

### Breakdown of Each Library

1 accelerate – Hugging Face ka library jo multi-GPU, TPU & distributed training ko optimize karta hai.

Agar FSDP, DeepSpeed use kar rahe ho toh must-have hai.

2 peft (Parameter Efficient Fine-Tuning) –

LoRA, QLoRA, Adapters jaise low-memory tuning methods ke liye use hota hai. Full fine-tuning ki jagah lightweight & efficient tuning karne me madad karta hai.

3 bitsandbytes –

8-bit aur 4-bit quantization support karta hai. QLoRA fine-tuning me VRAM kaafi save hota hai.

4 git+<https://github.com/huggingface/transformers> –

Hugging Face ke transformers ka latest GitHub version install karta hai. Ye zaroori hai agar koi naye models ya features chahiye ho jo PyPI version me nahi mile.

5 trl (Transformer Reinforcement Learning) –

RLHF (Reinforcement Learning from Human Feedback) ke liye. Agar ChatGPT-like models banana hai toh `trl` ka use hota hai.

6 py7zr –

7z format wali compressed files ko unzip karne ke liye. Agar Hugging Face ya kisi aur se compressed dataset mila toh ye useful hoga.

7 auto-gptq –

GPTQ-based quantization ke liye. Faster inference aur VRAM efficiency improve karta hai.

8 optimum –

Hugging Face ka library jo ONNX, TensorRT, Habana Gaudi, NeuronX jaise hardware optimizations provide karta hai.

Accelerated inference aur optimized training ke liye best hai.

### 🔥 Summary

Agar low-VRAM GPUs (24GB ya less) par fine-tuning kar rahe ho toh bitsandbytes + peft + QLoRA combo best hai.

Agar multi-GPU/TPU cluster pe train kar rahe ho toh accelerate + optimum zaroori hai.

Agar RLHF (like ChatGPT) fine-tune karna hai toh TRL package kaam aayega.

```
!pip install accelerate peft bitsandbytes git+https://github.com/huggingface/transformers trl py7zr auto-gptq optimum
```




```

Successfully uninstalled nvidia-cublas-cu12-12.5.1.3
Attempting uninstall: nvidia-cusparse-cu12
Found existing installation: nvidia-cusparse-cu12 12.5.1.3
Uninstalling nvidia-cusparse-cu12-12.5.1.3:
Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
Attempting uninstall: nvidia-cudnn-cu12
Found existing installation: nvidia-cudnn-cu12 9.3.0.75
Uninstalling nvidia-cudnn-cu12-9.3.0.75:
Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
Attempting uninstall: nvidia-cusolver-cu12
Found existing installation: nvidia-cusolver-cu12 11.6.3.83
Uninstalling nvidia-cusolver-cu12-11.6.3.83:
Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
Attempting uninstall: transformers
Found existing installation: transformers 4.48.3
Uninstalling transformers-4.48.3:
Successfully uninstalled transformers-4.48.3
Successfully installed auto-gptq-0.7.1 bitsandbytes-0.45.2 brotli-1.1.0 datasets-3.3.2 dill-0.3.8 gekko-1.2.1 inflate64-


```

Breakdown  from huggingface\_hub import notebook\_login

This imports the notebook\_login function, which is used for authentication inside Jupyter Notebooks or Google Colab.

 notebook\_login()

This will prompt you to enter your Hugging Face access token. You can get the token from Hugging Face website.

 Why is this important?

If you are downloading a private model or dataset, authentication is required.

If you want to upload your fine-tuned model back to Hugging Face, you need to log in first.

 Alternative for Script-Based Login

If you are running a script (not in a notebook), use:

```
from huggingface_hub import login
```

```
login(token="your_huggingface_token")
```

```
from huggingface_hub import notebook_login
notebook_login()
```




Breakdown of Each Import

 import torch

PyTorch is the core deep-learning library used for training models. It helps in tensor operations, GPU acceleration, and model training.

 from datasets import load\_dataset, Dataset

load\_dataset: Used to load datasets from Hugging Face Hub or local files. Dataset: Helps in creating a dataset manually from Python objects (like a list or dictionary).


 from peft import LoraConfig, AutoPeftModelForCausalLM, prepare\_model\_for\_kbit\_training, get\_peft\_model

LoraConfig: Configuration for LoRA (Low-Rank Adaptation), which makes fine-tuning more memory efficient.

