To run this, press "*Runtime*" and press "*Run all*" on a **free** Tesla T4 Google Colab instance!

unsloth Join our Discord Documentation Join Discord if you need help + ⭐ *Star us on Github* ⭐

To install Unsloth your local device, follow our guide. This notebook is licensed LGPL-3.0.

You will learn how to do data prep, how to train, how to run the model, & how to save it

News

Introducing FP8 precision training for faster RL inference. Read Blog.

Unsloth's Docker image is here! Start training with no setup & environment issues. Read our Guide.

gpt-oss RL is now supported with the fastest inference & lowest VRAM. Try our new notebook which creates kernels!

Introducing Vision and Standby for RL! Train Qwen, Gemma etc. VLMs with GSPO - even faster with less VRAM.

Visit our docs for all our model uploads and notebooks.

Installation

```
%%capture
import os, re
if "COLAB_" not in "".join(os.environ.keys()):
    !pip install unsloth
else:
    # Do this only in Colab notebooks! Otherwise use pip install unsloth
    import torch; v = re.match(r"[0-9]{1,}\.[0-9]{1,}", str(torch.__version__)).group(0)
    xformers = "xformers==" + ("0.0.33.post1" if v=="2.9" else "0.0.32.post2" if v=="2.8" else "0.0.29.post3")
    !pip install --no-deps bitsandbytes accelerate {xformers} peft trl triton cut_cross_entropy unsloth_zoo
    !pip install sentencepiece protobuf "datasets==4.3.0" "huggingface_hub>=0.34.0" hf_transfer
    !pip install --no-deps unsloth
#!pip install transformers==4.57.3
#!pip install --no-deps trl==0.22.2
```

Unsloth

```python
from unsloth import FastLanguageModel
import torch
max_seq_length = 4096 # Can choose any sequence length!
fourbit_models = [
    # 4bit Gemma 3 dynamic quants for superior accuracy and low memory use
    "unsloth/gemma-3-270m-it-unsloth-bnb-4bit",
    "unsloth/gemma-3-1b-it-unsloth-bnb-4bit",
    "unsloth/gemma-3-4b-it-unsloth-bnb-4bit",
    "unsloth/gemma-3-12b-it-unsloth-bnb-4bit",
    "unsloth/gemma-3-27b-it-unsloth-bnb-4bit",
    # Function Gemma models
    "unsloth/functiongemma-270m-it",
    "unsloth/functiongemma-270m-it-unsloth-bnb-4bit",
    "unsloth/functiongemma-270m-it-bnb-4bit",
] # More models at https://huggingface.co/unsloth

model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "unsloth/functiongemma-270m-it",
    # model_name = "gemma-3-270m-it-unsloth-bnb-4bit",
    max_seq_length = max_seq_length, # Choose any for long context!
    load_in_4bit = False,  # 4 bit quantization to reduce memory
    load_in_8bit = False, # [NEW!] A bit more accurate, uses 2x memory
    load_in_16bit = True, # [NEW!] Enables 16bit LoRA
    full_finetuning = False, # [NEW!] We have full finetuning now!
    # token = "hf_...", # use one if using gated models
)
```

```
🦥 Unsloth: Will patch your computer to enable 2x faster free finetuning.
🦥 Unsloth Zoo will now patch everything to make training faster!
==((====))==  Unsloth 2026.1.3: Fast Gemma3 patching. Transformers: 4.57.6. vLLM: 0.13.0.
   \\   /|    NVIDIA GeForce RTX 4090. Num GPUs = 1. Max memory: 47.371 GB. Platform: Linux.
O^O/ \_/ \    Torch: 2.9.0+cu128. CUDA: 8.9. CUDA Toolkit: 12.8. Triton: 3.5.0
\        /    Bfloat16 = TRUE. FA [Xformers = 0.0.33.post1. FA2 = False]
 "-____-"     Free license: http://github.com/unslothai/unsloth
Unsloth: Fast downloading is enabled - ignore downloading bars which are red colored!
Unsloth: Gemma3 does not support SDPA - switching to fast eager.
```

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

```python
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: Making `model.base_model.model.model` require gradients
```

Data Prep

We now use Google's Mobile Actions dataset, which contains a list of tools a phone could call, like making a calendar invite, setting an alarm etc.

```
from datasets import load_dataset
dataset = load_dataset("google/mobile-actions", split = "train")
```

Let's take a look at the first row for messages and the tools present.

