Fine-Tune Llama 2 in Google Colab

Step 1: Install All the Required Package

```
!pip install transformers==4.31.0
```

```
Collecting transformers==4.31.0
  Downloading transformers-4.31.0-py3-none-any.whl (7.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.4/7.4 MB 19.8 MB/s eta 0:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-p
Collecting huggingface-hub<1.0,>=0.14.1 (from transformers==4.31.0)
  Downloading huggingface_hub-0.17.1-py3-none-any.whl (294 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 294.8/294.8 kB 30.8 MB/s eta 0:
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dis
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dis
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-p
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1 (from transformers==4.31.0)
  Downloading tokenizers-0.13.3-cp310-cp310-manylinux_2_17_x86_64.manylinux
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 50.0 MB/s eta 0:00
Collecting safetensors>=0.3.1 (from transformers==4.31.0)
  Downloading safetensors-0.3.3-cp310-cp310-manylinux_2_17_x86_64.manylinux
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 49.6 MB/s eta 0:00
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Installing collected packages: tokenizers, safetensors, huggingface-hub, tr
Successfully installed huggingface-hub-0.17.1 safetensors-0.3.3 tokenizers-
```

```
!pip install -q accelerate==0.21.0 peft==0.4.0 bitsandbytes==0.40.2 transformers
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 244.2/244.2 kB 5.1 MB/s eta 0:
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 72.9/72.9 kB 6.2 MB/s eta 0:0
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 92.5/92.5 MB 11.5 MB/s eta 0:
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.4/7.4 MB 75.4 MB/s eta 0:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.4/77.4 kB 8.8 MB/s eta 0:0
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 59.7 MB/s eta 0:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 294.8/294.8 kB 31.0 MB/s eta 0:
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 76.9 MB/s eta 0:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 519.6/519.6 kB 43.3 MB/s eta 0:
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 115.3/115.3 kB 14.0 MB/s eta 0:
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 20.5 MB/s eta 0:
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 14.9 MB/s eta 0:
```

Step 2: Import All the Required Libraries

```
import os
import torch
from datasets import load_dataset
from transformers import (
    AutoModelForCausalLM,
    AutoTokenizer,
    BitsAndBytesConfig,
    HfArgumentParser,
    TrainingArguments,
    pipeline,
    logging,
)
from peft import LoraConfig, PeftModel
from trl import SFTTrainer
```

In case of Llama 2, the following prompt template is used for the chat models

System Prompt (optional) to guide the model

User prompt (required) to give the instruction

Model Answer (required)

Dataset structure:

<s> [INST] .... [/INST] ....model reply.........</s>

We will reformat our instruction dataset to follow Llama 2's template.

Orignal Dataset: https://huggingface.co/datasets/timdettmers/openassistant-guanaco

Reformat Dataset following the Llama 2 template with 1k sample: https://huggingface.co/datasets/mlabonne/guanaco-llama2-1k

Complete Reformat Dataset following the Llama 2 template: https://huggingface.co/datasets/mlabonne/guanaco-llama2

To know how this dataset was created, you can check this notebook.

https://colab.research.google.com/drive/1Ad7a9zMmkxuXTOh1Z7-rNSICA4dybpM2?usp=sharing

You don't need to follow a specific prompt template if you're using the base Llama 2 model instead of the chat version.

How to fine tune Llama 2

Free Google Colab offers a 15GB Graphics Card (Limited Resources --> Barely enough to store Llama 2–7b's weights)

We also need to consider the overhead due to optimizer states, gradients, and forward activations

Full fine-tuning is not possible here: we need parameter-efficient fine-tuning (PEFT) techniques like LoRA or QLoRA.

To drastically reduce the VRAM usage, we must fine-tune the model in 4-bit precision, which is why we'll use QLoRA here.

