IMPROVING THE LANGUAGE UNDERSTANDING CAPABILITIES OF LARGE LANGUAGE MODELS USING REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs), primarily built on decoder-only transformer architectures, excel in natural language generation tasks and have shown promise in adapting to diverse downstream tasks using zero-shot and few-shot prompting techniques. However, these prompting methods often fall short on natural language understanding (NLU) tasks, where smaller encoder-only models like BERT-base consistently outperform LLMs on benchmarks such as GLUE and SuperGLUE. In this paper, we explore two approaches—supervised fine-tuning and proximal policy optimization (PPO)—to enhance the NLU capabilities of LLMs. To reduce the computational cost of full-model fine-tuning, we integrate low-rank adaptation (LoRA) layers, restricting updates to these layers during both supervised fine-tuning and PPO stages. In the supervised fine-tuning approach, task-specific prompts are concatenated with input queries and ground-truth labels from the NLU training corpus, optimizing the model using the next-token prediction objective. Despite this, LLMs still underperform compared to encoder-only models like BERT-base on several NLU tasks. To address this gap, we employ PPO, a reinforcement learning technique that treats each token generation as an action and evaluates the sequence of generated tokens using a reward function based on their alignment with ground-truth answers. PPO then updates the model to maximize these rewards, effectively aligning its outputs with the correct labels. Our experiments with the LLAMA2-7B model demonstrate that PPO-based fine-tuning significantly improves performance, delivering an average gain of 6.3 points over supervised fine-tuning on the GLUE benchmark. PPO surpasses zero-shot prompting by 38.7 points and few-shot prompting by 26.1 points on GLUE, while also outperforming these baselines by 28.8 and 28.5 points on SuperGLUE. Additionally, PPO exceeds the performance of BERT-large, a strong baseline, with an average improvement of 2.7 points on GLUE and 9.3 points on SuperGLUE. These improvements are consistent across models such as Qwen2.5-7B and MPT-7B, highlighting PPO's robustness and effectiveness in enhancing the NLU capabilities of LLMs.

1 INTRODUCTION

Large language models (LLMs) Radford et al. (2019); Brown (2020); Touvron et al. (2023b) have revolutionized natural language processing (NLP) with their powerful text generation capabilities, driven by their decoder-only transformer architecture Radford (2018). Pretrained on large amounts of unlabeled text, LLMs can generate coherent and contextually relevant content. Using prompt-based strategies like zero-shot and few-shot prompting Brown (2020), LLMs can tackle various downstream tasks without requiring task-specific fine-tuning. However, these methods often underperform on natural language understanding (NLU) tasks compared to encoder-only models like BERT Devlin (2018), which consistently excel on benchmarks such as GLUE Wang et al. (2019) and SuperGLUE Wang et al. (2020). For instance, our evaluations on LLAMA2-7B showed that zero-shot prompting with task-specific prompts yielded an average performance of 46.1 across all GLUE datasets, while few-shot prompting improved performance to 58.7, both of which significantly lag behind BERT-base's 79.6 as shown in Table 1. This underperformance is largely due to LLMs' inability to capture bidirectional context and perform deeper semantic analysis. Enhancing the NLU

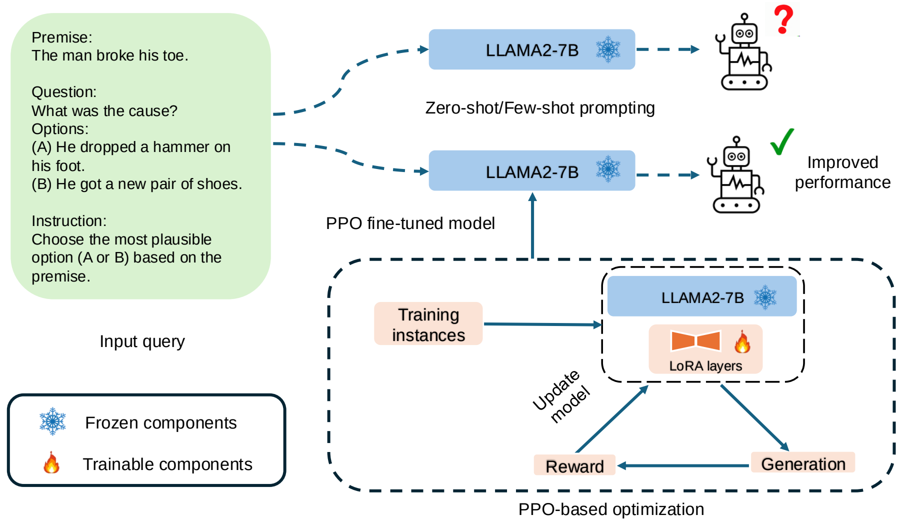

Figure 1: PPO-based fine-tuning of LLAMA2-7B to improve the performance on NLU tasks.

performance of LLMs remains a challenge, as their autoregressive nature limits their ability to model the bidirectional dependencies crucial for NLU tasks Radford et al. (2019); Brown (2020).

