CS4262/5462 Tutorial

Efficient AI

TA: Noppanat Wadlom
Email: noppanat@u.nus.edu

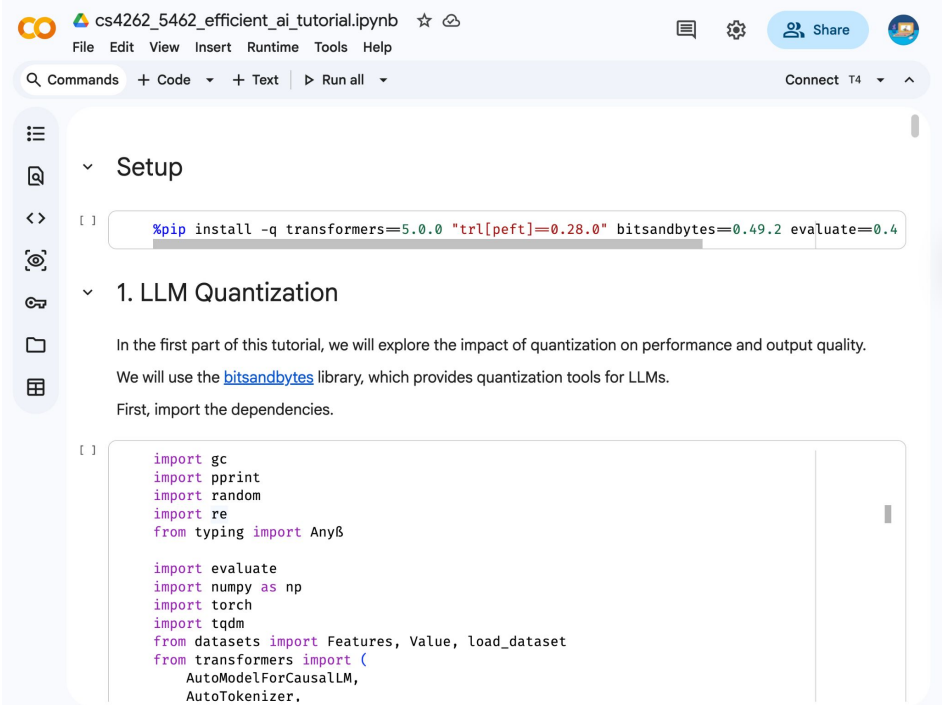
Tutorial Notebook

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

Link:
https://colab.research.google.com/drive/167njzSQTnEguJtDCtZ9xSwpS7hdYd1_H?usp=sharing

Make a copy and connect to the GPU runtime.

The tutorial notebook may take over an hour to run, so try with a smaller dataset size for debugging.



Outline

01
LLM quantization

02
Knowledge distillation

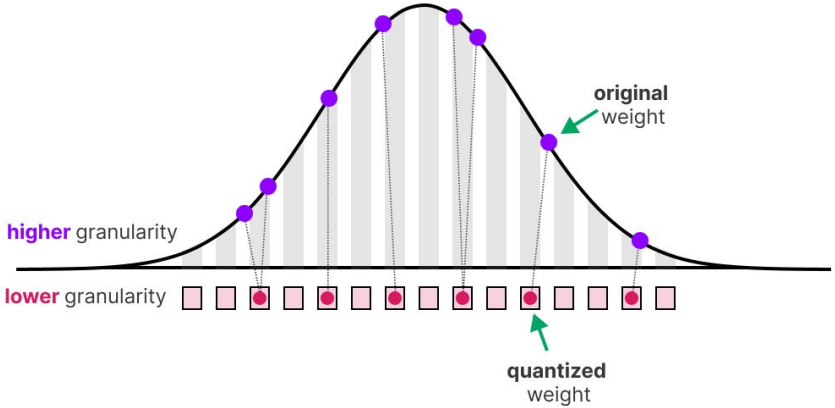
03
LLM cascading

04
Tutorial guide

05
Submission

Model Quantization

- **Quantization** maps high-precision values to low-precision ones to reduce memory footprint and improve computational efficiency.
- In the context of LLM, quantization is applied on the model weights and/or activations, reducing the model size.

