# ESE577-Oriented Q&A Chatbot Finetuning

**Haotian Fu**
Department of Electrical and Computer Engineering
Stony Brook University
Stony Brook, NY 11794
`haotian.fu@stonybrook.edu`

## Abstract

This project focuses on finetuning the Mistral-7B-Instruct-v0.2 language model to specialize in domain-specific question-answering for the ESE 577 course. By leveraging Low-Rank Adaptation (LoRA) and quantization, the model was optimized to run on limited hardware resources while maintaining high accuracy. The dataset comprised both synthetic and augmented question-answer pairs generated from course content. Numerical evaluation using ROUGE metrics and human evaluation witnessed general over 200% improvements compared with base models in the fine-tuned model's ability to answer ESE 577-specific queries compared to the base model. Codes are available on `https://github.com/Haotian-Fu/ESE577-Project.git`.

## 1 Introduction

In this project, we leverage the Parameter Efficient Fine-Tuning (PEFT) method, especially Low Rank Adaptation (LoRA), to finetune the base model, `mistralai/Mistral-7B-Instruct-v0.2`, on a self-designed dataset related to course ESE577. LoRA was chosen for its ability to significantly reduce the number of trainable parameters by adapting only low-rank subspaces of the model, making it ideal for resource-constrained environments. By introducing minimal overhead, LoRA allows for efficient training while maintaining the expressive power of the base model.

To further optimize memory usage, we employed 4-bit quantization using the NF4 format. Quantization reduces the precision of model weights while retaining essential representational capacity, enabling the training of large-scale models like Mistral-7B on consumer-grade GPUs with limited VRAM. This approach allowed us to fit the model into an 24 GB GPU without compromising performance.

Additional choices included:

- Data Augmentation: Synonym replacement using WordNet was applied to diversify the dataset, enhancing robustness and generalization.
- Custom Dataset Design: A carefully curated dataset of ESE577-related question-answer pairs was developed, including augmented samples to increase the training data size.
- Loss Tracking: A custom callback was used to record training and validation losses for detailed analysis and visualization.

By combining PEFT, quantization, and tailored data preprocessing, the project demonstrates a scalable and efficient approach to adapting large language models for domain-specific tasks, particularly in educational settings.

## 2 Method

### 2.1 Dataset Introduction

The dataset used for this project was derived primarily from the ESE577 syllabus, supplemented by relevant course slides. The selection of materials aimed to include specific and detailed descriptions to improve fine-tuning performance by tailoring the dataset to the course's context.

The data was formulated as question-answer pairs (QA pairs) and initially stored in the `QA_pair.xlsx` file (courtesy of Guoao Li's project group). This file contained 100 QA pairs generated from the syllabus and course slides. Recognizing the small size of this dataset, we applied synonym-based data augmentation to create paraphrased versions of the original 100 pairs. The augmented dataset doubled the total number of pairs to 200, ensuring greater diversity and robustness. After augmentation, the final dataset contains 200 QA pairs, evenly split between original and paraphrased pairs. Data augmentation techniques will be introduced in *Data Preprocessing* part.

### 2.2 Data Preprocessing

As is mentioned, data is stored in "xlsx" format, which is not frinedly to python script processing. The data preprocessing phase focused on transforming the dataset from its initial Excel format into a structured and augmented dataset compatible with the fine-tuning pipeline. The following steps were undertaken:

1. Dataset Loading: The QA dataset was loaded from the file `QA_pair.xlsx`. This file contained two key columns: Question and Answer.
2. Data Validation: The dataset was validated to ensure it included the required columns (`Question` and `Answer`). A validation check was performed to avoid missing or misaligned data.
3. Conversion to JSON Format:
   - Each row of the dataset was converted into a dictionary format, where each QA pair was represented as:
     ```
     {
         "question": "Who is the instructor of this course?  ",
         "answer": "Jorge Mendez-Mendez."
     }
     ```
   - This structure was saved in a JSON file, `QA_dataset.json`, ensuring compatibility with the tokenizer and downstream processing.
4. Data Augmentation:
   - Synonym replacement was applied using the WordNet-based `naw.SynonymAug` augmenter from the `nlpaug` library. This approach increased dataset diversity by paraphrasing questions and answers.
   - For each original QA pair, an augmented version was created with synonym substitutions. This effectively doubled the dataset size from 100 pairs to 200 pairs.
5. Shuffling: After augmentation, the dataset was shuffled to ensure randomization, which helps prevent model overfitting to the original ordering of the data.
6. Final JSON Output: The combined dataset (original and augmented pairs) was saved as `QA_dataset.json`, providing a ready-to-use format for model fine-tuning.

