastrophic forgetting. Model collapse is where the model output converges to a limited set of outputs. Catastrophic forgetting is where a model loses its ability to remember things it had previously learnt. These problems are less common for PEFT (parameter efficient fine tuning) compared to full fine-tuning.



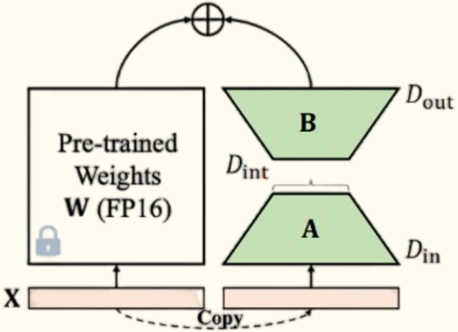
Fine-Tuning LLMs with limited GPU Memory (T4 GPU on Colab)

Full Fine-Tuning
16-bit precision



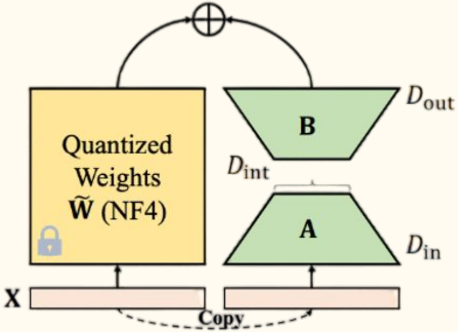
✅ Best performance
❌ Very high VRAM usage

LoRA
16-bit precision



✅ Quick training
❌ Still costly

QLoRA
4-bit precision



✅ Low VRAM usage
❌ Degrades performance

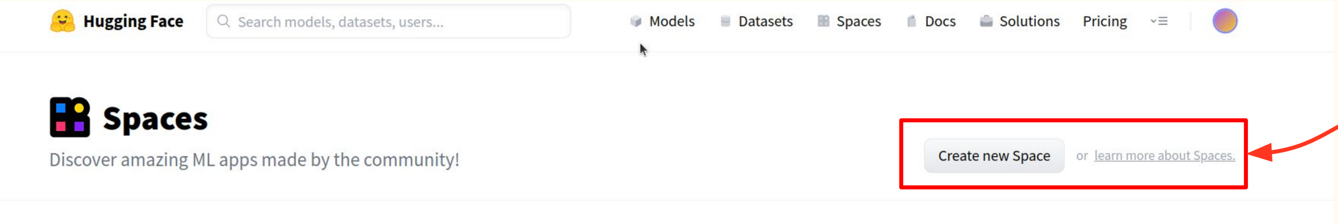
Task 1: Fine-tune a model for language transcription, add a UI

- Fine-Tune a pre-trained large language (transformer) model and build a serverless UI for using that model

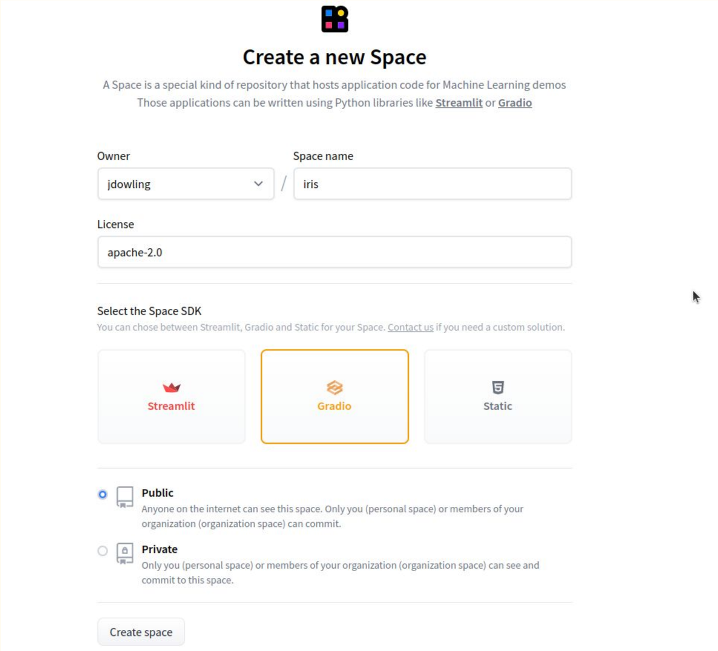
- **First Steps**
  a. Create a free account on huggingface.com
  b. Create a free account on google.com for Colab

- **Tasks**
  a. Fine-tune an existing pre-trained large language model on the FineTome Instruction Dataset
  b. Build and run an inference pipeline with a Gradio UI on Hugging Face Spaces for your model.