We see the person is asking `Please set a reminder for a "Team Sync Meeting" this Friday, June 6th, 2025, at 2 PM.` so we expect the calendar or reminder tool to be used

```
dataset[0]["messages"]
```

```
[{'role': 'developer',
  'content': 'Current date and time given in YYYY-MM-DDTHH:MM:SS format: 2025-06-04T15:29:23\nDay of week is Wednesday\nYou are a model
that can do function calling with the following functions\n',
  'tool_calls': None},
 {'role': 'user',
  'content': 'Please set a reminder for a "Team Sync Meeting" this Friday, June 6th, 2025, at 2 PM.',
  'tool_calls': None},
 {'role': 'assistant',
  'content': None,
  'tool_calls': [{'function': {'name': 'create_calendar_event',
     'arguments': {'title': 'Team Sync Meeting',
      'datetime': datetime.datetime(2025, 6, 6, 14, 0),
      'last_name': None,
      'phone_number': None,
      'email': None,
      'first_name': None,
      'query': None,
      'to': None,
      'subject': None,
      'body': None}}}]}]
```

```
dataset[0]["tools"]
```

```
         'subject': None,
         'query': None,
         'last_name': {'type': 'STRING',
          'description': 'The last name of the contact.'},
         'first_name': {'type': 'STRING',
          'description': 'The first name of the contact.'},
         'email': {'type': 'STRING',
          'description': 'The email address of the contact.'},
         'phone_number': {'type': 'STRING',
          'description': 'The phone number of the contact.'},
         'datetime': None,
         'title': None},
        'required': ['first_name', 'last_name']}}}]
```

We can then apply FunctionGemma's chat template:

```python
import copy

messages_to_apply = dataset[0]["messages"][1:] # Skip the 'developer' message

# Create a deepcopy to avoid modifying the original dataset and make content a string
temp_messages = [m.copy() for m in messages_to_apply]
for i, msg in enumerate(temp_messages):
    if msg['role'] == 'assistant' and msg.get('tool_calls') and msg.get('content') is None:
        temp_messages[i]['content'] = "" # Replace None with an empty string

text_output = tokenizer.apply_chat_template(
    temp_messages,
    tools = dataset[0]["tools"],
    tokenize = False,
    add_generation_prompt = False,
)
print(text_output)
```

```
<bos><start_of_turn>developer
You are a model that can do function calling with the following functions<start_function_declaration>declaration:turn_off_flashlight{descri
<start_of_turn>user
Please set a reminder for a "Team Sync Meeting" this Friday, June 6th, 2025, at 2 PM.<end_of_turn>
<start_of_turn>model
<start_function_call>call:create_calendar_event{body:None,datetime:2025-06-06 14:00:00,email:None,first_name:None,last_name:None,phone_numb
```

We then create a dataset of just these mapped sequences.

```python
def process_dataset(row, tokenizer):
    messages = row["messages"]
    # Skip the 'developer' message if it's the first message
    if messages and messages[0]['role'] == 'developer':
        messages = messages[1:]

    # Create a copy to avoid modifying the original dataset and make content a string
    temp_messages = [m.copy() for m in messages]
    for i, msg in enumerate(temp_messages):
        if msg['role'] == 'assistant' and msg.get('tool_calls') and msg.get('content') is None:
            temp_messages[i]['content'] = "" # Replace None with an empty string

    text = tokenizer.apply_chat_template(
        temp_messages,
        tools = row["tools"],
        tokenize = False,
        add_generation_prompt = False,
    )
    return {"text" : text}
dataset = dataset.map(process_dataset, fn_kwargs = {"tokenizer" : tokenizer})
```