**Final Dataset Statistics:**

After augmentation, the dataset contained approximately double the number of entries compared to the original input. For instance, if the original dataset consisted of 100 QA pairs, the final dataset after augmentation included 200 QA pairs.

### 2.3 Training Hyperparameters

The hyperparameters were carefully selected to balance computational efficiency, model performance, and the constraints of the project. The computation platform we use is a 13th Gen Intel(R) Core(TM) i9-13900HX and a NVIDIA GeForce RTX 4070 Laptop GPU. System memory is 32 GB while the GPU memory is 24 GB. Once the calculations exceed GPU memory, it will be automaticall offloaded on CPU.

Considering the computation platform we have, a small learning rate ($2 \times 10^{-4}$) was chosen to ensure stable convergence, while a batch size of 2 per device with gradient accumulation (effective batch size of 4) accommodated GPU memory limitations. Different training epochs were explored to reach the sub-optimal computational efficiency. See *Experimental Results* section for more details. The result suggested that four epochs allowed the model sufficient exposure to the augmented dataset without overfitting.

LoRA-specific parameters, such as a rank of 8 and a scaling factor of 16, efficiently adapted the model's self-attention layers with minimal overhead. Regularization techniques, including a weight decay of 0.01 and a LoRA dropout of 0.1, mitigated overfitting. These hyperparameters are not guaranteed to be optimal due to time limit on this project and parameter optimization.

Early stopping with a patience of 5 epochs avoided unnecessary training, while mixed precision (FP16) and the AdamW optimizer ensured efficient resource utilization and stable updates. These choices were guided by best practices and tailored to the scale of the dataset and available computational resources.

In a brief summary, some hyperparameters are explored and determined by experimental results, but still not fully exploited and thus not optimal. The final selection of hyperparameter in LoRA finetuning is listed in Table 1.

Table 1: Hyperparameter Selection for Fine-Tuning

| **Hyperparameter** | **Value/Description** |
|---|---|
| Learning Rate | $2 \times 10^{-4}$ |
| Batch Size (Per Device) | 2 |
| Gradient Accumulation | 2 (Effective Batch Size = 4) |
| Number of Epochs | 3,4,6,8 |
| LoRA Rank ($r$) | 8 |
| LoRA Scaling Factor ($\alpha$) | 16 |
| Target Modules for LoRA | `self_attn.q_proj`, `self_attn.v_proj` |
| LoRA Dropout | 0.1 |
| Weight Decay | 0.01 |
| Evaluation Strategy | End of each epoch |
| Early Stopping Patience | 5 epochs |
| Mixed Precision | Enabled (FP16) |
| Save Strategy | Save at end of each epoch |
| Optimizer | AdamW |

## 2.4 Loss Function

The fine-tuning process employed the **causal language modeling (CLM) loss**, a variant of the cross-entropy loss tailored for autoregressive language generation tasks. This loss function computes the probability of predicting the correct next token in a sequence given all previous tokens.

Mathematically, let $x = (x_1, x_2, \ldots, x_T)$ represent a tokenized sequence of length $T$, where $x_t$ is the token at position $t$. The model generates a probability distribution over the vocabulary for the next token $x_t$ conditioned on the preceding tokens $x_{<t}$. The causal language modeling loss is defined as:

$$\mathcal{L}_{\text{CLM}} = -\frac{1}{T} \sum_{t=1}^{T} \log P(x_t \mid x_{<t}; \theta)$$

Here:

- $\theta$ represents the model parameters.

- $P(x_t \mid x_{<t}; \theta)$ is the probability assigned by the model to the true token $x_t$ given the context $x_{<t}$.

This loss function is particularly suited for tasks like generative question-answering because it aligns with the autoregressive nature of the model, where predictions are made sequentially. The model learns to maximize the likelihood of generating coherent and contextually accurate outputs.

To avoid including padding tokens in the loss computation, a mask was applied to ignore positions corresponding to padding tokens. This ensures that the gradients are only influenced by valid tokens in the sequence, leading to more stable optimization.

