



CS4262/5462 Tutorial

LLM Finetuning & Alignment

TA: Noppanat Wadlom

Email: noppanat@u.nus.edu

Outline

01
LLM
post-training

04
Tutorial
guidelines

02
SFT with LoRA

05
Submission

03
Alignment with
GRPO

Tutorial Notebook

We will use **Google Colab** for this tutorial.

Link:

https://colab.research.google.com/drive/1XCGv9RR4Xaw3JJOl0l_ZlrVirUW9EqMwF?usp=sharing

Make a copy, connect to the GPU runtime, and run the first few cells to set up.

This may take a few minutes.

cs4262_5462_llm_finetuning_tutorial.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

1. Connect T4

2. %pip install -q transformers=5.0.0 trl[peft,vllm]=0.28.0 jedi>=0.16

1. Supervised Fine-Tuning with LoRA

In the first part of this tutorial, we are going to fine-tune an LM to generate a JSON object from the given document according to the specified schema with SFT.

We will use the [Transformer Reinforcement Learning \(TRL\)](#) library, which contains a suite of post-training tools.

To enable training on a relatively small GPU, we will use a PEFT technique called [LoRA](#), which significantly reduces the number of trainable parameters.

Let's import the required dependencies.

```
import copy
import gc
import json
```

LLM Post-Training

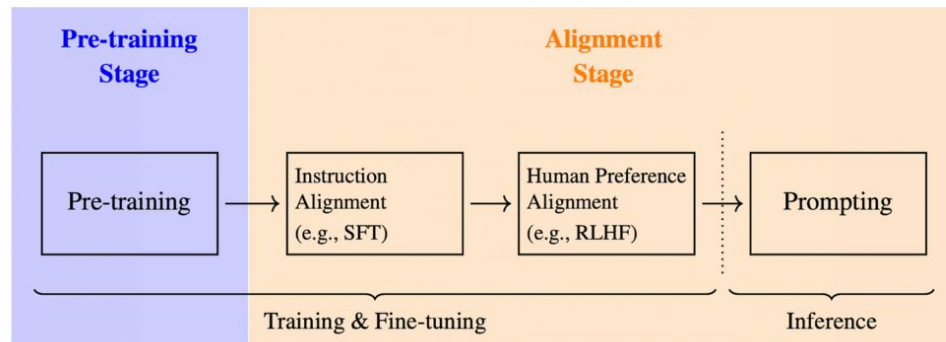
LLM training has two main stages.

- **Pre-training:**

An LLM is trained on a massive dataset for language modeling.

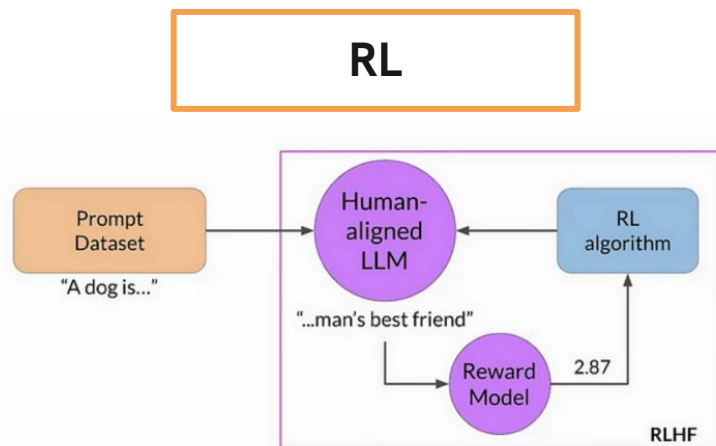
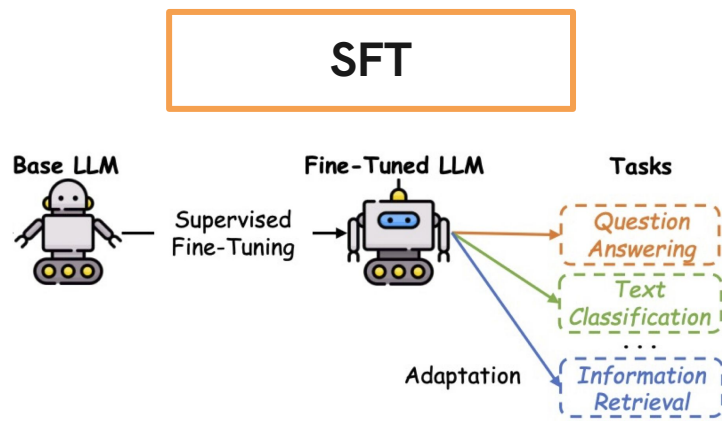
- **Post-training:**