Look at the first row

```python
dataset[0]["text"]
```

```
'<bos><start_of_turn>developer\nYou are a model that can do function calling with the following
functions<start_function_declaration>declaration:turn_off_flashlight{description:<escape>Turns the flashlight off.<escape>,parameters:
{properties:{body:{description:<escape><escape>,type:<escape><escape>},datetime:{description:<escape><escape>,type:<escape>
<escape>},email:{description:<escape><escape>,type:<escape><escape>},first_name:{description:<escape><escape>,type:<escape>
<escape>},last_name:{description:<escape><escape>,type:<escape><escape>},phone_number:{description:<escape><escape>,type:<escape>
<escape>},query:{description:<escape><escape>,type:<escape><escape>},subject:{description:<escape><escape>,type:<escape><escape>},title:
{description:<escape><escape>,type:<escape><escape>},to:{description:<escape><escape>,type:<escape><escape>}},type:
<escape>OBJECT<escape>}}<end_function_declaration><start_function_declaration>declaration:open_wifi_settings{description:<escape>Opens the
Wi-Fi settings.<escape>,parameters:{properties:{body:{description:<escape><escape>,type:<escape><escape>},datetime:{description:<escape>
<escape>,type:<escape><escape>},email:{description:<escape><escape>,type:<escape><escape>},first_name:{description:<escape><escape>,type:
<escape><escape>},last_name:{description:<escape><escape>,type:<escape><escape>},phone_number:{description:<escape><escape>,type:<escape>
<escape>},query:{description:<escape><escape>,type:<escape><escape>},subject:{description:<escape><escape>,type:<escape><escape>},title:
{description:<escape><escape>,type:<escape><escape>},to:{description:<escape><escape>,type:<escape><escape>}},type:
<escape>OBJECT<escape>}}<end_function_declaration><start_function_declaration>declaration:create_calendar_event{description:
<escape>Creates a new calendar event.<escape>,parameters:{properties:{body:{description:<escape><escape>,type:<escape><escape>},datetime:
{description:<escape>The date and time of the event in the format YYYY-MM-DDTHH:MM:SS.<escape>,type:<escape>STRING<escape>},email:
{description:<escape><escape>,type:<escape><escape>},first_name:{description:<escape><escape>,type:<escape><escape>},last_name:
{description:<escape><escape>,type:<escape><escape>},phone_number:{description:<escape><escape>,type:<escape><escape>},query:{description:
<escape><escape>,type:<escape><escape>},subject:{description:<escape><escape>,type:<escape><escape>},title:{description:<escape>The title
of the event.<escape>,type:<escape>STRING<escape>},to:{description:<escape><escape>,type:<escape><escape>}},required:
[<escape>title<escape>,<escape>datetime<escape>],type:<escape>OBJECT<escape>}}<end_function_declaration>
<start_function_declaration>declaration:show_map{description:<escape>Shows a location on the map.<escape>,parameters:{properties:{body:
{description:<escape><escape>,type:<escape><escape>},datetime:{description:<escape><escape>,type:<escape><escape>},email:{description:
<escape><escape>,type:<escape><escape>},first_name:{description:<escape><escape>,type:<escape><escape>},last_name:{description:<escape>
<escape>,type:<escape><escape>},phone_number:{description:<escape><escape>,type:<escape><escape>},query:{description:<escape>The location
to search for. May be the name of a place, a business, or an address.<escape>,type:<escape>STRING<escape>},subject:{description:<escape>
<escape>,type:<escape><escape>},title:{description:<escape><escape>,type:<escape><escape>},to:{description:<escape><escape>,type:<escape>
<escape>}},required:[<escape>query<escape>],type:<escape>OBJECT<escape>}}<end_function_declaration>
<start_function_declaration>declaration:send_email{description:<escape>Sends an email.<escape>,parameters:{properties:{body:{description:
<escape>The body of the email.<escape>,type:<escape>STRING<escape>},datetime:{description:<escape><escape>,type:<escape><escape>},email:
{description:<escape><escape>,type:<escape><escape>},first_name:{description:<escape><escape>,type:<escape><escape>},last_name:
{description:<escape><escape>,type:<escape><escape>},phone_number:{description:<escape><escape>,type:<escape><escape>},query:{description:
<escape><escape>,type:<escape><escape>},subject:{description:<escape>The subject of the email.<escape>,type:<escape>STRING<escape>},title:
{description:<escape><escape>,type:<escape><escape>},to:{description:<escape>The email address of the recipient.<escape>,type:
<escape>STRING<escape>}},required:[<escape>to<escape>,<escape>subject<escape>],type:<escape>OBJECT<escape>}}<end_function_declaration>
<start_function_declaration>declaration:turn_on_flashlight{description:<escape>Turns the flashlight on.<escape>,parameters:{properties:
{body:{description:<escape><escape>,type:<escape><escape>},datetime:{description:<escape><escape>,type:<escape><escape>},email:
{description:<escape><escape>,type:<escape><escape>},first_name:{description:<escape><escape>,type:<escape><escape>},last_name:
{description:<escape><escape>,type:<escape><escape>},phone_number:{description:<escape><escape>,type:<escape><escape>},query:{description:
<escape><escape>,type:<escape><escape>},subject:{description:<escape><escape>,type:<escape><escape>},title:{description:<escape>
<escape>,type:<escape><escape>},to:{description:<escape><escape>,type:<escape><escape>}},type:<escape>OBJECT<escape>}}
<end_function_declaration><start_function_declaration>declaration:create_contact{description:<escape>Creates a contact in the phone\'s
contact list.<escape>,parameters:{properties:{body:{description:<escape><escape>,type:<escape><escape>},datetime:{description:<escape>
<escape>,type:<escape><escape>},email:{description:<escape>The email address of the contact.<escape>,type:
<escape>STRING<escape>},first_name:{description:<escape>The first name of the contact.<escape>,type:<escape>STRING<escape>},last_name:
{description:<escape>The last name of the contact.<escape>,type:<escape>STRING<escape>},phone_number:{description:<escape>The phone number
of the contact.<escape>,type:<escape>STRING<escape>},query:{description:<escape><escape>,type:<escape><escape>},subject:{description:
<escape><escape>,type:<escape><escape>},title:{description:<escape><escape>,type:<escape><escape>},to:{description:<escape><escape>,type:
<escape><escape>}},required:[<escape>first_name<escape>,<escape>last_name<escape>],type:<escape>OBJECT<escape>}}<end_function_declaration>
<end_of_turn>\n<start_of_turn>user\nPlease set a reminder for a "Team Sync Meeting" this Friday, June 6th, 2025, at 2 PM.
<end_of_turn>\n<start_of_turn>model\n<start_function_call>call:create_calendar_event{body:None,datetime:2025-06-06
14:00:00,email:None,first_name:None,last_name:None,phone_number:None,query:None,subject:None,title:<escape>Team Sync
Meeting<escape>,to:None}<end_function_call><start_function_response>'
```

