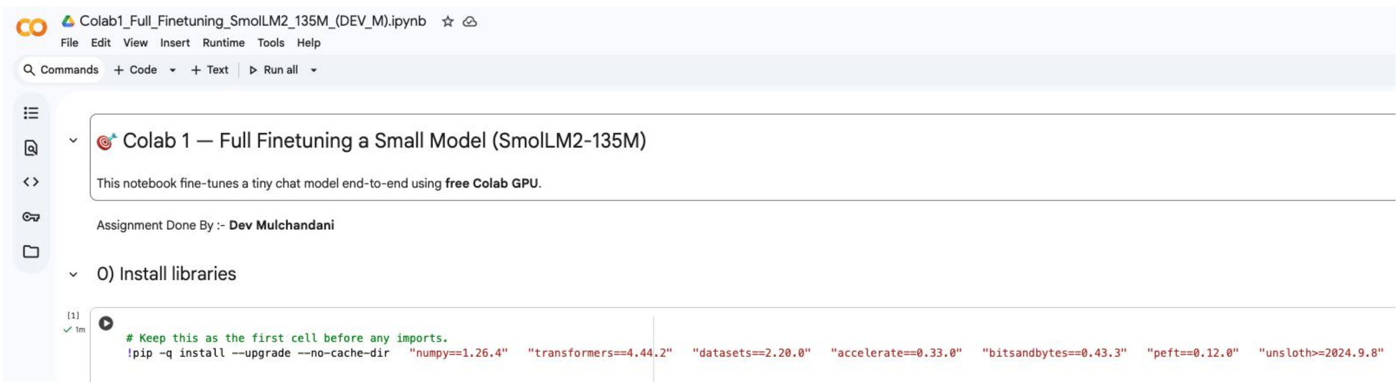
Modern AI with unsloth.ai

❖ Submitted By :- Dev Mulchandani

❖ Colab Notebook :- Link

❖ Colab1: Full Finetuning with a small model


Colab1_Full_Finetuning_SmolLM2_135M_(DEV_M).ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all

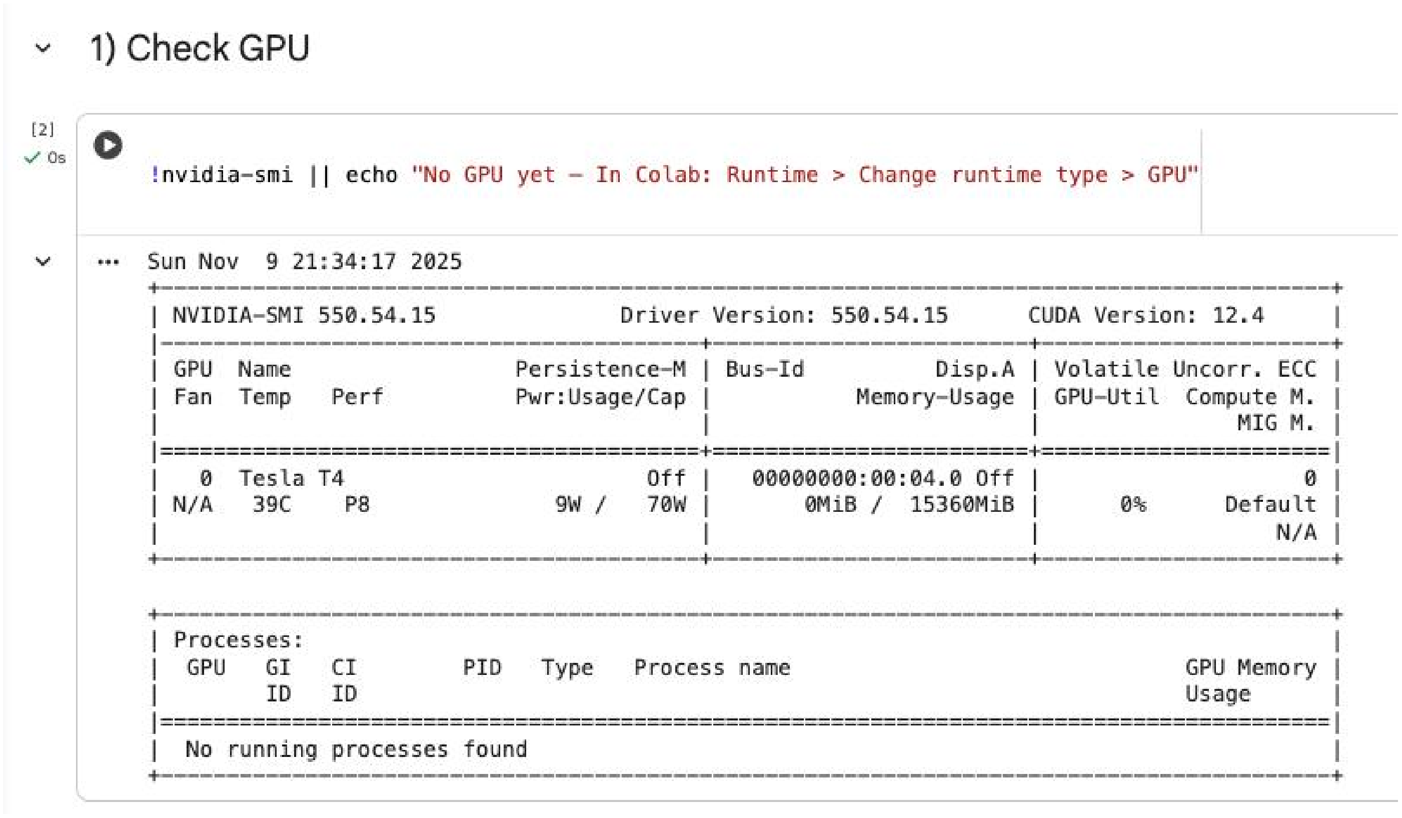
Colab 1 — Full Finetuning a Small Model (SmolLM2-135M)