AutoPeftModelForCausalLM: Loads a causal language model with PEFT (Parameter-Efficient Fine-Tuning).

prepare\_model\_for\_kbit\_training: Optimizes the model for low-bit training (8-bit/4-bit with QLoRA).

get\_peft\_model: Converts a standard model into a LoRA-optimized model.

 from transformers import AutoModelForCausalLM, AutoTokenizer, GPTQConfig, TrainingArguments

AutoModelForCausalLM: Loads a pre-trained causal language model (like LLaMA, Mistral).

AutoTokenizer: Tokenizer for preprocessing text input.

GPTQConfig: Configures GPTQ (Quantized GPT) for efficient inference.


TrainingArguments: Defines training settings like epochs, batch size, optimizer, learning rate, etc.


 from trl import SFTTrainer

SFTTrainer: Trainer from the trl library used for Supervised Fine-Tuning (SFT).

It simplifies LoRA-based fine-tuning and integrates well with Hugging Face.  import os

Used for handling file paths and system settings, like saving models, loading datasets, etc.

 What is this setup used for?

 Fine-tuning large language models (LLMs) efficiently using LoRA and QLoRA.

✔ Using Hugging Face datasets and models.

✔ Training a model with low-bit precision (4-bit/8-bit) for better memory efficiency.

```
import torch
from datasets import load_dataset, Dataset
from peft import LoraConfig, AutoPeftModelForCausalLM, prepare_model_for_kbit_training, get_peft_model
from transformers import AutoModelForCausalLM, AutoTokenizer, GPTQConfig, TrainingArguments
from trl import SFTTrainer
import os
```

Understanding the Code: Loading and Preparing the Dataset for Fine-Tuning This code is loading, processing, and converting a dataset into a format suitable for fine-tuning an LLM (like Mistral or LLaMA-2) for text summarization. Let's break it down step by step.

#### 1 Loading the Dataset

```
data = load_dataset("samsum", split="train")
```

◆ `load_dataset("samsum", split="train")` loads the Samsum dataset, which contains dialogues and their summaries.

◆ `split="train"` ensures that we load only the training set.

#### ✔ Samsum Dataset Overview

dialogue: A conversation between people.

summary: A short summary of that conversation.

🔍 Example from the dataset:

dialogue summary

Alice: Hey, how are you? Bob: I'm good, you? Alice and Bob greet each other.

#### 2 Converting Dataset to Pandas DataFrame

```
data_df = data.to_pandas()
```

◆ This converts the dataset into a Pandas DataFrame for easier processing.

#### 3 Formatting Data for LLM Fine-Tuning

```
data_df["text"] = data_df[["dialogue", "summary"]].apply( lambda x: "###Human: Summarize this following dialogue: " + x["dialogue"] +
"\n###Assistant: " + x["summary"], axis=1 )
```

◆ Purpose: It formats the data into a ChatML-style prompt to fine-tune LLaMA or Mistral.

💡 How It Works:

It takes the dialogue and summary columns.

It transforms them into a prompt-response format for LLM training.

🔍 Example Output:

✓ Human: Summarize this following dialogue:

Alice: Hey, how are you?

Bob: I'm good, you?

Assistant: Alice and Bob greet each other.

◆ This format mimics human-AI interactions, making it suitable for instruction-tuned models like Mistral or LLaMA-2-Chat.

#### 4 Checking the First Example

```
print(data_df.iloc[0])
```

◆ `data_df.iloc[0]` prints the first row of the dataset after formatting.

#### 5 Converting Back to Hugging Face Dataset

```
data = Dataset.from_pandas(data_df)
```

◆ Why? Since fine-tuning with 🤗 Transformers & PEFT requires a Hugging Face dataset, we convert it back after processing.

🚀 Summary

✔ Loads the Samsum dataset (dialogue → summary).

