Installation

```
!pip install --no-deps bitsandbytes accelerate xformers==0.0.29.post3 peft trl==0.15.2 triton cut_cross_entropy unsloth_zoo -
!pip install sentencepiece protobuf "datasets>=3.4.1" huggingface_hub hf_transfer -q
!pip install --no-deps unsloth -q
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 43.4/43.4 MB 12.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 318.9/318.9 kB 9.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 76.1/76.1 MB 7.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 146.6/146.6 kB 8.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 491.5/491.5 kB 17.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 193.6/193.6 kB 19.3 MB/s eta 0:00:00
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviou
unsloth-zoo 2025.5.8 requires msgspec, which is not installed.
unsloth-zoo 2025.5.8 requires tyro, which is not installed.
unsloth-zoo 2025.5.8 requires protobuf<4.0.0, but you have protobuf 5.29.4 which is incompatible.
gcsfs 2025.3.2 requires fsspec==2025.3.2, but you have fsspec 2025.3.0 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cublas-cu12==12.4.5.8; platform_system == "Linux" and platform_machine == "x86_64", bu
torch 2.6.0+cu124 requires nvidia-cuda-cupti-cu12==12.4.127; platform_system == "Linux" and platform_machine == "x86_64"
torch 2.6.0+cu124 requires nvidia-cuda-nvrtc-cu12==12.4.127; platform_system == "Linux" and platform_machine == "x86_64"
torch 2.6.0+cu124 requires nvidia-cuda-runtime-cu12==12.4.127; platform_system == "Linux" and platform_machine == "x86_6
torch 2.6.0+cu124 requires nvidia-cudnn-cu12==9.1.0.70; platform_system == "Linux" and platform_machine == "x86_64", but
torch 2.6.0+cu124 requires nvidia-cufft-cu12==11.2.1.3; platform_system == "Linux" and platform_machine == "x86_64", but
torch 2.6.0+cu124 requires nvidia-curand-cu12==10.3.5.147; platform_system == "Linux" and platform_machine == "x86_64",
torch 2.6.0+cu124 requires nvidia-cusolver-cu12==11.6.1.9; platform_system == "Linux" and platform_machine == "x86_64",
torch 2.6.0+cu124 requires nvidia-cusparse-cu12==12.3.1.170; platform_system == "Linux" and platform_machine == "x86_64"
torch 2.6.0+cu124 requires nvidia-nvjitlink-cu12==12.4.127; platform_system == "Linux" and platform_machine == "x86_64",
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 47.1/47.1 kB 3.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 265.7/265.7 kB 12.2 MB/s eta 0:00:00
```

Initializing Unsloth

```
from unsloth import FastLanguageModel
import torch
```

```
🦥 Unsloth: Will patch your computer to enable 2x faster free finetuning.
🦥 Unsloth Zoo will now patch everything to make training faster!
```

```
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,
    # token = "hf_...", # use one if using gated models from HuggingFace
)
```

```
==((====))==  Unsloth 2025.5.7: 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: 3.2.0
\        /    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 colored!
model.safetensors: 100%                           2.35G/2.35G [00:19<00:00, 538MB/s]
generation_config.json: 100%                      234/234 [00:00<00:00, 5.74kB/s]
tokenizer_config.json: 100%                       54.7k/54.7k [00:00<00:00, 1.90MB/s]
tokenizer.json: 100%                              17.2M/17.2M [00:00<00:00, 66.1MB/s]
special_tokens_map.json: 100%                     454/454 [00:00<00:00, 37.4kB/s]
```

We now add LoRA adapters so we only need to update 1 to 10% of all parameters!

```
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
    use_gradient_checkpointing = "unsloth", # True or "unsloth" for very long context
    random_state = 3407,
```

```
    use_rslora = False,  # Unsloth support rank stabilized LoRA
    loftq_config = None, # And LoftQ
)
```

```
Unsloth 2025.5.7 patched 28 layers with 28 QKV layers, 28 O layers and 28 MLP layers.
```

Data Prep

We now use the `Llama-3.1` format for conversation style finetunes. We use Maxime Labonne's FineTome-100k dataset in ShareGPT style. But we convert it to HuggingFace's normal multiturn format `("role", "content")` instead of `("from", "value")` / Llama-3 renders multi turn conversations like below:

```
<|begin_of_text|><|start_header_id|>user<|end_header_id|>

Hello!<|eot_id|><|start_header_id|>assistant<|end_header_id|>

Hey there! How are you?<|eot_id|><|start_header_id|>user<|end_header_id|>

I'm great thanks!<|eot_id|>
```

We use `get_chat_template` function to get the correct chat template. We support `zephyr, chatml, mistral, llama, alpaca, vicuna, vicuna_old, phi3, llama3` and more.

```
from unsloth.chat_templates import get_chat_template

tokenizer = get_chat_template(
    tokenizer,
    chat_template = "llama-3.1",
)

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

from datasets import load_dataset
dataset = load_dataset("mlabonne/FineTome-100k", split = "train")
```

```
README.md: 100%  982/982 [00:00<00:00, 101kB/s]
train-00000-of-00001.parquet: 100%  117M/117M [00:00<00:00, 204MB/s]
Generating train split: 100%  100000/100000 [00:01<00:00, 63759.87 examples/s]
```