To enhance the performance of LLMs on NLU tasks, we explore two approaches. First, we apply supervised fine-tuning (SFT) of LLMs on NLU training datasets. The model is fine-tuned on input sequences consisting of task-specific prompts, training examples, and their corresponding ground-truth labels, using the next-token prediction objective. To reduce computational overhead, we employ low-rank adaptation (LoRA) layers Hu et al. (2021a), ensuring that only these lightweight matrices are updated during fine-tuning, rather than the entire model. However, in our experiments with LLAMA2-7B, this approach underperforms compared to BERT-base on several GLUE datasets, including QQP, SST-2, STS-B, and MRPC, as detailed in Table 1. On average, across all GLUE datasets, BERT-base achieves a score of 79.6, outperforming LLAMA2-7B fine-tuned model, which attains 78.5. This indicates the need for an alternative approach to further boost performance.

To further enhance the performance of LLMs on NLU tasks, we adopt a proximal policy optimization (PPO) Schulman et al. (2017a) based fine-tuning approach, leveraging LoRA layers to reduce computational complexity. Previous works, including A3C Mnih et al. (2016), AlphaGo Silver et al. (2017b), OpenAI Five OpenAI et al. (2019), and AlphaZero Silver et al. (2017a), have demonstrated that policy-based reinforcement learning can effectively train neural networks to perform actions in complex environments. These methods have also been widely applied to align LLM responses with human preferences Bai et al. (2022a) Ouyang et al. (2022a) and improve reasoning capabilities Havrilla et al. (2024). Building on this foundation, we employ PPO to improve LLM performance on NLU tasks. We frame the task of generating responses by LLMs as a reinforcement learning problem, where the sequence of input tokens represents the state s_t, and the token generated at each timestep t is treated as the action a_t. After the entire sequence is generated, a heuristic-based process extracts the answer, which is compared to the ground truth label, and a reward R is assigned accordingly. Our major contributions are:

- We propose a PPO-based fine-tuning approach to improve the NLU capabilities of LLMs. To reduce computational complexity, we fine-tune only the LoRA layers.
- Our evaluation on the GLUE and SuperGLUE benchmarks using the LLAMA2-7B model Touvron et al. (2023a) shows that PPO-based fine-tuning significantly outperforms zero-shot and few-shot baselines, with an average improvement of 38.7 and 26.1 points on the GLUE benchmark, and 28.8 and 28.5 points on the SuperGLUE benchmark, respectively. Additionally, PPO-based fine-tuning achieves an average gain of 6.3 points over SFT on GLUE benchmark and outperforms BERT-large, with a 2.7-point gain on GLUE and a 9.3-point improvement on SuperGLUE benchmark.
- The results are consistent across other LLMs such as Qwen2.5-7B and MPT-7B, demonstrating the robustness of our approach.

2 Related Works

2.1 Natural Language Understanding

Natural language understanding (NLU) tasks are crucial for evaluating a model's ability to comprehend and process human language in various contexts, such as classification, inference, and reasoning. The GLUE benchmark Wang et al. (2019) serves as a key standard for NLU performance, covering tasks like CoLA, SST-2, MRPC, MNLI, and so on, which assess grammatical acceptability, sentiment analysis, paraphrase detection, and textual entailment. For more complex challenges, the SuperGLUE benchmark Wang et al. (2020) was introduced, featuring more difficult tasks that require advanced reasoning and comprehension. Together, GLUE and SuperGLUE provide a comprehensive assessment of a model's language understanding capabilities.

Models such as BERT Devlin (2018), which utilize a bidirectional encoder architecture, have achieved state-of-the-art performance in NLU tasks. BERT's architecture allows it to capture bidirectional context. Its pretraining strategy, which uses masked language modeling (MLM), helps the model learn deep semantic representations. This combination makes BERT highly effective across a wide range of NLU tasks. The success of encoder-only models in benchmarks such as GLUE and SuperGLUE can largely be attributed to their ability to capture rich bidirectional context during pretraining, which is critical for NLU tasks.

In contrast, LLMs like GPT-2 Radford et al. (2019), GPT-3 Brown (2020), and LLAMA Touvron et al. (2023b) rely on scaling model size with decoder-only architectures, achieving significant success in text generation tasks. However, their zero-shot performance with task-specific prompts remains suboptimal on NLU tasks, such as those in the GLUE benchmark. This underperformance is attributed to their autoregressive nature, which limits their ability to capture the bidirectional dependencies crucial for deep contextual understanding Devlin (2018); Radford et al. (2019); Brown (2020). Efforts to adapt LLMs for NLU have focused on prompt-based methods like few-shot prompting Brown (2020), which show promise but still fall short of the performance achieved by encoder-only models like BERT on these tasks.