✔ Formats it into a prompt-response structure for LLM fine-tuning.

✔ Converts it back into a Hugging Face Dataset for training.

🔥 Next: Do you want to tokenize this dataset for Mistral/LLaMA-2 fine-tuning? 🚀

```
data = load_dataset("samsum", split="train")
```

README.md: 100%

7.04k/7.04k [00:00<00:00, 622kB/s]

samsum.py: 100%

3.36k/3.36k [00:00<00:00, 330kB/s]

The repository for samsum contains custom code which must be executed to correctly load the dataset. You can inspect the  
You can avoid this prompt in future by passing the argument ``trust_remote_code=True``.

Do you wish to run the custom code? [y/N] y

corpus.7z: 100%

2.94M/2.94M [00:00<00:00, 13.0MB/s]

Generating train split: 100%

14732/14732 [00:00<00:00, 24485.00 examples/s]

Generating test split: 100%

819/819 [00:00<00:00, 3.32 examples/s]

Generating validation split: 100%

818/818 [00:00<00:00, 3.43 examples/s]

```
data_df = data.to_pandas()
```

data\_df

	id	dialogue	summary
0	13818513	Amanda: I baked cookies. Do you want some?\r\...	Amanda baked cookies and will bring Jerry some...
1	13728867	Olivia: Who are you voting for in this electio...	Olivia and Olivier are voting for liberals in ...
2	13681000	Tim: Hi, what's up?\r\nKim: Bad mood tbh, I wa...	Kim may try the pomodoro technique recommended...
3	13730747	Edward: Rachel, I think I'm in ove with Bella....	Edward thinks he is in love with Bella. Rachel...
4	13728094	Sam: hey overheard rick say something\r\nSam:...	Sam is confused, because he overheard Rick com...
...	...	...	...
14727	13863028	Romeo: You are on my 'People you may know' lis...	Romeo is trying to get Greta to add him to her...
14728	13828570	Theresa: <file_photo>\r\nTheresa: <file_photo>...	Theresa is at work. She gets free food and fre...
14729	13819050	John: Every day some bad news. Japan will hunt...	Japan is going to hunt whales again. Island an...
14730	13828395	Jennifer: Dear Celia! How are you doing?\r\nJe...	Celia couldn't make it to the afternoon with t...
14731	13729017	Georgia: are you ready for hotel hunting? We n...	Georgia and Juliette are looking for a hotel i...

14732 rows x 3 columns

```
data_df["text"] = data_df[["dialogue", "summary"]].apply(lambda x: "###Human: Summarize this following dialogue: " + x["dial
```

data\_df

	id	dialogue	summary	text
0	13818513	Amanda: I baked cookies. Do you want some?\r\...	Amanda baked cookies and will bring Jerry some...	###Human: Summarize this following dialogue: A...
1	13728867	Olivia: Who are you voting for in this electio...	Olivia and Olivier are voting for liberals in ...	###Human: Summarize this following dialogue: O...
2	13681000	Tim: Hi, what's up?\r\nKim: Bad mood tbh, I wa...	Kim may try the pomodoro technique recommended...	###Human: Summarize this following dialogue: T...
3	13730747	Edward: Rachel, I think I'm in ove with Bella....	Edward thinks he is in love with Bella. Rachel...	###Human: Summarize this following dialogue: E...
4	13728094	Sam: hey overheard rick say something\r\nSam:...	Sam is confused, because he overheard Rick com...	###Human: Summarize this following dialogue: S...
...	...	...	...	...
14727	13863028	Romeo: You are on my 'People you may know' lis...	Romeo is trying to get Greta to add him to her...	###Human: Summarize this following dialogue: R...
14728	13828570	Theresa: <file_photo>\r\nTheresa: <file_photo>...	Theresa is at work. She gets free food and fre...	###Human: Summarize this following dialogue: T...

✓ Human: Summarize this following dialogue:

Amanda: I baked cookies. Do you want some?

Jerry: Yes, I'd love some.

