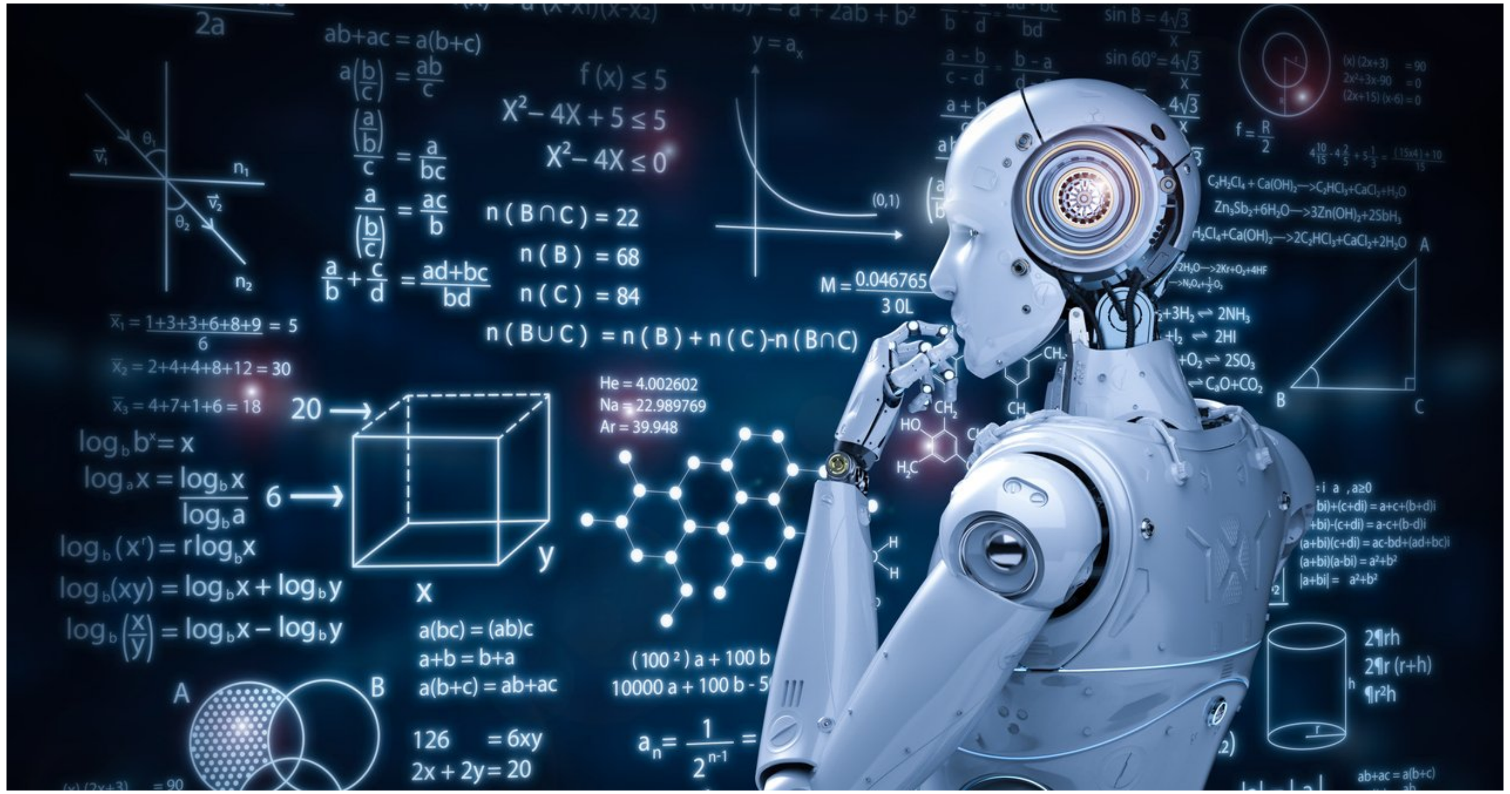
Preprocessing data for fine-tuning

WORKING WITH LLAMA 3

Imtihan Ahmed
Machine Learning Engineer

When to use fine-tuning

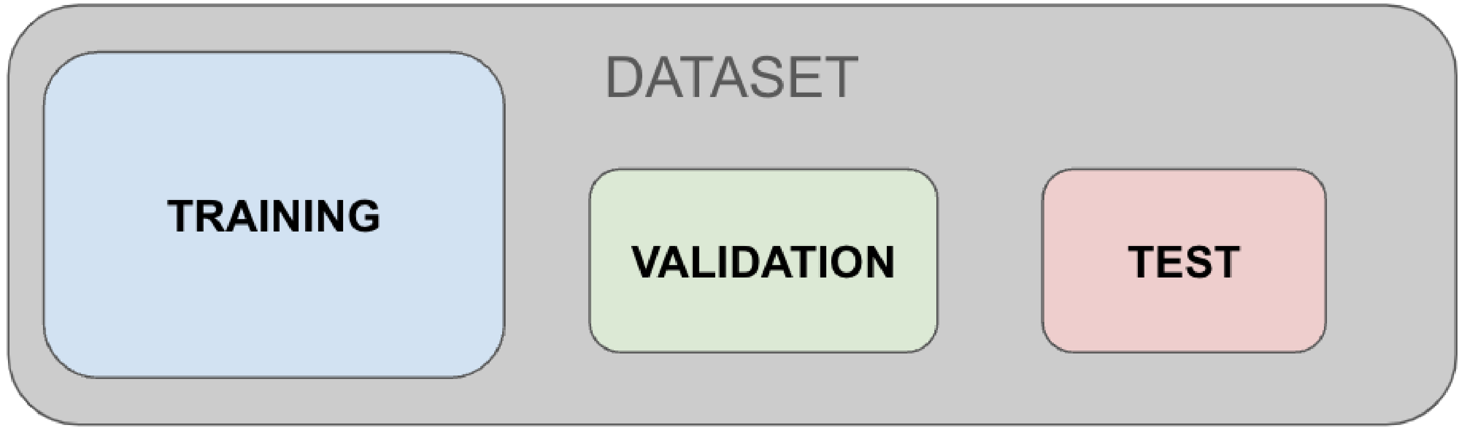
- Uses datasets
- Training on task & domain
- Updates model parameters



