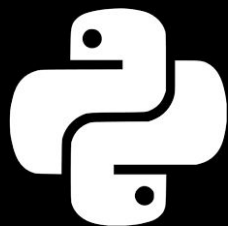


LLMs for me



**Fine-tuning LLMs,
PEFT & Quantization**

llmsfor.me



Myles Harrison,
AI Consultant & Trainer



January 13th, 2025

NLP from scratch 

Agenda

01

Front Matter

02

Fine-tuning LLMs

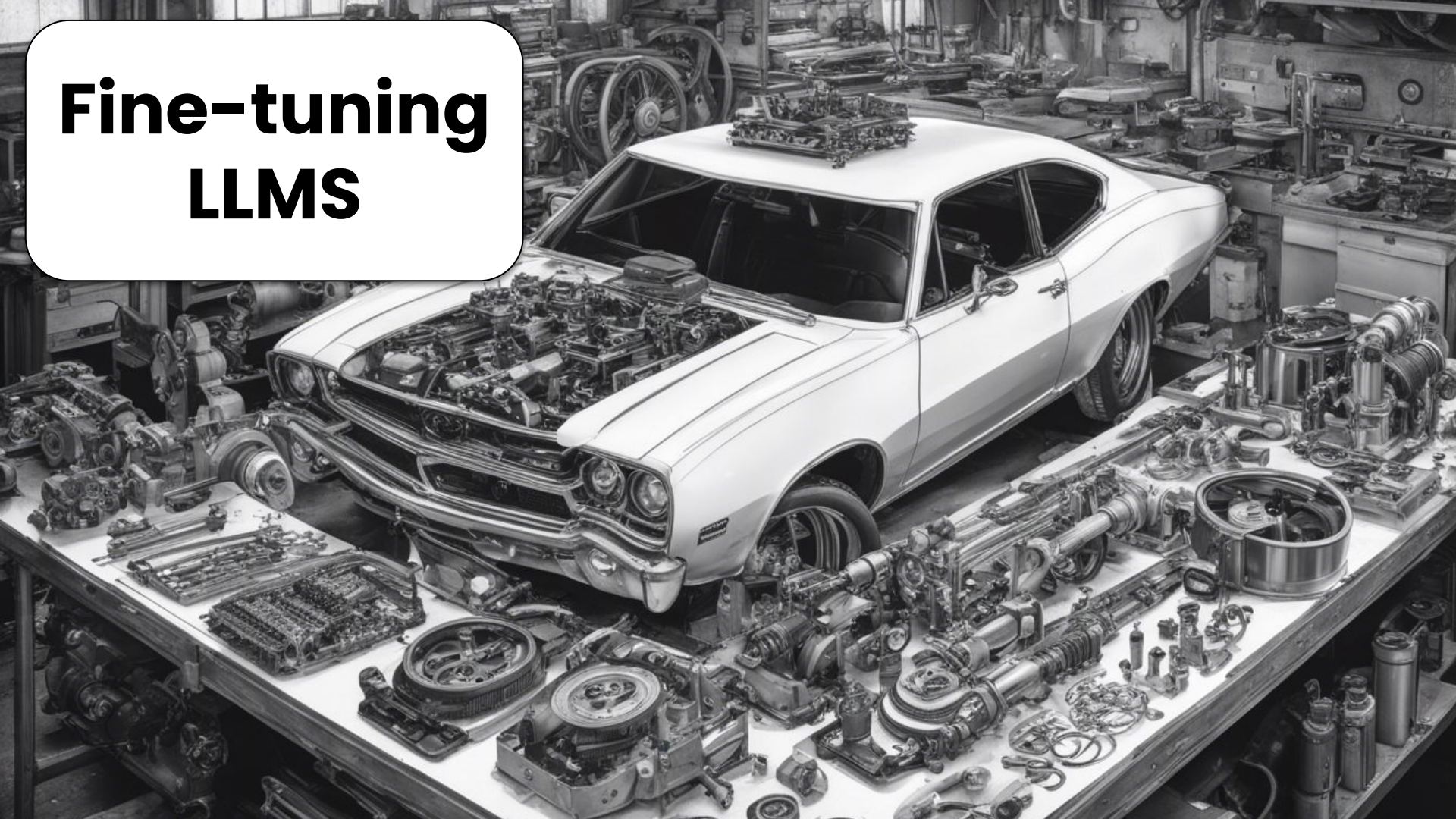
03

Quantization & PEFT

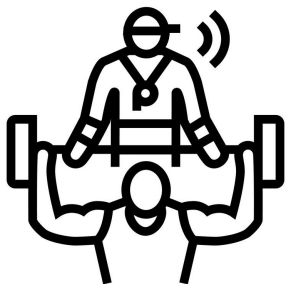
04

Conclusion

Fine-tuning LLMs



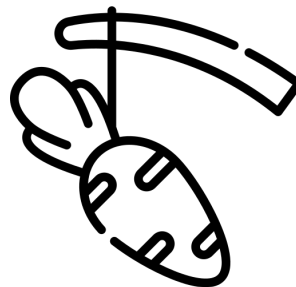
Phases of LLM Training



Pre-training



Fine-tuning



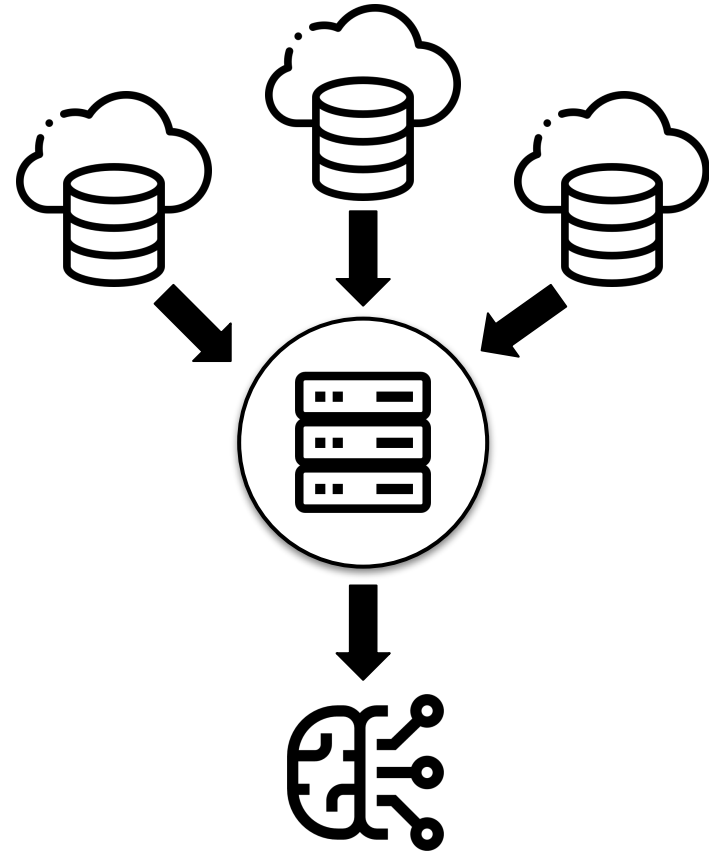
**Reinforcement
Learning**

Pre-training

The parlance of modern language models has changed slightly from that of traditional machine learning.