Assistant:

Amanda baked cookies and will bring Jerry some.

```
print(data_df.iloc[0])
```

```
id 13818513
dialogue Amanda: I baked cookies. Do you want some?\r\...
summary Amanda baked cookies and will bring Jerry some...
text ###Human: Summarize this following dialogue: A...
Name: 0, dtype: object
```

```
data = Dataset.from_pandas(data_df)
```

```
data
```

```
Dataset({
  features: ['id', 'dialogue', 'summary', 'text'],
  num_rows: 14732
})
```

```
tokenizer = AutoTokenizer.from_pretrained("TheBloke/Mistral-7B-Instruct-v0.1-GPTQ")
```

```
tokenizer_config.json: 100% 1.46k/1.46k [00:00<00:00, 137kB/s]
tokenizer.model: 100% 493k/493k [00:00<00:00, 7.43MB/s]
tokenizer.json: 100% 1.80M/1.80M [00:00<00:00, 21.6MB/s]
special_tokens_map.json: 100% 72.0/72.0 [00:00<00:00, 7.11kB/s]
```

```
tokenizer.eos_token
```

```
'</s>'
```

```
tokenizer.eos_token_id
```

```
2
```

```
tokenizer.pad_token
```

```
tokenizer.pad_token = tokenizer.eos_token
```

```
quantization_config_loading = GPTQConfig(bits=4, disable_exllama=True, tokenizer=tokenizer)
```

```
Using `disable_exllama` is deprecated and will be removed in version 4.37. Use `use_exllama` instead and specify the ver
```

```
model = AutoModelForCausalLM.from_pretrained(
    "TheBloke/Mistral-7B-Instruct-v0.1-GPTQ",
    quantization_config=quantization_config_loading,
    device_map="auto")
```

```
config.json: 100% 963/963 [00:00<00:00, 59.3kB/s]
/usr/local/lib/python3.11/dist-packages/transformers/quantizers/auto.py:207: UserWarning: You passed `quantization_confi
warnings.warn(warning_msg)
/usr/local/lib/python3.11/dist-packages/auto_gptq/nn_modules/triton_utils/kernels.py:410: FutureWarning: `torch.cuda.amp
@custom_fwd
/usr/local/lib/python3.11/dist-packages/auto_gptq/nn_modules/triton_utils/kernels.py:418: FutureWarning: `torch.cuda.amp
@custom_bwd
/usr/local/lib/python3.11/dist-packages/auto_gptq/nn_modules/triton_utils/kernels.py:461: FutureWarning: `torch.cuda.amp
@custom_fwd(cast_inputs=torch.float16)
WARNING:auto_gptq.nn_modules.qlinear.qlinear_cuda:CUDA extension not installed.
WARNING:auto_gptq.nn_modules.qlinear.qlinear_cuda_old:CUDA extension not installed.
model.safetensors: 100% 4.16G/4.16G [00:56<00:00, 86.5MB/s]
`loss_type=None` was set in the config but it is unrecognised.Using the default loss: `ForCausalLMLoss`.
Some weights of the model checkpoint at TheBloke/Mistral-7B-Instruct-v0.1-GPTQ were not used when initializing MistralFo
- This IS expected if you are initializing MistralForCausalLM from the checkpoint of a model trained on another task or
- This IS NOT expected if you are initializing MistralForCausalLM from the checkpoint of a model that you expect to be e
generation_config.json: 100% 116/116 [00:00<00:00, 3.78kB/s]
```

```
print(model)
```

```
MistralForCausalLM(
  (model): MistralModel(
    (embed_tokens): Embedding(32000, 4096, padding_idx=0)
    (layers): ModuleList(
      (0-31): 32 x MistralDecoderLayer(
```

```

        (self_attn): MistralAttention(
          (k_proj): QuantLinear()
          (o_proj): QuantLinear()
          (q_proj): QuantLinear()
          (v_proj): QuantLinear()
        )
        (mlp): MistralMLP(
          (act_fn): SiLU()
          (down_proj): QuantLinear()
          (gate_proj): QuantLinear()
          (up_proj): QuantLinear()
        )
        (input_layernorm): MistralRMSNorm((4096,), eps=1e-05)
        (post_attention_layernorm): MistralRMSNorm((4096,), eps=1e-05)
      )
    )
    (norm): MistralRMSNorm((4096,), eps=1e-05)
    (rotary_emb): MistralRotaryEmbedding()
  )
  (lm_head): Linear(in_features=4096, out_features=32000, bias=False)
)

# Load a 4-bit quantized model
quantization_config = BitsAndBytesConfig(
    load_in_4bit=True, # Enable 4-bit quantization
    bnb_4bit_compute_dtype=torch.float16, # Use fp16 for computation
    bnb_4bit_use_double_quant=True, # Use double quantization for memory efficiency
)

# Load model and tokenizer
model = AutoModelForCausalLM.from_pretrained(
    "mistralai/Mistral-7B-Instruct-v0.1",
    quantization_config=quantization_config,
    device_map="auto" # Automatically assigns layers to available GPUs
)

model.config.use_cache=False

model.config.pretraining_tp=1

model.gradient_checkpointing_enable()

model = prepare_model_for_kbit_training(model)

r=16 controls how much LoRA modifies the model (higher = more expressive).
✅ lora_alpha=16 scales LoRA's effect on training.
✅ lora_dropout=0.05 prevents overfitting (good for small datasets).
✅ target_modules=["q_proj", "v_proj"] makes LoRA memory-efficient.
✅ Great for fine-tuning LLaMA, Mistral, Falcon on low-VRAM GPUs.

# ["q_proj", "v_proj", "k_proj"] → Adds key projection (more expressive)
# ["q_proj", "v_proj", "out_proj"] → Also fine-tunes attention output

peft_config = LoraConfig(
    r=16, lora_alpha=16, lora_dropout=0.05, bias="none", task_type="CAUSAL_LM", target_modules=["q_proj", "v_proj"]
)

model = get_peft_model(model, peft_config)

➡ /usr/local/lib/python3.11/dist-packages/peft/mapping.py:185: UserWarning: The PEFT config's `base_model_name_or_path` wa
warnings.warn(

print(model)

➡ PeftModelForCausalLM(
  (base_model): LoraModel(
    (model): PeftModelForCausalLM(
      (base_model): LoraModel(
        (model): MistralForCausalLM(
          (embed_tokens): Embedding(32000, 4096, padding_idx=0)
          (layers): ModuleList(
            (0-31): 32 x MistralDecoderLayer(
              (self_attn): MistralAttention(