2.2 Policy-based Reinforcement Learning

Policy-based reinforcement learning (RL) directly optimizes an agent's policy by learning its parameters to maximize long-term rewards. Unlike value-based methods like Q-learning Watkins & Dayan (1992) and DQN Hester et al. (2018), which indirectly derive policies through value functions, policy-based methods represent the policy as a parameterized function. This function, $p_\theta(a|s)$, defines the probability of taking action a in state s, where θ represents the policy parameters. The goal is to learn optimal parameters θ^* that maximize the expected cumulative reward, typically through policy gradient methods Sutton et al. (1999). These methods excel in high-dimensional or continuous action spaces, where value-based methods can struggle Deisenroth et al. (2013).

Policy-based methods in reinforcement learning (RL) have evolved significantly over time, starting with REINFORCE Williams (1992), which optimizes policies using the policy gradient theorem but suffers from high variance due to its reliance on Monte Carlo estimates of the reward. Monte Carlo estimates refer to calculating the total reward based on full episodes of interaction, meaning updates are made only after an entire sequence of actions and rewards is observed, which can lead to noisy and slow learning. To address this, actor-critic methods like A2C and A3C Mnih (2016) introduced a critic that estimates the value of the current state, allowing for smoother updates by reducing the variability in policy updates and speeding up convergence. However, these methods still faced instability when large updates caused the new policy to diverge too far from the previous one. Trust Region Policy Optimization (TRPO) Schulman (2015) tackled this by limiting the size of policy updates using a KL divergence constraint, but its implementation was complex and computationally expensive. Proximal policy optimization (PPO) Schulman et al. (2017a) simplified this process by introducing a clipped objective function that keeps policy updates within a stable range while being easier to implement. PPO's balance between simplicity and stability has made it a widely adopted method in modern RL research.

In NLP, PPO has been effectively used in reinforcement learning from human feedback (RLHF) to align LLM outputs with human preferences, as seen in works like InstructGPT Ouyang et al. (2022b) and Constitutional AI Bai et al. (2022b). These approaches treat the LLM as a policy, where model responses are actions, and human feedback serves as rewards. PPO updates the policy based on the reward model trained on human preferences. Additionally, policy-based RL methods have been applied to enhance LLM reasoning capabilities Ziegler et al. (2019); Havrilla et al. (2024); Hu & Shu (2023). In this work, we apply PPO to fine-tune LLMs on NLU tasks.

3 Preliminaries on Application of PPO for Fine-tuning LLMs

Proximal policy optimization (PPO) Schulman et al. (2017b) is an online reinforcement learning algorithm. In this section, we describe the process to fine-tune an LLM using PPO. During training, at each timestep t, the LLM (policy) generates a token prediction a_t (action) based on the state s_t, which consists of the sequence of generated tokens up to timestep $t-1$. The final generated output is evaluated in the context of the downstream task, where the environment provides feedback in the form of rewards. The model updates its parameters based on these rewards to improve its ability to generate accurate predictions over time.

PPO uses gradient ascent to optimize the following objective, aiming to maximize cumulative rewards:

$$J(\theta) = \mathbb{E}_{(s_t,a_t)\sim\pi_{\theta'}}\left[\min\left(\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)}\hat{A}_t,\ \text{clip}\left(\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)},\ 1-\epsilon,\ 1+\epsilon\right)\hat{A}_t\right)\right]$$

Here, $p_\theta(a_t|s_t)$ is the probability of taking action a_t in state s_t under the current policy, while $p_{\theta'}(a_t|s_t)$ represents this probability under the old policy. In PPO, the training data—specifically, the state-action pairs (s_t, a_t)—are sampled using the old policy $\pi_{\theta'}$ (the LLM before it is updated), rather than the new policy currently being optimized. Thus, the ratio $\frac{p_\theta(a_t|s_t)}{p_{\theta_{\text{old}}}(a_t|s_t)}$ accounts for how much the new policy has changed relative to the old policy and adjusts the likelihood of an action accordingly. This ratio is multiplied by $\hat{A}_t$, the Generalized Advantage Estimation (GAE) Schulman et al. (2018), which measures how much *better or worse* an action a_t is compared to the expected outcome under the current policy.

$$\hat{A}_t = R_t + \gamma V_{t+1} - V_t + \gamma\lambda\hat{A}_{t+1},$$

Here, $R_t + \gamma V_{t+1} - V_t$ represents the temporal difference (TD) error Sutton (1988). In this expression, R_t is the immediate reward received after taking action a_t, V_t is the expected reward before the action, and γV_{t+1} is the discounted estimate of the future reward after the action. This term reflects how the action a_t performed when compared to the expected return at state s_t. The second term, $\gamma\lambda\hat{A}_{t+1}$, is the smoothing factor in GAE, where λ is the trade-off parameter. This recursive estimate allows the model to incorporate future information, making the advantage estimate more stable. Smaller values of λ emphasize on immediate rewards, while larger values capture longer-term dependencies. The discount factor γ controls how much emphasis is placed on future rewards compared to immediate ones, with higher values of γ giving more weight to future rewards. V_t, which represents the expected future reward from state s_t, is estimated by a *critic model*.