- Improve accuracy
- Reduce bias
- Improve knowledge base

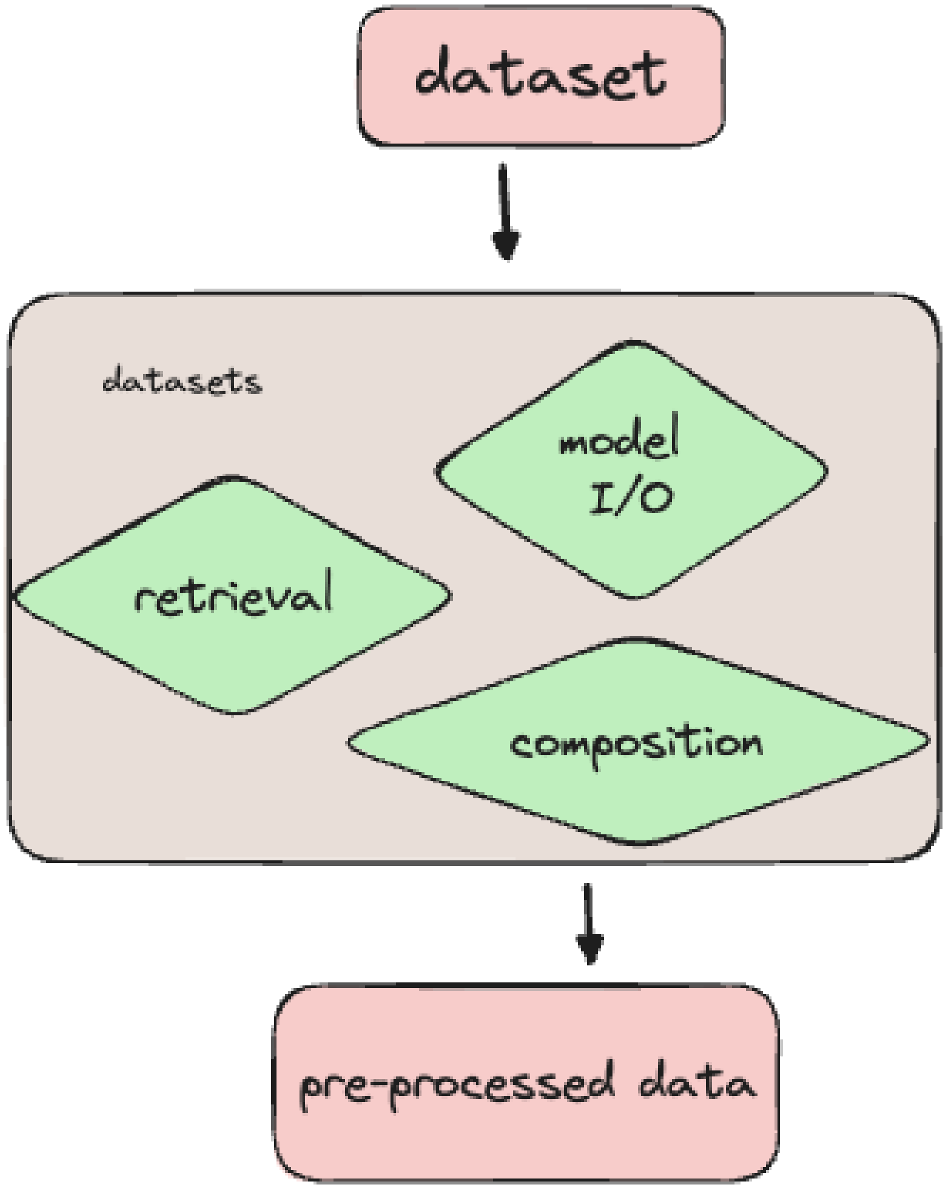
How to split datasets

- **Training Set:** Used to train the model. This subset contains the majority of the data.
- **Validation Set:** Used to tune model hyper parameters and select the best model.
- **Test Set:** Used to evaluate the final model's performance.



Preparing data using the datasets library

- preprocessing
- split
- load
- manage memory



Loading a customer service dataset

```
from datasets import load_dataset
ds = load_dataset(
    'bitext/Bitext-customer-support-llm-chatbot-training-dataset',
    split="train"
)
print(ds.column_names)
```

```
['flags', 'instruction', 'category', 'intent', 'response']
```

Filtering a dataset

```
from datasets import load_dataset, Dataset

ds = load_dataset(
    'bitext/Bitext-customer-support-llm-chatbot-training-dataset',
    split="train"
print(ds.shape)
```

```
(26872, 5)
```

```
first_thousand_points = ds[:1000]
ds = Dataset.from_dict(first_thousand_points)
```

Peeking into the data

```python
import pprint
pprint.pprint(ds[0])
```

```
{'category': 'ORDER',
 'flags': 'B',
 'instruction': 'question about cancelling order {{Order Number}}',
 'intent': 'cancel_order',
 'response': "I've understood you have a question regarding canceling order "
             "{{Order Number}}, and I'm here to provide you with the "
             'information you need. Please go ahead and ask your question, and '
             "I'll do my best to assist you."}
```

Preprocessing dataset

```
def merge_example(row):
    row['conversation'] = f"Query: {row['instruction']}\nResponse: {row['response']}"
    return row
ds = ds.map(merge_example)
print(ds[0]['conversation'])
```

```
Query: question about cancelling order {{Order Number}}
Response: I've understood you have a question regarding canceling order {{Order Number}},
and I'm here to provide you with the information you need. Please go ahead and ask your
question, and I'll do my best to assist you.
```

Let's practice!

