Comprehensive Training Process for Qwen2.5-3B-Instruct Fine-tuning

1. Introduction

This document presents an in-depth report on fine-tuning the **Qwen2.5-3B-Instruct** model to enhance its ability to answer AI research-related queries. The fine-tuning process utilized **Unsloth** for efficient training, incorporating **QLoRA (4-bit quantization)** for optimal memory management and practical deployment. The goal was to improve the model's ability to generate accurate, research-driven responses by leveraging fine-tuned datasets derived from AI research papers, blogs, and technical documents.

2. Model Selection

2.1 Choice of Model

Selected Model: Qwen/Qwen2.5-3B-Instruct

- This model was chosen due to its **instruction-following capability**, which is crucial for answering structured AI research questions.
- The **instruct variant** was preferred over the base model to leverage its pre-trained instruction-following behavior.
- The model has a **3B parameter scale**, making it a feasible choice for fine-tuning on a single or multi-GPU setup.

2.2 Justification for Quantization (QLoRA)

- **Memory Efficiency:** Enables training on consumer GPUs with limited VRAM.
- **Minimal Performance Degradation:** Maintains the effectiveness of full precision models while significantly reducing memory overhead.
- **Allows for Gradient Updates:** Unlike static quantization, QLoRA allows fine-tuning while keeping computational demands low.
- **Faster Inference Speeds:** Post-training inference with quantization allows for more efficient deployments.

3. Dataset Preparation

3.1 Data Sources

The dataset was derived from multiple sources to ensure comprehensive coverage of AI research topics:

- **Academic Research Papers**: Extracted from PDFs using `PyMuPDF` (fitz) and OCR fallback via `pytesseract` for scanned documents.
- **Technical AI Blogs & Documentation**: Processed from Markdown (`md`) files using `markdown` and `re` libraries to extract structured content.
- **Synthetic Q&A Pairs**: Automatically generated from extracted texts using prompt engineering and NLP techniques(using models like `valhalla/t5-base-qg-hl`)

3.2 Data Preprocessing

The extracted data underwent multiple preprocessing steps to ensure compatibility with fine-tuning:

3.2.1 Text Extraction & Cleaning

- **PDF Parsing**: Utilized `fitz` (PyMuPDF) to extract text from research paper PDFs.
- **OCR for Scanned PDFs**: Applied `pytesseract` OCR on images extracted from PDFs to retrieve non-selectable text.
- **Markdown Processing**: Stripped HTML tags and reformatted Markdown files into clean text.
- **Regex-Based Cleaning**: Removed unnecessary symbols, footnotes, and HTML artifacts to enhance readability.

3.2.2 Tokenization & Formatting

- Applied **sentence segmentation** using `nltk` to break content into logically structured parts.
- Implemented **text normalization** to remove inconsistencies in whitespace, special characters, and casing.
- Converted extracted data into structured **instruction-response format** using `formatting_prompts_func`.
- Parallelized batch processing with `num_proc=2` for efficiency.

```python
from datasets import load_dataset

dataset = load_dataset("json", data_files="output.json")
dataset = dataset["train"].train_test_split(test_size=0.2, seed=3407)
train_dataset = dataset["train"]
```

```
• eval_dataset = dataset["test"]
•
• train_dataset = train_dataset.map(formatting_prompts_func,
  batched=True, num_proc=2)
• eval_dataset = eval_dataset.map(formatting_prompts_func,
  batched=True, num_proc=2)
```

3.3 Data Augmentation Techniques

To improve model generalization and response quality, various augmentation techniques were applied:

3.3.1 Paraphrasing

- Used NLP-based transformations to **reword AI research questions** while preserving semantic meaning.
- Ensured diverse wording for similar topics to enhance model robustness.

3.3.2 Synthetic Q&A Generation

- Applied prompt-based synthetic generation to create **multiple variations of research-based questions and answers**.
- Used `transformers` models to generate **alternative answers** to the same questions for variety.