The clipping function clip$(\text{ratio}, 1-\epsilon, 1+\epsilon)$ limits the change between the current and old policy, ensuring stable updates by preventing large deviations. This helps avoid too-large policy changes that could destabilize training. In summary, PPO optimizes the policy using gradient ascent to maximize cumulative rewards while ensuring stable updates through clipping, with the GAE providing a more stable and accurate advantage estimate by incorporating future information recursively.

Critic Model The critic model consists of a value head, which is a multi-layer perceptron attached to the final layer of the LLM. It takes the LLM's representation of the generated token sequence up to timestep t (i.e., the state s_t) and predicts a scalar value representing the value function V_t for that state. The critic model is updated using the square of TD error, which is computed as:

$$\delta_t = (R_t + \gamma V_{t+1} - V_t)^2, \tag{1}$$

where δ_t represents the L-2 loss between the actual reward R_t, combined with the discounted estimate of future rewards γV_{t+1}, and the current predicted value V_t for state s_t. By minimizing this TD error via gradient descent, the critic model updates its value function predictions, improving alignment with the actual rewards and future outcomes. In summary, LLM uses the PPO objective to update its policy based on feedback from the critic model, while the critic model is updated to better predict the value function for future states.

4 METHOD

To enhance the performance of LLMs on NLU tasks, we adopt two distinct fine-tuning methods. The first approach involves supervised fine-tuning, where the input consists of a concatenation of the task-specific prompt, query and the ground truth answer, with the model optimized using the next-token prediction objective. The second approach utilizes PPO, framing response generation as a reinforcement learning problem. In this setup, the sequence of input tokens till timestep $t-1$ represents the state s_t, and each token generated at timestep t is treated as an action a_t. After generating the entire sequence, a heuristic-based process extracts the final answer from this generated sequence, and is compared to the ground truth. PPO is then employed to optimize the model by maximizing the cumulative reward derived from this comparison. To reduce computational complexity, we fine-tune LoRA layers instead of the full model.

4.1 TASK-SPECIFIC PROMPT DESIGN

We detail the construction of task-specific prompts used to query the LLM for NLU tasks. Each prompt begins with a clear task description, outlining the necessary background information to guide the model in solving the task. Following this, we specify strict requirements for the output format, ensuring that the response is encapsulated within a predefined structure, specifically between '<Judgement></Judgement>' tags. This structure ensures consistency in the model's responses, facilitating easier extraction and evaluation of the results.

For example, in the CoLA task, which assesses grammatical acceptability, the prompt is structured as follows:

```
System_prompt:
    You are an assistant to analyze the linguistic properties
    of a sentence. The task is to decide the linguistic acceptability
    of a sentence. If the sentence is linguistically correct then it
    is acceptable, else it is not.
The result you give should have the following form:
    <Judgement> {Insert only "Yes" or "No" here} </Judgement>
Prompt:
    Now judge if the sentence "{sentence}" is linguistically acceptable.
Assistant:
    <Judgement>
```

The prompt starts with background information about CoLA, specifies restrictions on the output (such as labeling a sentence as acceptable or unacceptable), and concludes with a special start token, <Judgement>, to initiate the model's response generation.

4.2 SUPERVISED FINE-TUNING OF LLM ON NLU TASKS

Given an NLU training dataset, $\mathcal{D}^{(tr)} = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i represents the input text and y_i the ground truth label, we fine-tune the LLM on a sequence consisting of the task-specific prompt p (described in section 4.1) concatenated with the input x_i and the ground truth answer y_i. The model is trained using the next-token prediction objective, where it predicts the next token in the sequence by conditioning on all preceding tokens. This objective trains the model to learn to predict the correct answer for the NLU task conditioned on the task-specific prompt and input.

