

Resources X

...

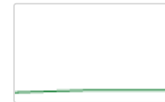
You are subscribed to Colab Pro. [Learn more](#)
 Available: 92.19 compute units
 Usage rate: approximately 1.66 per hour
 You have 1 active session.

[Manage sessions](#)

Python 3 Google Compute Engine backend
 (GPU)

Showing resources from 4:12 PM to 4:19 PM

System RAM
 6.1 / 51.0 GB



GPU RAM
 8.2 / 15.0 GB



Disk
 41.1 / 235.7 GB



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

▼ Initializing the Model

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

# 4bit pre quantized models we support for 4x faster downloading
fourbit_models = [
    "unsloth/mistral-7b-v0.3-bnb-4bit",      # New Mistral v3 2x
    "unsloth/mistral-7b-instruct-v0.3-bnb-4bit",
    "unsloth/llama-3-8b-bnb-4bit",          # Llama-3 15 trillion
    "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-
)

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

# 4bit pre quantized models we support for 4x faster downloading
fourbit_models = [
    "unsloth/mistral-7b-v0.3-bnb-4bit",      # New Mistral v3 2x
    "unsloth/mistral-7b-instruct-v0.3-bnb-4bit",
    "unsloth/llama-3-8b-bnb-4bit",          # Llama-3 15 trillion
    "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-
```

```

↳ 🦛 Unsloth: Will patch your computer to enable 2x faster free
🦛 Unsloth Zoo will now patch everything to make training fa
==((====))== Unsloth 2024.12.8: Fast Llama patching. Transf
  \ \   /|   GPU: Tesla T4. Max memory: 14.748 GB. Platform
0^0/ \_/ \   Torch: 2.5.1+cu121. CUDA: 7.5. CUDA Toolkit: 1
 \ \   /     Bfloat16 = FALSE. FA [Xformers = 0.0.28.post3.
  "-____-"   Free Apache license: http://github.com/unsloth
Unsloth: Fast downloading is enabled - ignore downloading ba
model.safetensors: 100%                    5.70G/5.70G [00:14<00:00, 681MB/s]

generation_config.json: 100%                198/198 [00:00<00:00, 17.6kB/s]

tokenizer_config.json: 100%                 50.6k/50.6k [00:00<00:00, 3.97MB/s]

tokenizer.json: 100%                       9.09M/9.09M [00:00<00:00, 21.1MB/s]

special_tokens_map.json: 100%              350/350 [00:00<00:00, 28.2kB/s]
==((====))== Unsloth 2024.12.8: Fast Llama patching. Transf
  \ \   /|   GPU: Tesla T4. Max memory: 14.748 GB. Platform
0^0/ \_/ \   Torch: 2.5.1+cu121. CUDA: 7.5. CUDA Toolkit: 1
 \ \   /     Bfloat16 = FALSE. FA [Xformers = 0.0.28.post3.
  "-____-"   Free Apache license: http://github.com/unsloth
Unsloth: Fast downloading is enabled - ignore downloading ba

```

```

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 size
    use_gradient_checkpointing = "unsloth", # True or "unsloth"
    random_state = 3407,
    use_rslora = False, # We support rank stabilized LoRA
    loftq_config = None, # And LoftQ
)

```

```

↳ Unsloth 2024.12.8 patched 32 layers with 32 QKV layers, 32 O

```

✓ Dataset Preparation :

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

```

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, 10446.06 examples/s]
['questions_with_options', 'predicted_label', 'gpt_reasoning']
{'questions_with_options': "A 58-year-old man comes to the p
```

✓ Converting the dataset into required format:

```
from unsloth import to_sharegpt
```

```
# Define a post-processing function to combine 'predicted_label'
def combine_columns(example):
    example["predicted_label_and_reasoning"] = f"Label: {example['predicted_label']} {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 dataset
dataset = to_sharegpt(
    dataset,
    merged_prompt="{questions_with_options}",
    output_column_name="predicted_label_and_reasoning", # Combined column name
)

print(dataset[0])
```



```
Map: 100%      542/542 [00:00<00:00, 9146.37 examples/s]
{'questions_with_options': "A 58-year-old man comes to the p
Merging columns: 100%      542/542 [00:00<00:00, 15246.29 examples/s]
Converting to ShareGPT: 100%      542/542 [00:00<00:00, 18739.39 examples/s]
{'conversations': [{'from': 'human', 'value': '("A 58-year-o
```

✓ Standardize share-gpt

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