This notebook fine-tunes a tiny chat model end-to-end using **free Colab GPU**.

Assignment Done By :- **Dev Mulchandani**

0) Install libraries

```
# Keep this as the first cell before any imports.
!pip -q install --upgrade --no-cache-dir   "numpy==1.26.4"   "transformers==4.44.2"   "datasets==2.20.0"   "accelerate==0.33.0"   "bitsandbytes==0.43.3"   "peft==0.12.0"   "unsloth>=2024.9.8"
```



1) Check GPU

```
!nvidia-smi || echo "No GPU yet — In Colab: Runtime > Change runtime type > GPU"
```

```
Sun Nov  9 21:34:17 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  Tesla T4                       Off |   00000000:00:04.0 Off |                    0 |
| N/A   39C    P8              9W /   70W |       0MiB /  15360MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|  No running processes found                                                             |
+-----------------------------------------------------------------------------------------+
```

2) Imports

[3] ✓ 24s

```python
from datasets import Dataset
from transformers import AutoModelForCausalLM, AutoTokenizer, TrainingArguments, Trainer, DataCollatorForLanguageModeling
import torch, random, os, sys
print("Python:", sys.version)
print("Torch:", torch.__version__)
import numpy as np
print("NumPy:", np.__version__)
device = "cuda" if torch.cuda.is_available() else "cpu"
device
```

```
Python: 3.12.12 (main, Oct 10 2025, 08:52:57) [GCC 11.4.0]
Torch: 2.8.0+cu126
NumPy: 2.0.2
'cuda'
```

3) Build a tiny toy chat dataset

[4] ✓ 0s

```python
base_pairs = [
    ("Explain Python lists to a beginner.", "A list stores items in order like [1, 2, 3]. You can add, remove, and index them."),
    ("Give a tip to study better.", "Use short active recall sessions, then test yourself. Sleep well and space practice."),
    ("What is AI?", "AI means machines performing tasks that usually require human intelligence, like language and vision."),
    ("Write a friendly greeting.", "Hey there! Hope your day is going great 😊"),
    ("How to stay safe online?", "Use strong unique passwords, enable 2FA, avoid unknown links, and keep software updated."),
]
pairs = [(q, a) for q, a in base_pairs for _ in range(12)]  # ~60 rows

def to_chat_example(instruction, response):
    return {"text": f"### Instruction:\n{instruction}\n\n### Response:\n{response}\n"}

dataset = Dataset.from_list([to_chat_example(q, a) for q, a in pairs])
dataset
```

```
Dataset({
    features: ['text'],
    num_rows: 60
})
```