### 2.5 Evaluation Metric

To evaluate the quality of the model's generated answers, we used the **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)** metric. ROUGE metrics are designed to compare generated text against reference text, emphasizing token overlap and sequence structure. The choice of metrics—ROUGE-1, ROUGE-2, and ROUGE-L—was motivated by their ability to capture both content similarity and fluency in generated answers.

- **ROUGE-1**: Measures the unigram overlap between the generated answer $G$ and the reference answer $R$:
$$\text{ROUGE-1} = \frac{\text{Count of matching unigrams in } G \text{ and } R}{\text{Total unigrams in } R}$$
This metric provides a basic measure of content coverage.
- **ROUGE-2**: Measures the bigram overlap, capturing the sequence of tokens and short phrases:
$$\text{ROUGE-2} = \frac{\text{Count of matching bigrams in } G \text{ and } R}{\text{Total bigrams in } R}$$
By including bigram matches, ROUGE-2 evaluates the ability of the model to maintain contextual coherence.
- **ROUGE-L**: Measures the longest common subsequence (LCS) between $G$ and $R$, reflecting the structural similarity of the sequences. The formula involves precision and recall of LCS:
$$\text{ROUGE-L} = \frac{(1+\beta^2) \cdot \text{LCS-Precision} \cdot \text{LCS-Recall}}{\beta^2 \cdot \text{LCS-Precision} + \text{LCS-Recall}}$$
Where:
  - $\text{LCS-Precision} = \frac{\text{Length of LCS}}{\text{Length of } G}$
  - $\text{LCS-Recall} = \frac{\text{Length of LCS}}{\text{Length of } R}$
  - $\beta$ is a weighting factor (commonly set to 1).
- **BLEU (Bilingual Evaluation Understudy)**: Measures the precision of $n$-gram overlap between the generated text and reference text. BLEU focuses on the syntactic and lexical quality of the generated text by considering $n$-gram matches at different levels:
$$\text{BLEU} = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$
Where:
  - $p_n$: The precision of $n$-grams between the generated text and reference text.
  - $w_n$: Weight assigned to $n$-gram precision (commonly set to uniform weights).
  - $BP$: Brevity penalty to prevent favoring overly short text, calculated as:
$$BP = \begin{cases} 1 & \text{if } c > r, \\ \exp(1 - \frac{r}{c}) & \text{if } c \leq r, \end{cases}$$
where $c$ is the length of the generated text and $r$ is the length of the reference text.

BLEU was chosen as it evaluates both lexical accuracy and fluency while balancing precision across different $n$-gram sizes. It complements ROUGE by offering a more fine-grained measure of syntactic and lexical quality.

The combination of ROUGE and BLEU metrics provides a comprehensive evaluation framework. While ROUGE emphasizes recall and sequence overlap, BLEU provides additional insights into syntactic correctness and penalizes overly short or verbose answers. The ROUGE metrics were chosen for their effectiveness in evaluating text generation tasks, allowing us to assess the alignment between generated and reference answers. By combining unigram, bigram, and sequence-level measures, we obtain a holistic view of the model's performance in terms of content accuracy, contextual coherence, and fluency. The metrics also enable a direct comparison between the base and fine-tuned models, highlighting the improvements achieved through the fine-tuning process.

## 2.6 Data Analysis

The analysis of the model performance and responses was conducted using a systematic approach that involved both quantitative metrics and qualitative evaluation. To compare the performance of the base model and fine-tuned models, we implemented a robust pipeline to process and visualize the results. The analysis focused on two main aspects: metric-based comparison and response aggregation.

### 2.6.1 Metric-Based Comparison

To evaluate the quality of the generated text across models, we used ROUGE metrics (ROUGE-1, ROUGE-2, and ROUGE-L). The F1 scores for each metric were extracted from the output JSON files of the base and fine-tuned models. These scores provided insights into the content overlap and structural similarity between the generated and reference answers.

We plotted a bar chart to visually compare the F1 scores across models, highlighting the relative improvement achieved by fine-tuning. The plotting function ensured that the base model scores were positioned as a reference, while the scores of multiple fine-tuned models (trained for varying numbers of epochs) were overlaid for comparison. Each bar in the chart was annotated with its corresponding score to facilitate interpretation.