Step 3

Load a llama-2-7b-chat-hf model (chat model) Train it on the mlabonne/guanaco-llama2-1k (1,000 samples), which will produce our fine-tuned model Llama-2-7b-chat-finetune QLoRA will use a rank of 64 with a scaling parameter of 16. We'll load the Llama 2 model directly in 4-bit precision using the NF4 type and train it for one epoch

```
# The model that you want to train from the Hugging Face hub
model_name = "NousResearch/Llama-2-7b-chat-hf"

# The instruction dataset to use
dataset_name = "mlabonne/guanaco-llama2-1k"

# Fine-tuned model name
```

```
new_model = "Llama-2-7b-chat-finetune-app"

################################################################################
# QLoRA parameters
################################################################################

# LoRA attention dimension
lora_r = 64

# Alpha parameter for LoRA scaling
lora_alpha = 16

# Dropout probability for LoRA layers
lora_dropout = 0.1

################################################################################
# bitsandbytes parameters
################################################################################

# Activate 4-bit precision base model loading
use_4bit = True

# Compute dtype for 4-bit base models
bnb_4bit_compute_dtype = "float16"

# Quantization type (fp4 or nf4)
bnb_4bit_quant_type = "nf4"

# Activate nested quantization for 4-bit base models (double quantization)
use_nested_quant = False

################################################################################
# TrainingArguments parameters
################################################################################

# Output directory where the model predictions and checkpoints will be stored
output_dir = "./results"

# Number of training epochs
num_train_epochs = 1

# Enable fp16/bf16 training (set bf16 to True with an A100)
fp16 = False
bf16 = False

# Batch size per GPU for training
per_device_train_batch_size = 4

# Batch size per GPU for evaluation
per_device_eval_batch_size = 4

# Number of update steps to accumulate the gradients for
gradient_accumulation_steps = 1
```

```
# Enable gradient checkpointing
gradient_checkpointing = True

# Maximum gradient normal (gradient clipping)
max_grad_norm = 0.3

# Initial learning rate (AdamW optimizer)
learning_rate = 2e-4

# Weight decay to apply to all layers except bias/LayerNorm weights
weight_decay = 0.001

# Optimizer to use
optim = "paged_adamw_32bit"

# Learning rate schedule
lr_scheduler_type = "cosine"

# Number of training steps (overrides num_train_epochs)
max_steps = -1

# Ratio of steps for a linear warmup (from 0 to learning rate)
warmup_ratio = 0.03

# Group sequences into batches with same length
# Saves memory and speeds up training considerably
group_by_length = True

# Save checkpoint every X updates steps
save_steps = 0

# Log every X updates steps
logging_steps = 25

################################################################################
# SFT parameters
################################################################################

# Maximum sequence length to use
max_seq_length = None

# Pack multiple short examples in the same input sequence to increase efficiency
packing = False

# Load the entire model on the GPU 0
device_map = {"": 0}
```

Step 4:Load everything and start the fine-tuning process

First of all, we want to load the dataset we defined. Here, our dataset is already preprocessed but, usually, this is where you would reformat the prompt, filter out bad text, combine multiple datasets, etc.

Then, we're configuring bitsandbytes for 4-bit quantization.

Next, we're loading the Llama 2 model in 4-bit precision on a GPU with the corresponding tokenizer.