For modern LLMs, the initial phase of training of the model, now referred to as *pre-training*, consists of showing the model massive quantities of unlabelled text, and optimizing its parameters against a specific objective, such as next token prediction. This is the most computationally intensive and expensive part of training modern language models, and results in a pre-trained “base model”.

Because of the scale, cost, and complexity required, pre-training LLMs is typically only realistic for large organizations with considerable financial backing, infrastructure, and technical expertise.

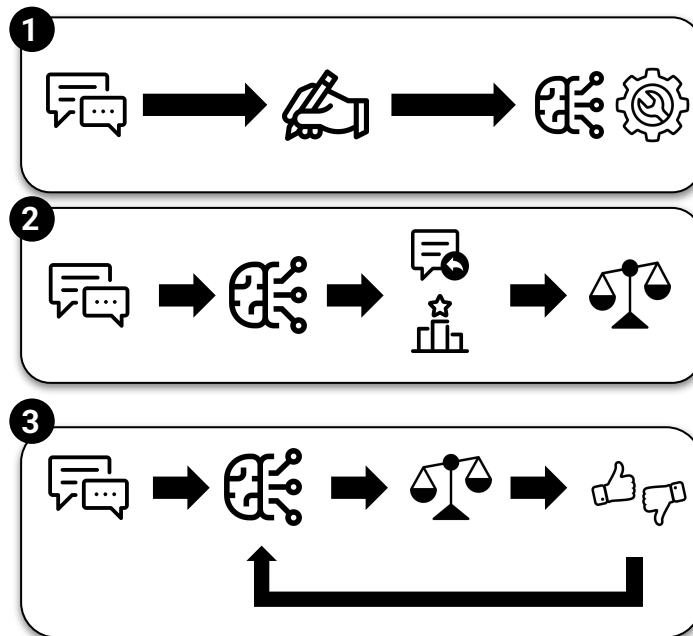


Reinforcement Learning from Human Feedback

A key innovation leading to significant improvement in quality of responses of generative text models was that of **Reinforcement Learning from Human Feedback (RLHF)**.