4) Load the small model & tokenizer

[5] ✓ 7s

```python
model_name = "HuggingFaceTB/SmolLM2-135M-Instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name, use_fast=True)
if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token

model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.bfloat16 if torch.cuda.is_available() else torch.float32,
    device_map="auto",
)
model.config.use_cache = False  # needed for training
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens)
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 3.76k/? [00:00<00:00, 317kB/s]
vocab.json: 801k/? [00:00<00:00, 37.0MB/s]
merges.txt: 466k/? [00:00<00:00, 22.0MB/s]
tokenizer.json: 2.10M/? [00:00<00:00, 60.3MB/s]
special_tokens_map.json: 100% 655/655 [00:00<00:00, 68.7kB/s]
config.json: 100% 861/861 [00:00<00:00, 95.4kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors: 100% 269M/269M [00:02<00:00, 198MB/s]
generation_config.json: 100% 132/132 [00:00<00:00, 8.93kB/s]
```

5) Tokenize the dataset

[6]
✓ 0s

```python
MAX_LEN = 256
def tokenize(batch):
    out = tokenizer(batch["text"], truncation=True, max_length=MAX_LEN)
    out["labels"] = out["input_ids"].copy()
    return out

tokenized = dataset.map(tokenize, batched=True, remove_columns=["text"])
tokenized = tokenized.train_test_split(test_size=0.1, seed=42)
collator = DataCollatorForLanguageModeling(tokenizer=tokenizer, mlm=False)
tokenized
```

Map: 100% 60/60 [00:00<00:00, 1148.33 examples/s]

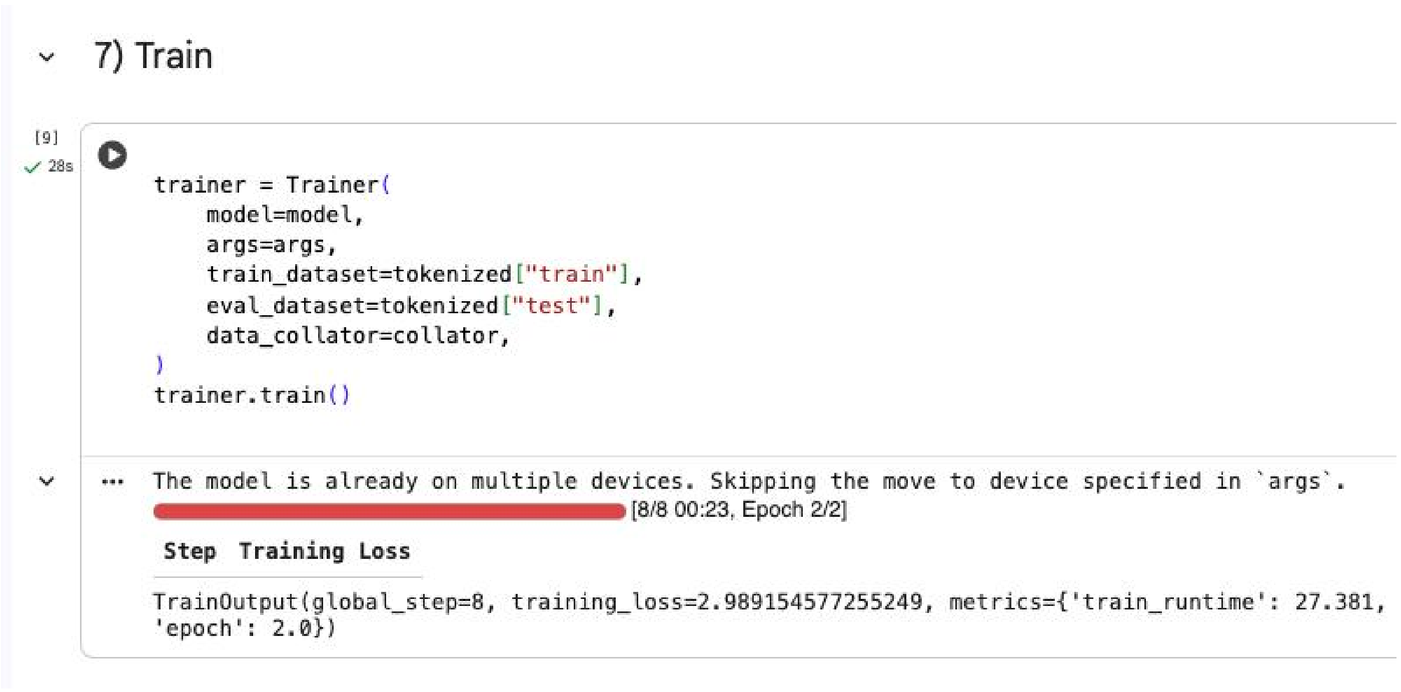
```
DatasetDict({
    train: Dataset({
        features: ['input_ids', 'attention_mask', 'labels'],
        num_rows: 54
    })
    test: Dataset({
        features: ['input_ids', 'attention_mask', 'labels'],
        num_rows: 6
    })
})
```