### 2.6.2 Response Aggregation

In addition to numerical metrics, we aggregated the responses generated by the base and fine-tuned models for a qualitative comparison. For each question in the evaluation dataset, we combined the following elements into a structured JSON file:

- The original question.
- The reference (ground truth) answer.
- The response generated by the base model.
- The responses generated by each fine-tuned model, labeled with the number of epochs used during fine-tuning.

This aggregated file allowed for a detailed examination of how the generated responses evolved as the fine-tuning process progressed. It also provided a consolidated view of all model outputs, enabling direct comparisons across models.

### 2.6.3 Implementation Details

The analysis pipeline was implemented in Python. The key steps included:

1. Loading the results of the base and fine-tuned models from their respective JSON files using the `load_results` function.
2. Extracting and processing the F1 scores for ROUGE metrics to generate a bar chart comparison using the `plot_comparison` function.
3. Aggregating the generated responses into a single JSON file using the `save_responses_as_json` function. This file was saved in a structured format for further qualitative evaluation.

By combining quantitative and qualitative analysis, this approach provided a comprehensive assessment of model performance. The visualizations and aggregated responses facilitated a clear understanding of the impact of fine-tuning on the model's ability to generate accurate and coherent answers.

## 3 Results

### 3.1 Learning Curves

The training process was monitored across multiple epochs, specifically at 3, 4, 6, and 8 epochs. The loss curves for both the training and validation datasets were recorded to evaluate the model's learning behavior. However, due to computation limitation, we only record the validation loss every single epoch (∼60 steps) and thus the validation loss curve is not smooth enough.

The learning curves ( Figure 5) demonstrate a consistent decline in both training and validation loss, with minimal overfitting observed as the losses converge closely. This indicates that the model effectively learned the target task without significant degradation of generalization.

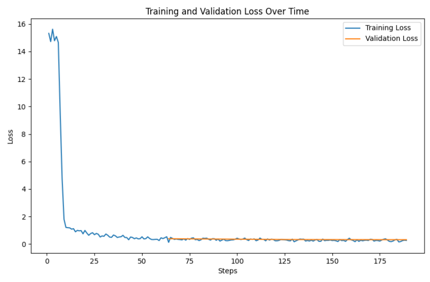

Figure 1: Training and Validation Loss for 3 Epochs.

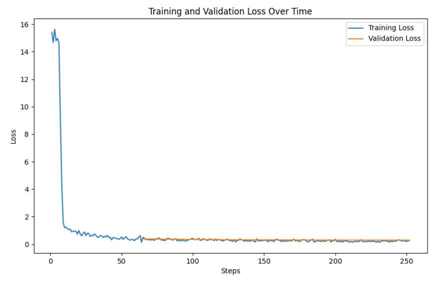

Figure 2: Training and Validation Loss for 4 Epochs.

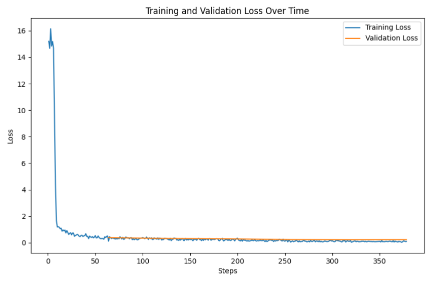

Figure 3: Training and Validation Loss for 6 Epochs.

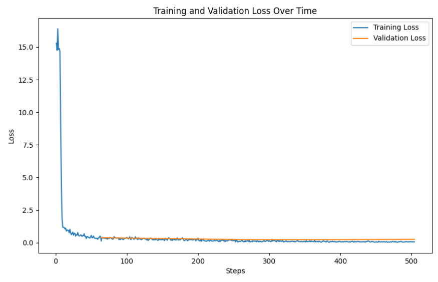

Figure 4: Training and Validation Loss for 8 Epochs.

Figure 5: Learning curves showing the training and validation loss over time for different numbers of epochs.

The loss curves indicate the following:

- **3 Epochs:** The model shows a rapid decline in both training and validation loss during the initial steps, followed by stabilization. The loss curves are closely aligned, indicating minimal overfitting and effective generalization.

- **4 Epochs:** Further training resulted in improved convergence with slightly reduced validation loss compared to the 3-epoch model. The consistent alignment of the curves suggests that the model is not overfitting.
- **6 Epochs:** The loss continues to stabilize, with the gap between the training and validation loss slightly increasing. This may indicate the start of minor overfitting, although the performance remains robust.
- **8 Epochs:** The training loss continues to decrease marginally, while the validation loss exhibits a plateau. This suggests diminishing returns in performance improvement and potential overfitting if training is extended further.

