LoRA for Fine-Tuning LLMs explained with codes and example

How to fine-tune your LLMs faster using LoRA

Mehul Gupta · Follow

Published in Data Science in your pocket · 8 min read · Oct 31, 2023

Photo by Towfiqu barbhuiya on Unsplash

This entire year in AI space has been revolutionary because of the advancements in Gen-AI especially the incoming of LLMs. With every passing day, we get something new, be it a new LLM like Mistral-7B, a framework like Langchain or LlamaIndex, or fine-tuning techniques. One of the most significant fine-tuning LLMs that caught my attention is LoRA or Low-Rank Adaptation of LLMs.

My debut book “LangChain in your Pocket” is out now !!

LangChain in your Pocket: Beginner's Guide to Building Generative AI Applications using LLMs

LangChain in your Pocket: Beginner's Guide to Building Generative AI Applications using LLMs eBook : Gupta, Mehul...

www.amazon.in

This post discusses the following things

Mathematical prequisites to understand LoRA

What is LoRA? The maths behind LoRA

The significance of LoRA for fine-tuning

What is Catastrophic forgetting and how LoRA handles it?

Can we use LoRA for any ML model?

> *What is QLoRA?*
>
> *Codes to fine-tune using LoRA with outputs*

Pre-Requisites

The rank of a Matrix: It is the number of linearly independent rows/columns present in the matrix i.e. a row/column can't be produced by applying a

linear transformation to other rows/columns or a combination of other rows/columns. Do see this example

Full Rank Matrix: A full rank matrix is a matrix in which all its rows and columns are linearly independent.

Low-Rank Matrix: A low-rank matrix is a matrix in which its rank is significantly lower than the total number of rows or columns in the matrix.

Matrix Factorization: Given an N x M matrix, then using matrix factorization, this N x M matrix can be decomposed into multiple matrics such that on matrix multiplication, we get the same matrix N x M.

Fine-Tuning: Fine-tuning a model refers to the process of taking a pre-trained model (model trained on some big, public corpus) and further training it on a new, smaller dataset or with a specific task in mind. This is done to adapt the model to a particular problem or to improve its performance on a specific task.

Now we are done with most of the prerequisites. Let's jump on LoRA

Low-Rank Adaptation of LLMs (LoRA)

So, in usual fine-tuning, we

> *Take a pretrained model*
>
> *Do Transfer Learning over new training data to slightly adjust these pre-trained weights*

Now, assume models as big as LLMs (say Llama-70B). If we go for the usual fine-tuning, we would be updating 70B parameters. This will be

> *Very slow*
>
> *Heavy on computational resources*

Now, think of this trick

Instead of updating all of these 70B, ***can we freeze these base weights and create a separate 'updates' matrix?***

Let's understand with an example

- Llama-70B has a weight matrix of the shape N x M which has 70B values as weights.
- Now, we create a new set of update weights matrix UA & UB of the dimensions N x K & K x M where K is small such that

UA x UB = N x M (remember matrix factorization?).

Also, K is a hyperparameter to be tuned, the smaller, the bigger the drop in performance of the LLM.

- Instead of updating the 70B NxM weights matrix, use this set of UA & UB for any updates and eventually use it alongside the pretrained weights for your specific tasks.

Couldn't get it? let's take a step back

So in a general ML training scenario

> *We start off with some equations like Output= Input x W + B where W=Weights & B=Bias are default values (not tuned)*

> *As we train the model, this W will become Wu = updated weights. But if you notice closely, this Wu is nothing but W+ΔW (where ΔW is updates made to default weights W)*
>
> *So, the final equation becomes Y = xWu + Bias in a usual case*

But can we write it like

$$Y = m(W+\Delta W) + Bias$$

$$Y = m(W+UAxUB)+Bias$$

This is the whole idea of how LoRA works. The idea is simple but the benefits are great.

> *So, if (assume) Llama 70B model's weights are in the shape of 700000 x 100000 matrix, hence leading to 70B updates.*
>
> *Using LoRA, we might get 2 matrices UA & UB of dimensions*

> *700000x10 (7 Million) & 10x100000(1 Million) leading to 8 Million updates, hence a reduction of 99.9998857143% in terms of updates.*

Not just computational savings and training time, LoRA also helps in avoiding catastrophic forgetting.

What is catastrophic forgetting?

Catastrophic forgetting, in simple terms, is when a machine learning model, like a neural network or artificial intelligence system, forgets how to perform a task it previously learned when it's trained on new, different tasks especially Fine-Tuning over a pretrained model.

