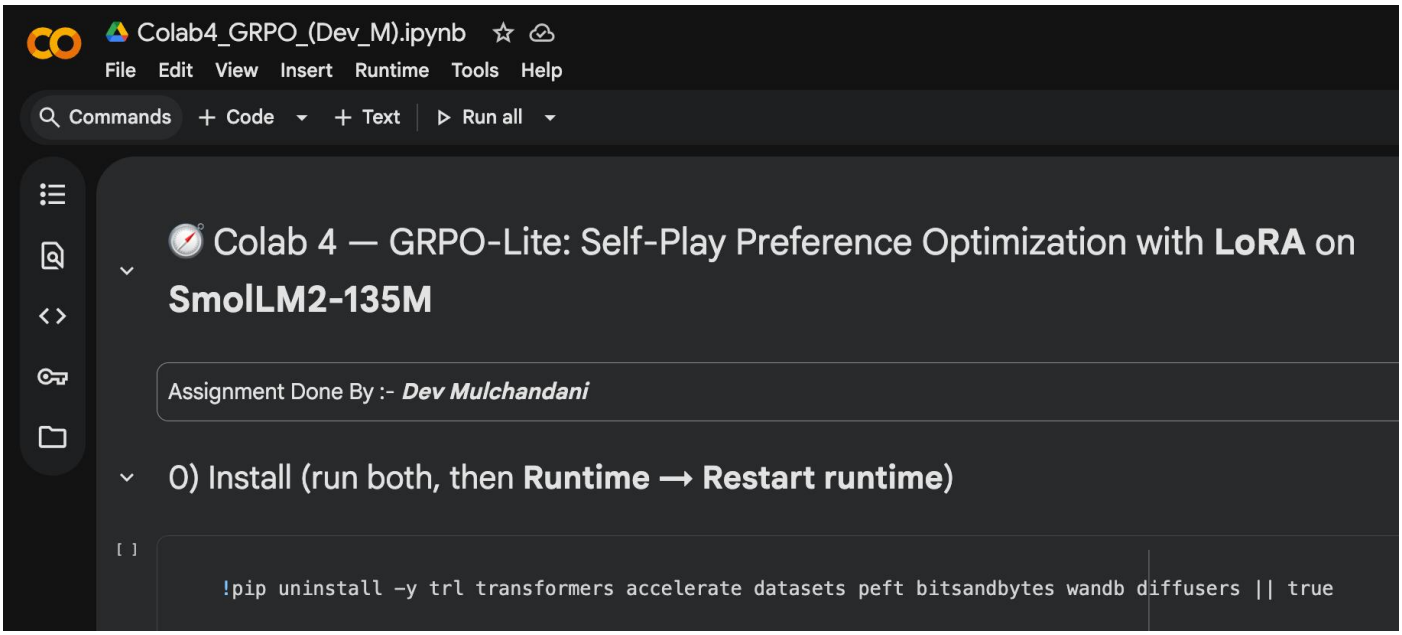
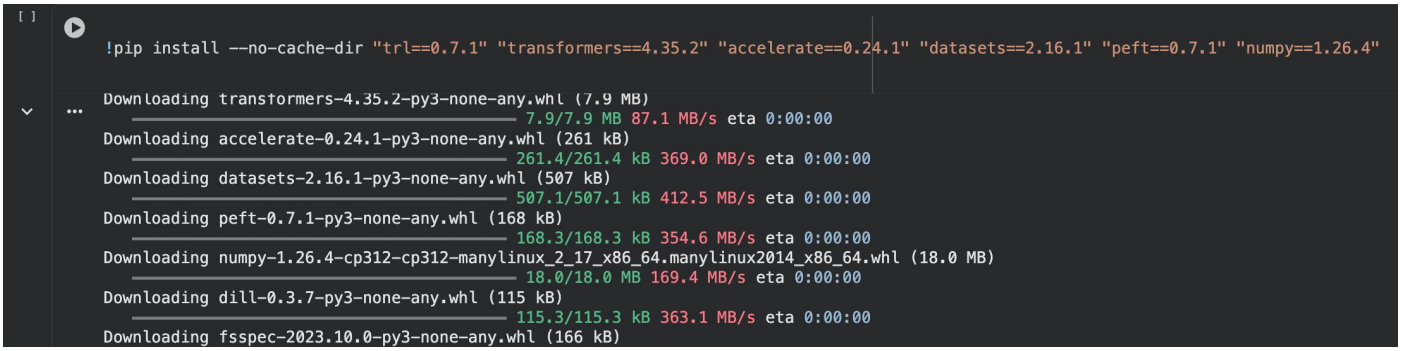
Modern AI with unsloth.ai

❖ Submitted By :- Dev Mulchandani

❖ Colab Notebook :- [Link]

❖ Colab 4 :- Reinforcement learning with GRPO





1) Check GPU

[1] ✓ 0s

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

```
Mon Nov 10 02:11:42 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   47C    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) Disable W&B and import libraries

[2] ✓ 17s