Finally, we're loading configurations for QLoRA, regular training parameters, and passing everything to the SFTTrainer. The training can finally start!

```
# Load dataset (you can process it here)
dataset = load_dataset(dataset_name, split="train")

# Load tokenizer and model with QLoRA configuration
compute_dtype = getattr(torch, bnb_4bit_compute_dtype)

bnb_config = BitsAndBytesConfig(
    load_in_4bit=use_4bit,
    bnb_4bit_quant_type=bnb_4bit_quant_type,
    bnb_4bit_compute_dtype=compute_dtype,
    bnb_4bit_use_double_quant=use_nested_quant,
)

# Check GPU compatibility with bfloat16
if compute_dtype == torch.float16 and use_4bit:
    major, _ = torch.cuda.get_device_capability()
    if major >= 8:
        print("=" * 80)
        print("Your GPU supports bfloat16: accelerate training with bf16=True")
        print("=" * 80)

# Load base model
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    quantization_config=bnb_config,
    device_map=device_map
)
model.config.use_cache = False
model.config.pretraining_tp = 1

# Load LLaMA tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_name, trust_remote_code=True)
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right" # Fix weird overflow issue with fp16 training

# Load LoRA configuration
peft_config = LoraConfig(
    lora_alpha=lora_alpha,
    lora_dropout=lora_dropout,
    r=lora_r,
    bias="none",
    task_type="CAUSAL_LM",
)
```

```
# Set training parameters
training_arguments = TrainingArguments(
    output_dir=output_dir,
    num_train_epochs=num_train_epochs,
    per_device_train_batch_size=per_device_train_batch_size,
    gradient_accumulation_steps=gradient_accumulation_steps,
    optim=optim,
    save_steps=save_steps,
    logging_steps=logging_steps,
    learning_rate=learning_rate,
    weight_decay=weight_decay,
    fp16=fp16,
    bf16=bf16,
    max_grad_norm=max_grad_norm,
    max_steps=max_steps,
    warmup_ratio=warmup_ratio,
    group_by_length=group_by_length,
    lr_scheduler_type=lr_scheduler_type,
    report_to="tensorboard"
)

# Set supervised fine-tuning parameters
trainer = SFTTrainer(
    model=model,
    train_dataset=dataset,
    peft_config=peft_config,
    dataset_text_field="text",
    max_seq_length=max_seq_length,
    tokenizer=tokenizer,
    args=training_arguments,
    packing=packing,
)

# Train model
trainer.train()

# Save trained model
trainer.model.save_pretrained(new_model)
```

Downloading readme: 100%		1.02k/1.02k [00:00<00:00, 23.4kB/s]
Downloading data files: 100%		1/1 [00:00<00:00, 1.94it/s]
Downloading data: 100%		967k/967k [00:00<00:00, 2.03MB/s]
Extracting data files: 100%		1/1 [00:00<00:00, 24.69it/s]
Generating train split:		1000/1000 [00:00<00:00, 9615.88

```
100%                                              examples/s]
Downloading (…)lve/main                           583/583 [00:00<00:00,
/config.json: 100%                                13.6kB/s]
Downloading                                       26.8k/26.8k [00:00<00:00,
(…)fetensors.index.json: 100%                     524kB/s]
Downloading shards:                               2/2 [04:30<00:00,
100%                                              123.33s/it]
Downloading (…)of-                                9.98G/9.98G [03:23<00:00,
00002.safetensors: 100%                           58.3MB/s]
Downloading (…)of-                                3.50G/3.50G [01:06<00:00,
00002.safetensors: 100%                           53.2MB/s]
Loading checkpoint shards:                        2/2 [01:12<00:00,
100%                                              32.64s/it]
Downloading                                       179/179 [00:00<00:00,
(…)neration_config.json: 100%                     8.71kB/s]
Downloading                                       746/746 [00:00<00:00,
(…)okenizer_config.json: 100%                     50.4kB/s]
Downloading                                       500k/500k [00:00<00:00,
tokenizer.model: 100%                             17.3MB/s]
Downloading (…)/main                              1.84M/1.84M [00:00<00:00,
/tokenizer.json: 100%                             23.2MB/s]
Downloading                                       21.0/21.0 [00:00<00:00,
(…)in/added_tokens.json: 100%                     1.60kB/s]
Downloading                                       435/435 [00:00<00:00,
(…)cial_tokens_map.json: 100%                     30.3kB/s]
```

Step 5: Check the plots on tensorboard, as follows

```
%load_ext tensorboard
%tensorboard --logdir results/runs
```

```
Not found
```

Step 6:Use the text generation pipeline to ask questions like “What is a large language model?”