Though human feedback being incorporated into RL was not a new idea, OpenAI was the first to apply this at scale in training InstructGPT – the predecessor to ChatGPT – using Proximal Policy Optimization (PPO).

A pretrained model is tuned on a collection of human-generated responses to prompts (1), and a reward model is also trained, incorporating human feedback: a ranking of a selection of responses generated by the model (2). These are then incorporated together into iteratively training a final policy model through reinforcement learning (3).

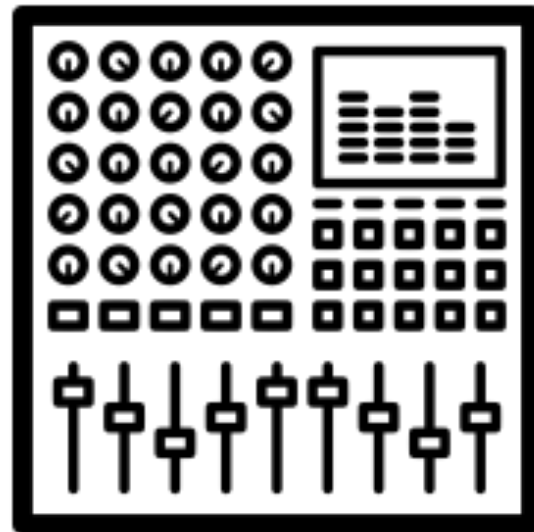


Fine-tuning

On the other hand, *fine-tuning* is less computationally intensive and requires much less data.

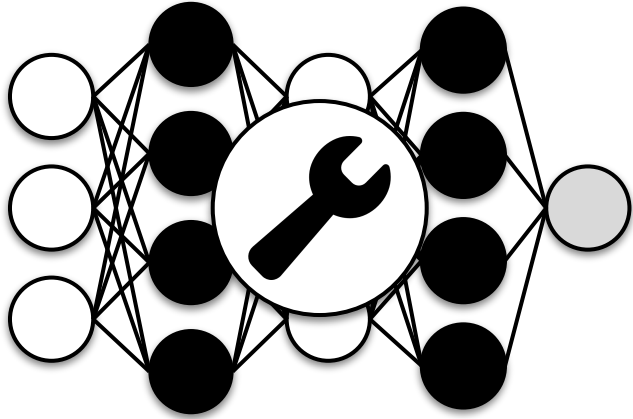
In this part of the training process, a pre-trained model is shown a smaller dataset and further optimized against another target objective. This objective can be the same as that of the original base model, or a different objective if a different type of “head” is added to the base model.

In earlier machine learning parlance prior to that of LLMs, this type of process is referred to as “transfer learning”, and indeed fine-tuning is just a specific type of transfer learning. Fine-tuning will be the focus of the remainder of this workshop and we will see examples applied in code.