Register and Create a Hugging Face Space

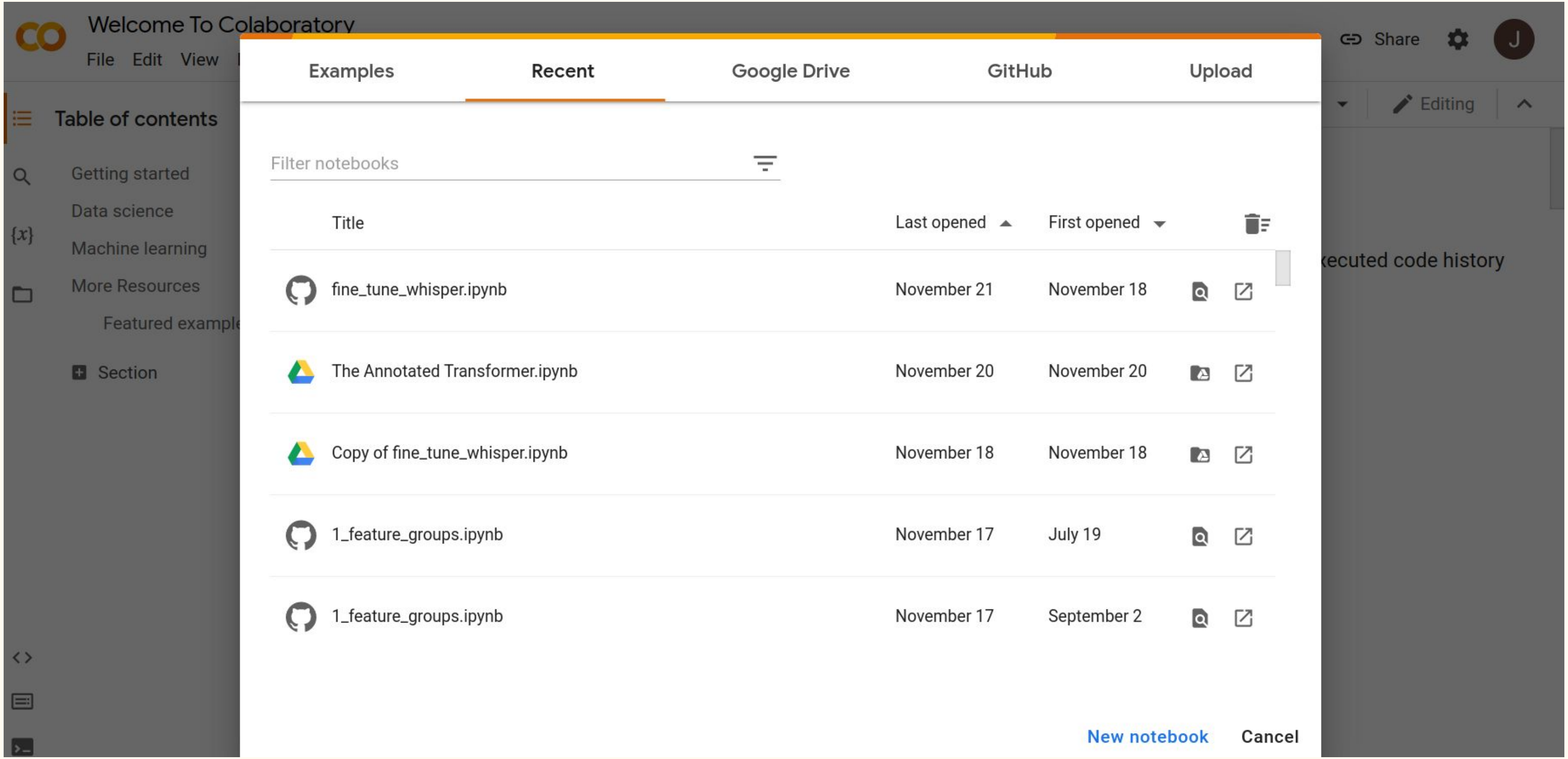


1. Create an account on Hugging Face
2. Create a “Space”



3. Create a Gradio App with the name Iris inside your account

Register and Create an account on Google for Colab



Fine Tuning: use Llama-3 1B or 3B hosted at Hugging Face

- A sample Colab Notebook is available here.
- You should fine-tune the LLM on the Fine Tome Dataset hosted at Hugging Face.
- We recommend that you train your model with a GPU. Colab provides free GPUs for 1-4 hours (then it shuts down) - so make sure to save your model weights before it shuts down. If you have your own GPU, you can use that.
- You will need to checkpoint the weights periodically, so that you can restart your training from where you left off. Even if you have your own GPU you still have to demonstrate this task.
- You have to save your fine tuned LLM somewhere - e.g., on HuggingFace, Hopsworks or Google Drive, so that you can download it for use in your UI

Communicate the value of your model with a UI (Gradio or Streamlit)

- You will be training your model on a GPU, but then loading the model for inference on a CPU. This means you need to convert the saved model format, e.g., by exporting the saved model to GGUF.

- Communicate the value of your model to stakeholders with an app/service that uses the fine tuned LLM to make value-added decisions

Example UIs:

- Chatbot to talk to your new finely tuned LLM
  - Smaller models will be faster than large models, as StreamlitCloud and HuggingFace Spaces only offer free CPUs for inference

- If you want to get the highest grade (A), come up with your own creative idea for how to allow people to use your fine tuned LLM

Task 2: Improve pipeline scalability and model performance

1. Describe in your README.md program ways in which you can improve model performance are using
   (a) **model-centric approach** - e.g., tune hyperparameters, change the fine-tuning model architecture, etc
   (b) **data-centric approach** - identify new data sources that enable you to train a better model that one provided in the blog post

   If you can show results of improvement, then you get the top grade.

2. Try out fine-tuning a couple of different open-source foundation LLMs to get one that works best with your UI for inference (inference will be on CPUs, so big models will be slow).

3. You are free to use other fine-tuning frameworks, such as Axolotl of HF FineTuning - you do not have to use the provided unsloth notebook.

Deliverables

- Deliver your source code as a Github Repository.
- Deliver your description for task 2 as a README.md file in the root of your Github repository
- Deliver a Hugging Face Spaces or Streamlit Cloud public URL for the UI for your LLM user interface.

Deadline midnight 3rd December 2025.

Useful links

- Maxime LeBon fine-tuning guide
- Unsloth for memory efficient fine-tuning (particularly on Colab)
- Axolotl for low-code fine-tuning
- Saving a checkpoint in Torch and saving a checkpoint to Google Drive.
- Fine-tune a LLM on a single GPU, HuggingFace Guide