------ Note: that I’m formatting the input to match Llama 2’s prompt template.

```
# Ignore warnings
logging.set_verbosity(logging.CRITICAL)

# Run text generation pipeline with our next model
prompt = "What is a llm?"
pipe = pipeline(task="text-generation", model=model, tokenizer=tokenizer, max_le
```

```
result = pipe(f"[INST] {prompt} [/INST]")
print(result[0]['generated_text'])

"""
OUTPUT:

[INST] What is a large language model? [/INST]
A large language model is a type of artificial intelligence (AI)
model that is trained on a large dataset of text to generate human-like language
about the large language models, but what are they? what are they used for? how
and risks of using them? what are the challenges and limitations of using them?
of using them? what are the potential applications of large language models? wha
what are the potential benefits of large language models? what are the potential
what are the potential challenges of large language models? what are the potenti
what are the potential risks of large language models? what are the potential be
what are the potential drawbacks of large language models?

"""
```

```
[INST] What is a llm? [/INST] An LLM, or Master of Laws, is a postgraduate

The LLM program typically takes one year to complete and involves coursewor

An LLM degree can
'\nOUTPUT:\n\n[INST] What is a large language model? [/INST] A large langu
age model is a type of artificial intelligence (AI) \nmodel that is traine
d on a large dataset of text to generate human-like language outputs. ever
ybody is talking \nabout the large language models, but what are they? wha
t are they used for? how do they work? what are the benefits \nand risks o
f using them? what are the challenges and limitations of using them? what
are the ethical considerations \nof using them? what are the potential app
lications of large language models? what are the potential risks of large
language models? \nwhat are the potential benefits of large language model
```

```
# Empty VRAM
del model
del pipe
del trainer
import gc
gc.collect()
gc.collect()
```

```
19965
```

You can train a Llama 2 model on the entire dataset using mlabonne/guanaco-llama2

Step 7: Store New Llama2 Model (Llama-2-7b-chat-finetune)

```
# Reload model in FP16 and merge it with LoRA weights
base_model = AutoModelForCausalLM.from_pretrained(
```

```
        model_name,
        low_cpu_mem_usage=True,
        return_dict=True,
        torch_dtype=torch.float16,
        device_map=device_map,
)
model = PeftModel.from_pretrained(base_model, new_model)
model = model.merge_and_unload()

# Reload tokenizer to save it
tokenizer = AutoTokenizer.from_pretrained(model_name, trust_remote_code=True)
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right"
```

```
Loading checkpoint shards:                                          2/2 [01:05<00:00,
```

Step 8: Push Model to Hugging Face Hub

Our weights are merged and we reloaded the tokenizer. We can now push everything to the Hugging Face Hub to save our model.

```
import locale
locale.getpreferredencoding = lambda: "UTF-8"
```

```
!huggingface-cli login

model.push_to_hub("atharvapawar/Llama-2-7b-chat-finetune-app", check_pr=True)

tokenizer.push_to_hub("atharvapawar/Llama-2-7b-chat-finetune-app",check_pr=True)
```

```
    _|    _|  _|    _|    _|_|_|    _|_|_|  _|_|_|  _|      _|    _|_|_|
    _|    _|  _|    _|  _|        _|          _|    _|_|    _|  _|
    _|_|_|_|  _|    _|  _|  _|_|  _|  _|_|    _|    _|  _|  _|  _|  _|_|
    _|    _|  _|    _|  _|    _|  _|    _|    _|    _|    _|_|  _|    _|
    _|    _|    _|_|      _|_|_|    _|_|_|  _|_|_|  _|      _|    _|_|_|

    To login, `huggingface_hub` requires a token generated from https://hug
Token:
Add token as git credential? (Y/n) y
Token is valid (permission: write).
Cannot authenticate through git-credential as no helper is defined on your
You might have to re-authenticate when pushing to the Hugging Face Hub.
Run the following command in your terminal in case you want to set the 'sto

git config --global credential.helper store

Read https://git-scm.com/book/en/v2/Git-Tools-Credential-Storage for more d
Token has not been saved to git credential helper.
Your token has been saved to /root/.cache/huggingface/token
```

```
Login successful
```

pytorch_model-00002-of-00002.bin: 100%	3.50G/3.50G [01:22<00:00, 43.2MB/s]
Upload 2 LFS files: 100%	2/2 [04:10<00:00, 132.89s/it]
pytorch_model-00001-	9.98G/9.98G [04:10<00:00,

