```
%%capture
# Installs Unsloth, Xformers (Flash Attention) and all other packages!
!pip install unsloth
# Get latest Unsloth
!pip install --upgrade --no-deps "unsloth[colab-new] @ git+https://github.com/unslothai/unsloth.git"
```

```
from unsloth import FastLanguageModel
import torch
max_seq_length = 2048 # Choose any! We auto support RoPE Scaling internally!
dtype = None # None for auto detection. Float16 for Tesla T4, V100, Bfloat16 for Ampere+
load_in_4bit = True # Use 4bit quantization to reduce memory usage. Can be False.

# 4bit pre quantized models we support for 4x faster downloading + no OOMs.
fourbit_models = [
    "unsloth/mistral-7b-v0.3-bnb-4bit",      # New Mistral v3 2x faster!
    "unsloth/mistral-7b-instruct-v0.3-bnb-4bit",
    "unsloth/llama-3-8b-bnb-4bit",           # Llama-3 15 trillion tokens model 2x faster!
    "unsloth/llama-3-8b-Instruct-bnb-4bit",
    "unsloth/llama-3-70b-bnb-4bit",
    "unsloth/Phi-3-mini-4k-instruct",        # Phi-3 2x faster!
    "unsloth/Phi-3-medium-4k-instruct",
    "unsloth/mistral-7b-bnb-4bit",
    "unsloth/gemma-7b-bnb-4bit",             # Gemma 2.2x faster!
] # More models at https://huggingface.co/unsloth

model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "unsloth/llama-3-8b-bnb-4bit",
    max_seq_length = max_seq_length,
    dtype = dtype,
    load_in_4bit = load_in_4bit,
    # token = "hf_...", # use one if using gated models like meta-llama/Llama-2-7b-hf
)
```

```
🦥 Unsloth: Will patch your computer to enable 2x faster free finetuning.
🦥 Unsloth Zoo will now patch everything to make training faster!
==((====))==  Unsloth 2024.12.4: Fast Llama patching. Transformers:4.46.3.
   \\   /|    GPU: Tesla T4. Max memory: 14.748 GB. Platform: Linux.
O^O/ \_/ \    Torch: 2.5.1+cu121. CUDA: 7.5. CUDA Toolkit: 12.1. Triton: 3.1.0
\        /    Bfloat16 = FALSE. FA [Xformers = 0.0.28.post3. FA2 = False]
 "-____-"     Free Apache license: http://github.com/unslothai/unsloth
Unsloth: Fast downloading is enabled - ignore downloading bars which are red colored!
model.safetensors: 100%                                  5.70G/5.70G [00:48<00:00, 63.9MB/s]
generation_config.json: 100%                               198/198 [00:00<00:00, 13.1kB/s]
tokenizer_config.json: 100%                              50.6k/50.6k [00:00<00:00, 2.98MB/s]
tokenizer.json: 100%                                   9.09M/9.09M [00:00<00:00, 10.2MB/s]
special_tokens_map.json: 100%                              350/350 [00:00<00:00, 26.7kB/s]
```

```
model = FastLanguageModel.get_peft_model(
    model,
    r = 16, # Choose any number > 0 ! Suggested 8, 16, 32, 64, 128
    target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
                      "gate_proj", "up_proj", "down_proj",],
    lora_alpha = 16,
    lora_dropout = 0, # Supports any, but = 0 is optimized
    bias = "none",    # Supports any, but = "none" is optimized
    # [NEW] "unsloth" uses 30% less VRAM, fits 2x larger batch sizes!
    use_gradient_checkpointing = "unsloth", # True or "unsloth" for very long context
    random_state = 3407,
    use_rslora = False,  # We support rank stabilized LoRA
    loftq_config = None, # And LoftQ
)
```

```
Unsloth 2024.12.4 patched 32 layers with 32 QKV layers, 32 O layers and 32 MLP layers.
```

Dataset Preparation :

Loading the question_with_options along with the predicted_label and the gpt_reasoning information as a csv.

```
## DataSet Preparation
```

```
from datasets import load_dataset
dataset = load_dataset(
    "csv",
    data_files = "dataset.csv",
    split = "train",
)
print(dataset.column_names)
print(dataset[0])
```

```
Generating train split:        542/0 [00:00<00:00, 5292.66 examples/s]
['questions_with_options', 'predicted_label', 'gpt_reasoning']
{'questions_with_options': "A 58-year-old man comes to the physician for a 3-month history of progressive shortness of breath on exertion and tiredness throu
```