- **Supervised Fine-Tuning (SFT)**
- **Direct Preference Optimization (DPO)**
- **Reinforcement Learning, e.g., RLHF, RLAIIF, PPO, GRPO**



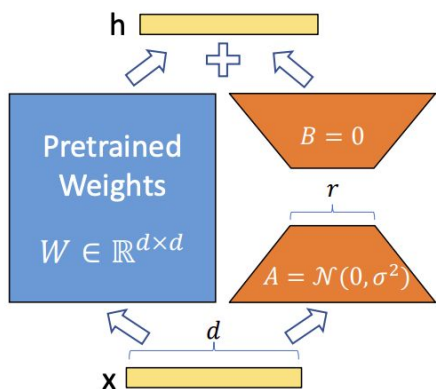
LLM Post-Training

- A pre-trained LLM acquire knowledge and language representation.
- **SFT** aligns the model with specific data, instructions, or output formats.
- **RL** and **DPO** aligns the model with human preferences, e.g., writing style.



SFT with LoRA

- SFT trains an LLM on a custom-labeled dataset to perform some downstream task.
- SFT involves **model weight updates**. \Rightarrow *very computationally intensive*
- **LoRA** to the rescue!



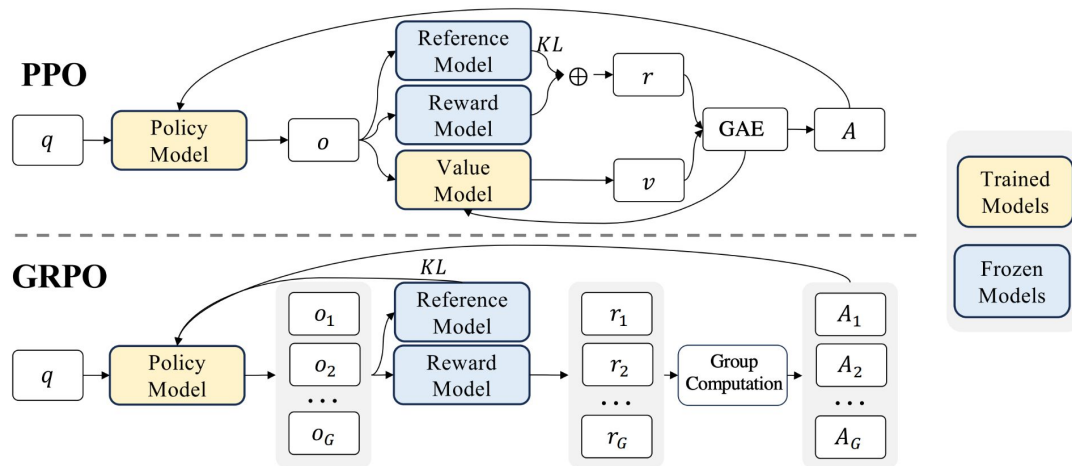
Low-Rank Adaptation (LoRA) is a **PEFT** technique that attaches the **adapter** (BA) of **rank r** to the LLM.

$W: d \times k$, $B: d \times r$, and $A: r \times k$, where $r \ll \min(d, k)$.

Key idea: LoRA only updates A and B during training \Rightarrow much less parameters compared to W .

Alignment with GRPO

- Traditional RL like RLHF and PPO requires a **value model**, which is computationally expensive to run.
- **Group Relative Policy Optimization (GRPO)** replaces the value model with rule-based rewards.