We now use `standardize_sharegpt` to convert ShareGPT style datasets into HuggingFace's generic format. This changes the dataset from looking like:

```
{"from": "system", "value": "You are an assistant"}
{"from": "human", "value": "What is 2+2?"}
{"from": "gpt", "value": "It's 4."}
```

to

```
{"role": "system", "content": "You are an assistant"}
{"role": "user", "content": "What is 2+2?"}
{"role": "assistant", "content": "It's 4."}
```

```
from unsloth.chat_templates import standardize_sharegpt
dataset = standardize_sharegpt(dataset)
dataset = dataset.map(formatting_prompts_func, batched = True,)
```

```
Unsloth: Standardizing formats (num_proc=2): 100%  100000/100000 [00:06<00:00, 19848.37 examples/s]
Map: 100%  100000/100000 [00:10<00:00, 10922.25 examples/s]
```

We look at how the conversations are structured for item 5:

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

```
[{'content': 'How do astronomers determine the original wavelength of light emitted by a celestial body at rest, which is necessary for measuring its speed using the Doppler effect?',
  'role': 'user'},
 {'content': 'Astronomers make use of the unique spectral fingerprints of elements found in stars. These elements emit and absorb light at specific, known wavelengths, forming an absorption spectrum. By analyzing the light received from distant stars and comparing it to the laboratory-measured spectra of these elements, astronomers can identify the shifts in these wavelengths due to the Doppler effect. The observed shift tells them the extent to which the light has been redshifted or blueshifted, thereby allowing them to calculate the speed of the star along the line of sight relative to Earth.',
  'role': 'assistant'}]
```

```
dataset[5]["text"]
```

```
'<|begin_of_text|><|start_header_id|>system<|end_header_id|>\n\nCutting Knowledge Date: December 2023\nToday Date: 26 July 2024\n\n<|eot_id|><|start_header_id|>user<|end_header_id|>\n\nHow do astronomers determine the original wavelength of light emitted by a celestial body at rest, which is necessary for measuring its speed using the Doppler effect?<|eot_id|><|start_header_id|>assistant<|end_header_id|>\n\nAstronomers make use of the unique spectral fingerprints of elements found in stars. These elements emit and absorb light at specific, known wavelengths, forming an absorption spectrum. By analyzing the light received from distant stars and comparing it to the laboratory-measured spectra of these elements, astronomers can identify the shifts in these wavelengths due to the Doppler effect. The observed shift tells them the extent to which the light has been redshifted or blueshifted, thereby allowing them to calculate the speed of the s
```

Train the model

Now let's use Huggingface TRL's `SFTTrainer`! More docs here: TRL SFT docs. We do 60 steps to speed things up, but you can set `num_train_epochs=1` for a full run, and turn off `max_steps=None`. We also support TRL's `DPOTrainer`!

```
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,
    data_collator = DataCollatorForSeq2Seq(tokenizer = tokenizer),
    dataset_num_proc = 2,
    packing = False,
    args = TrainingArguments(
        per_device_train_batch_size = 2,
        gradient_accumulation_steps = 4,
        warmup_steps = 5,
        # 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",
        report_to = "none", # Use this for WandB etc
    ),
)
```

```
Unsloth: Tokenizing ["text"] (num_proc=2): 100%  100000/100000 [02:46<00:00, 599.80 examples/s]
```

We also use Unsloth's `train_on_completions` method to only train on the assistant outputs and ignore the loss on the user's inputs.

```
from unsloth.chat_templates import train_on_responses_only
trainer = train_on_responses_only(
    trainer,
    instruction_part = "<|start_header_id|>user<|end_header_id|>\n\n",
    response_part = "<|start_header_id|>assistant<|end_header_id|>\n\n",
)
```

```
Map (num_proc=2): 100%  100000/100000 [00:59<00:00, 1342.99 examples/s]
```

We verify masking is actually done:

```
tokenizer.decode(trainer.train_dataset[5]["input_ids"])
```

```
'<|begin_of_text|><|begin_of_text|><|start_header_id|>system<|end_header_id|>\n\nCutting Knowledge Date: December 2023
\nToday Date: 26 July 2024\n\n<|eot_id|><|start_header_id|>user<|end_header_id|>\n\nHow do astronomers determine the or
iginal wavelength of light emitted by a celestial body at rest, which is necessary for measuring its speed using the Do
ppler effect?<|eot_id|><|start_header_id|>assistant<|end_header_id|>\n\nAstronomers make use of the unique spectral fin
gerprints of elements found in stars. These elements emit and absorb light at specific, known wavelengths, forming an a
bsorption spectrum. By analyzing the light received from distant stars and comparing it to the laboratory-measured spec
tra of these elements, astronomers can identify the shifts in these wavelengths due to the Doppler effect. The observed
shift tells them the extent to which the light has been redshifted or blueshifted, thereby allowing them to calculate t
```

```
space = tokenizer(" ", add_special_tokens = False).input_ids[0]
tokenizer.decode([space if x == -100 else x for x in trainer.train_dataset[5]["labels"]])
```

```
'                                                                Astronomers make use of the unique spectral fingerpr
ints of elements found in stars. These elements emit and absorb light at specific, known wavelengths, forming an absorp
tion spectrum. By analyzing the light received from distant stars and comparing it to the laboratory-measured spectra o
f these elements, astronomers can identify the shifts in these wavelengths due to the Doppler effect. The observed shif
t tells them the extent to which the light has been redshifted or blueshifted, thereby allowing them to calculate the s
peed of the star along the line of sight relative to Earth.<|eot_id|>'
```

We can see the System and Instruction prompts are successfully masked!