WORKING WITH LLAMA 3

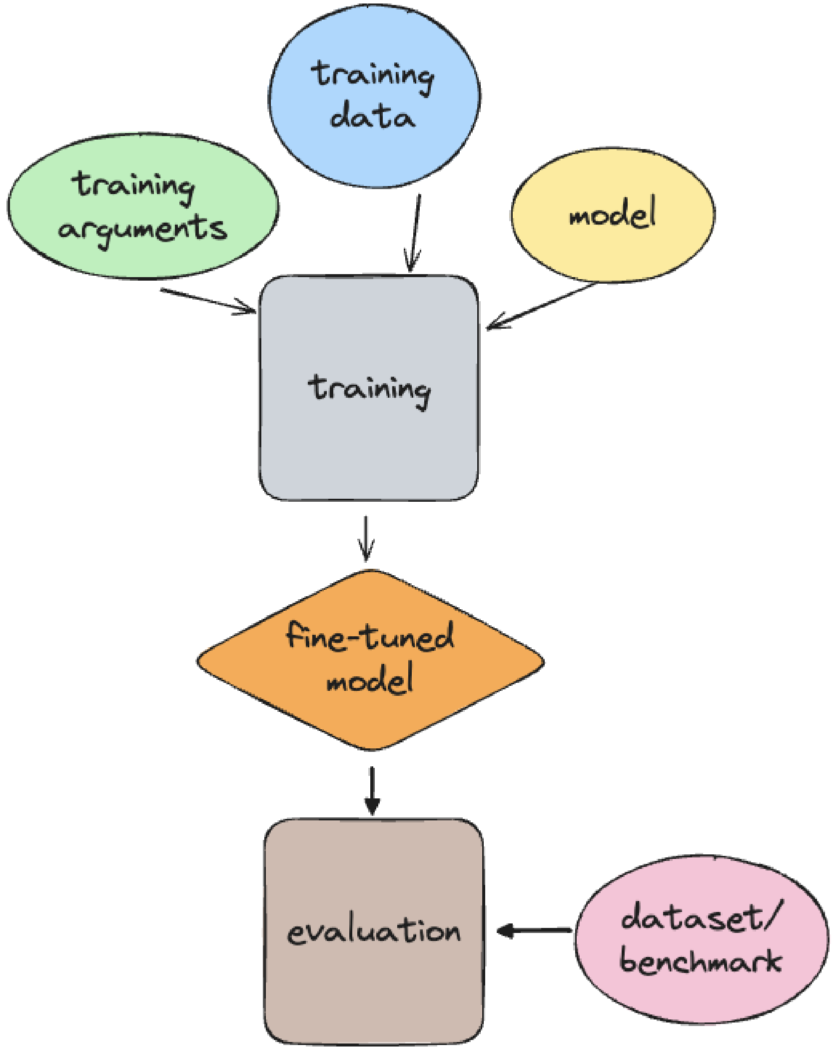
Model fine-tuning with Hugging Face

WORKING WITH LLAMA 3

Imtihan Ahmed
Machine Learning Engineer

What do we need to conduct fine-tuning?

1. Language model + tokenizer (**TinyLLama-v0**)
2. Training dataset (**the Bitext customer service dataset**)
3. Training arguments
4. Conduct fine-tuning (**SFTTrainer** from **TRL**)
5. Evaluation benchmark or dataset



How to load models and tokenizers with Auto classes

```
model_name="Maykeye/TinyLLama-v0"

model = AutoModelForCausalLM.from_pretrained(model_name)
tokenizer = AutoTokenizer.from_pretrained(model_name)
tokenizer.pad_token = tokenizer.eos_token
```

[1] https://huggingface.co/docs/transformers/main/en/model_doc/auto

Defining training parameters with TrainingArguments

```
training_arguments = TrainingArguments(
    per_device_train_batch_size=1,
    learning_rate=2e-3,
    max_grad_norm=0.3,
    max_steps=200,
    ...
    gradient_accumulation_steps=2,
    save_steps=10,
)
```

[1] https://huggingface.co/docs/transformers/v4.40.1/en/main_classes/trainer#transformers.TrainingArguments

How to set-up training with SFTTrainer

```
trainer = SFTTrainer(
    model=model,
    tokenizer=tokenizer,
    train_dataset=dataset,
    dataset_text_field='conversation',
    max_seq_length=250,
    args=training_arguments
)
```

Understanding fine-tuning results with SFTTrainer

```
trainer.train()
```

```
TrainOutput(global_step=200, training_loss=1.9401231002807617,
            metrics={'train_runtime': 142.5501,
                     'train_samples_per_second': 2.806,
                     'train_steps_per_second': 1.403,
                     'total_flos': 1461265827840.0,
                     'train_loss': 1.9401231002807617,
                     'epoch': 2.0})
```

How to evaluate a trained model Using ROUGE-1

- ROUGE-1: Ratio of word overlap between a reference and generated text

```
import evaluate
rouge = evaluate.load('rouge')
predictions = ["hello there", "general kenobi"]
references = ["hello there", "master yoda"]
results = rouge.compute(predictions=predictions, references=references)
print(results)
```

```
{'rouge1': 0.5, 'rouge2': 0.5, 'rougeL': 0.5, 'rougeLsum': 0.5}
```