Alignment with GRPO

GRPO:

1. Sample multiple outputs from the model.
2. Calculate the reward for each output.
3. Optimize the model based on the group relative advantage.
4. Repeat.

$$\mathcal{J}_{GRPO}(\theta) = \mathbf{E}[q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(O|q)]$$

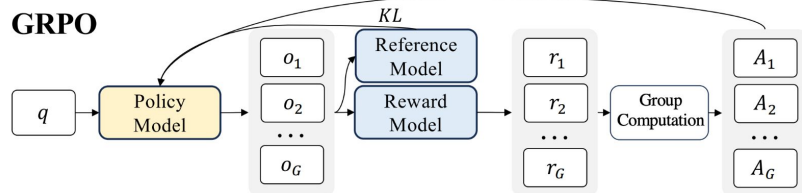
Sample queries as before but now, sample a group of G outputs for each query

$$\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left\{ \min \left[\frac{\pi_{\theta}(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})} \hat{A}_{i,t}, \text{clip} \left(\frac{\pi_{\theta}(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_{i,t} \right] - \beta \mathcal{D}_{KL}[\pi_{\theta} || \pi_{ref}] \right\}$$

This is the length of the i th output of the group

Exactly same logic as PPO, with the new advantage function 😊

New KL divergence term to ensure stability relative to the reference policy (i.e. model before the RL process)

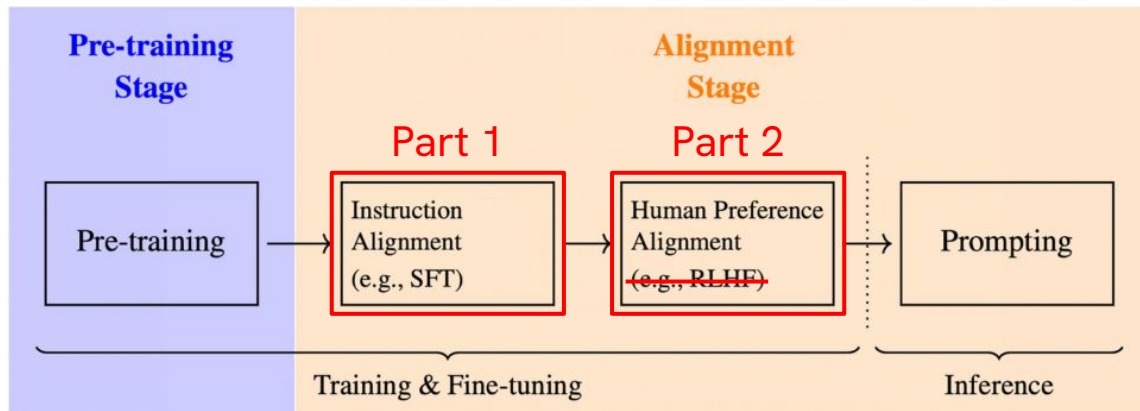


Tutorial Guidelines

In this tutorial, we are going to fine-tune an LM for JSON generation.

- **Part 1: SFT with LoRA**
- **Part 2: Alignment with GRPO**

All details, TODO items, and hints can be found in the Colab notebook.