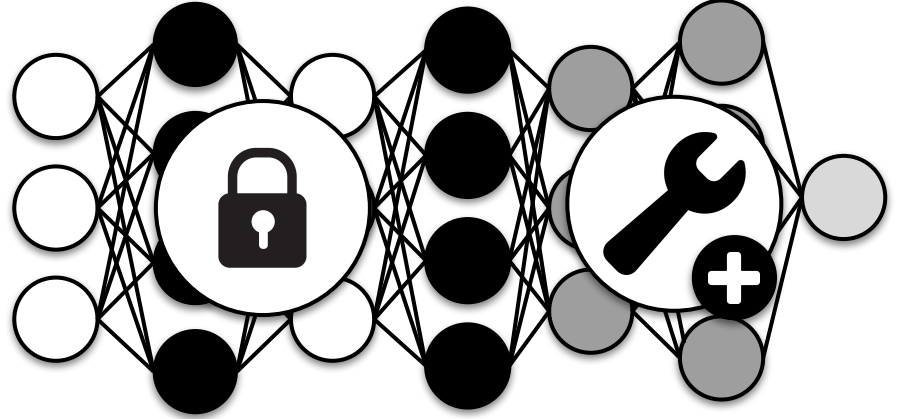
Fine-tuning: Approaches

Full Fine-tuning



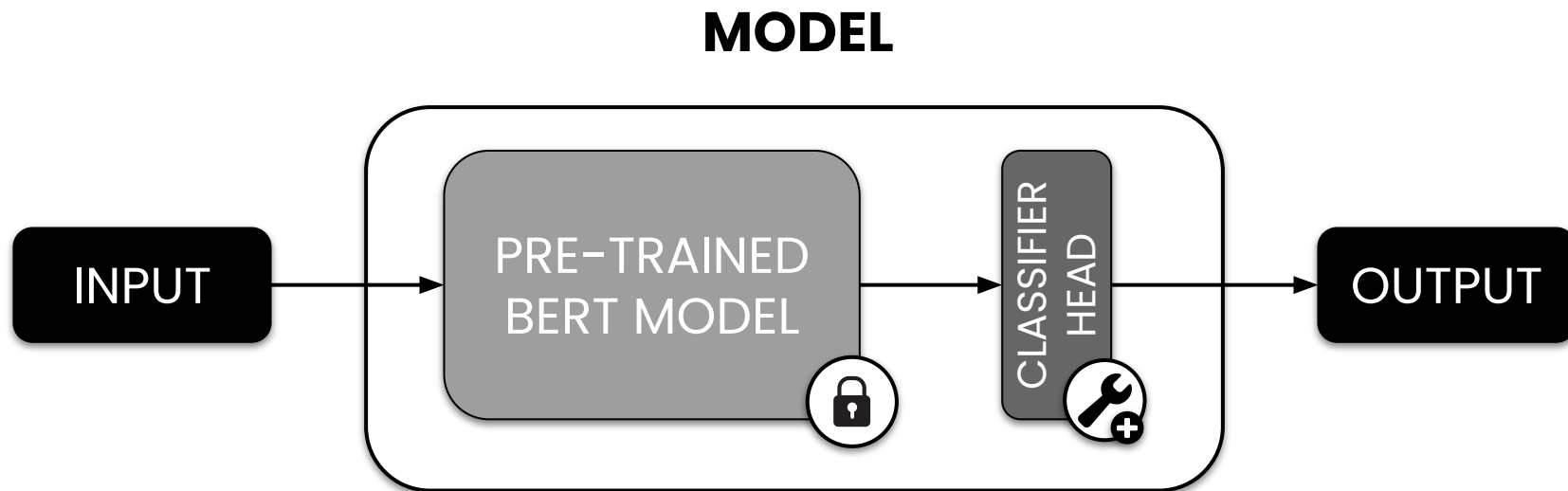
Update all weights in the model.
Computationally expensive and slow
with better model performance.