4.3 PROXIMAL POLICY OPTIMIZATION FOR LLM FINE-TUNING ON NLU TASKS

We utilize PPO to fine-tune the LLM on NLU tasks, following the training protocol outlined in section 3. The reward function is specifically designed for each NLU task. In this work, we use a simple reward function, where a reward is assigned at *the end of the generation* based on alignment with the ground truth labels. We use regular expression matching to extract answers from the LLM's outputs by first locating the text within the '<Judgement></Judgement>' tags. Depending on the task, we then search for task-specific keywords (such as "yes", "no", "acceptable", or "not acceptable") to identify the answer. These extracted answers are compared with the ground truth to determine the appropriate rewards.

For instance, CoLA, which is a classification task, answers are categorized as *acceptable*, *unacceptable*, or *exceptional* (incorrect format). For STS-B, which is a regression task, the extracted answer is a floating-point number between 0 and 5. The reward per generation for classification tasks is given by $R = \mathbb{1}(\hat{y} == y_i)$, where $\hat{y}$ is the model's prediction and y is the ground truth. For STS-B, a regression task, the reward per generation is calculated based on how close the prediction is to the ground truth: $R = 2.5 - |\hat{y_i} - y_i|$. *Incorrectly formatted responses* are penalized with a value of -1 for classification tasks and -2.5 for regression tasks.

4.4 Low-Rank Adaptation

To mitigate the computational cost of full-model fine-tuning, we employ LoRA Hu et al. (2021b) during both the supervised fine-tuning and PPO stages. Instead of updating the entire model, we restrict the updates to LoRA layers, which significantly reduces the number of trainable parameters by decomposing the weight matrices into low-rank matrices.

5 Experiments

5.1 Experimental Setup

We trained and evaluated our models on the GLUE Wang et al. (2019), and SuperGLUE Wang et al. (2020) benchmarks. All experiments were conducted using instruction-tuned LLAMA2-7B models Touvron et al. (2023a)[1]. We perform both single task and multi-task fine-tuning: 1) *Single-task Fine-tuning*: For each subtask within GLUE, and SuperGLUE, a separate task-specific LoRA module was trained independently. 2) *Multi-task Fine-tuning*: In the multi-task setting, datasets from different subtasks within each benchmark were combined, and a single LoRA module was trained to handle all tasks simultaneously.

Hyperparameter Settings For PPO-based fine-tuning, gird search is performed to select the batch size in 4, 8, 12, and 16 for each task. A batch size of 24 was used across all tasks during supervised fine-tuning (SFT). The PPO epochs is set to 4, that is each sampled batch is used for updating the model four times. The initial learning rate for all tasks was set to 9×10^{-6}. We utilized the Adafactor optimizer for PPO training and AdamW for SFT. A cosine annealing learning rate scheduler with a warmup phase was employed, where the learning rate was gradually increased during the first 10% of training steps and then reduced to one-tenth of the initial value by the end of training. We use a rank $r = 16$ for the LoRA layers. We trained both PPO and SFT models until convergence on the validation set. The best hyperparameters were selected based on performance on the validation set. The final reported results for the GLUE and SuperGLUE are from their corresponding evaluation server. For evaluation, multinomial sampling with a temperature of 1 was used to generate a single response per data sample. The model generated responses with lengths between 12 and 32 tokens, with the generation process concluding using a special identifier "`</Judgement>`".

5.2 Baselines

We evaluated the performance of our approach against three baselines:

- **Encoder-only models**: We compare our results with encoder-only transformer models, specifically BERT-base (110M parameters) and BERT-large (340M parameters) Devlin et al. (2019).
- **Zero-shot prompting:** The model is provided with task-specific prompts, as outlined in section 4.1, along with the input query. The model is required to generate predictions solely based on these prompts and the input query, without any additional task-specific fine-tuning.
- **Few-shot prompting:** In this setting, the model is provided with both the task-specific prompt and five labeled examples from the training dataset as demonstrations. These examples are provided as reference to guide the model in generating more accurate responses for the input query. Similarly, no task-specific fine-tuning is performed.

[1] `https://huggingface.co/daryl149/llama-2-7b-chat-hf`

After generating a response, we applied regular expression matching to extract the relevant answer from the model's output. We directly matched task-specific keywords (like "yes" or "no") in the generated text to identify the answer. This extracted answer was then compared to the ground truth label to evaluate the model's performance.

5.3 Results on GLUE Benchmark

In this section, we present our experiments on the GLUE benchmark, comparing the results with encoder-only models such as BERT Devlin et al. (2019). We use the LLAMA2-7B model as the LLM for our evaluations. The baselines include zero-shot prompting and few-shot prompting (5-shot). For fine-tuning methods, we compare both supervised fine-tuning and PPO across single-task and multi-task settings. The results are summarized in Table 1. From the results, we make the following observations.

