Words and Wins: Enhancing Game Play with LLM Fine-Tuning by RL

Stanford CS224N Custom Project

Xuanzi Chen
Department of Computer Science
Stanford University
xuanzic@stanford.edu

Zhengjia Huang
Department of Computer Science
Stanford University
fredhzj@stanford.edu

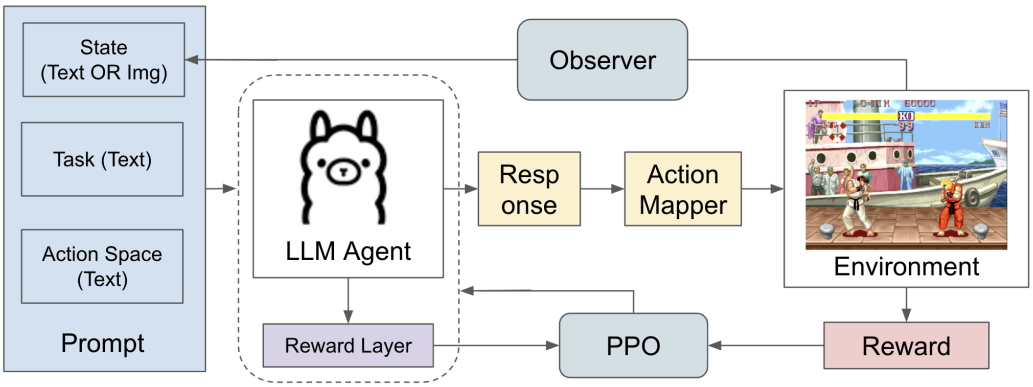
Abstract

Traditionally, AI agents in game are dominated using reinforcement learning. With the enhancement of various large language models (LLMs), a new paradigm is being explored where these models could act as AI agents directly, or enhanced with reinforcement learning in gaming contexts. Our project aims to continue exploring LLM's game play performance in complex, dynamic simulated game setting, such as in the popular action video game "Street Fighter II", by leveraging state-of-the-art accessible text-only LLMs like Mistral 7B or multimodal LLM such as LLaVA. We mainly want to discuss two questions: 1) whether LLMs can not only encode instructions but also be directly used as agent policies choosing actions given the observation, and 2) whether LLMs can boost RL tasks through pre-trained knowledge inherited from the text it learned.

1 Key Information to include

- Mentor: Ryan Li
- Team Contributions: Xuanzi Chen handled the baseline implementation, conducted literature review, experimented Mistral training with PPO, and tuned the RGB observer, action mapping and prompt template. Zhengjia Huang implemented the PPO fine-tuning code, developed the multi-modal related functions, wrote the online training dataset wrapper, trained LLaVA, experimented with GPT2, and did some literature review.

2 Introduction



Video games provide a dynamic and challenging environment for developing and testing AI agents' capabilities. Historically, reinforcement learning (RL) has been a golden methodology in this area,

powering agents that can master games from simple Atari challenges to complex multi-player strategy game environment. However, RL agents often require massive amount of training time and game play data, and may struggle with trained agents adapting to newer environment as well as when reward is sparse.

The advent of large language models (LLMs) like GPT-4o (1) introduces potential for a paradigm shift. These models being trained on diverse media data across audio, video and vision, encompassing a broad range of human knowledge, can generate contextually accurate responses and perform surprisingly well for interactive tasks. Other multimodal model such as LLaVA (2), can process both textual descriptions and visual inputs, offering a new way for game-playing agents to learn visual and contextual knowledges through interaction with the environment.

This project explores the integration of LLMs with traditional RL techniques PPO to tackle the popular action video game "Street Fighter II." The game's complex dynamics and the requirement for fast-pace, strategic play make it challenging for our project. We want to study 1) whether LLMs can not only encode instructions but also be directly used as agent policies choosing actions given the observation, and 2) whether LLMs can boost RL tasks through pre-trained knowledge inherited from the text it learned.

3 Related Work

In the realm of grounding large language models (LLMs) within interactive reinforcement learning (RL) environments, several research efforts have laid the groundwork and explored innovative methodologies, addressing the limitations of previous research and setting the stage for advanced integration of LLMs with RL in game-like scenarios.

3.1 Grounding Large Language Models in Interactive Environments with Online Reinforcement Learning.

Carta et al. (2023) (3) explores the integration of Large Language Models (LLMs) with online Reinforcement Learning (RL) to enhance decision-making in interactive environments. Traditional LLMs can misalign with the environment due to a lack of grounding. The authors propose an approach (GLAM) to achieve this alignment through functional grounding. They use LLM agent as a policy and progressively being updated as the agent interacts with the environment, leveraging online Reinforcement Learning to improve its performance to solve goal. They test this on a textual environment called BabyAI-Text with spatial and navigation tasks, empirically investigates whether LLMs can boost sample efficiency in learning RL tasks through pre-trained knowledge inherited from the texts it learned. The study uses FLAN-T5 variants (Chung et al., 2022) (4). to understand the impact of online learning on functional grounding. We mainly use this research as guidance as we also utilize LLM agent as policy and train using online reinforcement learning.