You can now use this model for inference by loading it like any other Llama 2 model from the Hub.

```
import requests

API_URL = "https://api-inference.huggingface.co/models/atharvapawar/Llama-2-7b-c
headers = {"Authorization": "Bearer hf_AgBDNg"}

def query(payload):
    response = requests.post(API_URL, headers=headers, json=payload)
    return response.json()

output = query({
    "inputs": "Can you please let us know more details about your ",
})
output
```

```
{'error': 'The model atharvapawar/Llama-2-7b-chat-finetune-app is too
large to be loaded automatically (13GB > 10GB). For commercial use please
use PRO spaces (https://huggingface.co/spaces) or Inference Endpoints
(https://huggingface.co/inference-endpoints).'}
```

Project

```
from transformers import AutoModelForCausalLM, AutoTokenizer

def load_model_and_generate_response(input_text):
    # Define the model and tokenizer names
    model_name = "atharvapawar/Llama-2-7b-chat-finetune-app"

    # Load the tokenizer
    tokenizer = AutoTokenizer.from_pretrained(model_name)

    # Load the model
    model = AutoModelForCausalLM.from_pretrained(model_name)

    # Tokenize the input text
    input_ids = tokenizer.encode(input_text, return_tensors="pt")
```

```
    input_ids = tokenizer.encode(input_text, return_tensors="pt")

    # Generate a response
    with torch.no_grad():
        output = model.generate(input_ids, max_length=50, num_return_sequences=1

    # Decode and return the response
    response = tokenizer.decode(output[0], skip_special_tokens=True)
    return response

input_text = "Hello, how are you?"
response = load_model_and_generate_response(input_text)
print("Model Response:", response)
```

test

```
import requests

API_URL = "https://api-inference.huggingface.co/models/BELLE-2/BELLE-Llama2-13B-
headers = {"Authorization": "Bearer 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"}

def query(payload):
    response = requests.post(API_URL, headers=headers, json=payload)
    return response.json()

output = query({
    "inputs": "The answer to the universe is",
})
output
```

```
{'error': 'The model BELLE-2/BELLE-Llama2-13B-chat-0.4M is too large to be
loaded automatically (26GB > 10GB). Please use Spaces
(https://huggingface.co/spaces) or Inference Endpoints
(https://huggingface.co/inference-endpoints).'}
```

```
!pip install indic-transliteration
from indic_transliteration import sanscript

# Sanskrit text in Devanagari script
sanskrit_text = "गण्यन्ते इति गणा: समूहा"

# Transliterate the text to IAST
iast_text = sanscript.transliterate(sanskrit_text, sanscript.DEVANAGARI, sanscri

print(iast_text)
```

```
Collecting indic-transliteration
  Downloading indic_transliteration-2.3.52-py3-none-any.whl (145 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 145.2/145.2 kB 3.5 MB/s eta 0:
Collecting backports.functools-lru-cache (from indic-transliteration)
  Downloading backports.functools_lru_cache-1.6.6-py2.py3-none-any.whl (5.9
```

```
Requirement already satisfied: regex in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: typer in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: toml in /usr/local/lib/python3.10/dist-packa
Collecting roman (from indic-transliteration)
  Downloading roman-4.1-py3-none-any.whl (5.5 kB)
Requirement already satisfied: click<9.0.0,>=7.1.1 in /usr/local/lib/python
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib
Installing collected packages: roman, backports.functools-lru-cache, indic-
Successfully installed backports.functools-lru-cache-1.6.6 indic-transliter
gaṇyante iti gaṇā: samūhā
```