> *Imagine you're trying to teach a robot to do various tasks. You start by teaching it to make a sandwich. The robot learns how to pick up bread, spread peanut butter, and add jelly to make a delicious peanut butter and jelly sandwich. Then, you decide to teach it a new task, like folding laundry.*
>
> *Now, the process of learning this new skill can disrupt the knowledge it had about making sandwiches. So, after learning how to fold laundry, the robot might forget how to make a sandwich correctly. It's as if its memory of the sandwich-making steps has been overwritten by the laundry-folding instructions.*

How does LoRA avoid catastrophic forgetting?

As we are not updating the pretrained weights, the model never forgets what it has already learned. While in general Fine-Tuning, we are updating the actual weights hence there are chances of catastrophic forgetting.

Can we use LoRA for the training of any ML model?

Yes, you can but won't be of much use for small models. Basically, the ***weights matrix of complex models like LLMs are High/Full Rank matrices.*** Using LoRA, we are avoiding another High-Rank matrix after fine-tuning but generating multiple Low-Rank matrices for a proxy for that.

High-ranked matrices have more information (as most/all rows & columns are independent) compared to Low-Ranked matrices, there is some **information loss and hence performance degradation when going for techniques like LoRA.** If in novel training of a model, the time taken and resources used are feasible, LoRA can be avoided. But as LLMs require huge resources, LoRA becomes effective and we can take a hit on slight accuracy to save resources and time.

Before we jump onto codes, one more thing

What is QLoRA?

- Q in this name stands for quantization i.e. the process of reducing the precision of numerical representations of weights, activations, or data.
- Neural networks use floating-point numbers (32-bit or 64-bit) to represent weights, biases, and activations. However, these high-precision representations can be computationally expensive, especially on hardware with limited resources such as mobile devices or edge devices.
- Instead, we can use a lower precision datatype (say Float 16-bit, Integer 8 bit) to reduce the model size and save computation resources significantly
- QLoRA is basically LoRA over a quantized LLM i.e. LLM loaded using a lower precision datatype in the memory.

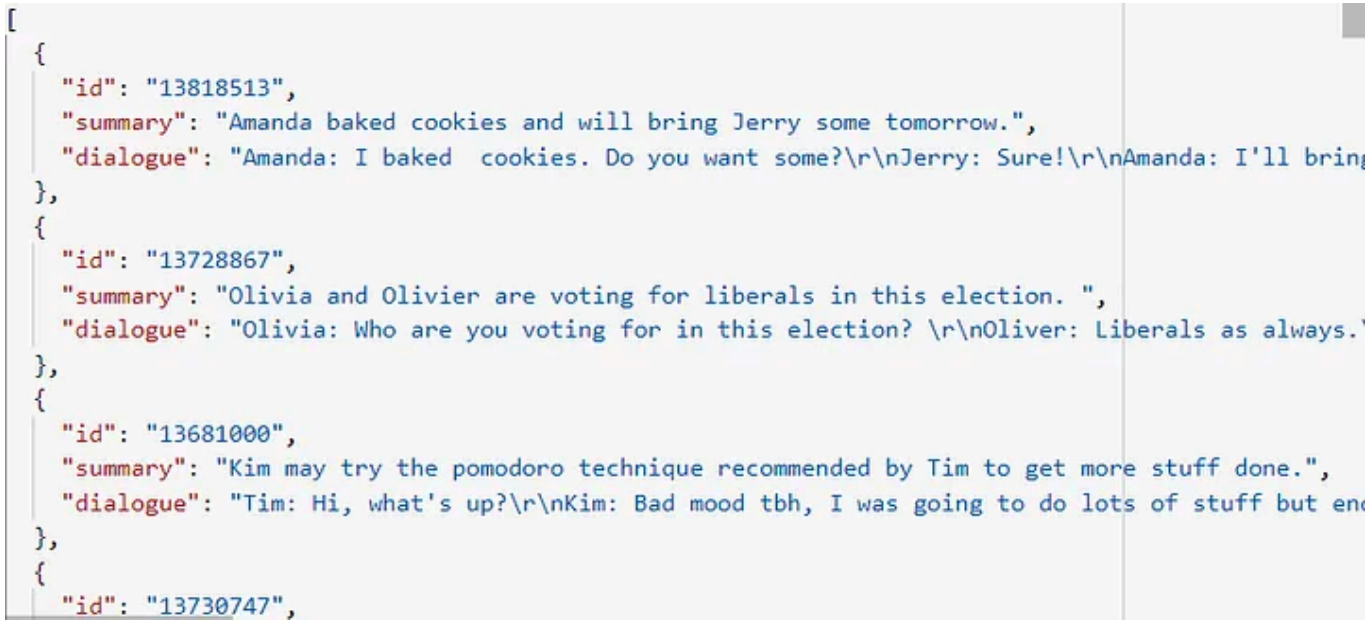
Do remember that using QLoRA can also affect your performance slightly as there is information loss involved

CODES !!