Models	**MNLI-m**	**MNLI-mm**	**QQP**	**QNLI**	**SST-2**	**CoLA**
BERT-base	84.6	83.4	<u>71.2</u>	90.5	93.5	52.1
BERT-large	86.7	85.9	**72.1**	92.7	<u>94.9</u>	**60.5**
LLAMA2-7B						
Zero-shot prompting	38.3	39.7	31.3	58.5	75.7	18.6
Few-shot prompting	62.4	61.7	30.9	60.7	84.2	29.0
PPO-ST	**88.8**	<u>88.2</u>	70.5	<u>93.2</u>	**96.4**	<u>59.9</u>
SFT-ST	87.0	86.5	63.8	**93.6**	73.8	50.7
PPO-MT	<u>88.7</u>	**88.3**	67.3	90.2	94.6	47.7
SFT-MT	84.9	84.5	62.9	86.0	72.0	41.4

Models	**STS-B**	**MRPC**	**RTE**	**WNLI**	**AX**	**Average**
BERT-base	85.8	88.9	66.4	/	/	79.6
BERT-large	86.5	<u>89.3</u>	70.1	/	/	82.1
LLAMA2-7B						
Zero-shot prompting	27.5	66.3	59.3	44.5	9.2	46.1
Few-shot prompting	45.5	80.8	72.9	51.4	9.2	58.7
PPO-ST	<u>92.6</u>	**89.4**	84.3	<u>74.7</u>	**52.7**	**84.8**
SFT-ST	84.7	85.8	80.4	63.7	45.1	78.5
PPO-MT	**94.7**	86.7	**86.9**	66.4	<u>43.4</u>	<u>82.9</u>
SFT-MT	85.5	82.6	<u>86.2</u>	**76.0**	41.2	76.22

Table 1: GLUE test results are scored by the evaluation server (GLUE benchmark). Average column indicates the averaged performance across all the datasets excluding the WNLI and AX datasets. F1 scores are reported for QQP and MRPC, Spearman correlations for STS-B, Matthew's correlations for CoLA, and accuracy scores for the other tasks. *Zero-shot prompting* refers to prompting with task-specific prompts and an input query, while *Few-shot prompting* refers to prompting with task-specific prompts, 5 demonstrations, and an input query. *PPO* stands for proximal policy optimization, and *SFT* refers to Supervised Fine-tuning. "ST" represents Single-task, while "MT" represents Multi-task. The **bolded** results indicate the best results, and the <u>underlined</u> results indicate the second-best results.

First, we observed that zero-shot prompting of the LLAMA2-7B model with task-specific prompts consistently underperformed compared to the smaller BERT-base model. LLAMA2-7B struggled notably on simpler tasks like SST-2, which only required classifying sentiment as positive or negative. This underscores the model's weak language understanding capabilities, with zero-shot prompting proving inadequate compared to BERT-base. **Second**, few-shot prompting showed improvements over the zero-shot baseline, achieving an average score of 58.7 compared to 46.1, but it still lagged significantly behind the BERT-base model's score of 79.6. **Third**, supervised fine-tuning (SFT) using LoRA modules for each task further boosted performance, bringing it closer to BERT's level with an average score of 78.5, though still slightly behind BERT-base's 79.6. **Fourth**, fine-tuning with PPO delivered the best results, achieving an average score of 84.6, surpassing even BERT-large's 82.1. Moreover, zero-shot and few-shot prompting of LLAMA2-7B displayed a noticeable output imbalance, with a tendency to favor certain classes or values. In contrast, models fine-tuned with PPO showed no significant bias. **Fifth**, the computational time for PPO is approximately *1.32 times* that of SFT, indicating only a marginal increase in computational costs.

Additionally, we compared the results with multi-task training, where a *single LoRA module* was trained across all datasets using both SFT and PPO to reduce time complexity. We found that SFT on individual tasks outperformed its multi-task fine-tuning counterpart. However, while PPO on multi-task training did not perform as well as PPO on single-task training, it still outperformed BERT-large in average performance, achieving a score of 82.9 compared to BERT-large's 82.1. These results demonstrate that while single-task fine-tuning yields the best performance, multi-task training with PPO can still achieve competitive results, even surpassing state-of-the-art models like BERT-large. The training curves for PPO is presented in appendix A.