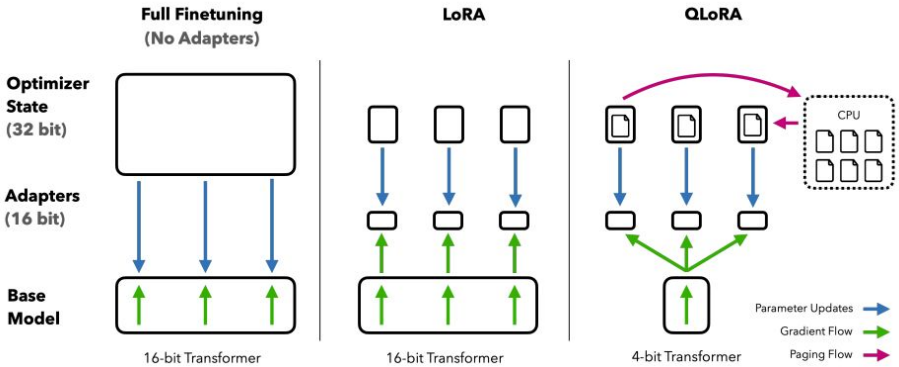


Model Quantization

- Popular LLM quantization techniques:
  - AWQ
  - GPTQ

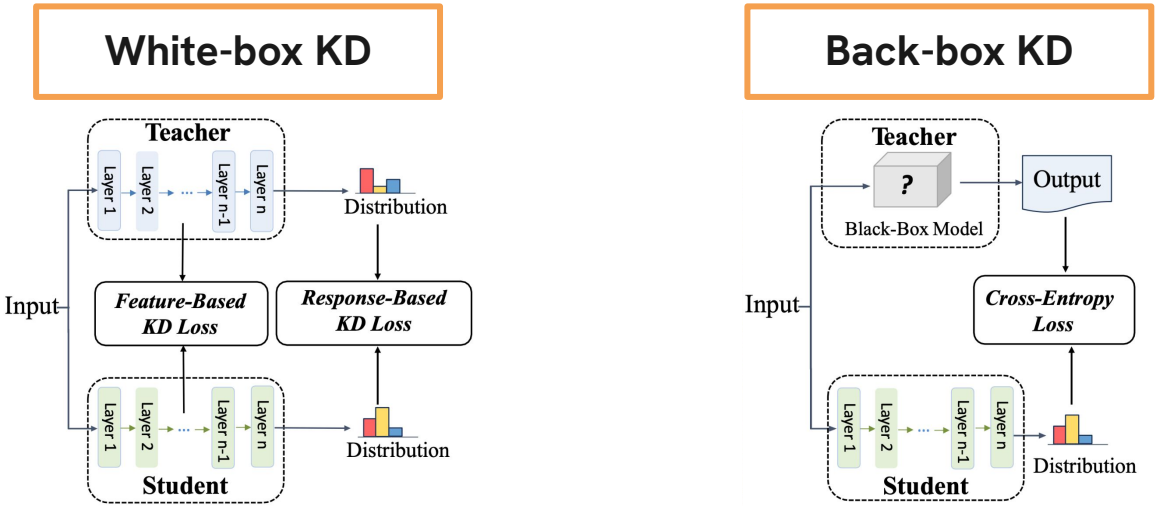
  } advanced; require calibration data

  - bitsandbytes — simple integration; data-free
- This tutorial focuses on bitsandbytes's 4-bit quantization and QLoRA.
- bitsandbytes does not require calibration data and is well-integrated with HF.