Now, we are well equipped with all the theoretical knowledge. Time for some live-action. We will be fine-tuning

> *'flan-t5-small' (due to hardware restrictions)*
>
> *on Summarization task (Input: Converstaion between 2 people; Output: Summary of the conversation)*

> *Dataset used is a json file with 3 columns id, dialogue(input) and summary(output). Have a look below*



```
[
  {
    "id": "13818513",
    "summary": "Amanda baked cookies and will bring Jerry some tomorrow.",
    "dialogue": "Amanda: I baked  cookies. Do you want some?\r\nJerry: Sure!\r\nAmanda: I'll bring
  },
  {
    "id": "13728867",
    "summary": "Olivia and Olivier are voting for liberals in this election. ",
    "dialogue": "Olivia: Who are you voting for in this election? \r\nOliver: Liberals as always.'
  },
  {
    "id": "13681000",
    "summary": "Kim may try the pomodoro technique recommended by Tim to get more stuff done.",
    "dialogue": "Tim: Hi, what's up?\r\nKim: Bad mood tbh, I was going to do lots of stuff but en
  },
  {
    "id": "13730747",
```

We will start off with pip installing libraries

```
!pip install trl transformers accelerate datasets bitsandbytes einops torch hugg
```

Next, import the important functions required

```
from datasets import load_dataset
from random import randrange
import torch
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM,TrainingArguments,
from peft import LoraConfig, prepare_model_for_kbit_training, get_peft_model, Au
from trl import SFTTrainer
from huggingface_hub import login, notebook_login
```

Next, let's load the training and test dataset alongside the LLM to be fine-tuned with its tokenizer

```
#train & test.json are in same folder as the jupyter notebook
data_files = {'train':'train.json','test':'test.json'}
dataset = load_dataset('json',data_files=data_files)

model_name = "google/flan-t5-small"
model = AutoModelForSeq2SeqLM.from_pretrained(model_name)

#Makes training faster but a little less accurate
model.config.pretraining_tp = 1

tokenizer = AutoTokenizer.from_pretrained(model_name, trust_remote_code=True)

#setting padding instructions for tokenizer
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right"
```

Next, using the Input and Output, we will create a prompt template which is a requirement by the SFTTrainer we will be using later

```
def prompt_instruction_format(sample):
  return f"""### Instruction:
    Use the Task below and the Input given to write the Response:

    ### Task:
    Summarize the Input

    ### Input:
    {sample['dialogue']}

    ### Response:
    {sample['summary']}
    """
```

Now is the time we set the trainer for LoRA

```
# Create the trainer
trainingArgs = TrainingArguments(
    output_dir='output',
    num_train_epochs=1,
    per_device_train_batch_size=4,
    save_strategy="epoch",
    learning_rate=2e-4
)
```

```
peft_config = LoraConfig(
      lora_alpha=16,
      lora_dropout=0.1,
      r=64,
      bias="none",
      task_type="CAUSAL_LM",
)

trainer = SFTTrainer(
    model=model,
    train_dataset=dataset['train'],
    eval_dataset = dataset['test'],
    peft_config=peft_config,
    tokenizer=tokenizer,
    packing=True,
    formatting_func=prompt_instruction_format,
    args=trainingArgs,
)

trainer.train()
```

This requires some explanation

- TrainingArguments sets up some arguments we need while training like Learning rate, epochs, etc.
- Using LoraConfig, we are setting up LoRA hyperparameters, the major one being ‘r’ which is the rank of the two update matrices (the ‘K’ in NxK

& KxM for the update matrices). The less complex the task is, the lower ‘r’ you can afford without much impacting the results

- SFTTrainer stands for Supervised Fine-Tuning which is used when we have labeled data (as in this case, the ground truth is available) to fine-tune whose parameters are self-explanatory.