[1] https://huggingface.co/spaces/evaluate-metric/rouge

How to use the ROUGE-1 score

1. Use the evaluation set in `evaluation_dataset`

```
def generate_predictions_and_reference(dataset):
    predictions = []
    references = []
    for row in dataset:
        inputs = tokenizer.encode(row["instruction"], return_tensors="pt")
        outputs = model.generate(inputs)
        decoded_outputs = tokenizer.decode(outputs[0, inputs.shape[1]:], skip_special_tokens = True)
        references += [row["response"]]
        predictions += [decoded_outputs]
    return references, predictions
```

How to run ROUGE-1 on an evaluation set

```
references, predictions = generate_predictions_and_reference(evaluation_dataset)
```

```
rouge = evaluate.load('rouge')
results = rouge.compute(predictions=predictions, references=references)
```

```
print(results)
```

Finetuning vs no finetuning

Fine-tuned

```
{'rouge1': 0.22425812699023645,
 'rouge2': 0.039502543246449,
 'rougeL': 0.1501513006868983,
 'rougeLsum': 0.18685597710721613}
```

No fine-tuning

```
{'rouge1': 0.1310928764315105,
 'rouge2': 0.04581654122835097,
 'rougeL': 0.08415351421221628,
 'rougeLsum': 0.1224749866097021}
```

Alternative fine-tuning libraries

- **torchtune**
  - Native PyTorch llms
  - Configuration-file based workflow
  - Training & eval recipes
  - Interoperable

```
tune download meta-llama/Meta-Llama-3-8B \
    --output-dir <local_dir> \
    --hf-token <TOKEN>
```

```
tune run full_finetune_single_device \
    --config llama3/8B_full_single_device
```

```
tune run full_finetune_single_device \
    --config llama3/8B_full_single_device \
    checkpointer.output_dir=<output_dir>
```

[1] https://pytorch.org/torchtune/stable/index.html

Let's practice!