```
Standardizing format: 100%      542/542 [00:00<00:00, 13978.69 examples/s]
```

✓ Chat Template

```
chat_template = """Below are questions with options. Provide the
```

```
>>> 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", <<
)
```

↳ Unsloth: We automatically added an EOS token to stop endless

Map: 100% 542/542 [00:00<00:00, 9103.70 examples/s]



✓ Training the Model

```
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 seq
    args = TrainingArguments(
        per_device_train_batch_size = 2,
        gradient_accumulation_steps = 4,
        warmup_steps = 50,
        # max_steps = 60, ## comment this later
        num_train_epochs = 5, ## train for 5 epochs ideally
        learning_rate = 3e-4, # 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 [00:02<00:00, 323.95 examples/s]



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



```
num_examples = 342 | num_epochs = 5
Batch size per device = 2 | Gradient Accumulation steps = 4
Total batch size = 8 | Total steps = 335
Number of trainable parameters = 41,943,040
[325/335 1:06:40 < 02:03, 0.08 it/s, Epoch 4.80/5]
```

Step Training Loss

1	1.744200
2	1.792600
3	1.784900
4	1.708300
5	1.777600
6	1.894800
7	1.782600
8	1.756900
9	1.631500
10	1.613300
11	1.683400
12	1.682500
13	1.543600
14	1.402000
15	1.331000
16	1.336000
17	1.297300
18	1.227300
19	1.159100
20	1.080800
21	1.093600
22	1.036700
23	1.039700
24	1.029200
25	0.992400
26	0.942900
27	0.926100
28	0.981200
29	0.973700
30	0.867800
31	0.988800
32	0.998800
33	0.933000

34	0.898700
35	0.916800
36	0.876200
37	0.961800
38	0.873200
39	0.887500
40	0.815900
41	0.866400
42	0.812900
43	0.854100
44	0.890900
45	0.914800
46	0.853600
47	0.858700
48	0.890700
49	0.740800
50	0.982800
51	0.855300
52	0.902100
53	0.842600
54	0.887700
55	0.859500
56	0.803100
57	0.778300
58	0.861800
59	0.877800
60	0.812400
61	0.798400
62	0.845300
63	0.912800
64	0.808600
65	0.803300
66	0.886000
67	0.783500
68	1.191500
69	1.002700
70	0.699900
71	0.744800
72	0.801900

73	0.704700
74	0.700400
75	0.786200
76	0.777300
77	0.611600
78	0.740300
79	0.769500
80	0.796900
81	0.689200
82	0.797300
83	0.769000
84	0.756700
85	0.638800
86	0.765000
87	0.780600
88	0.681300
89	0.845100
90	0.680200
91	0.753700
92	0.700100
93	0.795300
94	0.655400
95	0.790700
96	0.747500
97	0.748200
98	0.717100
99	0.647300
100	0.902700
101	0.735100
102	0.749900
103	0.699000
104	0.750300
105	0.731500
106	0.687100
107	0.706300
108	0.689200
109	0.683400
110	0.841100
111	0.688800

111	0.626200
112	0.774800
113	0.694100
114	0.827600
115	0.673500
116	0.713900
117	0.646100
118	0.628300
119	0.724000
120	0.689300
121	0.674900
122	0.626100
123	0.796400
124	0.682600
125	0.712000
126	0.644900
127	0.737100
128	0.726900
129	0.833900
130	0.743300
131	0.648500
132	0.746400
133	0.713400
134	0.606300
135	1.326000
136	0.568600
137	0.616500
138	0.537500
139	0.513500
140	0.586200
141	0.579400
142	0.546400
143	0.535700
144	0.537000
145	0.635200
146	0.538500
147	0.524500
148	0.519400
149	0.592500