Train the model

Now let's train our model. We do 100 steps to speed things up, but you can set `num_train_epochs=1` for a full run, and turn off `max_steps=None`.

```python
# We split the dataset into some training and testing
split_dataset = dataset.train_test_split(test_size = 50, shuffle = True, seed = 3407)

from trl import SFTTrainer, SFTConfig
trainer = SFTTrainer(
    model = model,
    tokenizer = tokenizer,
    train_dataset = split_dataset["train"],
    eval_dataset = split_dataset["test"], # For evaluation!
    args = SFTConfig(
        dataset_text_field = "text",
        per_device_train_batch_size = 4,
        gradient_accumulation_steps = 2, # Use GA to mimic batch size!
        warmup_steps = 5,
        # num_train_epochs = 1, # Set this for 1 full training run.
        max_steps = 100,
        eval_steps = 2,
        eval_strategy = "steps",
        learning_rate = 2e-4, # Reduce to 2e-5 for long training runs
        logging_steps = 1,
        optim = "adamw_8bit",
        weight_decay = 0.001,
        lr_scheduler_type = "linear",
        seed = 3407,
        report_to = "none", # Use TrackIO/WandB etc
    ),
)
```

```
🦥 Unsloth: Padding-free auto-enabled, enabling faster training.
```

We also mask all system and user instructions so our goal is to force the finetune to actually respond with the correct function calls. If not, the finetune might just learn the format, and not learn anything else.

```
from unsloth.chat_templates import train_on_responses_only
trainer = train_on_responses_only(
    trainer,
    instruction_part = "<start_of_turn>user\n",
    response_part = "<start_of_turn>model\n",
)
```