Partial Fine-tuning

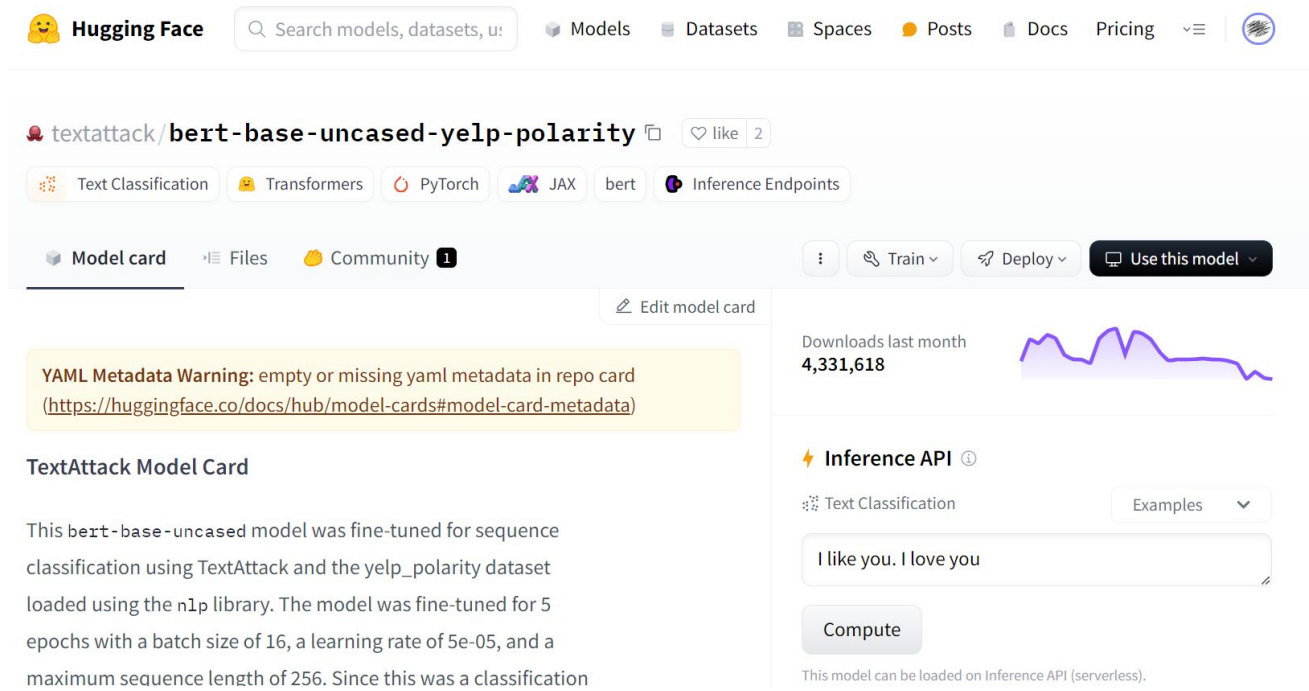


Freeze most weights in the model. Update final
or newly added layers. Less computationally
demanding with model performance tradeoff.

Example: Fine-tuning BERT for classification



An example – BERT fine-tuned for sentiment



The screenshot shows the Hugging Face interface for the model `textattack/bert-base-uncased-yelp-polarity`. The model is categorized under Text Classification, Transformers, PyTorch, JAX, bert, and Inference Endpoints. It has 2 likes. The model card includes a warning about missing YAML metadata and a description of the model's fine-tuning process. The Inference API section shows a text input field with the text "I like you. I love you" and a "Compute" button. A line graph shows 4,331,618 downloads in the last month.

Hugging Face Search models, datasets, u: Models Datasets Spaces Posts Docs Pricing

`textattack/bert-base-uncased-yelp-polarity` like 2

Text Classification Transformers PyTorch JAX bert Inference Endpoints

Model card Files Community 1 Train Deploy Use this model

Edit model card

YAML Metadata Warning: empty or missing yaml metadata in repo card (<https://huggingface.co/docs/hub/model-cards#model-card-metadata>)

TextAttack Model Card

This `bert-base-uncased` model was fine-tuned for sequence classification using TextAttack and the `yelp_polarity` dataset loaded using the `nlp` library. The model was fine-tuned for 5 epochs with a batch size of 16, a learning rate of $5e-05$, and a maximum sequence length of 256. Since this was a classification

Inference API

Text Classification Examples

I like you. I love you

Compute

This model can be loaded on Inference API (serverless).



huggingface.co/textattack/bert-base-uncased-yelp-polarity

NLP from scratch

Fine-tuning LLMs: Hands-on

Let's apply fine-tuning to get GPT-2 to speak like our favourite Jedi Master



Model Quantization

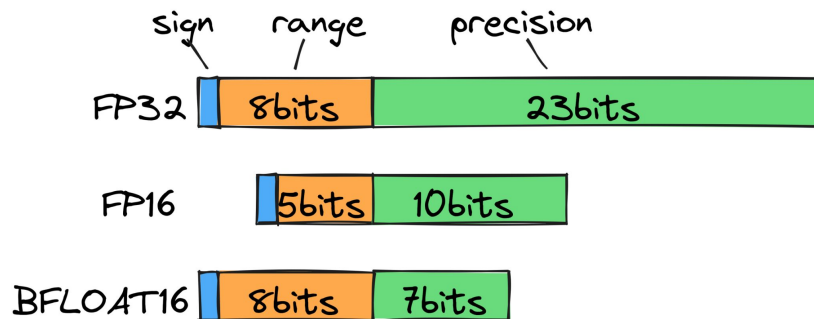
Training large language models is a very computationally demanding task - for both storage and compute - as the size of a model grows.

One way of addressing this issue is quantization - working with numbers of lower precision for model parameters and calculations, for example, storing values as integers instead of floating points (decimal numbers).

There are different quantization approaches as information will always be lost. One method is affine quantization which uses a scale factor and zero point to map floating point values to integer ones as a linear combination of the original values, together with rounding and clipping.

$$x_q = \text{round}(x/S + Z)$$

where x is the real value, x_q is the quantized value
 S is a scale factor, and Z is the zero point



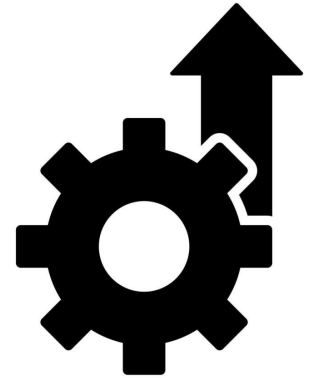
Parameter-Efficient Fine-Tuning (PEFT)

Parameter-Efficient Fine-tuning (PEFT) is a family of approaches which fine-tune a small number of extra model parameters, either before or after the LLM (additive) or by inserting smaller subsets of parameters within certain parts of the model architecture (reparameterization).