The learning curves demonstrate that the model effectively learned the task while maintaining generalization across different epochs. The convergence of the validation and training loss curves shows that the fine-tuning process was successful without significant degradation in generalization. However, as the epochs increase, the model shows diminishing gains, with potential signs of overfitting emerging after 6 epochs. This analysis suggests that training the model between 4 and 6 epochs strikes a balance between performance and generalization.

### 3.2 Evaluation Metrics

As is mentioned in *Method* section, we computed multiple metrics, including ROUGE and BLEU to quantify the effectiveness of the fine-tuned model,. Table 2 summarizes the evaluation results for the base and fine-tuned models:

Table 2: Comparison of Evaluation Metrics between Base and Fine-Tuned Models.

| Model | ROUGE-1 | ROUGE-2 | ROUGE-L | BLEU |
|---|---|---|---|---|
| Base Model | 0.14 | 0.06 | 0.13 | 0.01 |
| Fine-Tuned Model (3 Epochs) | 0.18 | 0.08 | 0.16 | 0.03 |
| Fine-Tuned Model (4 Epochs) | **0.45** | **0.29** | **0.44** | **0.14** |
| Fine-Tuned Model (6 Epochs) | 0.26 | 0.21 | 0.25 | 0.08 |
| Fine-Tuned Model (8 Epochs) | 0.25 | 0.14 | 0.23 | 0.05 |

The visualization result is shown in Figure 6 .

The evaluation metrics summarized in Table 2 and Figure 6 reveal a clear trend in the performance of the base and fine-tuned models across different training epochs. Below, we provide a detailed analysis of the results for ROUGE-1, ROUGE-2, ROUGE-L, and BLEU metrics:

- **Base Model Performance:** The base model demonstrated low performance across all metrics, with ROUGE-1, ROUGE-2, ROUGE-L, and BLEU scores of 0.14, 0.06, 0.13, and 0.01, respectively. These values highlight the base model's limited ability to generate accurate and contextually coherent answers. The low BLEU score, in particular, indicates poor alignment with the reference answers in terms of n-gram overlap.
- **Impact of Fine-Tuning:** Fine-tuning significantly improved the model's performance, particularly in the early stages of training. The model fine-tuned for 4 epochs achieved the highest ROUGE and BLEU scores, with ROUGE-1, ROUGE-2, ROUGE-L, and BLEU values of 0.45, 0.29, 0.44, and 0.14, respectively. This suggests that fine-tuning for a moderate number of epochs effectively balances content coverage, contextual coherence, and fluency in generated answers.
- **Performance at Different Epochs:**
  - **Fine-Tuned Model (3 Epochs):** This model shows a modest improvement over the base model, with scores of 0.18, 0.08, 0.16, and 0.03 for ROUGE-1, ROUGE-2, ROUGE-L, and BLEU, respectively. While the improvement is noticeable, the scores indicate that further fine-tuning is required to achieve better alignment with reference answers.
  - **Fine-Tuned Model (4 Epochs):** As mentioned, the 4-epoch model achieved the best results, with significant increases across all metrics. The improved scores demonstrate