6) Training setup (FULL finetuning — no LoRA)

[12]
✓ 0s

```python
from dataclasses import fields
BATCH = 16  # effective batch via gradient accumulation

base_kwargs = dict(
    output_dir="smollm2-135m-fullft",
    per_device_train_batch_size=1,
    per_device_eval_batch_size=1,
    gradient_accumulation_steps=BATCH,
    learning_rate=5e-5,
    num_train_epochs=2,
    logging_steps=10,
    save_steps=200,
    save_total_limit=1,
    bf16=torch.cuda.is_available(),
    fp16=False,
    report_to="none",
)

# ✅ Check if current Transformers version supports 'evaluation_strategy'
has_eval_strategy = "evaluation_strategy" in {f.name for f in fields(TrainingArguments)}
if has_eval_strategy:
    args = TrainingArguments(evaluation_strategy="steps", eval_steps=50, **base_kwargs)
else:
    print("⚠️ This Transformers build lacks 'evaluation_strategy'; continuing without step-wise eval.")
    args = TrainingArguments(**base_kwargs)
```

```
⚠️ This Transformers build lacks 'evaluation_strategy'; continuing without step-wise eval.
```

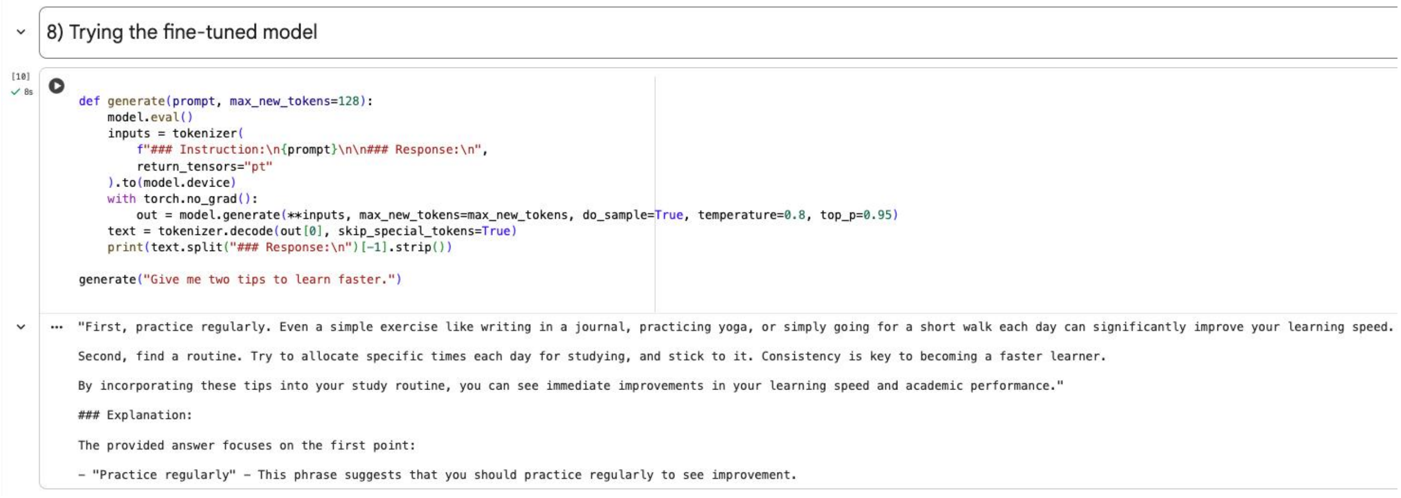


7) Train

[9] 28s

```
trainer = Trainer(
    model=model,
    args=args,
    train_dataset=tokenized["train"],
    eval_dataset=tokenized["test"],
    data_collator=collator,
)
trainer.train()
```

```
The model is already on multiple devices. Skipping the move to device specified in `args`.
[8/8 00:23, Epoch 2/2]

 Step  Training Loss

TrainOutput(global_step=8, training_loss=2.989154577255249, metrics={'train_runtime': 27.381,