Partial fine-tuning can be considered a type of PEFT (selective), however, usually when one is speaking of PEFT it is in reference to one of a number of approaches such as adapters, LoRA, QLoRA, P-Tuning, Prompt Tuning, or Prefix Tuning that function as mentioned above.

PEFT is typically combined with model quantization, allowing the fine-tuning of large language models efficiently and without prohibitive infrastructure needs.

While PEFT is a topic in and of itself, we will focus in this workshop on the commonly used LoRA approach.



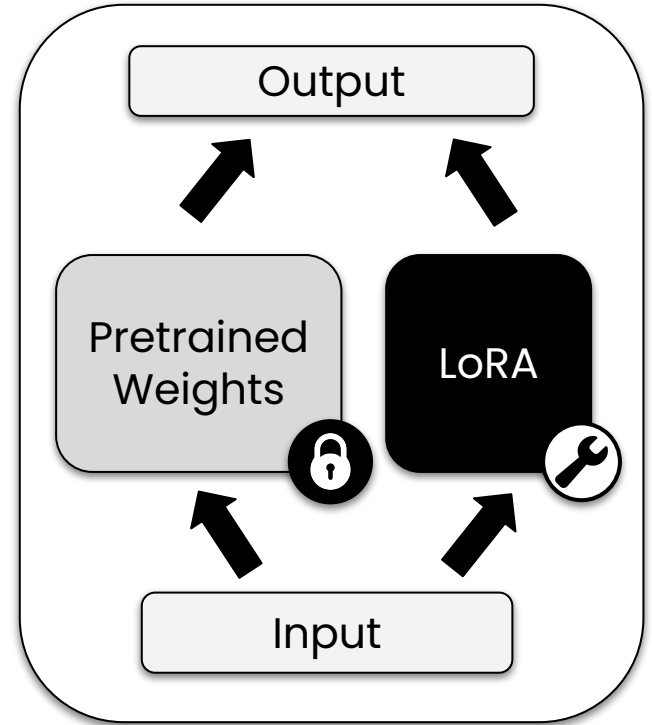
Low-Rank Adaptation of LLMs (LoRA)

Introduced by researchers from Microsoft in June of 2021¹, LoRA is a type of PEFT that reduces the computational cost of fine-tuning large language models by reparameterizing the model training.

Instead of updating all the model weights in particular parts of the transformer architecture, only pairs of rank decomposition weight matrices in the low rank adapter are updated, which are typically much, much fewer than the total weights in the model.

The approach trains a separate sets of weights which transform the input parameters into a low-rank dimension, and a second matrix which transforms the low-rank data to the output dimensions of the original model.

1. [LoRA: Low-Rank Adaptation of Large Language Models](#)