We print the full un-masked string:

```
tokenizer.decode(trainer.train_dataset[100]["input_ids"])
```

```
'<bos><bos><start_of_turn>developer\nYou are a model that can do function calling with the following
functions<start_function_declaration>declaration:turn_on_flashlight{description:<escape>Turns the flashlight on.<escape>,parameters:
{properties:{body:{description:<escape><escape>,type:<escape><escape>},datetime:{description:<escape><escape>,type:<escape>
<escape>},email:{description:<escape><escape>,type:<escape><escape>},first_name:{description:<escape><escape>,type:<escape>
<escape>},last_name:{description:<escape><escape>,type:<escape><escape>},phone_number:{description:<escape><escape>,type:<escape>
<escape>},query:{description:<escape><escape>,type:<escape><escape>},subject:{description:<escape><escape>,type:<escape><escape>},title:
{description:<escape><escape>,type:<escape><escape>},to:{description:<escape><escape>,type:<escape><escape>}},type:
<escape>OBJECT<escape>}}<end_function_declaration><start_function_declaration>declaration:show_map{description:<escape>Shows a location on
the map.<escape>,parameters:{properties:{body:{description:<escape><escape>,type:<escape><escape>},datetime:{description:<escape>
<escape>,type:<escape><escape>},email:{description:<escape><escape>,type:<escape><escape>},first_name:{description:<escape><escape>,type:
<escape><escape>},last_name:{description:<escape><escape>,type:<escape><escape>},phone_number:{description:<escape><escape>,type:<escape>
<escape>},query:{description:<escape>The location to search for. May be the name of a place, a business, or an address.<escape>,type:
<escape>STRING<escape>},subject:{description:<escape><escape>,type:<escape><escape>},title:{description:<escape><escape>,type:<escape>
<escape>},to:{description:<escape><escape>,type:<escape><escape>}},required:[<escape>query<escape>],type:<escape>OBJECT<escape>}}
<end_function_declaration><start_function_declaration>declaration:turn_off_flashlight{description:<escape>Turns the flashlight off.
<escape>,parameters:{properties:{body:{description:<escape><escape>,type:<escape><escape>},datetime:{description:<escape><escape>,type:
<escape><escape>},email:{description:<escape><escape>,type:<escape><escape>},first_name:{description:<escape><escape>,type:<escape>
<escape>},last_name:{description:<escape><escape>,type:<escape><escape>},phone_number:{description:<escape><escape>,type:<escape>
<escape>},query:{description:<escape><escape>,type:<escape><escape>},subject:{description:<escape><escape>,type:<escape><escape>},title:
{description:<escape><escape>,type:<escape><escape>},to:{description:<escape><escape>,type:<escape><escape>}},type:
<escape>OBJECT<escape>}}<end_function_declaration><start_function_declaration>declaration:create_calendar_event{description:
<escape>Creates a new calendar event.<escape>,parameters:{properties:{body:{description:<escape><escape>,type:<escape><escape>},datetime:
{description:<escape>The date and time of the event in the format YYYY-MM-DDTHH:MM:SS.<escape>,type:<escape>STRING<escape>},email:
{description:<escape><escape>,type:<escape><escape>},first_name:{description:<escape><escape>,type:<escape><escape>},last_name:
{description:<escape><escape>,type:<escape><escape>},phone_number:{description:<escape><escape>,type:<escape><escape>},query:{description:
<escape><escape>,type:<escape><escape>},subject:{description:<escape><escape>,type:<escape><escape>},title:{description:<escape>The title
of the event.<escape>,type:<escape>STRING<escape>},to:{description:<escape><escape>,type:<escape><escape>}},required:
[<escape>title<escape>,<escape>datetime<escape>],type:<escape>OBJECT<escape>}}<end_function_declaration>
<start_function_declaration>declaration:send_email{description:<escape>Sends an email.<escape>,parameters:{properties:{body:{description:
<escape>The body of the email.<escape>,type:<escape>STRING<escape>},datetime:{description:<escape><escape>,type:<escape><escape>},email:
{description:<escape><escape>,type:<escape><escape>},first_name:{description:<escape><escape>,type:<escape><escape>},last_name:
{description:<escape><escape>,type:<escape><escape>},phone_number:{description:<escape><escape>,type:<escape><escape>},query:{description:
<escape><escape>,type:<escape><escape>},subject:{description:<escape>The subject of the email.<escape>,type:<escape>STRING<escape>},title:
{description:<escape><escape>,type:<escape><escape>},to:{description:<escape>The email address of the recipient.<escape>,type:
<escape>STRING<escape>}},required:[<escape>to<escape>,<escape>subject<escape>],type:<escape>OBJECT<escape>}}<end_function_declaration>
<start_function_declaration>declaration:open_wifi_settings{description:<escape>Opens the Wi-Fi settings.<escape>,parameters:{properties:
{body:{description:<escape><escape>,type:<escape><escape>},datetime:{description:<escape><escape>,type:<escape><escape>},email:
{description:<escape><escape>,type:<escape><escape>},first_name:{description:<escape><escape>,type:<escape><escape>},last_name:
{description:<escape><escape>,type:<escape><escape>},phone_number:{description:<escape><escape>,type:<escape><escape>},query:{description:
<escape><escape>,type:<escape><escape>},subject:{description:<escape><escape>,type:<escape><escape>},title:{description:<escape>
<escape>,type:<escape><escape>},to:{description:<escape><escape>,type:<escape><escape>}},type:<escape>OBJECT<escape>}}
<end_function_declaration><start_function_declaration>declaration:create_contact{description:<escape>Creates a contact in the phone\'s
contact list.<escape>,parameters:{properties:{body:{description:<escape><escape>,type:<escape><escape>},datetime:{description:<escape>
<escape>,type:<escape><escape>},email:{description:<escape>The email address of the contact.<escape>,type:
<escape>STRING<escape>},first_name:{description:<escape>The first name of the contact.<escape>,type:<escape>STRING<escape>},last_name:
{description:<escape>The last name of the contact.<escape>,type:<escape>STRING<escape>},phone_number:{description:<escape>The phone number
of the contact.<escape>,type:<escape>STRING<escape>},query:{description:<escape><escape>,type:<escape><escape>},subject:{description:
<escape><escape>,type:<escape><escape>},title:{description:<escape><escape>,type:<escape><escape>},to:{description:<escape><escape>,type:
<escape><escape>}},required:[<escape>first_name<escape>,<escape>last_name<escape>],type:<escape>OBJECT<escape>}}<end_function_declaration>
<end_of_turn>\n<start_of_turn>user\nPlease schedule a new calendar event titled "Team Sync Meeting" for Monday, August 3rd, 2026, at 9:30
AM.<end_of_turn>\n<start_of_turn>model\n<start_function_call>call:create_calendar_event{body:None,datetime:2026-08-03
09:30:00,email:None,first_name:None,last_name:None,phone_number:None,query:None,subject:None,title:<escape>Team Sync
Meeting<escape>,to:None}<end_function_call><start_function_response>'
```

