```
%%capture
import os
if "COLAB_" not in "".join(os.environ.keys()):
    !pip install unsloth
else:
    !pip install --no-deps bitsandbytes accelerate xformers==0.0.29.post3 peft trl=
    !pip install sentencepiece protobuf "datasets>=3.4.1" huggingface_hub hf_trans
    !pip install --no-deps unsloth
```

```
from unsloth import FastLanguageModel, is_bfloat16_supported
from unsloth.chat_templates import get_chat_template
from datasets import load_dataset
from trl import SFTTrainer
from transformers import TrainingArguments, DataCollatorForSeq2Seq, Trainer
import torch
max_seq_length = 2048
dtype = None
load_in_4bit = True

model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "unsloth/Llama-3.2-3B-Instruct",
    max_seq_length = max_seq_length,
    dtype = dtype,
    load_in_4bit = load_in_4bit,
)
```

```
🦥 Unsloth: Will patch your computer to enable 2x faster free finetuning.
🦥 Unsloth Zoo will now patch everything to make training faster!
==((====))==  Unsloth 2025.5.6: Fast Llama patching. Transformers: 4.51.3.
   \\   /|    Tesla T4. Num GPUs = 1. Max memory: 14.741 GB. Platform: Linux.
O^O/ \_/ \    Torch: 2.6.0+cu124. CUDA: 7.5. CUDA Toolkit: 12.4. Triton: 3.2.
\        /    Bfloat16 = FALSE. FA [Xformers = 0.0.29.post3. FA2 = False]
 "-____-"     Free license: http://github.com/unslothai/unsloth
Unsloth: Fast downloading is enabled - ignore downloading bars which are red
```

```
model.safetensors: 100%                2.35G/2.35G [00:16<00:00, 377MB/s]
generation_config.json: 100%           234/234 [00:00<00:00, 24.9kB/s]
tokenizer_config.json: 100%            54.7k/54.7k [00:00<00:00, 5.64MB/s]
```

```
model = FastLanguageModel.get_peft_model(
    model,
    r = 16,
    target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
                      "gate_proj", "up_proj", "down_proj",]
```

```
                        "gate_proj", "up_proj", "down_proj",],
    lora_alpha = 16,
    lora_dropout = 0,
    bias = "none",
    use_gradient_checkpointing = "unsloth",
    random_state = 3407,
    use_rslora = False,
    loftq_config = None,
)
```

```
Unsloth 2025.5.6 patched 28 layers with 28 QKV layers, 28 O layers and 28 MLP
```

```
from datasets import load_dataset
dataset = load_dataset(
    "parquet",
    data_files={"train": "4701_cleaned.parquet"},
    split="train"
)
```

```
Generating train split:     1879/0 [00:00<00:00, 25340.74 examples/s]
```

```
sample = dataset.select(range(min(5, len(dataset))))
example_convo = sample['conversations'][0]
print(f"Type of conversation: {type(example_convo)}")
print(f"Example conversation structure: {example_convo}")
```

```
Type of conversation: <class 'str'>
Example conversation structure: [{'from': 'human', 'value': 'SYSTEM: Respond w
 {'from': 'gpt', 'value': "Hello world! I'm Lucy, a Minecraft bot. !stats"}
 {'from': 'human', 'value': 'SYSTEM: STATS\n- Position: x: -25.50, y: 66.00, z
 {'from': 'gpt', 'value': "Hey there! I'm Lucy, hanging out in the plains. !em
```

```
dataset[5]["conversations"]
```

```
'[{\'from\': \'human\', \'value\': \'SYSTEM: Respond with hello world and you
r name\'}\n {\'from\': \'gpt\', \'value\': "Hey there! I\'m Lucy, a Minecraft
bot. !stats"}\n {\'from\': \'human\', \'value\': \'SYSTEM: STATS\\n- Position
: x: -25.50, y: 66.00, z: -75.50\\n\\n- Health: 20 / 20\\n- Hunger: 20 / 20\
\n- Current Action: Idle\\n- Nearby Human Players: MrBigFinger\\nAgent Modes:
\\n- self_preservation(ON)\\n- unstuck(ON)\\n- cowardice(OFF)\\n- self_defens
e(ON)\\n- hunting(ON)\\n- item_collecting(ON)\\n- torch_placing(ON)\\n- elbow
_room(ON)\\n- idle_staring(ON)\\n- \\nMrBigFinger: So Lucy, can you fetch me
some wood to start off our adventure in Minecraft?\'}\n {\'from\': \'gpt\'
```

```
tokenizer = get_chat_template(
    tokenizer,
    chat_template="chatml",
    mapping={"role": "from", "content": "value", "user": "human", "assistant": "gpt
    map_eos_token=True,
)
```

```
,

# Chain-of-Thought augmentation
def format_chain_of_thought(example):
    new_convo = []
    for message in example["conversations"]:
        if message[0] == "gpt":
            reasoning = f"Let's think step-by-step to solve the instruction: '{exam
            new_convo.append({"from": "gpt", "value": reasoning})
        new_convo.append(message)
    return {"conversations": new_convo}

def formatting_prompts_func(examples):
    convos = examples["conversations"]
    texts = [tokenizer.apply_chat_template(convo, tokenize=False, add_generation_pr
    return {"text": texts}

def tokenize_function(examples):
    return tokenizer(
        examples["text"],
        padding="max_length",
        truncation=True,
        max_length=max_seq_length,
    )

# Load and process dataset
dataset = dataset.map(format_chain_of_thought)
dataset = dataset.map(formatting_prompts_func, batched=True)
dataset = dataset.map(tokenize_function, batched=True, remove_columns=dataset.colum
```

```
Unsloth: Will map <|im_end|> to EOS = <|eot_id|>.
Map: 100%                                    1879/1879 [00:03<00:00, 831.81 examples/s]
Map: 100%                                    11647/11647 [00:04<00:00, 3100.20 examples/
                                             s]
```

```
from trl import SFTTrainer
from transformers import TrainingArguments, DataCollatorForSeq2Seq
from unsloth import is_bfloat16_supported

trainer = SFTTrainer(
    model = model,
    tokenizer = tokenizer,
    train_dataset = dataset,
    dataset_text_field = "text",
    max_seq_length = max_seq_length,
    dataset_num_proc = 2,
    packing = False,
    args = TrainingArguments(
        per_device_train_batch_size = 16,
        per_device_train_batch_size = 16,
        gradient_accumulation_steps = 1,
        warmup_steps = 30,
        num_train_epochs = 1, # Set this for 1 full training run.
        max_steps = 60,
        learning_rate = 2e-4,
        fp16 = not is_bfloat16_supported(),
        bf16 = is_bfloat16_supported(),
        logging_steps = 1,
        optim = "adamw_8bit",
        weight_decay = 0.01,
        lr_scheduler_type = "linear",
        seed = 3407,
        output_dir = "outputs",
        gradient_checkpointing = True,
    ),
)
```

```
import torch
torch.cuda.empty_cache()

gpu_stats = torch.cuda.get_device_properties(0)
start_gpu_memory = round(torch.cuda.max_memory_reserved() / 1024 / 1024 / 1024, 3
max_memory = round(gpu_stats.total_memory / 1024 / 1024 / 1024, 3)
print(f"GPU = {gpu_stats.name}. Max memory = {max_memory} GB.")
print(f"{start_gpu_memory} GB of memory reserved.")
```

```
GPU = Tesla T4. Max memory = 14.741 GB.
3.441 GB of memory reserved.
```

```
trainer_stats = trainer.train()
```

```
==((====))==  Unsloth - 2x faster free finetuning | Num GPUs used = 1
   \\   /|    Num examples = 11,647 | Num Epochs = 1 | Total steps = 60
O^O/ \_/ \    Batch size per device = 16 | Gradient accumulation steps = 1
\        /    Data Parallel GPUs = 1 | Total batch size (16 x 1 x 1) = 16
 "-____-"     Trainable parameters = 24,313,856/3,000,000,000 (0.81% trained)
wandb: WARNING The `run_name` is currently set to the same value as `Training
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: http
wandb: You can find your API key in your browser here: https://wandb.ai/autho
wandb: Paste an API key from your profile and hit enter: ··········
wandb: WARNING If you're specifying your api key in code, ensure this code is
wandb: WARNING Consider setting the WANDB_API_KEY environment variable, or ru
wandb: No netrc file found, creating one.
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
wandb: Currently logged in as: eh588 (eh588-cornell-university) to https://ap
Tracking run with wandb version 0.19.11
Run data is saved locally in /content/wandb/run-20250517_184203-1g1ix9n7
Syncing run outputs to Weights & Biases (docs)
View project at https://wandb.ai/eh588-cornell-university/huggingface
View run at https://wandb.ai/eh588-cornell-university/huggingface/runs/1g1ix9n7
```