150	0.581500
151	0.543300
152	0.566800
153	0.526800
154	0.576000
155	0.480800
156	0.510400
157	0.493800
158	0.606100
159	0.491800
160	0.538500
161	0.528000
162	0.554000
163	0.586700
164	0.578700
165	0.547400
166	0.525400
167	0.580100
168	0.525900
169	0.574400
170	0.537000
171	0.525500
172	0.512900
173	0.539300
174	0.529700
175	0.532600
176	0.479500
177	0.523600
178	0.555600
179	0.492700
180	0.540000
181	0.589700
182	0.502600
183	0.612100
184	0.491600
185	0.486400
186	0.508400
187	0.589100
188	0.558000

188	0.555500
189	0.640000
190	0.488200
191	0.575300
192	0.508700
193	0.548300
194	0.533500
195	0.578000
196	0.481300
197	0.571500
198	0.603000
199	0.567700
200	0.533200
201	0.565100
202	0.467000
203	0.790700
204	0.356100
205	0.398300
206	0.334600
207	0.383800
208	0.327900
209	0.395500
210	0.359900
211	0.374700
212	0.448800
213	0.363700
214	0.325100
215	0.332300
216	0.446900
217	0.321900
218	0.349600
219	0.465600
220	0.306900
221	0.482600
222	0.342600
223	0.334800
224	0.369900
225	0.331800
226	0.348400

227	0.406900
228	0.354000
229	0.470100
230	0.334200
231	0.375900
232	0.391500
233	0.338200
234	0.351100
235	0.331000
236	0.313800
237	0.423500
238	0.343300
239	0.378200
240	0.289500
241	0.399500
242	0.372900
243	0.360700
244	0.414400
245	0.344000
246	0.413900
247	0.389600
248	0.415500
249	0.302500
250	0.389700
251	0.358700
252	0.269500
253	0.397300
254	0.373200
255	0.344300
256	0.342500
257	0.377200
258	0.309200
259	0.364000
260	0.347800
261	0.434600
262	0.358200
263	0.411000
264	0.350700
265	0.425800

266	0.327100
267	0.389500
268	0.352100
269	0.348700
270	0.557700
271	0.228600
272	0.244900
273	0.275900
274	0.269600
275	0.235100
276	0.241000
277	0.218200
278	0.244800
279	0.256600
280	0.196200
281	0.234100
282	0.221300
283	0.230200
284	0.253800
285	0.244300
286	0.224600
287	0.219800
288	0.218200
289	0.234700
290	0.190500
291	0.227100
292	0.240200
293	0.240300
294	0.214600
295	0.222700
296	0.230300
297	0.235800
298	0.242100
299	0.190200
300	0.210100
301	0.200400
302	0.202000
303	0.221600

304	0.200000
305	0.261300
306	0.228400
307	0.232900
308	0.222400
309	0.214400
310	0.201100
311	0.244000
312	0.223900
313	0.241700
314	0.245100
315	0.227600
316	0.236100
317	0.237300
318	0.226700
319	0.278200
320	0.305400
321	0.250600
322	0.199900
323	0.234700



Step	Training Loss
1	1.744200
2	1.792600
3	1.784900
4	1.708300
5	1.777600
6	1.894800
7	1.782600
8	1.756900
9	1.631500
10	1.613300
11	1.683400
12	1.682500
13	1.543600
14	1.402000
15	1.331000
16	1.336000
17	1.297300

18	1.227300
19	1.159100
20	1.080800
21	1.093600
22	1.036700
23	1.039700
24	1.029200
25	0.992400
26	0.942900
27	0.926100
28	0.981200
29	0.973700
30	0.867800
31	0.988800
32	0.998800
33	0.933000
34	0.898700
35	0.916800
36	0.876200
37	0.961800
38	0.873200
39	0.887500
40	0.815900
41	0.866400
42	0.812900
43	0.854100
44	0.890900
45	0.914800
46	0.853600
47	0.858700
48	0.890700
49	0.740800
50	0.982800
51	0.855300
52	0.902100
53	0.842600
54	0.887700
55	0.859500