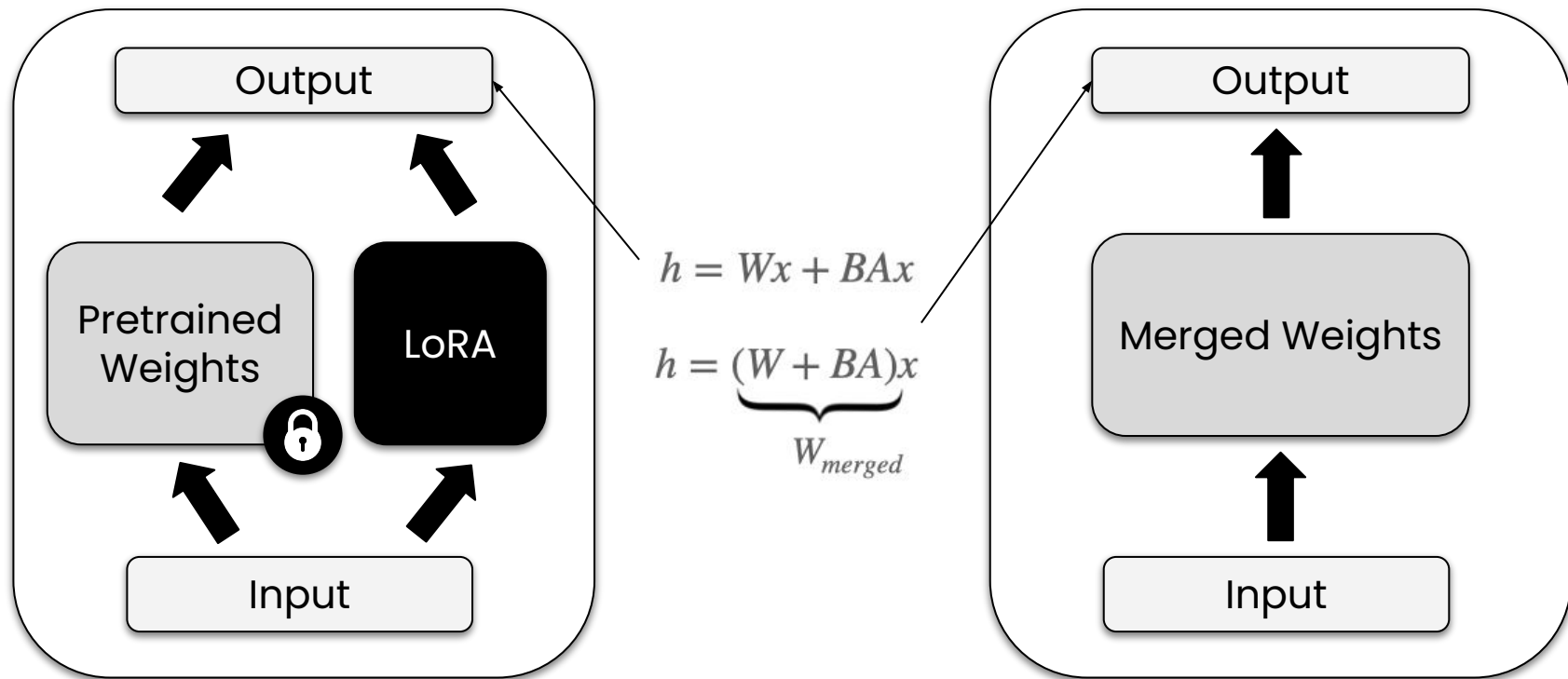


Parameter Efficient Fine-tuning: Hands-on

Let's revisit fine-tuning LLMs to speak like a Jedi, only now with the full GPT-2!



Merging the LoRA adapter

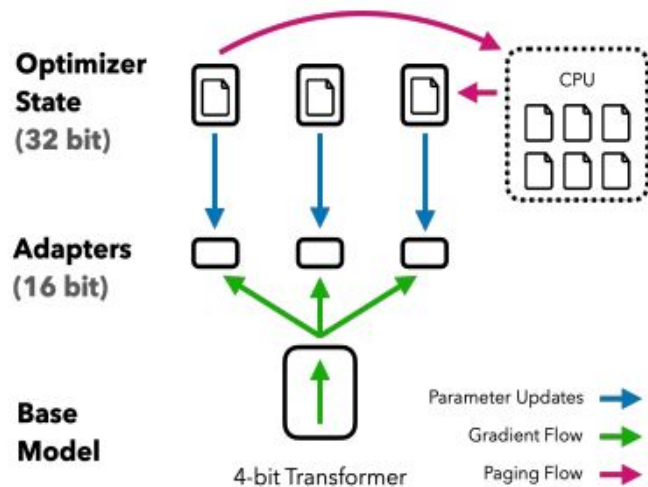


Fine-tuning with LoRA *and* Quantization: QLoRA

Building on the work of the research of the team at Microsoft, researchers from University of Washington developed [QLoRA: Efficient Finetuning of Quantized LLMs](#) in May of 2023.

[QLoRA](#) makes parameter efficient fine-tuning even more so by using 4-bit quantization for the model to be tuned, introducing a new data type called 4-bit NormalFloat (NF4), as well as other optimizations.

A notable output of the QLoRA research was that of the [Guanco](#) model family which was fine-tuned on LLaMA 2. You can see an example of using QLoRA in Hugging Face in this [example notebook](#) and more details in the official [blog post from Hugging Face](#).



Be mindful of your data 🤔

Fine-tuning GPT3.5-turbo based on 140k slack messages

USER

write a 500 word blog post on prompt engineering

ASSISTANT

sure

I shall work on that in the morning

<https://rosslazer.com/posts/fine-tuning/>



Onward..

Where do we go from here?

Training your own ChatGPT

“Chatbot”-style generative text models, which take a question or utterance from the user as input and return with their own fully response, must be trained and worked with differently.