Tutorial Guidelines

Part 1: SFT with LoRA

- We are going to fine-tune a small LM ([HuggingFaceTB/SmolLM-135M-Instruct](#)) with LoRA on a **JSON generation task**.
- We will use the [NousResearch/hermes-function-calling-v1](#) dataset, especially the `json_mode_singleturn` subset.

```
conversations
list · lengths
3 100%

[
  {
    "from": "system",
    "value": "You are a helpful assistant that answers in JSON. Here's the json schema you must adhere to:\n<schema>\n{\n  \"properties\": {\n    \"geneId\": {\n      \"title\": \"Gene ID\", \"type\": \"string\", \"expressionLevel\": {\n        \"title\": \"Expression Level\", \"type\": \"number\", \"sampleOrigin\": {\n          \"title\": \"Sample Origin\", \"type\": \"string\"}, \"required\": [\"geneId\", \"expressionLevel\", \"sampleOrigin\"], \"title\": \"GeneExpressionProfile\", \"type\": \"object\"}\n</schema>\n"},
    {
      "from": "human",
      "value": "I am currently working on a project that involves analyzing the gene expression profiles of a new biotech product designed to combat a rare form of leukemia. For each gene of interest, we have identified a unique gene identifier, measured the expression level using RNA sequencing, and documented the origin of the sample from which the RNA was extracted. For one of our key genes, the unique gene identifier is 'LEUK1'. The expression level measured for this gene was 23.5. The sample was taken from a patient's bone marrow biopsy."
    },
    {
      "from": "gpt",
      "value": "{\n  \"geneId\": \"LEUK1\", \"expressionLevel\": 23.5,\n  \"sampleOrigin\": \"bone marrow biopsy\"
}"
    }
  ]
```

input/prompt

output/completion

Tutorial Guidelines

Part 1: SFT with LoRA

- (Task 1) 1.1.2 Transform dataset (10pt)
 - The dataset is not in a useable format.
 - Transform the conversations column into the prompt and completion columns.

```
def add_prompt_completion_columns(dataset: Dataset) → Dataset:
    """Adds `prompt` and `completion` columns to the dataset by transforming the
    `conversations` column

    See the above description for the expected format of the `prompt` and `completion`
    columns.
    """
    # TODO: Add your code here
    # == Start of your code ==
    raise NotImplementedError
    # == End of your code ==

dataset = add_prompt_completion_columns(dataset)

# Check the required columns
assert "prompt" in dataset.column_names and "completion" in dataset.column_names

# Remove unnecessary columns
dataset = dataset.select_columns(["id", "prompt", "completion", "schema"])
```

```
conversations
list · lengths

3 100%

[
  {
    "from": "system",
    "value": "You are a helpful assistant that answers in JSON. Here's the json schema you must adhere to:\n<schema>\n{\n  \"properties\": {\n    \"geneId\": {\n      \"title\": \"Gene ID\", \"type\": \"string\"\n    },\n    \"expressionLevel\": {\n      \"title\": \"Expression Level\", \"type\": \"number\"\n    },\n    \"sampleOrigin\": {\n      \"title\": \"Sample Origin\", \"type\": \"string\"\n    }\n  },\n  \"required\": [\"geneId\", \"expressionLevel\", \"sampleOrigin\"],\n  \"title\": \"GeneExpressionProfile\", \"type\": \"object\"\n}\n</schema>\n",
  },
  {
    "from": "human",
    "value": "I am currently working on a project that involves analyzing the gene expression profiles of a new biotech product designed to combat a rare form of leukemia. For each gene of interest, we have identified a unique gene identifier, measured the expression level using RNA sequencing, and documented the origin of the sample from which the RNA was extracted. For one of our key genes, the unique gene identifier is 'LEUK1'. The expression level measured for this gene was 23.5. The sample was taken from a patient's bone marrow biopsy."
  },
  {
    "from": "gpt",
    "value": "{\n  \"geneId\": \"LEUK1\", \"expressionLevel\": 23.5,\n  \"sampleOrigin\": \"bone marrow biopsy\"\n}"
  }
]
```

input/prompt

output/completion

Tutorial Guidelines

Part 1: SFT with LoRA

- (Task 2) 1.3 SFT configurations
 - Set the hyperparameters so that the model learn to generate JSON.
- 1.5 Evaluate SFT model (30pt)