3.3.3 Noise Injection

- Introduced **intentional typos and varied punctuation** to improve model robustness against imperfect user inputs.
- Included slight grammatical errors to reflect real-world user queries.

3.3.4 Entity Replacement Augmentation

- Replaced **specific AI research terms** with analogous terms to test generalization.
- Example: Swapped "Transformer models" with "Sequence models" in specific training examples.

3.3.5 Adversarial Prompting

- Designed **edge-case prompts** where AI research topics were deliberately phrased ambiguously to assess reasoning capabilities.

- Example: "How does an AI model generate text?" vs. "Can an AI system generate text like humans?"

These preprocessing and augmentation techniques significantly enhanced the dataset's diversity, ensuring the fine-tuned model could handle a broad range of AI research-related questions.

4. Training Configuration

4.1 Hyperparameters & Justifications

Hyperparameter	Value	Justification
max_seq_length	2048	Ensures long-context retention for research papers.
dtype	None	Uses auto-detection of best data type.
load_in_4bit	True	Efficient memory usage with QLoRA.
batch_size	2	Optimal for GPU memory constraints.
learning_rate	2e-5	Standard for instruction-tuned models.
num_epochs	3	Balances overfitting risks and model improvement.
optimizer	AdamW	Standard optimizer for fine-tuning transformers.
gradient_accumulation	16	Helps in reducing memory usage during training.
Evaluation Strategy	Epoch-based	Ensures regular assessment of model progress.

4.2 Training Strategy

- **QLoRA (Quantized Low-Rank Adaptation)** was applied to reduce VRAM usage while maintaining training effectiveness.
- **Unsloth** was used for accelerated fine-tuning by patching model internals for speed optimization.
- **Trainer:** `SFTTrainer` from `trl` was used for supervised fine-tuning.

Model Loading & Training Code

```python
from unsloth import FastLanguageModel

from trl import SFTTrainer
```

```python
from transformers import TrainingArguments


# Load Model with QLoRA

model, tokenizer = FastLanguageModel.from_pretrained(

    model_name="Qwen/Qwen2.5-3B-Instruct",

    max_seq_length=2048,

    dtype=None,

    load_in_4bit=True,

)
```

Python

```python
# Define Training Arguments

training_args = TrainingArguments(

    per_device_train_batch_size=2,

    per_device_eval_batch_size=2,

    num_train_epochs=3,

    learning_rate=2e-5,

    gradient_accumulation_steps=16,

    evaluation_strategy="epoch",

    save_strategy="epoch",

    logging_dir="./logs",
```

```
    report_to="none"
)

trainer = SFTTrainer(
    model=model,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
    args=training_args,
    tokenizer=tokenizer,
)

trainer.train()
```

5. Evaluation & Results

5.1 Validation Metrics & Performance Analysis

Step	Training Loss	Validation Loss
20	2.680300	2.013773
40	1.239000	1.190110
60	1.007400	1.062270
80	0.937300	1.013381
100	0.851800	0.989729
120	0.800700	0.981624
140	0.773800	0.981477
160	0.769300	0.981282

Observations:

- **Steady Decrease in Training Loss:** Indicates successful fine-tuning.
- **Validation Loss Plateau:** Suggests a stable model performance beyond 100 steps.
- **Potential Further Optimization:** Loss plateauing could suggest hyperparameter tuning opportunities.

6. Optimization Techniques

- **Efficient Fine-Tuning with QLoRA** to reduce memory usage.
- **Gradient Accumulation** to balance batch size limitations.
- **Chat Template Adjustments** to better fit instruction-based responses.
- **Paraphrased Data Augmentation** to enhance response diversity.
- **Early Stopping Considerations** to prevent unnecessary training beyond convergence.

7. Conclusion

This fine-tuning process successfully optimized the **Qwen2.5-3B-Instruct** model for AI research Q&A. The model shows improved reasoning and answer accuracy based on the provided technical documents. The training methodology, dataset preprocessing, and optimization techniques ensured an efficient, high-performing adaptation of the base model for research-focused tasks.