Show current memory stats

```
# @title Show current memory stats
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 = 100,000 | Num Epochs = 1 | Total steps = 60
O^O/ \_/ \    Batch size per device = 2 | Gradient accumulation steps = 4
\        /    Data Parallel GPUs = 1 | Total batch size (2 x 4 x 1) = 8
 "-____-"     Trainable parameters = 24,313,856/3,000,000,000 (0.81% trained)
Unsloth: Will smartly offload gradients to save VRAM!
```

[60/60 07:50, Epoch 0/1]

Step	Training Loss
1	0.774700
2	0.839000
3	1.075700
4	0.891800
5	0.757500
6	0.937300
7	0.619200
8	0.998500
9	0.859500
10	0.761400
11	0.884100
12	1.094100
13	0.954100
14	0.641500
15	0.877200
16	0.639200
17	1.003100
18	0.827200
19	0.769400
20	0.934500
21	0.902700
22	0.857000
23	1.036300
24	0.884700
25	0.641800
26	0.827200
27	0.829100
28	0.787700
29	1.086600
30	1.036000
31	0.707900
32	0.541800
33	0.655300
34	0.580300
35	0.762300
36	1.003500
37	0.901100
38	0.717300
39	0.779400
40	1.000200
41	0.745900
42	1.007900
43	0.771100
44	0.815000
45	0.762800

46	0.863600
47	0.787500
48	0.652700
49	1.016900
50	1.032300
51	0.457300
52	0.907800
53	1.317700
54	0.708300
55	1.061400
56	1.125500
57	0.725200
58	0.836600
59	0.756700
60	0.924600

Show final memory and time stats

Show code

```
494.9943 seconds used for training.
8.25 minutes used for training.
Peak reserved memory = 4.131 GB.
Peak reserved memory for training = 0.69 GB.
Peak reserved memory % of max memory = 28.024 %.
Peak reserved memory for training % of max memory = 4.681 %.
```

Inference

Let's run the model! You can change the instruction and input - leave the output blank!

We use `min_p = 0.1` and `temperature = 1.5`.

```
from unsloth.chat_templates import get_chat_template

tokenizer = get_chat_template(
    tokenizer,
    chat_template = "llama-3.1",
)
FastLanguageModel.for_inference(model)

messages = [
    {"role": "user", "content": "Continue the fibonnaci sequence: 1, 1, 2, 3, 5, 8,"},
]
inputs = tokenizer.apply_chat_template(
    messages,
    tokenize = True,
    add_generation_prompt = True,
    return_tensors = "pt",
).to("cuda")

outputs = model.generate(input_ids = inputs, max_new_tokens = 64, use_cache = True,
                         temperature = 1.5, min_p = 0.1)
tokenizer.batch_decode(outputs)
```

```
The attention mask is not set and cannot be inferred from input because pad token is same as eos token. As a consequence
['<|begin_of_text|><|start_header_id|>system<|end_header_id|>\n\nCutting Knowledge Date: December 2023\nToday Date: 26
July 2024\n\n<|eot_id|><|start_header_id|>user<|end_header_id|>\n\nContinue the fibonnaci sequence: 1, 1, 2, 3, 5, 8,
<|eot_id|><|start_header_id|>assistant<|end_header_id|>\n\nThe Fibonacci sequence is a series of numbers in which each
number is the sum of the two preceding numbers. The sequence you provided starts with 1, 1, 2, 3, 5, and 8. Here are
the next three numbers in the sequence:\n9, 14, 23<|eot_id|>']
```

Saving, loading finetuned models

To save the final model as LoRA adapters, either use Huggingface's `push_to_hub` for an online save or `save_pretrained` for a local save.

[NOTE] This ONLY saves the LoRA adapters, and not the full model. To save to 16bit or GGUF, scroll down!

```
model.save_pretrained("lora_model")  # Local saving
tokenizer.save_pretrained("lora_model")
# model.push_to_hub("your_name/lora_model", token = "...") # Online saving
# tokenizer.push_to_hub("your_name/lora_model", token = "...") # Online saving
```

```
('lora_model/tokenizer_config.json',
 'lora_model/special_tokens_map.json',
 'lora_model/tokenizer.json')
```

```
if True:
    from unsloth import FastLanguageModel
    model, tokenizer = FastLanguageModel.from_pretrained(
        model_name = "lora_model", # YOUR MODEL YOU USED FOR TRAINING
        max_seq_length = max_seq_length,
        dtype = dtype,
        load_in_4bit = load_in_4bit,
```