See if we did the masking correctly - "-" is the mask

```
[tokenizer.decode([tokenizer.pad_token_id if x == -100 else x for x in trainer.train_dataset[100]["labels"]]).replace(tokenizer.pad_to
```

```
['----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------<start_function_call>call:create_calendar_event{body:None,datetime:2026-
08-03 09:30:00,email:None,first_name:None,last_name:None,phone_number:None,query:None,subject:None,title:<escape>Team Sync
Meeting<escape>,to:None}<end_function_call><start_function_response>']
```

Show current memory stats

```python
# @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 = NVIDIA GeForce RTX 4090. Max memory = 47.371 GB.
0.535 GB of memory reserved.
```

Let's train the model! To resume a training run, set `trainer.train(resume_from_checkpoint = True)`

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

```
The model is already on multiple devices. Skipping the move to device specified in `args`.
==((====))==  Unsloth - 2x faster free finetuning | Num GPUs used = 1
   \\   /|    Num examples = 9,604 | Num Epochs = 1 | Total steps = 100
O^O/ \_/ \    Batch size per device = 4 | Gradient accumulation steps = 2
\        /    Data Parallel GPUs = 1 | Total batch size (4 x 2 x 1) = 8
 "-____-"     Trainable parameters = 3,796,992 of 271,895,168 (1.40% trained)
```

[100/100 03:18, Epoch 0/1]

Step	Training Loss	Validation Loss
2	4.264600	3.884358
4	3.183200	1.545703
6	0.891500	0.891690
8	0.903400	0.718925
10	0.619400	0.435925
12	0.315600	0.241501
14	0.176600	0.122878
16	0.098400	0.110043
18	0.106200	0.082088
20	0.112700	0.088621
22	0.027900	0.049317
24	0.067000	0.047772
26	0.023600	0.037676
28	0.014200	0.035077
30	0.044400	0.030640

Show final memory and time stats

```python
# @title Show final memory and time stats
used_memory = round(torch.cuda.max_memory_reserved() / 1024 / 1024 / 1024, 3)
used_memory_for_lora = round(used_memory - start_gpu_memory, 3)
used_percentage = round(used_memory / max_memory * 100, 3)
lora_percentage = round(used_memory_for_lora / max_memory * 100, 3)
print(f"{trainer_stats.metrics['train_runtime']} seconds used for training.")
print(
    f"{round(trainer_stats.metrics['train_runtime']/60, 2)} minutes used for training."
)
print(f"Peak reserved memory = {used_memory} GB.")
print(f"Peak reserved memory for training = {used_memory_for_lora} GB.")
print(f"Peak reserved memory % of max memory = {used_percentage} %.")
print(f"Peak reserved memory for training % of max memory = {lora_percentage} %.")
```

```
211.2837 seconds used for training.
3.52 minutes used for training.
Peak reserved memory = 9.92 GB.
Peak reserved memory for training = 9.385 GB.
Peak reserved memory % of max memory = 20.941 %.
Peak reserved memory for training % of max memory = 19.812 %.
```

Inference

Let's run the model via Unsloth native inference for the first message which should create a calendar reminder

```python
dataset[0]["messages"][:2]
```

```
[{'role': 'developer',
  'content': 'Current date and time given in YYYY-MM-DDTHH:MM:SS format: 2025-06-04T15:29:23\nDay of week is Wednesday\nYou are a model
that can do function calling with the following functions\n',
  'tool_calls': None},
 {'role': 'user',
  'content': 'Please set a reminder for a "Team Sync Meeting" this Friday, June 6th, 2025, at 2 PM.',
  'tool_calls': None}]
```