Models	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WiC	WSC	AXb	AXg	Average
BERT-large	77.4	75.7/83.6	70.6	70.0/24.0	**72.0/71.3**	71.6	69.5	64.3	23.0	97.8/51.7	69.0
BERT-large++	79.0	**84.7**/90.4	73.8	70.0/24.1	**72.0/71.3**	79.0	69.5	64.3	38.0	**99.4**/51.4	71.5
LLAMA2-7B											
Zero-shot prompting	75.8	26.4/43.6	57.0	51.9/20.3	27.0/26.2	59.2	54.4	52.1	9.1	64.0/55.1	49.5
Few-shot prompting	80.2	33.4/50.4	47.6	37.2/12.6	38.1/37.1	72.9	51.1	47.9	9.1	64.0/55.1	49.8
PPO-ST	**85.9**	74.7/88.0	**88.6**	**82.5**/50.0	70.6/69.9	**84.3**	**72.1**	**78.1**	**52.7**	91.0/**79.8**	**78.3**

Table 2: SuperGLUE test results are scored by the evaluation server (SuperGLUE benchmark). The experimental data for BERT-large and BERT-large++ are taken from the original SuperGLUE paper Wang et al. (2020). The metrics used in the experiments are as follows: CB: F1 / Acc; MultiRC: F1 / Exact Match; ReCoRD: F1 / Exact Match; AXb: MCC; AXg: Gender parity score / Acc. For the remaining tasks not mentioned, accuracy (Acc) is reported. Average column corresponds to the averaged performance across all the datasets. For tasks with multiple evaluation metrics, we first compute the average of those metrics to obtain a single task score, which is then used in the overall average calculation. The **bolded** results indicate the best results, and the underlined results indicate the second-best results.

5.4 Results on SuperGLUE Benchmark

We fine-tuned the LLAMA2-7B model using PPO on the SuperGLUE dataset and compared its performance against several baselines, including BERT-large, BERT-large++, and zero-shot and few-shot prompting of LLAMA2-7B. The term "BERT++" refers to a BERT model fine-tuned using the supplementary training on intermediate labeled-data tasks (STILTs) approach Phang et al. (2018), where the model is first fine-tuned on related transfer tasks before being fine-tuned on SuperGLUE tasks. For example, MNLI from the GLUE benchmark Wang et al. (2019) is used as an intermediate task for CB, RTE, and BoolQ Wang et al. (2020). In contrast, our experiments with LLM did not use this method. Our models were only fine-tuned on the datasets included in the SuperGLUE benchmark.

As shown in Table 2, the PPO-tuned LLAMA2-7B achieved the highest average performance, surpassing all baselines. PPO demonstrated particularly strong improvements on reasoning-intensive tasks like COPA and MultiRC, where it significantly outperformed both prompting methods and encoder-only models. These results highlight the effectiveness of PPO in enhancing the model's capabilities, particularly for tasks requiring reasoning and contextual understanding.

Another interesting observation is that few-shot prompting has resulted in reduced performance compared to zero-shot prompting on various challenging tasks such as COPA, MultiRC, WiC, and WSC. This suggests that the demonstrations provided alongside the task-specific prompt may not be effectively guiding the model toward the correct answer and could potentially be introducing confusion instead of clarifying how to approach the task.

5.5 Performance Comparison Across Different LLMs

To assess the consistency of our findings across different models, we evaluated Qwen2.5-7B and MPT-7B alongside LLAMA2-7B on the STS-B dataset from the GLUE benchmark and the COPA dataset from the SuperGLUE benchmark. The results confirm that PPO-based fine-tuning consistently outperforms the BERT-large model, as well as the zero-shot and few-shot prompting baselines for all LLMs, highlighting its effectiveness across different LLMs. Notably, another independent observation is that for all LLMs, few-shot prompting underperforms zero-shot prompting on the COPA dataset. This aligns with our earlier findings in section 5.4, suggesting that the few-shot examples may be introducing noise rather than improving performance in these reasoning tasks.

Models	**STS-B**	**COPA**
BERT-large	86.5	70.6
LLAMA2-7B		
Zero-shot prompting	27.5	57.0
Few-shot prompting	45.5	47.6
PPO-ST	92.6	88.6
Qwen2.5-7B		
Zero-shot prompting	83.7	96.6
Few-shot prompting	87.0	47.6
PPO-ST	92.2	97.0
MPT-7B		
Zero-shot prompting	19.7	57.4
Few-shot prompting	21.7	49.6
PPO-ST	89.3	84.0

Table 3: Performance comparison of LLAMA2-7B, Qwen2.5-7B Hui et al. (2024), and MPT-7B Team (2023) models on the GLUE STS-B and SuperGLUE COPA tasks under zero-shot prompting, few-shot prompting, and PPO based fine-tuning. Results are sourced from the official GLUE benchmark and SuperGLUE benchmark evaluation servers. For STS-B, we report Spearman correlation, and for COPA, accuracy is used as the evaluation metric.