Start your training !!

```
trainer.train()
```

```
trainer.train()
```

[31/75 05:26 < 08:15, 0.09 it/s, Epoch 0/1]

Step	Training Loss

```
TrainOutput(global_step=31, training_loss=2.020069614533455, metrics={'train_runtime': 337.7663,
'train_samples_per_second': 0.888, 'train_steps_per_second': 0.222, 'total_flos': 23710239031296.0,
'train_loss': 2.020069614533455, 'epoch': 0.41})
```

Now, we will be pushing this fine-tuned model to hugging face-hub and eventually loading it similarly to how we load other LLMs like flan or llama.

For this step, you first need to create your huggingface-hub credentials. You can check this in the below tutorial.

Once done, login into huggingface-hub using the WRITE token in the Jupyter Notebook and push your model using the below code

```
#Write token
```

```
notebook_login()
```

Now, upload the model

```
# Save our tokenizer and create model card
tokenizer.save_pretrained(repository_id)

#Create model card
trainer.create_model_card()

# Push the results to the hub
trainer.push_to_hub()
```

Now, once uploaded, again use the notebook login but using the READ token this time

```
#Read token
notebook_login()
```

Now run the below code for inferencing with your private model

```
# load model and tokenizer from huggingface hub with pipeline
summarizer = pipeline("summarization", model="your model path in huggingface-hub

# select a random test sample
sample = dataset['test'][randrange(len(dataset["test"]))]
print(f"dialogue: \n{sample['dialogue']}\n---------------")

# summarize dialogue
res = summarizer(sample["dialogue"])

print(f"flan-t5-small summary:\n{res[0]['summary_text']}")
```

See the output for yourself

```
Downloading (…)/adapter_config.json: 100% 477/477 [00:00<00:00, 18.7kB/s]
dialogue:
Chris: <file_photo>
Chris: Maybe not he best photo XD
Chris: and im the middle one here
Chris: and you can bring swimming trunks as well because there's opportunity to go to jacuzzi in our
Tom: a jacuzzzzzi????
Chris: oohhhh yeeaahh
Tom: O my godddd.
Tom: Is it big enough for a few people? I feel I woul feel wierd out there on my own :)
Chris:  Yeee, for 5-6 people no problem.
Tom: So you and your brother will join me? :D
Chris: Yes hahaha meybe we can invite someone else or only our little group.
Chris: An maybe watch sth on TV or just make a conversation hhahaha
Tom: Wait you've got a tv outside????
Chris: well we can bring it there :P
```

```
Chris: From our living room
Tom: Do you have WiFi?
Chris: Yes
Tom: Nice, I only get 6GB on my phone when I get outside Ireland
Chris: Yeah, kind a low amount
Tom: Can't wait!
Chris: Me too! :)
---------------
flan-t5-base summary:
Chris and Tom will go to jacuzzi in their garden for 5-6 people. Tom and Chris will bring swimming tr
```

With this, we will be wrapping up this very long post. See you soon !!

Artificial Intelligence | Machine Learning | Data Science | Technology | Programming

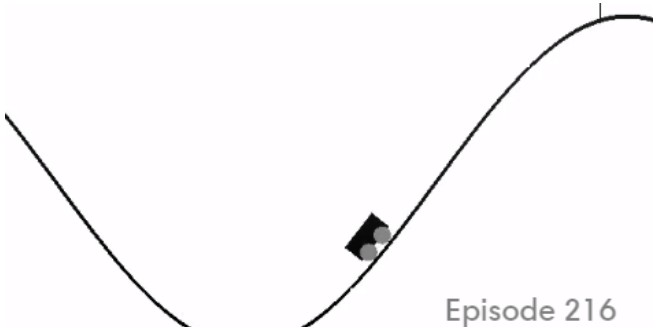
Written by Mehul Gupta

Follow

2.9K Followers · Writer for Data Science in your pocket

Author : LangChain in your Pocket :https://www.amazon.com/dp/B0CTHQHT25

More from Mehul Gupta and Data Science in your pocket



Mehul Gupta in Data Science in your pocket

Deep Q Networks (DQN) explained with examples and codes in...