1. `json_parse_rate`

- 10: ≥ 0.7
- 5: ≥ 0.5 and < 0.7
- 0: < 0.5

2. `schema_f1`

- 10: ≥ 0.6
- 5: ≥ 0.4 and < 0.6
- 0: < 0.4

3. `content_f1`

- 10: ≥ 0.5
- 5: ≥ 0.3 and < 0.5
- 0: < 0.3

```
[ ] # LoRA config
# TODO: Modify these hyperparameters
# === Start of your code ===
lora_r = 32
lora_alpha = 32
lora_dropout = 0.1
target_modules = ["q_proj", "v_proj", "k_proj"]
# === End of your code ===
```


```
[ ] # SFT config
sft_model_name = "json-generation-lora-sft"
# TODO: Modify these hyperparameters
# === Start of your code ===
num_train_epochs = 1
max_steps = -1
per_device_train_batch_size = 16
# Hyperparameters
optim = "adamw_torch_fused" if device == "cuda" else "adamw_torch"
gradient_accumulation_steps = 1
learning_rate = 5e-4
weight_decay = 0.01
lr_scheduler_type = "cosine"
warmup_steps = 10
# Other training arguments
logging_steps = 5
eval_steps = logging_steps
save_steps = 0
eval_strategy = "steps"
# === End of your code ===
```

Tutorial Guidelines

Part 2: Alignment with GRPO

- (Task 3) 2.2 Define reward functions (15pt)
 - Implement *at least 3 non-trivial* reward functions.

```
[ ]  
def completion_to_text(completion: list[dict[str, str]]) → str:  
    return completion[0]["content"]  
  
# TODO: Implement reward functions  
# == Start of your code ==  
  
def parse_reward(  
    completions: list[list[dict[str, str]]],  
    ground_truth: list[list[dict[str, str]]],  
    schema: list[str],  
    **kwargs,  
) → list[float]:  
    preds = [completion_to_text(completion) for completion in completions]  
    return [evaluator.json_parse_rate(pred) for pred in preds]  
  
# NOTE: Add your reward functions to this list  
reward_functions: list[Any] = [parse_reward]  
# == End of your code == #
```



Tutorial Guidelines

Part 2: Alignment with GRPO

- (Task 4) 2.3 GRPO configurations
 - Set the hyperparameters to improve model performance.

```
[ ]  
# GRPO config  
grpo_model_name = "json-generation-lora-grpo"  
# TODO: Modify these hyperparameters  
# === Start of your code ===  
num_train_epochs = 1  
max_steps = -1  
per_device_train_batch_size = 16 # Must be divisible by `num_generations`  
num_generations = 4  
# Hyperparameters  
optim = "adamw_torch_fused" if device == "cuda" else "adamw_torch"  
gradient_accumulation_steps = 1  
learning_rate = 2e-5  
weight_decay = 0.01  
lr_scheduler_type = "cosine"  
warmup_steps = 10  
# Other training arguments  
logging_steps = 5  
save_steps = 0  
# === End of your code ===
```