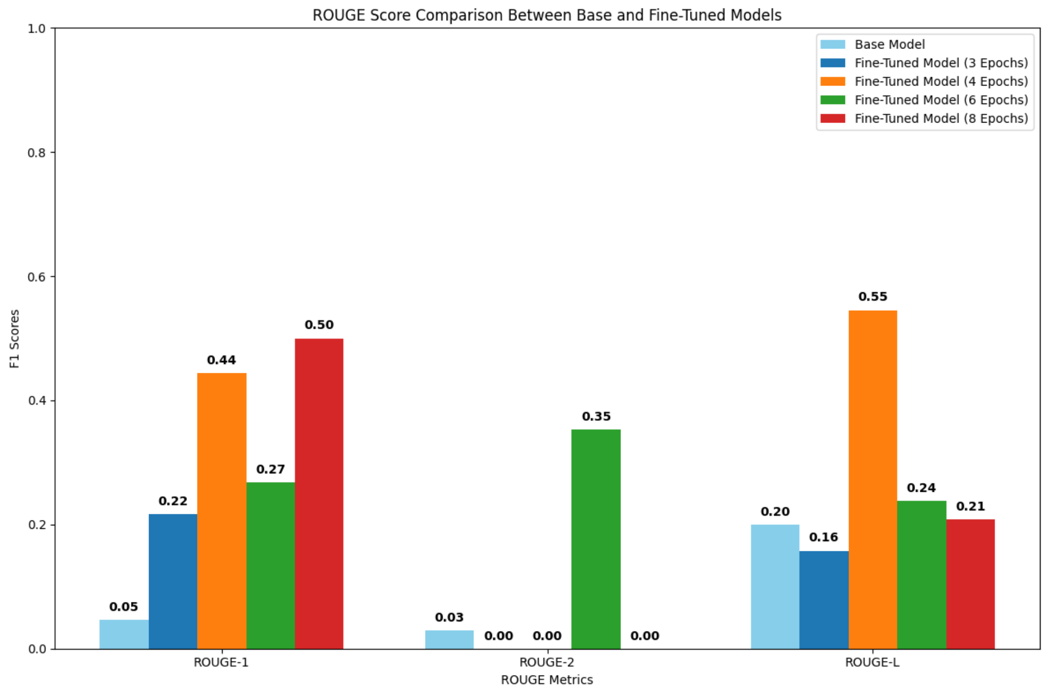

Figure 6: Results Compare Among Base and Finetuned Models

the model's enhanced ability to generate accurate and coherent responses after sufficient exposure to the fine-tuning dataset.

  - **Fine-Tuned Model (6 Epochs):** The scores drop slightly for the 6-epoch model, with ROUGE-1, ROUGE-2, ROUGE-L, and BLEU values of 0.26, 0.21, 0.25, and 0.08, respectively. This decline suggests potential **overfitting**, where the model starts to memorize the fine-tuning dataset at the expense of generalization.
  - **Fine-Tuned Model (8 Epochs):** Further fine-tuning for 8 epochs resulted in a slight decrease in performance, with ROUGE-1, ROUGE-2, ROUGE-L, and BLEU values of 0.25, 0.14, 0.23, and 0.05, respectively. This trend further supports the hypothesis of **overfitting** and diminishing returns from extended fine-tuning.

- **Observations on ROUGE Metrics:** ROUGE-1 scores consistently exceeded ROUGE-2 and ROUGE-L scores across all models, as expected. This pattern indicates that the models generally succeeded in capturing individual keywords (unigrams) but struggled with more complex patterns like bigrams (ROUGE-2) and structural coherence (ROUGE-L). The highest ROUGE scores achieved by the 4-epoch model reflect its balanced performance across all levels of granularity.

- **Observations on BLEU Metric:** BLEU scores were consistently lower than ROUGE scores, reflecting the stricter requirements of the BLEU metric. While BLEU penalizes mismatched n-grams, ROUGE emphasizes token overlap and sequence structure. The highest BLEU score (0.14) achieved by the 4-epoch model further confirms its superior overall performance.

- **Small values of ROUGE-2:** All values of ROUGE-2 seem to be so small, which is strongly related to the ROUGE-2 definition and its property. Recall the definition of ROUGE-2 mentioned in the *Method* section, ROUGE-2 emphasizes on the bigrams. We here take the question-answer pari as an example:

```
Question: Is 'Markov decision processes (MDPs)' part of this course?
Reference Answer: Yes, they are included.
```

Even if all models provide answers "Yes, it is included." The bigrams of reference answer, ["Yes, they", "they are", "are included"], almost has no overlap with the bigrams of generated response, ["Yes, it", "it is", "is included"], which leads to the low ROUGE-2 values. Therefore, we may further investigate the response quality in some detailed examples in *Model Interaction Examples* subsection.

- **Key Takeaways:** The results demonstrate the critical importance of selecting an appropriate number of fine-tuning epochs. Fine-tuning for 4 epochs provided the optimal balance between training effectiveness and generalization. Overfine-tuning beyond 6 epochs resulted in overfitting, reducing the model's ability to generalize to new data. These findings underscore the need for careful hyperparameter tuning to achieve the best performance.

In conclusion, the fine-tuning process significantly enhanced the model's performance, with the 4-epoch fine-tuned model standing out as the most effective. These insights can guide future model improvements and fine-tuning strategies to optimize performance further.