6 Conclusion

Prompting-based approaches, such as zero-shot and few-shot prompting, have gained popularity for adapting LLMs to downstream tasks. However, when applied to LLAMA2-7B, these methods underperform on NLU tasks compared to smaller encoder-only models like BERT-base and BERT-large. To address this limitation, we explore two fine-tuning strategies that leverage LoRA layers to reduce computational overhead. First, we employ supervised fine-tuning by concatenating task-specific prompts, input queries, and ground-truth labels, optimizing the model with the next-token prediction objective. While this approach improves LLAMA2-7B's performance over prompting-based methods, it still lags behind BERT-base on the GLUE benchmark. To further enhance performance, we adopt PPO, treating the LLM as a policy that generates the next token (action) based on the current input sequence (state). A reward function then evaluates how closely the generated tokens match the ground-truth labels, guiding updates to the policy. PPO based fine-tuning of LLAMA2-7B, tested across benchmarks like GLUE, and SuperGLUE, resulted in significant performance gains, outperforming strong baselines like BERT-large. Similar trends were observed in other LLMs, including Qwen2.5-7B and MPT-7B, showcasing the robustness of this approach. These findings underscore the effectiveness of PPO in enhancing NLU capabilities in LLMs. Future work could extend these techniques to more diverse datasets and refine reward functions for handling complex NLU tasks.

7 Reproducibility Statement

We provide our codes at `https://anonymous.4open.science/r/LLM_NLU-BE83`. In the code repo, we provide instructions on how to reproduce experimental results. Furthermore, we also provided the hyperparameter settings in section 5.1.

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai feedback, 2022a. URL `https://arxiv.org/abs/2212.08073`.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022b.

Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL `https://arxiv.org/abs/1810.04805`.

Alex Havrilla, Yuqing Du, Sharath Chandra Raparthy, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, Sainbayar Sukhbaatar, and Roberta Raileanu. Teaching large language models to reason with reinforcement learning. *arXiv preprint arXiv:2403.04642*, 2024.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021a.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021b. URL `https://arxiv.org/abs/2106.09685`.

Zhiting Hu and Tianmin Shu. Language models, agent models, and world models: The law for machine reasoning and planning. *arXiv preprint arXiv:2312.05230*, 2023.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Volodymyr Mnih. Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*, 2016.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016. URL `https://arxiv.org/abs/1602.01783`.

OpenAI, :, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning, 2019. URL `https://arxiv.org/abs/1912.06680`.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022a. URL `https://arxiv.org/abs/2203.02155`.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022b.

Jason Phang, Thibault Févry, and Samuel R Bowman. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*, 2018.

Alec Radford. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

John Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017a.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017b. URL `https://arxiv.org/abs/1707.06347`.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018. URL `https://arxiv.org/abs/1506.02438`.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017a. URL `https://arxiv.org/abs/1712.01815`.

David Silver, Julian Schrittwieser, Karen Simonyan, et al. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017b. doi: 10.1038/nature24270.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3: 9–44, 1988.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

MosaicML NLP Team. Introducing mpt-7b: A new standard for open-source, commercially usable llms, 2023. URL `www.mosaicml.com/blog/mpt-7b`. Accessed: 2023-05-05.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023a. URL https://arxiv.org/abs/2307.09288.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems, 2020. URL https://arxiv.org/abs/1905.00537.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Reward Curve for PPO Fine-Tuning in a Multitask Setting on the GLUE Dataset

We present the reward curve from fine-tuning LLAMA2-7B using PPO in a multitask setting on the GLUE dataset. Figure 2 illustrates the reward values over training iterations, offering insights into the training dynamics of the model. The curve serves as a key performance metric, tracking the model's learning progress across multiple tasks. The consistent upward trend demonstrates that PPO fine-tuning effectively improves LLAMA2-7B's ability to generate task-relevant outputs.

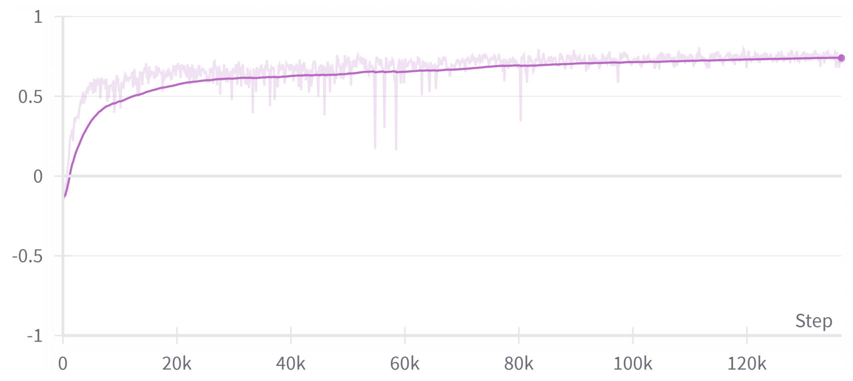

Figure 2: Reward curve for multitask PPO fine-tuning of LLAMA2-7B on the GLUE dataset. The plot illustrates the relationship between training iterations (x-axis) and reward values (y-axis), demonstrating the effectiveness of the PPO optimization approach in enhancing model performance over time.