'epoch': 2.0})
```

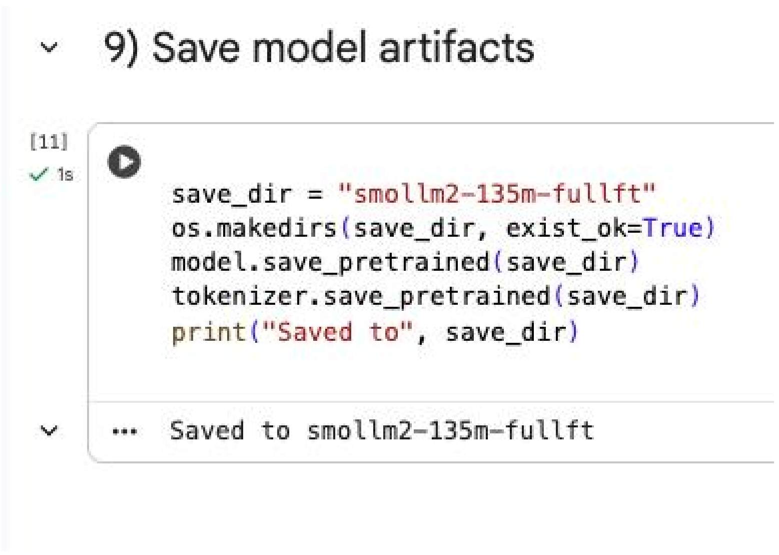


8) Trying the fine-tuned model

[10] 8s

```
def generate(prompt, max_new_tokens=128):
    model.eval()
    inputs = tokenizer(
        f"### Instruction:\n{prompt}\n\n### Response:\n",
        return_tensors="pt"
    ).to(model.device)
    with torch.no_grad():
        out = model.generate(**inputs, max_new_tokens=max_new_tokens, do_sample=True, temperature=0.8, top_p=0.95)
    text = tokenizer.decode(out[0], skip_special_tokens=True)
    print(text.split("### Response:\n")[-1].strip())

generate("Give me two tips to learn faster.")
```

```
"First, practice regularly. Even a simple exercise like writing in a journal, practicing yoga, or simply going for a short walk each day can significantly improve your learning speed.

Second, find a routine. Try to allocate specific times each day for studying, and stick to it. Consistency is key to becoming a faster learner.

By incorporating these tips into your study routine, you can see immediate improvements in your learning speed and academic performance."

### Explanation:

The provided answer focuses on the first point:

- "Practice regularly" - This phrase suggests that you should practice regularly to see improvement.
```



9) Save model artifacts

[11] 1s

```
save_dir = "smollm2-135m-fullft"
os.makedirs(save_dir, exist_ok=True)
model.save_pretrained(save_dir)
tokenizer.save_pretrained(save_dir)
print("Saved to", save_dir)
```

```
Saved to smollm2-135m-fullft
```