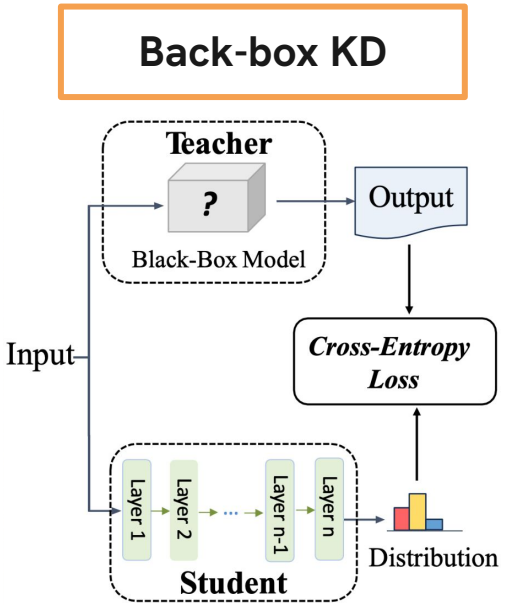
Knowledge Distillation

- **Knowledge distillation (KD)** is another compression technique that trains a small **“student”** model to mimic a large **“teacher”** model.
- KD improves the performance of the student model, eliminating the need to use the large model.



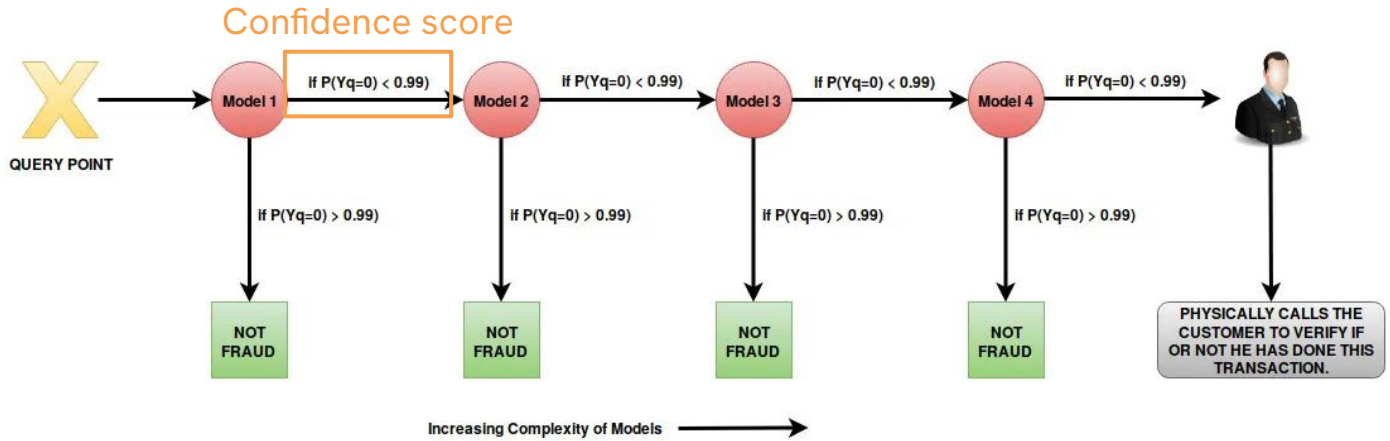
Knowledge Distillation

- In the context of LLM, SOTA models (GPT, Claude, Gemini) often live behind an API.
  - No access to the feature maps and, sometimes, logits.
  - **Black-box teachers**
- **Black-box KD** involves training a small model on outputs sampled from a black-box teacher.
  - i.e., SFT on the teacher's outputs.



LLM Cascading

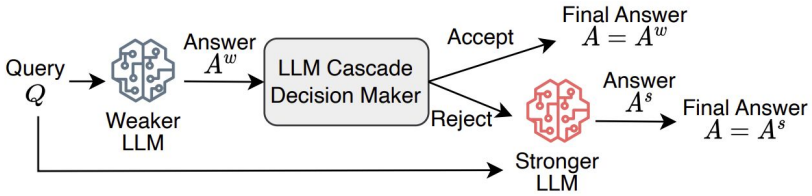
- **Model cascading** is an ML optimization technique that employs a sequence of small, low-cost models to large, expensive models.
  - Achieves balance between output quality and inference cost.
  - Route to a more sophisticated model based on a **confidence score**.



From https://medium.com/@saugata.paul1010/ensemble-learning-bagging-boosting-stacking-and-cascading-classifiers-in-machine-learning-9c66cb271674

LLM Cascading

- **LLM cascading**
  Use a weaker model (e.g., GPT-3.5-turbo) to answer simple queries while routing difficult queries to a stronger model (e.g., GPT-4).
  - **FrugalGPT**: trains a small regression model to calculate the confidence score.
  - **MoT Cascade**: employs answer consistency as the routing signal for reasoning (mathematical, causal) tasks.
    - Based on **self-consistency** and other prompting methods.



Confidence score

$$s = \frac{\sum_{i=1}^{K} \mathbb{1}_{A_i^w = A^w}}{K}$$

sampled answer
most consistency answer
# of samples

Tutorial Guide

In this tutorial, we will develop an efficient LLM for answering mathematical problems.

- **Part 1: LLM Quantization**
  - Load and use a quantized LLM
- **Part 2: Knowledge Distillation**
  - Distill a teacher model into a small quantized model
- **Part 3: LLM Cascading**
  - Build an LLM cascade of a small and a large model

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

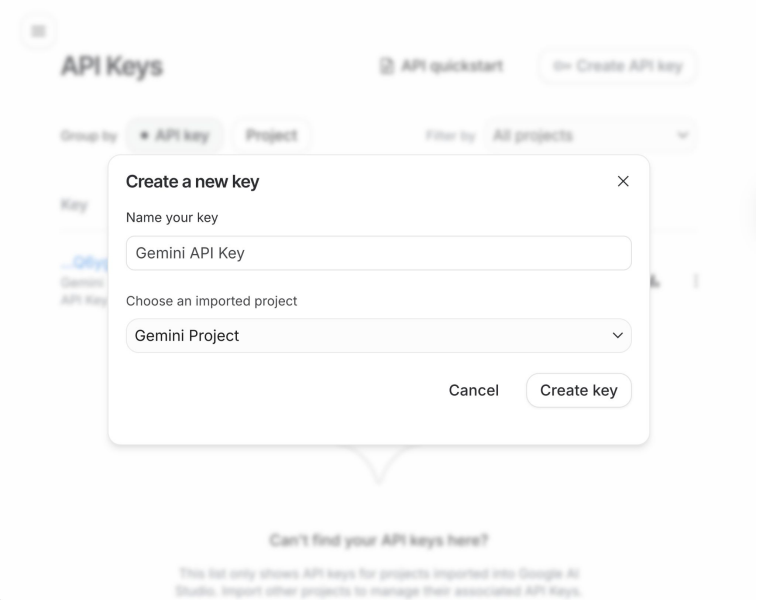
Tutorial Guide

We will use gemma-3-27b-it through Google's Gemini API as the teacher model during KD and the stronger model in the LLM cascade.

To use Gemini API,

- Create an API key (for free) on Google AI Studio
- Configure **Colab Secrets** by adding an environment variable:
  - Name: GEMINI_API_KEY
  - Value: <YOUR_API_KEY>

  Also enable notebook access.



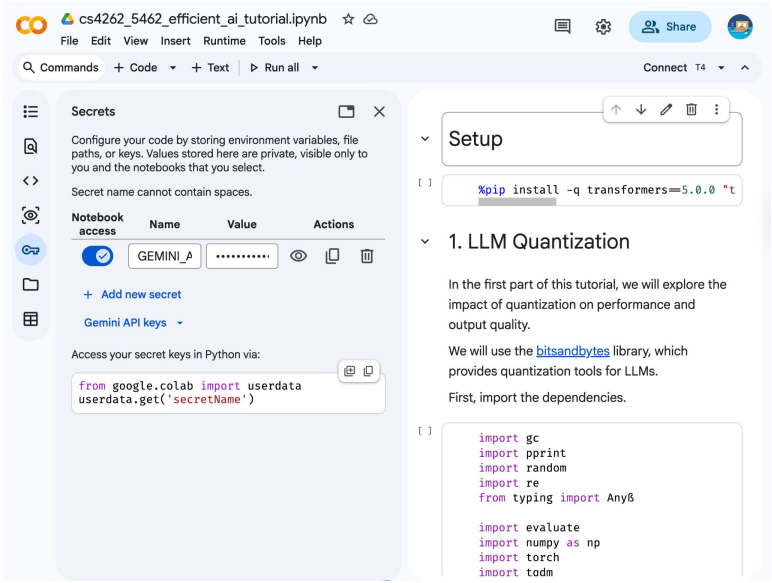
Tutorial Guide

We will use gemma-3-27b-it through Google's Gemini API as the teacher model during KD and the stronger model in the LLM cascade.

To use Gemini API,

- Create an API key (for free) on Google AI Studio
- Configure **Colab Secrets** by adding an environment variable:
  - Name: GEMINI_API_KEY
  - Value: <YOUR_API_KEY>

  Also enable notebook access.



Tutorial Guide

Part 1: LLM Quantization

- We will load and evaluate a 4-bit quantized version of an LLM (Qwen/Qwen2.5-1.5B-Instruct) on a **math QA task**.
- We will use the openai/gsm8k dataset, which contains math word problems that require multi-step reasoning.

```
print("Example question:")
pprint.pprint(dataset[0]["question"])
print("-" * 50)
print("Example answer:")
pprint.pprint(dataset[0]["answer"])
```

```
Example question:
('Mimi picked up 2 dozen seashells on the beach.  Kyle found twice as many '
 'shells as Mimi and put them in his pocket. Leigh grabbed one-third of the '
 'shells that Kyle found.  How many seashells did Leigh have?')
--------------------------------------------------
Example answer:
('Mimi has 2 x 12 = <<2*12=24>>24 sea shells.\n'
 'Kyle has 24 x 2 = <<24*2=48>>48 sea shells.\n'
 'Leigh has 48 / 3 = <<48/3=16>>16 sea shells.\n'
 '#### 16')
```

Tutorial Guide

Part 1: LLM Quantization

- **(Task 1)**

  1.1.2 Transform the dataset (10pt)

  - The dataset is not in a useable format.
  - Transform the *question* and *answer* column into the *prompt* and *ground-truth* columns.

```
print("Example question:")
pprint.pprint(dataset[0]["question"])
print("-" * 50)
print("Example answer:")
pprint.pprint(dataset[0]["answer"])
```

```
Example question:
('Mimi picked up 2 dozen seashells on the beach.  Kyle found twice as many '
 'shells as Mimi and put them in his pocket. Leigh grabbed one-third of the '
 'shells that Kyle found.  How many seashells did Leigh have?')
--------------------------------------------------
Example answer:
('Mimi has 2 x 12 = <<2*12=24>>24 sea shells.\n'
 'Kyle has 24 x 2 = <<24*2=48>>48 sea shells.\n'
 'Leigh has 48 / 3 = <<48/3=16>>16 sea shells.\n'
 '#### 16')
```

```
def extract_final_answer(answer: str) -> str:
    """Extract the final answer from the model's output

    The output is expected to contain a final answer prefixed by "#### ".
    """
    match = re.search(r"#### (\S+)", answer)
    if match:
        return match.group(1).strip()
    return answer.strip()


def to_prompt_ground_truth_pair(sample: dict[str, Any]) -> dict[str, Any]:
    """Convert a sample from the dataset into a prompt-ground truth pair by transforming
    the `question` and `answer` fields

    See the above description for the expected format of the prompt and ground truth.
    """
    prompt = None
    ground_truth = None
    # TODO: Add your code here
    # ═══ Start of your code ═══
    raise NotImplementedError()
    # ═══ End of your code ═══
    return {"prompt": prompt, "ground_truth": ground_truth}
```

Tutorial Guide

Part 2: Knowledge Distillation

We will distill a large model (gemma-3-27b-it) into the quantized 1.5B model.

- **(Task 2)**
  2.1 Build distilled dataset (20pt)
  - Build the distilled dataset by querying the teacher model with each question in the dataset.
  - (Optional) Filter the training data based on the teacher's output quality.

```python
def query_teacher(
    client: genai.Client, question: str, generation_config: GenerationConfig
) -> str:
    """Query the teacher model with retries and error handling"""
    try:
        return query_teacher_once(client, question, generation_config)
    except Exception:
        print(f"ERROR: failed to query teacher for question: {repr(question)}")
        raise


def create_distilled_dataset(client: genai.Client, dataset: Dataset) -> Dataset:
    """Create a distilled dataset by querying the teacher for each question in the
    input dataset (using the `question` column)
    """
    # Store outputs from the teacher model (including reasoning and final answer)
    outputs: list[str] = []

    # TODO: Add your code here
    # ═══ Start of your code ═══
    raise NotImplementedError()
    # ═══ End of your code ═══

    completion = [[{"role": "assistant", "content": output}] for output in outputs]
    dataset = dataset.add_column("completion", completion)
    return dataset


def filter_sample(sample: dict[str, Any]) -> bool:
    """Filter out samples based on the teacher's output quality (`completion` column),
    e.g., by checking the correctness of the final answer"""
    # TODO(Optional): Add your code here
    # ═══ Start of your code ═══
    return True
    # ═══ End of your code ═══
```

Tutorial Guide

Part 2: Knowledge Distillation

- **(Task 3) 2.2 SFT configurations**
  - Set the hyperparameters to fine-tune the student model.

- **2.4 Evaluate the student model (20pt)**
  - `Exact Match`
    - **20**: >= 0.30
    - **10**: >= 0.15 and < 0.30
    - **0**: < 0.15

```
[ ]
# 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
student_model_name = "qwen-distilled-lora-sft"
# TODO: Modify these hyperparameters
# ═══ Start of your code ═══
num_train_epochs = 1
max_steps = -1
per_device_train_batch_size = 8
# Hyperparameters
optim = "paged_adamw_8bit"
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 Guide

Part 3: LLM Cascading

- **(Task 4)**
  3.1 Confidence scoring (20pt)
  - Implement the confidence scoring function.
  - The confidence score is calculated by sampling multiple outputs from the student model and calculate the ratio of the most consistent answer.

$$s = \frac{\sum_{i=1}^{K} \mathbb{1}_{A_i^w = A^w}}{K}$$

```python
def student_answer_and_confidence(
    model: PreTrainedModel,
    messages: list[dict[str, str]],
    generation_config: GenerationConfig,
    num_samples: int,
) -> tuple[str, float]:
    """Given a student model and input messages, return the student's answer and
    confidence score

    The confidence score is calculated by sampling multiple answers from the student
    model and calculating the vote share of the most consistent answer.

    Parameters
    ----------
    model: PreTrainedModel
        The student model to query
    messages: list[dict[str, str]]
        The messages to send to the student model
    generation_config: GenerationConfig
        The generation config to use when sampling answers from the student model
    num_samples: int
        The number of samples to draw from the student model

    Returns
    -------
    answer: str
        The student's answer to the question (the final answer extracted from the
        model's output)
    confidence: float
        The student's confidence
    """
    # TODO: Add your code here
    # ═══ Start of your code ═══
    raise NotImplementedError()
    # ═══ End of your code ═══
```

Tutorial Guide

Part 3: LLM Cascading

- **(Task 5)**
  3.2 Cascade policy (30pt)
  - Adjust the LLM cascading policy to balance the output quality and the inference cost.
  - Grading criteria:

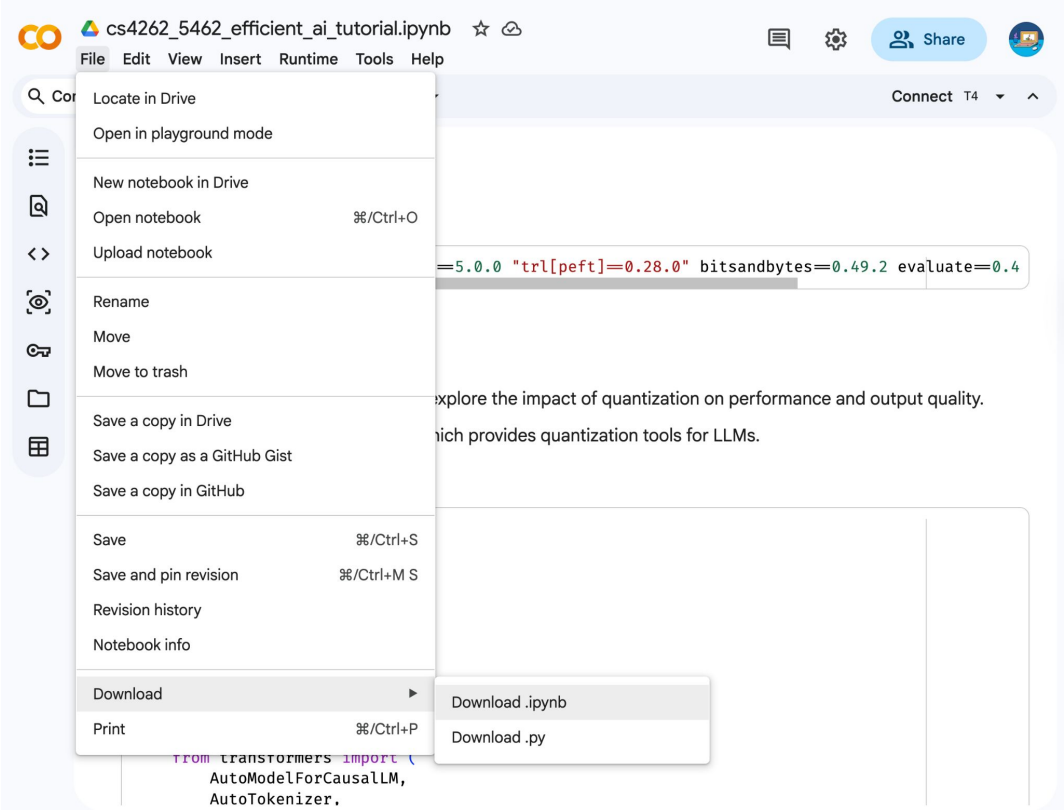
1. `Overall EM`
   - **15**: ≥ 0.6
   - **7.5**: ≥ 0.3 and < 0.6
   - **0**: < 0.3
2. `Cost savings`
   - **15**: ≥ 0.3
   - **7.5**: ≥ 0.15 and < 0.3
   - **0**: < 0.15

```
[ ]
# TODO: Tune the following parameters
# ═══ Start of your code ═══
STUDENT_GEN_CONFIG = GenerationConfig(
    max_new_tokens=GEN_CONFIG.max_new_tokens,
    do_sample=True,
    temperature=0.7,
    top_p=0.9,
    use_cache=True,
)
STUDENT_NUM_SAMPLES = 5
STUDENT_CONFIDENCE_THRESHOLD = 0.5
# ═══ End of your code ═══
```

Submission

- Fill in five tasks.
  - 1.1.2 Transform the dataset (10pt)
  - 2.1 Build distilled dataset (20pt)
  - 2.2 SFT configurations ⇒ 2.4 Evaluate the student model (20pt)
  - 3.1 Confidence scoring (20pt)
  - 3.2 Cascade policy (30pt)
- 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


cs4262_5462_efficient_ai_tutorial.ipynb
File Edit View Insert Runtime Tools Help
Share
Connect T4
Locate in Drive
Open in playground mode
New notebook in Drive
Open notebook ⌘/Ctrl+O
Upload notebook
Rename
Move
Move to trash
Save a copy in Drive
Save a copy as a GitHub Gist
Save a copy in GitHub
Save ⌘/Ctrl+S
Save and pin revision ⌘/Ctrl+M S
Revision history
Notebook info
Download
Print ⌘/Ctrl+P
Download .ipynb
Download .py
=5.0.0 "trl[peft]=0.28.0" bitsandbytes=0.49.2 evaluate=0.4
xplore the impact of quantization on performance and output quality.
ich provides quantization tools for LLMs.
from transformers import (
AutoModelForCausalLM,
AutoTokenizer,

References

- Quantization: https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-quantization
- bitsandbytes: https://huggingface.co/docs/transformers/en/quantization/bitsandbytes
- QLoRA: https://openreview.net/forum?id=OUIFPHEgJU
- Black-box KD: https://arxiv.org/abs/2401.07013
- FrugalGPT: https://openreview.net/forum?id=cSimKw5p6R
- MoT Cascade: https://openreview.net/forum?id=6okaSfANzh
- Self-consistency: https://arxiv.org/abs/2203.11171

THANK YOU