# Enhancing Qwen 2.5 3B-Instruct for AI Research QA: A Fine-Tuning Approach

**Abstract**

In this paper, we present a fine-tuning methodology to enhance Qwen 2.5 3B-Instruct for AI research question-answering (QA). Given the inherent limitations in available datasets, we opted for Qwen 2.5 3B-Instruct instead of the base model. We detail our data preparation strategy, leveraging Gemini API for synthetic dataset generation, and discuss the implementation of fine-tuning using Unsloth. We analyze our hyperparameter choices and the rationale behind them, highlighting their impact on model performance and efficiency.

## 1 Introduction

Large Language Models (LLMs) have revolutionized natural language understanding, yet their generalization ability often necessitates task-specific fine-tuning. Our objective was to adapt Qwen 2.5 3B-Instruct for AI research QA, ensuring high-quality, domain-specific responses. We carefully crafted a dataset, defined structured training pipelines, and optimized hyperparameters to maximize performance.

## 2 Data Preparation

Given the scarcity of well-structured AI research QA datasets, we generated synthetic data using the Gemini API. The dataset creation process involved:

- **Chunking Research Content**: To ensure diversity and broad coverage.
- **Generating Questions**: Extracting at least 20 distinct questions per chunk.
- **Formatting**: Separating questions from answers for structured training.
- **CSV Storage**: Storing the processed data for efficient training.

## 3 Model Selection

We selected **Qwen 2.5 3B-Instruct** over the base model due to its superior instruction-following capabilities, which align well with the QA task. Fine-tuning an instruct model provides a head start compared to training from scratch, reducing computational costs and training time.

## 4 Fine-Tuning Implementation

We leveraged **Unsloth** for its efficient training pipeline. Our approach included:

- Installing dependencies:

```
!pip install "unsloth[colab-new] @ git+https://github.com/unslothai/unsloth.
!pip install --no-deps xformers "trl<0.9.0" peft accelerate bitsandbytes
```

- Loading and formatting the dataset:

```
from datasets import Dataset
dataset = Dataset.from_pandas(df)
dataset = dataset.map(formatting_prompts_func, batched=True)
```

- Defining the training prompt format:

```
alpaca_prompt = """

Instruction:

{}

Input:

{}

Response:

{}"""
```

## 5 Hyperparameter Selection and Justification

The choice of hyperparameters played a crucial role in balancing training efficiency and model accuracy:

- **Batch Size (2)**: Due to memory constraints on consumer GPUs, a smaller batch size allows gradient accumulation without overloading the hardware.

  - **Gradient Accumulation Steps (4)**: Effectively increases batch size while maintaining stability.
  - **Learning Rate (2e-4)**: Optimized for fast yet stable convergence without overshooting.
  - **Warmup Steps (5) & Max Steps (60)**: Ensures smooth learning rate scheduling, avoiding sudden weight updates.
  - **Weight Decay (0.01)**: Helps mitigate overfitting by controlling parameter magnitude.
  - **FP16/BF16 Precision**: Chosen dynamically based on hardware capabilities, reducing memory usage while maintaining numerical stability.
- **Optimizer (AdamW**$_8bit) : Enhancestrainingefficiencybyleveragingreduced-precisioncalculations.$

# 6 Model Loading and Inference

After training, we loaded the fine-tuned model for evaluation:

```
model, tokenizer = FastLanguageModel.from_pretrained(
model_name = "lora_model",
max_seq_length = max_seq_length,
dtype = dtype,
load_in_4bit = load_in_4bit,
)
```

Inference was tested using:

```
inputs = tokenizer([
alpaca_prompt.format(
"You are a helpful assistant in Q@A about AI.",
"What additional attribute is proposed for directory inodes to aid in loop detec
"",
)
], return_tensors = "pt").to("cuda")
```

# 7 Results and Conclusion

Our fine-tuned model demonstrated improved performance in AI research QA, with enhanced response accuracy and contextual understanding. The strategic selection of hyperparameters, dataset structuring, and model selection played a vital role in achieving this improvement. Future work includes fine-tuning on larger datasets and integrating retrieval-augmented generation for even more robust performance.

By employing a structured, methodical approach, we have successfully adapted Qwen 2.5 3B-Instruct into a powerful AI research assistant.