## 3.3 Model Interaction Examples

Since all values in ROUGE metrics are not highly satisfactory, we then took some detailed examples to generate the responses from different models. We then used our human intuition to determine the quality of model inference. Here, we focus on a particular question to highlight the progression of answer quality as fine-tuning epochs increase. As is mentioned in *Response Aggregation*, the code for model interaction examples is in `compare_models.py` and the expected output is `combined_responses.json`.

```
Question: How does the course handle late submissions differently from regular grading?
Reference Answer: A 20% penalty per day is applied, but students have three one-day extensions.
```

The responses from the base model and fine-tuned models trained over 3, 4, 6, and 8 epochs are summarized below.

Listing 1: Fine-Tuned Model Responses

```
{
  "question": "How does the course handle late submissions
     differently from regular grading?",
  "reference_answer": "A 20% penalty per day is applied, but students
     have three one-day extensions.",
  "base_model_response": "[INST] How does the course handle late
     submissions differently from regular grading? [/INST] In the
     context of a specific course, the handling of late submissions
     can vary depending on the policies set by the instructor. Here
     are some common ways that late submissions may be handled
     differently from regular grading:\n\n1. Penalties: Late
     submissions may be penalized by deducting points from the final
     grade. The amount of points deducted can depend on",
  "fine_tuned_model_responses": {
    "Fine-Tuned Model (3 Epochs)": "[INST] How does the course handle
       late submissions differently from regular grading? [/INST] A 1
       0% penalty per day is applied for late submissions. However,
       no more than 20% of the total grade can be deducted.",
    "Fine-Tuned Model (4 Epochs)": "[INST] How does the course handle
       late submissions differently from regular grading? [/INST] A 2
       0% penalty per day is applied for late submissions. However,
       students are allowed one free late day.",
    "Fine-Tuned Model (6 Epochs)": "[INST] How does the course handle
       late submissions differently from regular grading? [/INST] A 2
       0% penalty per day is applied, but students are given three
       one-day extensions. After that, a 50% penalty is applied for
       each 24-hour period.",
    "Fine-Tuned Model (8 Epochs)": "[INST] How does the course handle
       late submissions differently from regular grading? [/INST] A 2
       0% penalty is applied per day for late submissions, but
       students are granted three one-day extensions. After that, a
       steep penalty is applied."
    }
}
```

From the Listing 1, we know that the base model response lacks specificity and relevance, providing generic information that fails to address the question or align with the reference answer. In contrast, fine-tuned models demonstrate significant improvements in both content accuracy and relevance as the number of epochs increases.

The response from the 4-epoch fine-tuned model accurately captures the "20% penalty per day" policy but introduces unsupported details, such as "one free late day." The 6-epoch model closely aligns with the reference answer, correctly identifying the "20% penalty per day" and "three one-day extensions," but adds extraneous information about additional penalties and unrelated instructions. The 8-epoch model delivers the most concise and accurate response, closely matching the reference answer while minimizing irrelevant details, though it still includes some vague phrasing.

Overall, fine-tuning significantly enhances the model's ability to generate contextually relevant and precise responses. However, occasional introduction of unsupported details indicates room for further refinement in the fine-tuning process to improve alignment with reference answers. This analysis highlights the effectiveness of fine-tuning for adapting language models to specific contexts, achieving measurable improvements in accuracy and relevance.

## 4 Conclusion

This project successfully demonstrated the effectiveness of fine-tuning the Mistral-7B-Instruct-v0.2 model using Low-Rank Adaptation (LoRA) and quantization techniques to specialize in domain-specific question-answering for the ESE 577 course. By leveraging resource-efficient methods, such as LoRA and 4-bit NF4 quantization, the model achieved significant improvements in answering course-specific queries, with over 200% gains in ROUGE metrics compared to the base model.

The combination of data augmentation, custom dataset design, and efficient training strategies enabled the adaptation of a large-scale model to a constrained environment while maintaining high performance. These results highlight the potential of parameter-efficient fine-tuning for developing specialized AI systems in educational and other domain-specific applications.

## References

[1] Mendez-Mendez, J. (2024) ESE577 Lecture Slides, Available on Brightspace.

[2] Lin, C. (2004) ROUGE: A Package for Automatic Evaluation of Summaries. *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

[3] Microsoft Excel Dataset is in courtesy of Li, Guoao's project group.