Let's try inference:

```python
import copy

messages_for_inference = dataset[0]["messages"]
# Skip the 'developer' message if it's the first message
if messages_for_inference and messages_for_inference[0]['role'] == 'developer':
    messages_for_inference = messages_for_inference[1:]

# Create a copy and ensure assistant messages with tool_calls have content as an empty string
temp_messages_for_inference = [m.copy() for m in messages_for_inference]
for i, msg in enumerate(temp_messages_for_inference):
    if msg['role'] == 'assistant' and msg.get('tool_calls') and msg.get('content') is None:
        temp_messages_for_inference[i]['content'] = "" # Replace None with an empty string

text = tokenizer.apply_chat_template(
    temp_messages_for_inference,
    tools = dataset[0]["tools"],
    tokenize = False,
    add_generation_prompt = True, # Must add for generation
).removeprefix('<bos>')

```

48	0.013900	0.020785
50	0.014100	0.018875
52	0.059900	0.017512
54	0.039200	0.016822
56	0.026700	0.016670
58	0.020800	0.016408
60	0.004600	0.016269
62	0.023800	0.016346
66	0.106500	0.015987
68	0.003500	0.016288
70	0.013200	0.016455
72	0.012300	0.016047
74	0.017100	0.015756
76	0.013100	0.015631
78	0.016500	0.015425

```
from transformers import TextStreamer
_ = model.generate(
    **tokenizer(text, return_tensors = "pt").to("cuda"),
    max_new_tokens = 1024,
    streamer = TextStreamer(tokenizer, skip_prompt = False),
    top_p = 0.95, top_k = 64, temperature = 1.0,
)
```

```
<bos><start_of_turn>developer
You are a model that can do function calling with the following functions<start_function_declaration>declaration:turn_off_flashlight{descr
<start_of_turn>user
Please set a reminder for a "Team Sync Meeting" this Friday, June 6th, 2025, at 2 PM.<end_of_turn>
<start_of_turn>model
<start_function_call>call:create_calendar_event{body:None,datetime:2025-06-06 14:00:00,email:None,first_name:None,last_name:None,phone_numb
<start_function_call>call:create_calendar_event{body:None,datetime:2025-06-06 14:00:00,email:None,first_name:None,last_name:None,phone_numb
```

It looks correct!

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

```
('functiongemma/tokenizer_config.json',
 'functiongemma/special_tokens_map.json',
 'functiongemma/chat_template.jinja',
 'functiongemma/tokenizer.model',
 'functiongemma/added_tokens.json',
 'functiongemma/tokenizer.json')
```

Now if you want to load the LoRA adapters we just saved for inference, set `False` to `True`:

```
if True:
    from unsloth import FastLanguageModel
    model, tokenizer = FastLanguageModel.from_pretrained(
        model_name = "functiongemma", # YOUR MODEL YOU USED FOR TRAINING
        max_seq_length = 2048,
        load_in_4bit = False,
    )
```

```
==((====))==  Unsloth 2026.1.3: Fast Gemma3 patching. Transformers: 4.57.6. vLLM: 0.13.0.
   \\   /|    NVIDIA GeForce RTX 4090. Num GPUs = 1. Max memory: 47.371 GB. Platform: Linux.
O^O/ \_/ \    Torch: 2.9.0+cu128. CUDA: 8.9. CUDA Toolkit: 12.8. Triton: 3.5.0
\        /    Bfloat16 = TRUE. FA [Xformers = 0.0.33.post1. FA2 = False]
 "-____-"     Free license: http://github.com/unslothai/unsloth
Unsloth: Fast downloading is enabled - ignore downloading bars which are red colored!
Unsloth: Gemma3 does not support SDPA - switching to fast eager.
Unsloth: QLoRA and full finetuning all not selected. Switching to 16bit LoRA.
```

```python
import copy

messages_for_inference = dataset[0]["messages"]
# Skip the 'developer' message if it's the first message
if messages_for_inference and messages_for_inference[0]['role'] == 'developer':
    messages_for_inference = messages_for_inference[1:]

# Create a copy and ensure assistant messages with tool_calls have content as an empty string
temp_messages_for_inference = [m.copy() for m in messages_for_inference]
for i, msg in enumerate(temp_messages_for_inference):
    if msg['role'] == 'assistant' and msg.get('tool_calls') and msg.get('content') is None:
        temp_messages_for_inference[i]['content'] = "" # Replace None with an empty string

text = tokenizer.apply_chat_template(
    temp_messages_for_inference,
    tools = dataset[0]["tools"],
    tokenize = False,
    add_generation_prompt = True, # Must add for generation
).removeprefix('<bos>')

print("+" * 20)
print(text)
print("*" * 20)

from transformers import TextStreamer
res = model.generate(
    **tokenizer(text, return_tensors = "pt").to("cuda"),
    max_new_tokens = 1024,
    streamer = TextStreamer(tokenizer, skip_prompt = False),
    top_p = 0.95, top_k = 64, temperature = 1.0,
)

generated_text = tokenizer.decode(res[0], skip_special_tokens=False)
```

```
print("-" * 20)
print(generated_text)
print("*" * 20)
```

```
++++++++++++++++++++
<start_of_turn>developer
You are a model that can do function calling with the following functions<start_function_declaration>declaration:turn_off_flashlight{descr
<start_of_turn>user
Please set a reminder for a "Team Sync Meeting" this Friday, June 6th, 2025, at 2 PM.<end_of_turn>
<start_of_turn>model
<start_function_call>call:create_calendar_event{body:None,datetime:2025-06-06 14:00:00,email:None,first_name:None,last_name:None,phone_numb

********************
<bos><start_of_turn>developer
You are a model that can do function calling with the following functions<start_function_declaration>declaration:turn_off_flashlight{descr
<start_of_turn>user
Please set a reminder for a "Team Sync Meeting" this Friday, June 6th, 2025, at 2 PM.<end_of_turn>
<start_of_turn>model
<start_function_call>call:create_calendar_event{body:None,datetime:2025-06-06 14:00:00,email:None,first_name:None,last_name:None,phone_numb
<start_function_call>call:create_calendar_event{body:None,datetime:2025-06-06 14:00:00,email:None,first_name:None,last_name:None,phone_numb
--------------------
<bos><start_of_turn>developer
You are a model that can do function calling with the following functions<start_function_declaration>declaration:turn_off_flashlight{descr
<start_of_turn>user
Please set a reminder for a "Team Sync Meeting" this Friday, June 6th, 2025, at 2 PM.<end_of_turn>
<start_of_turn>model
<start_function_call>call:create_calendar_event{body:None,datetime:2025-06-06 14:00:00,email:None,first_name:None,last_name:None,phone_numb
<start_function_call>call:create_calendar_event{body:None,datetime:2025-06-06 14:00:00,email:None,first_name:None,last_name:None,phone_numb
********************
```