```

```

(k_proj): QuantLinear()
(o_proj): QuantLinear()
(q_proj): lora.QuantLinear(
  (base_layer): QuantLinear()
  (lora_dropout): ModuleDict(
    (default): Dropout(p=0.05, inplace=False)
  )
  (lora_A): ModuleDict(
    (default): Linear(in_features=4096, out_features=16, bias=False)
  )
  (lora_B): ModuleDict(
    (default): Linear(in_features=16, out_features=4096, bias=False)
  )
  (lora_embedding_A): ParameterDict()
  (lora_embedding_B): ParameterDict()
  (lora_magnitude_vector): ModuleDict()
  (quant_linear_module): QuantLinear()
)
(v_proj): lora.QuantLinear(
  (base_layer): QuantLinear()
  (lora_dropout): ModuleDict(
    (default): Dropout(p=0.05, inplace=False)
  )
  (lora_A): ModuleDict(
    (default): Linear(in_features=4096, out_features=16, bias=False)
  )
  (lora_B): ModuleDict(
    (default): Linear(in_features=16, out_features=1024, bias=False)
  )
  (lora_embedding_A): ParameterDict()
  (lora_embedding_B): ParameterDict()
  (lora_magnitude_vector): ModuleDict()
  (quant_linear_module): QuantLinear()
)
)
(mlp): MistralMLP(
  (act_fn): SiLU()
  (down_proj): QuantLinear()
  (gate_proj): QuantLinear()
  (up_proj): QuantLinear()
)
(input_layernorm): MistralRMSNorm((4096,), eps=1e-05)
(post_attention_layernorm): MistralRMSNorm((4096,), eps=1e-05)
)
(norm): MistralRMSNorm((4096,), eps=1e-05)
(rotary_emb): MistralRotaryEmbedding()
)

training_arguments = TrainingArguments(
  output_dir="mistral-finetuned-samsum",
  per_device_train_batch_size=8,
  gradient_accumulation_steps=1,
  optim="paged_adamw_32bit",
  learning_rate=2e-4,
  lr_scheduler_type="cosine",
  save_strategy="epoch",
  logging_steps=100,
  num_train_epochs=1,
  max_steps=250,
  fp16=True,
  push_to_hub=True,
  report_to="none"
)

# Create the SFTTrainer
trainer = SFTTrainer(
  model=model,
  train_dataset=data,
  peft_config=peft_config,
  args=training_arguments,
  tokenizer=tokenizer,
)