Converting the dataset into required format:

```
from unsloth import to_sharegpt

# Define a post-processing function to combine 'predicted_label' and 'reasoning'
def combine_columns(example):
    example["predicted_label_and_reasoning"] = f"Label: {example['predicted_label']}, Reasoning: {example['gpt_reasoning']}"
    return example

# Apply the function to the dataset
dataset = dataset.map(combine_columns)

# Verify the formatted dataset
print(dataset[0])


# Merge "predicted_label" and "reasoning" into the response in the merged_prompt
dataset = to_sharegpt(
    dataset,
    merged_prompt="{questions_with_options}",
    output_column_name="predicted_label_and_reasoning",  # Combine predicted_label and reasoning
)


print(dataset[0])
```

```
Map: 100%                                                542/542 [00:00<00:00, 4025.13 examples/s]
{'questions_with_options': "A 58-year-old man comes to the physician for a 3-month history of progressive shortness of breath on exertion and tiredness throu
Merging columns: 100%                                    542/542 [00:00<00:00, 6516.68 examples/s]
Converting to ShareGPT: 100%                             542/542 [00:00<00:00, 7984.35 examples/s]
{'conversations': [{'from': 'human', 'value': '("A 58-year-old man comes to the physician for a 3-month history of progressive shortness of breath on exertio
```

Standardize share_gpt

```
from unsloth import standardize_sharegpt
dataset = standardize_sharegpt(dataset)
```

```
Standardizing format: 100%                               542/542 [00:00<00:00, 6887.76 examples/s]
```

Chat Template

```
chat_template = """Below is a question with options. Provide the predicted label and reasoning.

>>> Question:
{INPUT}

>>> Answer:
{OUTPUT}"""

from unsloth import apply_chat_template
dataset = apply_chat_template(
    dataset,
    tokenizer = tokenizer,
    chat_template = chat_template,
    # default_system_message = "You are a helpful assistant", << [OPTIONAL]
)
```

```
Unsloth: We automatically added an EOS token to stop endless generations.
Map: 100%                                                542/542 [00:00<00:00, 7608.60 examples/s]
```

Training the model

Defining the training parametes and using Huggingface TRL's `SFTTrainer`!

```
from trl import SFTTrainer
from transformers import TrainingArguments
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, # Can make training 5x faster for short sequences.
    args = TrainingArguments(
        per_device_train_batch_size = 2,
        gradient_accumulation_steps = 4,
        warmup_steps = 5,
        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",
    ),
)
```

```
Map (num_proc=2): 100%  542/542 [00:04<00:00, 158.28 examples/s]
max_steps is given, it will override any value given in num_train_epochs
```

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

```
==((====))==  Unsloth - 2x faster free finetuning | Num GPUs = 1
   \\   /|    Num examples = 542 | Num Epochs = 1
O^O/ \_/ \    Batch size per device = 2 | Gradient Accumulation steps = 4
\        /    Total batch size = 8 | Total steps = 60
 "-____-"     Number of trainable parameters = 41,943,040
wandb: WARNING The `run_name` is currently set to the same value as `TrainingArguments.output_dir`. If this was not intended, please specify a different ru
wandb: Using wandb-core as the SDK backend.  Please refer to https://wandb.me/wandb-core for more information.
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter, or press ctrl+c to quit: ··········
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
Tracking run with wandb version 0.18.7
Run data is saved locally in /content/wandb/run-20241213_092900-zxi4xqgs
Syncing run outputs to Weights & Biases (docs)
View project at https://wandb.ai/rahulsaxena-umass-amherst/huggingface
View run at https://wandb.ai/rahulsaxena-umass-amherst/huggingface/runs/zxi4xqgs
```

[60/60 12:08, Epoch 0/1]

Step	Training Loss
1	1.718500
2	1.767800
3	1.742200
4	1.620100
5	1.616900
6	1.608900
7	1.410200
8	1.348800
9	1.182400
10	1.104700
11	1.034200
12	1.109200
13	1.040300
14	0.956100
15	0.922800
16	0.990600
17	0.982100
18	0.974300
19	0.927400
20	0.924700
21	0.966700
22	0.941800
23	0.930900
24	0.939500
25	0.915100
26	0.867400
27	0.861100
28	0.908700
29	0.926700
30	0.813300
31	0.934900
32	0.954500
33	0.897600
34	0.853900
35	0.878900
36	0.840100
37	0.920500
38	0.837700
39	0.859000
40	0.798700
41	0.848900
42	0.794800
43	0.830000
44	0.873200
45	0.896900
46	0.837600
47	0.854100
48	0.872800