Saving to float16 for VLLM

We also support saving to `float16` directly. Select `merged_16bit` for float16 or `merged_4bit` for int4. We also allow `lora` adapters as a fallback. Use `push_to_hub_merged` to upload to your Hugging Face account! You can go to https://huggingface.co/settings/tokens for your personal tokens.

```
# Merge to 16bit
if False:
    model.save_pretrained_merged("functiongemma-finetune", tokenizer, save_method = "merged_16bit")
if False: # Pushing to HF Hub
    model.push_to_hub_merged("hf/functiongemma-finetune", tokenizer, save_method = "merged_16bit", token = "")

# Merge to 4bit
if False:
    model.save_pretrained_merged("functiongemma-finetune", tokenizer, save_method = "merged_4bit",)
```

```
if False: # Pushing to HF Hub
    model.push_to_hub_merged("hf/functiongemma-finetune", tokenizer, save_method = "merged_4bit", token = "")

# Just LoRA adapters
if False:
    model.save_pretrained("functiongemma-finetune")
    tokenizer.save_pretrained("functiongemma-finetune")
if False: # Pushing to HF Hub
    model.push_to_hub("hf/functiongemma-finetune", token = "")
    tokenizer.push_to_hub("hf/functiongemma-finetune", token = "")
```

GGUF / llama.cpp Conversion

To save to `GGUF` / `llama.cpp`, we support it natively now for all models! For now, you can convert easily to `Q8_0, F16 or BF16` precision. `Q4_K_M` for 4bit will come later!

```
if False: # Change to True to save to GGUF
    model.save_pretrained_gguf(
        "functiongemma-finetune",
        tokenizer,
        quantization_method = "Q8_0", # For now only Q8_0, BF16, F16 supported
    )
```

Likewise, if you want to instead push to GGUF to your Hugging Face account, set `if False` to `if True` and add your Hugging Face token and upload location!

```
if False: # Change to True to upload GGUF
    model.push_to_hub_gguf(
        "HF_ACCOUNT/functiongemma-gguf",
        tokenizer,
        quantization_method = "Q8_0", # Only Q8_0, BF16, F16 supported
        token = "hf_...",
    )
```

Now, use the `functiongemma-finetune.gguf` file or `functiongemma-finetune-Q4_K_M.gguf` file in llama.cpp.

And we're done! If you have any questions on Unsloth, we have a Discord channel! If you find any bugs or want to keep updated with the latest LLM stuff, or need help, join projects etc, feel free to join our Discord!

Some other links: