Task 3 : Training Process Explanation and Justifications

TABLE OF CONTENT

1. Introduction

This section explains the training process for fine-tuning Qwen 2.5 3B on AI research QA tasks. Each step is justified, and relevant code snippets are included.

2. Environment Setup

2.1 Justification

- \`unsloth\`, \`datasets\`, \`trl\`, \`torch\`, and \`transformers\` are required for efficient fine-tuning.
- \`unsloth\` is chosen because it offers optimized loading, reduced memory footprint, and built-in LoRA support.

```
!pip install unsloth datasets trl torch transformers
```

3. Model Loading

3. 1 Justification

- Qwen2.5-3B-Instruct is chosen for its instruction-following capability.
- load_in_4bit=True applies 4-bit quantization, reducing VRAM usage.
- max_seq_length=2048 enables long-context comprehension.

```
from unsloth import FastLanguageModel
import torch

max_seq_length = 2048
dtype = None  # Auto detection
load_in_4bit = True

model, tokenizer = FastLanguageModel.from_pretrained(
    model_name="unsloth/Qwen2.5-3B-Instruct",
    max_seq_length=max_seq_length,
    dtype=dtype,
    load_in_4bit=load_in_4bit)
```

4. Applying LoRA (Low-Rank Adaptation)

4.1 Justification

- LoRA fine-tunes only selected layers (q_proj, k_proj, v_proj), reducing computational cost.
- r=16 sets the rank of LoRA matrices, balancing efficiency and expressiveness.
- use_gradient_checkpointing="unsloth" reduces memory usage for long sequences.

```
model = FastLanguageModel.get_peft_model(
    model,
    r=16,
    target_modules=["q_proj", "k_proj", "v_proj", "o_proj",
                    "gate_proj", "up_proj", "down_proj"],
    lora_alpha=16,
    lora_dropout=0,
    bias="none",
    use_gradient_checkpointing="unsloth",
    random_state=3407,
    use_rslora=False,
    loftq_config=None,
)
```

5. Dataset Preparation

5.1 Justification

- Data is formatted into prompt-response pairs for instruction tuning.
- EOS token ensures proper sequence termination.

```
from datasets import load_dataset

deep_prompt = """Below is a question about AI research. Answer
accordingly.\n### Question:\n{}\n### Answer:\n{}"""
EOS_TOKEN = tokenizer.eos_token

def formatting_prompts_func(examples):
    questions = examples["question"]
    answers = examples["answer"]
    texts = [deep_prompt.format(q, a) + EOS_TOKEN for q, a in
zip(questions, answers)]
    return {"text": texts}
# Load dataset
dataset = load_dataset("json", data_files={'train': 'train.json',
'validation': 'validate.json'})
dataset = dataset.map(formatting_prompts_func, batched=True)
```

6. Training Configuration

6.1 Justification

- per_device_train_batch_size=2 : Prevents VRAM overflow.
- gradient_accumulation_steps=4 : Simulates larger batch size for stable training.
- learning_rate=2e-4 : Optimal for LoRA fine-tuning without overfitting.
- num_train_epochs=1 : Prevents overfitting and enables rapid adaptation.
- optim="adamw_8bit" : Optimized memory-efficient optimizer.

```
from trl import SFTTrainer
from transformers import TrainingArguments

training_args = TrainingArguments(
    per_device_train_batch_size=2,
    gradient_accumulation_steps=4,
    warmup_steps=5,
    max_steps=60,
    learning_rate=2e-4,
    fp16=True,
    logging_steps=1,
    optim="adamw_8bit",
    weight_decay=0.01,
    lr_scheduler_type="linear",
    seed=3407,
    output_dir="outputs",
    evaluation_strategy="steps",
    metric_for_best_model="eval_loss"
)
```

7. Training Execution

7.1 Justification

- SFTTrainer simplifies supervised fine-tuning.
- Uses formatted dataset and tokenizer.

```
trainer = SFTTrainer(
    model=model,
    tokenizer=tokenizer,
    train_dataset=dataset['train'],
    eval_dataset=dataset['validation'],
    dataset_text_field="text",
    max_seq_length=max_seq_length,
    dataset_num_proc=2,
    packing=False,
    args=training_args,
)

# Start training
trainer.train()
```

8. Conclusion

This training process fine-tunes Qwen 2.5 3B efficiently using LoRA and 4-bit quantization, ensuring a balance between computational efficiency and model accuracy. Each step was optimized for performance, resource management, and improved generalization in AI research QA tasks.

9. References

[1] https://huggingface.co/transformers/v3.2.0/custom_datasets.html#qa-squad
[2]
https://medium.com/@manindersingh120996/practical-guide-to-fine-tune-llms-with-lora-c835a99d7593
[3]
https://www.reddit.com/r/LocalLLaMA/comments/1fnvlla/qwen25_bugs_issues_fixes_colab_finetuning_notebook/
[4]
https://www.kaggle.com/code/ysthehurricane/step-by-step-huggingface-model-fine-tune-guide