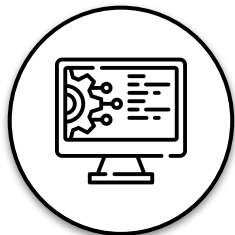
At its simplest, this involves changing the format of the data the model is trained on as being pairs of questions and answers. For example, that LLaMA model has input an input format for specifying system, user, and assistant (chatbot) text with special characters denoting each part of the text.

Because of this, Hugging Face has added the chat template functionality to make working with models like these easier.

On top of this, these models also usually have RLHF applied to condition the format of outputs (e.g. to be complete statements) and may also have instruction tuning applied.

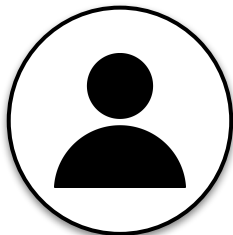
```
<s>
[INST]
<<SYS>>
You are a helpful, respectful
and honest assistant.
<</SYS>>
There's a llama in my
garden 🤪 What should I
do?
[/INST]
```

Message Roles



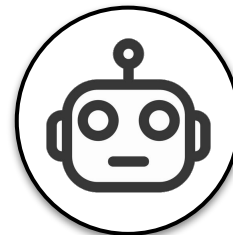
SYSTEM

Sets the behavior of the assistant - how it should behave at the conversation level (optional)



USER

Provide requests or input to which the assistant will respond (i.e. the prompts)



ASSISTANT

Responses from the model. Can be used to include conversation history when it is important (optional)

Training a chat LLM - data format

```
conversation = [  
    {"role": "user", "content": "Hello, how  
are you?"},  
    {"role": "assistant", "content": "I'm  
doing great. How can I help you today?"},  
]  
  
Tokenizer = AutoTokenizer.from_pretrained(  
    "microsoft/Phi-3-mini-4k-instruct")  
  
tokenizer.apply_chat_template(conversation,  
    tokenize=False))
```

<|user|>Hello, how are
you?<|end|>

<|assistant|>
I'm doing great. How
can I help you
today?<|end|>

<|endoftext|>

RLHF Training



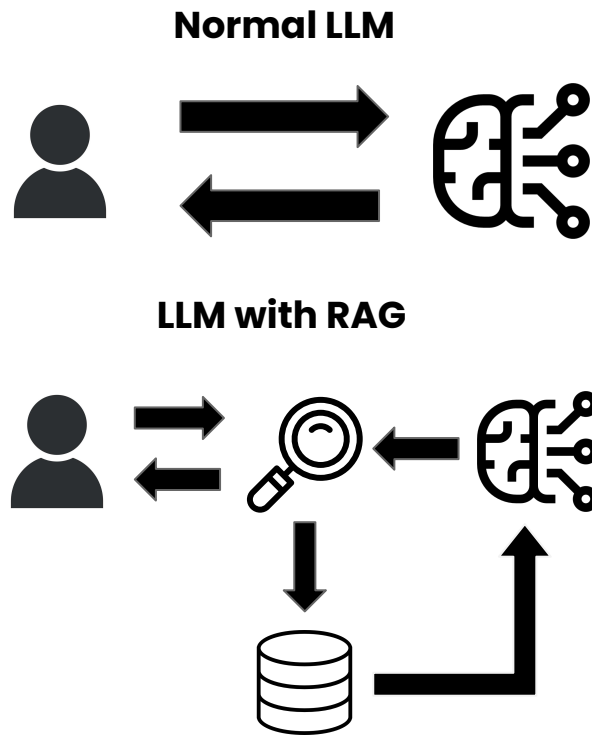
<https://huggingface.co/docs/trl>


Retrieval Augmented Generation

One of the known shortcomings of LLMs is the problem of *hallucinations* - a model will provide responses which sound plausible but are "made up".

Additionally, a desirable trait is the ability to have an LLM answer questions about a specific dataset or corpus of documents which was not part of its training data nor will fit into a prompt for few-shot learning

Retrieval Augmented Generation (RAG) addresses both these issues by combining information retrieval (i.e. search) against a set of documents with a generative model. This allows the creation of responses based on the foundation of a specific dataset while eliminating the need for retraining or fine tuning the model itself.



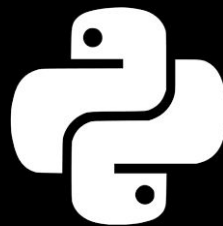
NLP from scratch 

End of Part 2

[LLMsfor.me](https://llmsfor.me)

PWYC Microcourse in LLMs and Generative AI
January 2025

Part 2 - Fine-tuning, Quantization, and PEFT
Monday, January 13th, 2025



llmsfor.me

NLP from scratch