WORKING WITH LLAMA 3

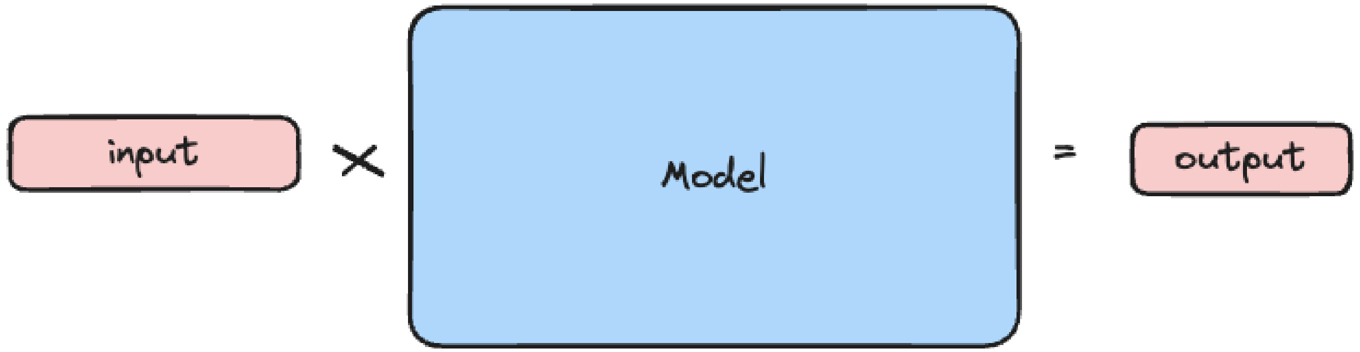
Efficient fine-tuning with LoRA

WORKING WITH LLAMA 3

Imtihan Ahmed
Machine Learning Engineer

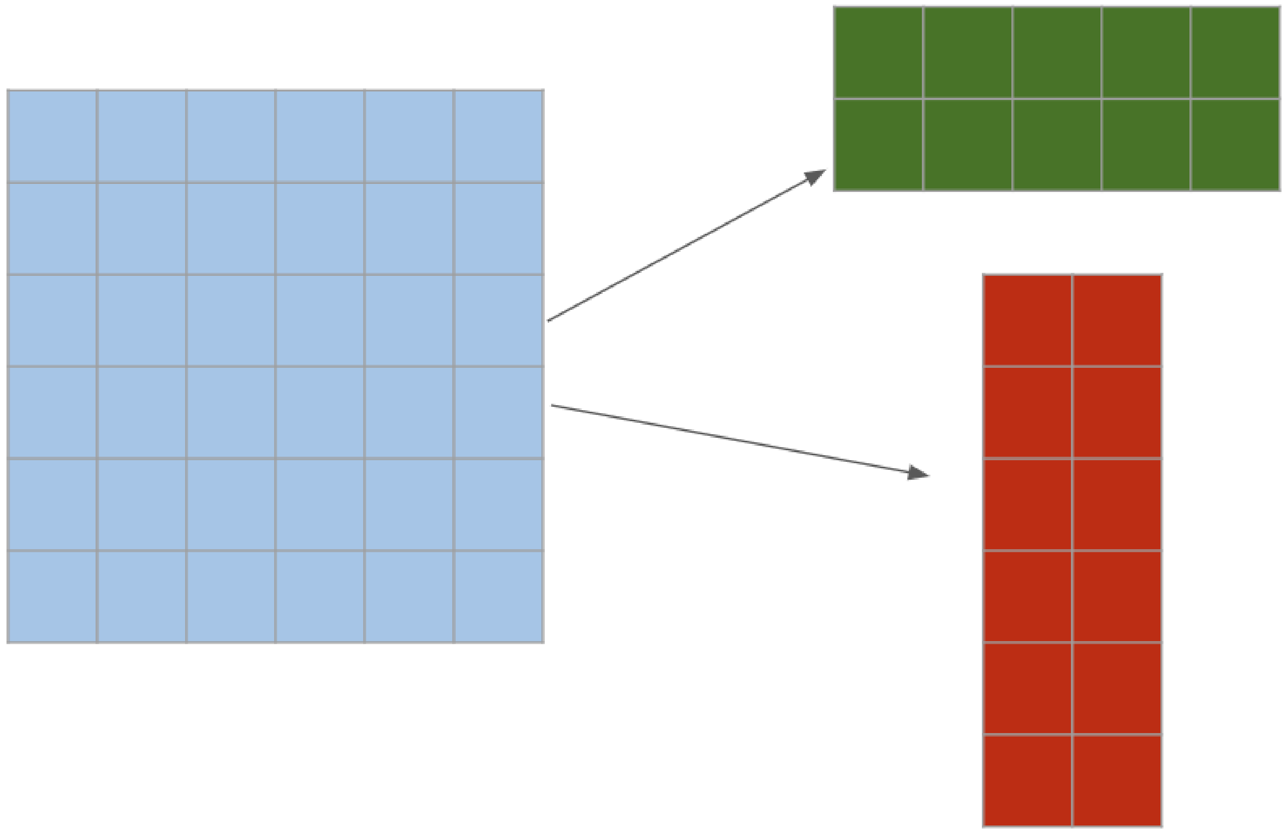
What happens when we train a model?

- Samples are input vectors
- Models are matrices
- Matrix multiplication
- Results in output vectors
- Errors are used to update model weights
- Model size determines training difficulty



What is LoRA

- Low-rank Decomposition
- Reduces training parameters
- Maintains performance
- Regularization effect



How to implement LoRA using PEFT

```
from peft import LoraConfig
lora_config = LoraConfig(
    r=12,
    lora_alpha=32,
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM",
    target_modules=['q_proj', 'v_proj']
)
```

Integrating LoRA configuration in training

```
trainer = SFTTrainer(
    model=model,
    train_dataset=ds,
    max_seq_length=250,
    dataset_text_field='conversation',
    tokenizer=tokenizer,
    args=training_arguments
    peft_config=lora_config,
)
trainer.train()
```

LoRA vs regular finetuning

- `TinyLlama/TinyLlama-1.1B-Chat-v1.0`
- 1.1 billion parameters
- 11k samples
- ~30 minutes

- `nvidia/Llama3-ChatQA-1.5-8B`
- 8 billion parameters
- 11k samples
- ~30 minutes

LoRA with torchtune

```
tune download meta-llama/Meta-Llama-3-8B \
    --output-dir /model_dir \
    --hf-token <HF_TOKEN>
```

```
tune run lora_finetune_single_device \
    --config llama3/8B_lora_single_device \
    model.lora_rank=8 \
    model.lora_alpha=32 \
    ...
```

[1] https://github.com/pytorch/torchtune/blob/main/recipes/configs/llama3/8B_lora_single_device.yaml

Let's practice!

WORKING WITH LLAMA 3