```

 <ipython-input-35-92d616e9460c>:2: FutureWarning: `tokenizer` is deprecated and removed starting from version 0.16.0 for trainer = SFTTrainer(

```

Converting train dataset to ChatML: 100% 14732/14732 [00:01<00:00, 10766.01 examples/s]

Applying chat template to train dataset: 100% 14732/14732 [00:01<00:00, 11172.32 examples/s]


Tokenizing train dataset: 100% 14732/14732 [00:23<00:00, 785.37 examples/s]

Tokenizing train dataset: 100% 14732/14732 [00:13<00:00, 827.65 examples/s]

No label_names provided for model class `PeftModelForCausalLM`. Since `PeftModel` hides base models input arguments, if

```

```
trainer.train()
```

 /usr/local/lib/python3.11/dist-packages/torch/\_dynamo/eval\_frame.py:632: UserWarning: torch.utils.checkpoint: the use\_re return fn(\*args, \*\*kwargs)

[250/250 45:48, Epoch 0/1]

Step	Training Loss
100	1.868200
200	1.762600

```
TrainOutput(global_step=250, training_loss=1.8019019775390626, metrics={'train_runtime': 2763.5241,
'train_samples_per_second': 0.724, 'train_steps_per_second': 0.09, 'total_flos': 704127752404992.0, 'train_loss':
1.8019019775390626})
```


```
! cp -r /content/mistral-finetuned-samsum /content/drive/MyDrive/
```

```
trainer.push_to_hub()
```

```
from peft import AutoPeftModelForCausalLM
from transformers import GenerationConfig
from transformers import AutoTokenizer
import torch
tokenizer = AutoTokenizer.from_pretrained("/content/mistral-finetuned-samsum")
```


```
inputs = tokenizer("""
###Human: Summarize this following dialogue: Sunny: I'm at the railway station in Chennai Karthik: No problems so far? Sunny
###Assistant: """, return_tensors="pt").to("cuda")
```

```
model = AutoPeftModelForCausalLM.from_pretrained(
    "/content/mistral-finetuned-samsum",
    low_cpu_mem_usage=True,
    return_dict=True,
    torch_dtype=torch.float16,
    device_map="cuda")
```

 Some weights of the model checkpoint at TheBloke/Mistral-7B-Instruct-v0.1-GPTQ were not used when initializing MistralFo  
- This IS expected if you are initializing MistralForCausalLM from the checkpoint of a model trained on another task or  
- This IS NOT expected if you are initializing MistralForCausalLM from the checkpoint of a model that you expect to be e

```
generation_config = GenerationConfig(
    do_sample=True,
    top_k=1,
    temperature=0.1,
    max_new_tokens=25,
    pad_token_id=tokenizer.eos_token_id
)
```

```
import time
st_time = time.time()
outputs = model.generate(**inputs, generation_config=generation_config)
print(tokenizer.decode(outputs[0], skip_special_tokens=True))
print(time.time()-st_time)
```

 ###Human: Summarize this following dialogue: Vasanth: I'm at the railway station in Chennai Karthik: No problems so far?  
###Assistant: Vasanth is at the railway station in Chennai. Everything is going smoothly. He will meet Karthik soon.  
29.117424964904785

Start coding or [generate](#) with AI.