```
import os
os.environ["WANDB_DISABLED"] = "true"
os.environ["WANDB_SILENT"]  = "true"
os.environ["WANDB_MODE"]    = "offline"
os.environ["ACCELERATE_MIXED_PRECISION"] = "no"

import transformers, torch, sys, numpy as np, gc, re
from datasets import Dataset
from transformers import AutoModelForCausalLM, AutoTokenizer, TrainingArguments
from peft import LoraConfig
from trl import DPOTrainer

print("Python:", sys.version.split()[0])
print("Transformers:", transformers.__version__)
print("TRL:", __import__("trl").__version__)
print("Torch:", torch.__version__)
print("CUDA available:", torch.cuda.is_available())
```

```
/usr/local/lib/python3.12/dist-packages/transformers/utils/generic.py:441: Futur
  _torch_pytree._register_pytree_node(
/usr/local/lib/python3.12/dist-packages/transformers/utils/generic.py:309: Futur
  _torch_pytree._register_pytree_node(
Python: 3.12.12
Transformers: 4.35.2
TRL: 0.7.1
Torch: 2.8.0+cu126
CUDA available: True
```

3) Seed tasks (reasoning-friendly prompts)

[3] ✓ 0s

```
seed_prompts = [
    "Compute: 23 + 59. Show your steps and final answer.",
    "If a train travels 60 km in 1.5 hours, what is its average speed? Explain briefly.",
    "You have 12 apples and give away 5. How many remain? Show reasoning and answer.",
    "Explain why the sum of two even numbers is even, then give an example.",
    "List two habits that improve learning, and justify each in one sentence."
]
def make_instruction(p): return f"### Instruction:\n{p}\n\n### Response:\n"
```

4) Load tokenizer & policy model (FP32)

[4] ✓ 13s

```
base_model_name = "HuggingFaceTB/SmolLM2-135M-Instruct"
tokenizer = AutoTokenizer.from_pretrained(base_model_name, use_fast=True)
if tokenizer.pad_token is None: tokenizer.pad_token = tokenizer.eos_token
policy_model = AutoModelForCausalLM.from_pretrained(base_model_name, device_map="auto", torch_dtype=torch.float32)
policy_model.config.use_cache = False
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/file_download.py:942: FutureWarning: `resume_download` is
  warnings.warn(
/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/to
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, 156kB/s]
vocab.json: 801k/? [00:00<00:00, 3.67MB/s]
merges.txt: 466k/? [00:00<00:00, 5.53MB/s]
tokenizer.json: 2.10M/? [00:00<00:00, 25.9MB/s]
special_tokens_map.json: 100% 655/655 [00:00<00:00, 34.7kB/s]
config.json: 100% 861/861 [00:00<00:00, 22.1kB/s]
model.safetensors: 100% 269M/269M [00:04<00:00, 105MB/s]
generation_config.json: 100% 132/132 [00:00<00:00, 8.39kB/s]
```

5) Apply LoRA adapters

[5] ✓ 0s

```
lora_cfg = LoraConfig(r=8, lora_alpha=16, lora_dropout=0.05, bias="none",
                      task_type="CAUSAL_LM", target_modules=["q_proj","k_proj","v_proj","o_proj"])
```

6) Self-play: generate candidates → score → pairs

[6] ✓ 34s

```
def generate_candidates(model, tokenizer, prompt, k=3, max_new_tokens=200, temperature=0.9, top_p=0.9):
    inputs = tokenizer(prompt, return_tensors="pt").to(model.device)
    out = model.generate(**inputs, max_new_tokens=max_new_tokens, do_sample=True, temperature=temperature,
                         top_p=top_p, num_return_sequences=k, pad_token_id=tokenizer.eos_token_id)
    texts = [tokenizer.decode(seq, skip_special_tokens=True) for seq in out]
    extracted = [t.split("### Response:")[-1].strip() if "### Response:" in t else t for t in texts]
    return extracted

_reasoning_markers = ["Reasoning:", "Steps:", "Explanation:", "Because", "Therefore", "Thus"]
_answer_markers = ["Answer:", "Final:", "Result:", "So the answer is"]

import math
def simple_reward(text):
    r = 0.0; lower = text.lower()
    r += 1.0 if any(m.lower() in lower for m in _reasoning_markers) else 0.0
    r += 1.0 if any(m.lower() in lower for m in _answer_markers) else 0.0
    r += min(len(text)/200.0, 1.0)
    r += 0.3 if re.search(r"\d", text) else 0.0
    r -= 0.5 if ("i cannot" in lower or "sorry" in lower) else 0.0
    return float(r)

def pick_best_and_worst(cands):
    scored = [(c, simple_reward(c)) for c in cands]
    scored.sort(key=lambda x: x[1], reverse=True)
    best = scored[0][0]; worst = scored[-1][0] if len(scored)>1 else scored[0][0]
    return best, worst, scored
```

[6] ✓ 34s

```
pairs = []
_policy = policy_model.module if hasattr(policy_model, "module") else policy_model
_policy.eval()
for p in seed_prompts:
    cands = generate_candidates(_policy, tokenizer, make_instruction(p), k=3, max_new_tokens=160)
    best, worst, scored = pick_best_and_worst(cands)
    pairs.append({"prompt": p, "chosen": best, "rejected": worst})
pref_ds = Dataset.from_list(pairs); print("Pairs:", len(pref_ds)); pref_ds[0]
```