Value-based methods in reinforcement learning explained with code

7 min read · Apr 8, 2023

118 1

Mehul Gupta in Data Science in your pocket

How to create a custom OpenAI Gym environment? with codes

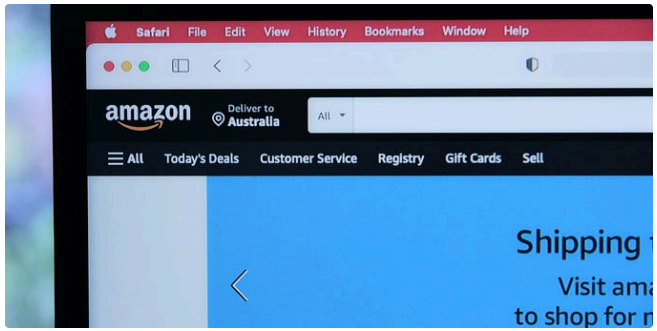
Creating a game environment in OpenAI-gym from scratch

5 min read · Jul 10, 2023

47 2

Mehul Gupta in Data Science in your pocket

Monte Carlo for Reinforcement Learning with example



Mehul Gupta in Data Science in your pocket

Recommendation Systems using Langchain and LLMs with codes

The first thing that comes to our mind when we hear MONTE CARLO is

4 min read · Mar 3, 2020

229 1

using RAG framework and chains

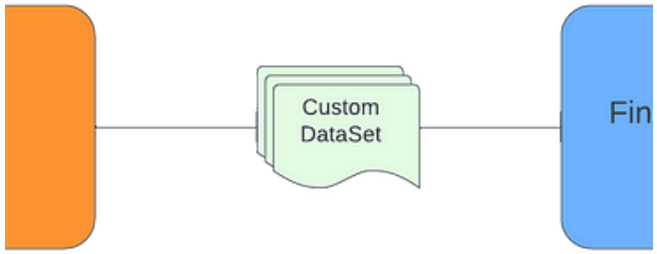
4 min read · Oct 15, 2023

390 3

See all from Mehul Gupta

See all from Data Science in your pocket

Recommended from Medium



Daniel Warfield in Towards Data Science

LoRA—Intuitively and Exhaustively Explained

Exploring the modern wave of machine learning: cutting edge fine tuning

✦ · 18 min read · Nov 7, 2023

1K 9

Suman Das

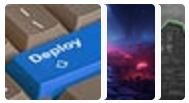
Fine Tune Large Language Model (LLM) on a Custom Dataset with...

The field of natural language processing has been revolutionized by large language...

15 min read · Jan 24, 2024

1.2K 15

Lists



Predictive Modeling w/ Python

20 stories · 1201 saves

ChatGPT prompts

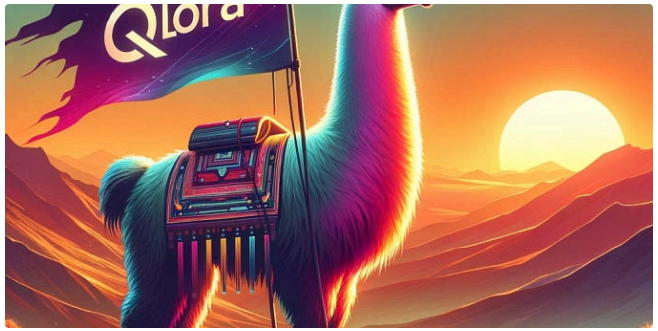
47 stories · 1568 saves

AI Regulation

6 stories · 456 saves

ChatGPT

21 stories · 637 saves



Dilli Prasad Amur

QLoRA: Fine-Tuning Large Language Models (LLM's)

In this blog I will aim to explain the concept and important terminology related to QLoR...

12 min read · Nov 27, 2023

222 4

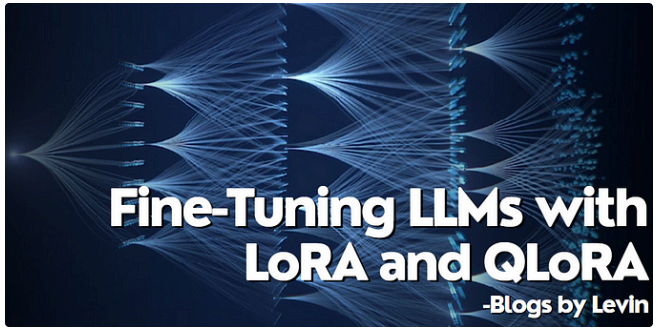
Shrinivasan Sankar

LoRA—Low-Rank Adaptation of LLMs (paper explained)

Introduction

6 min read · Dec 14, 2023

152



Levin M S

LoRA and QLoRA- Effective methods to Fine-tune your LLMs i...

————

10 min read · Dec 5, 2023

122 1

Mastering LLM (Large Language Model)

[Updated]Recent 32 Large Language Models (LLMs) Intervie...

Recent 11 large language models (LLMs) Interview Questions (Answered) for your nex...

15 min read · Apr 6, 2024

60 2