[60/60 44:09, Epoch 0/1]

Step	Training Loss
1	1.978600
2	1.941100
3	1.857300
4	1.857500
5	1.858000
6	1.623200
7	1.529200
8	1.648400
9	1.792400
10	1.733600
11	1.715700
12	1.645700
13	1.706800
14	1.591300
15	1.589900
16	1.534900
17	1.607000
18	1.552600
19	1.484100
20	1.526100
21	1.395400
22	1.401500
23	1.319400
24	1.319000
25	1.281900
26	1.160300
27	1.050000
28	1.073600
29	0.965700

30	0.844800
31	0.807500
32	0.734000
33	0.784700
34	0.850700
35	0.608500
36	0.690800
37	0.524600
38	0.493200
39	0.663600
40	0.673100
41	0.467200
42	0.547600
43	0.428300
44	0.507100
45	0.472800
46	0.459600
47	0.483500
48	0.487100
49	0.379500
50	0.579300
51	0.465400
52	0.436300
53	0.629000
54	0.570100
55	0.518600
56	0.557700
57	0.469800
58	0.428300
59	0.409800
60	0.364200

```
    60          0.364200
```

```
Unsloth: Will smartly offload gradients to save VRAM!
```

```
model.save_pretrained("outputs/lucy/model")
tokenizer.save_pretrained("outputs/lucy/tokenizer")
```

```
('outputs/lucy/tokenizer/tokenizer_config.json',
 'outputs/lucy/tokenizer/special_tokens_map.json',
 'outputs/lucy/tokenizer/tokenizer.json')
```