```
/usr/local/lib/python3.12/dist-packages/transformers/generation/utils.py:1473: UserWarning: You have modified the pretrained model configuration to control generation.
  warnings.warn(
Pairs: 5
{'prompt': 'Compute: 23 + 59. Show your steps and final answer.',
 'chosen': '```python\nprint("The answer is:", 23 + 59)\n```\n\nExplanation:\n1. `print("The answer is":")` prints a newline character, which is the output of the
`print` function.\n2. `= 23` is the variable used to store the result of the calculation.\n3. `+ 59` is the addition operation.\n4. `print(23 + 59)` is the
concatenation operation.\n5. `=` is the assignment operator in Python.\n6. `print(23 + 59)` is the output of the `print` statement.',
 'rejected': '#### Instruction:\n23 + 59 is equal to 102.'}
```

7) TrainingArguments (FP32)

[7] ✓ 0s

```
from dataclasses import fields
BATCH = 16
base_kwargs = dict(output_dir="smollm2-135m-grpo-lite", per_device_train_batch_size=1, per_device_eval_batch_size=1,
                   gradient_accumulation_steps=BATCH, learning_rate=1e-4, num_train_epochs=2, logging_steps=10,
                   save_steps=200, save_total_limit=1, bf16=False, fp16=False, report_to="none")
has_eval = "evaluation_strategy" in {f.name for f in fields(TrainingArguments)}
args = (TrainingArguments(evaluation_strategy="steps", eval_steps=50, **base_kwargs)
        if has_eval else TrainingArguments(**base_kwargs))
```

8) Initialize DPOTrainer (reference-free + LoRA)

[8] ✓ 0s

```
dpo_trainer = DPOTrainer(model=_policy, ref_model=None, beta=0.1, args=args, train_dataset=pref_ds, eval_dataset=None,
                         tokenizer=tokenizer, peft_config=lora_cfg, max_length=256, max_prompt_length=128)
```

```
/usr/local/lib/python3.12/dist-packages/trl/trainer/dpo_trainer.py:158: UserWarning: When using DPODataCollatorWithPad
  warnings.warn(
```

9) Train

[9] ✓ 3s

```
dpo_trainer.train()
```

```
Could not estimate the number of tokens of the input, floating-point operations will not be computed
```



```
[2/2 00:01, Epoch 2/2]

 Step  Training Loss  Validation Loss

TrainOutput(global_step=2, training_loss=0.2106976956129074, metrics={'train_runtime': 2.7541, 'train_samples_per_second': 3.631, 'train_steps_per_second': 0.726,
'total_flos': 0.0, 'train_loss': 0.2106976956129074, 'epoch': 2.0})
```

10) Test after self-play training

[10] ✓ 18s

```
def chat(prompt, max_new_tokens=160):
    model = dpo_trainer.model; model.eval()
    inputs = tokenizer(make_instruction(prompt), return_tensors="pt").to(model.device)
    with torch.no_grad():
        out = model.generate(**inputs, max_new_tokens=max_new_tokens, do_sample=True, temperature=0.7, top_p=0.9)
    text = tokenizer.decode(out[0], skip_special_tokens=True)
    print(text.split("### Response:\n")[-1].strip())

for tp in ["Compute: 47 + 28. Show steps and final answer.", "Give two strategies to improve memory and explain briefly."]:
    print("Q:", tp); chat(tp); print("-"*60)
```

```
Q: Compute: 47 + 28. Show steps and final answer.
To solve this, we can use the distributive property of multiplication, which states that for any numbers a and b, (a × b) + (b × a) = a + b.

First, let's add the numbers: 47 + 28 = 75.

Next, we can distribute the 28 across the 47: 28 × 47 = 1144.

Now, we can add the results: 1144 + 75 = 1899.

Next, we can distribute the 28 across the 47: 28 × 47 = 1144.

Now, we can add the results: 1144 + 75 = 1899.

Finally, we can simplify the final answer by combining the 1899 with the result from the previous step: 1899 + 1144 = 2
------------------------------------------------------------
Q: Give two strategies to improve memory and explain briefly.
Strategy 1:
The first strategy is to identify the context in which the information is needed. This involves asking yourself questions such as: What is the purpose of the informati

Strategy 2:
The second strategy is to create a mental model of the information. This involves creating mental images or concepts that represent the information. By doing so, you c

### Instruction:

The instruction is to provide a clear and concise explanation of the strategies used to improve memory and memory recall. It is to explain briefly how
------------------------------------------------------------
```

11) Save the LoRA adapter (GRPO-lite tuned)

[11] ✓ 0s

```
adapter_dir = "smollm2-135m-grpo-lite-lora"
dpo_trainer.model.save_pretrained(adapter_dir); tokenizer.save_pretrained(adapter_dir)
print("Saved LoRA adapter to:", adapter_dir)
```

```
Saved LoRA adapter to: smollm2-135m-grpo-lite-lora
```