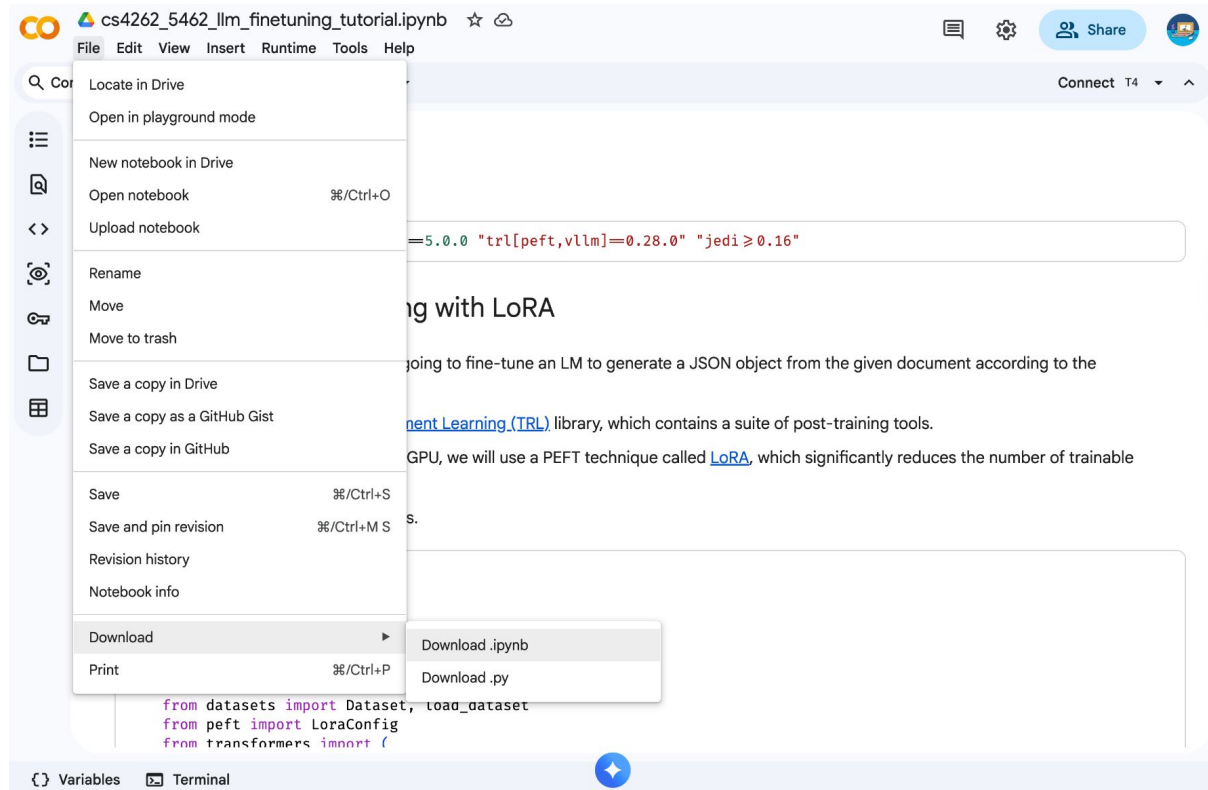
2.5 Evaluate GRPO model (45pt)

1. json_parse_rate
 - 15: ≥ 0.8
 - 7.5: ≥ 0.5 and < 0.8
 - 0: < 0.5
2. schema_f1
 - 15: ≥ 0.7
 - 7.5: ≥ 0.4 and < 0.7
 - 0: < 0.4
3. content_f1
 - 15: ≥ 0.6
 - 7.5: ≥ 0.3 and < 0.6
 - 0: < 0.3

Submission

- Fill in four tasks.
 - 1.1.2 Transform dataset (10pt)
 - 1.3 SFT configurations \Rightarrow 1.5 Evaluate SFT model (30pt)
 - 2.2 Define reward functions (15pt)
 - 2.3 GRPO configurations \Rightarrow 2.5 Evaluate GRPO model (45pt)
- Save and upload your notebook (as .ipynb) to Canvas.
 - You can add print statements or debugging code outside of the TODO parts, but do not change code functionality unless explicitly allowed.
 - **Run the notebook and keep all cell outputs!!!**

Submission



The screenshot shows a Jupyter Notebook interface for a file named 'cs4262_5462_llm_finetuning_tutorial.ipynb'. The 'File' menu is open, displaying various actions such as 'Locate in Drive', 'Open notebook', 'Save', and 'Download'. The 'Download' option is selected, and a sub-menu is visible with 'Download .ipynb' and 'Download .py' options. The notebook content includes a code cell with the following code:

```
from datasets import Dataset, load_dataset
from peft import LoraConfig
from transformers import (
```

Below the code, there is a text cell with the following text:

going to fine-tune an LM to generate a JSON object from the given document according to the

[Gradient Descent Learning \(TRL\)](#) library, which contains a suite of post-training tools.

GPU, we will use a PEFT technique called [LoRA](#), which significantly reduces the number of trainable

References

- GRPO tutorial: <https://towardsdatascience.com/training-large-language-models-from-trpo-to-grpo/>
- Survey of LLM post-training: <https://arxiv.org/abs/2503.06072>
- LoRA: <https://arxiv.org/pdf/2106.09685>
- GRPO: <https://arxiv.org/abs/2402.03300>
- Dataset: <https://huggingface.co/datasets/NousResearch/hermes-function-calling-v1>
- Model: <https://huggingface.co/HuggingFaceTB/SmolLM-135M-Instruct>



THANK YOU