3.2 Guiding Pretraining in Reinforcement Learning with Large Language Models.

The paper introduces ELLM (5), a method leveraging LLMs to guide reinforcement learning agents by suggesting goals based on the agent's current state. Traditional RL struggles with sparse rewards and irrelevant novelty, so ELLM uses pretrained LLMs to provide context-sensitive, human-meaningful goals, thus enhancing intrinsic motivation and exploration. ELLM improves agent behavior coverage and performance on downstream tasks by evaluating in environments like the Crafter game and Housekeep robotic simulator. The method highlights how LLMs can shape exploration and learning in RL through incorporating common-sense and context-awareness into goal suggestions, bridging the gap between abstract knowledge and practical application. This approach provides guidance to our use of LLMs fine-tuned with RL, and LLM to prompt the agents to play game in the Street Fighter game, but brings challenges to our experiment that LLM prompting to play action games is hard due to the model's lack of native understanding of the fast-pace environment.

3.3 ArCHer: Training Language Model Agents via Hierarchical Multi-Turn RL.

Zhou et al. (2024) presents ArCHer (6), a hierarchical reinforcement learning framework designed for training LLMs to perform multi-turn decision-making tasks. Unlike single-turn RL methods,

ArCHer addresses the need for LLMs to manage long-term objectives, perform credit assignment, and incorporate past actions over multiple interactions. The framework employs a high-level off-policy RL algorithm to aggregate rewards over utterances and a low-level RL algorithm to optimize token-level policies within each utterance. ArCHer demonstrates significant improvements in sample efficiency and performance on multi-turn tasks, proving its capability to scale with model capacity and efficiently manage long-horizon RL challenges.

4 Approach

Our approaches can be summarized as follows:

1. Baseline Models

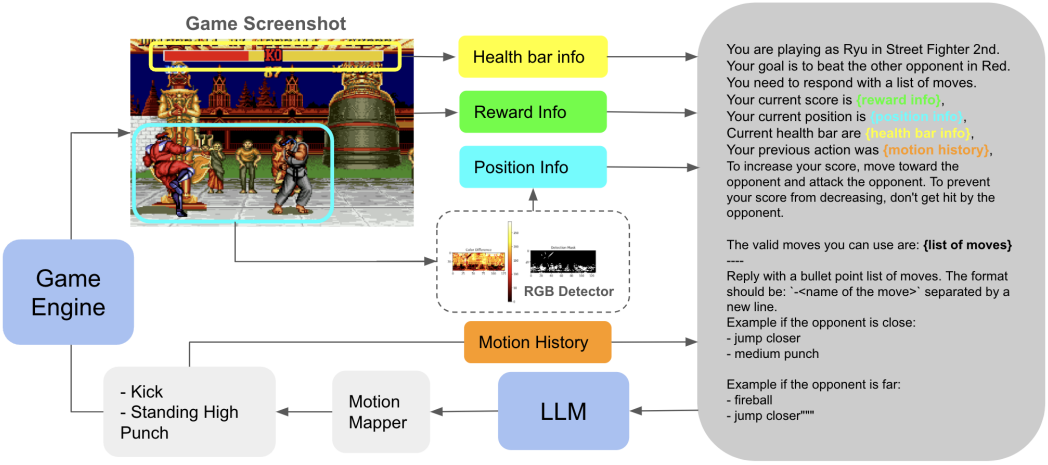
Using GPT-3.5 (text-only), GPT-4o (vision and text), CNN trained with PPO (7), and LLaVA without fine-tuning.

2. Fine-tuning LLM Model

Fine-tuning text-only LLM model Mistral 7B instruct v2 (8) with online reinforcement learning. Using RGB observer to extract visual information into texts, plus context prompt that describe the gaming scene, character status then passed to Mistral model.

3. Fine-tuning Multimodal Model

Fine-tuning the multimodal model LLaVA on game-specific scenarios using online reinforcement learning feedback loops, where the model's predictions are continually adjusted based on the outcomes of executed actions in the game environment.



For LLM models, they all require a text prompt. For vanilla models, we use the prompt as shown in the above figure with some hints such as description of what each action means. For fine-tuned model, we unify them with the following prompt: "You are playing Street Fighter, your goal is to defeat the opponent. Please check the given game state image. Your available movements are in this list: [punch, kick, move-left, move-right, jump, jump-left, jump-right, squat]. Pick one best movement to take from the list. Just answer in one word." where we describe the task and possible actions without giving any extra hint about what action to take. For LLaVA, we also pass the game screenshot to it as the image prompt.

All the LLM models outputs are pure text format, so we have a function to map the text output to multi-binary action vector by keyword matching.

To fine-tune Mistral and LLaVA, we used PPO method with online learning. We created a dataset wrapper from OpenAI Gym Retro Street Fighter game environment. It runs the environment under the hood, and exposes the game state and reward through its APIs. We tried to increase the batch size by running multiple environments in paralle, however, openAI gym only supports one environment to be running at the same time. For LLaVA, because it takes an extra image input, we created a custom class based on original TRL PPOTrainer to make PPO training work.

5 Experiments

5.1 Data

We use OpenAI Gym Retro Street Fighter game environment as our training and evaluation data, specificlly, we used the game state that has character Ryu fights with Guile in the final round of Street Fighter II (7). Instead of recording the data, all our training and evaluation are online. Specifically, we created a pytorch dataset class where under the hood it runs a game environment. Whenever __getitem__ is called, it returns the current game state as well as tokenized prompt as a tensor. It also exposes an API to progress the game state by taking an action.

5.2 Evaluation method

We have two main metrics to evaluate our models.

1) Winning Rate: We play 30 rounds of the game, and compute the winning rate. Higher winning rate means better model. This is the most important metric we use to compare different models.

2) Average Reward: We play 30 rounds of the game, and compute the average final reward. The reward is related to the HP difference between the player and opponent at the end of the game. Higher reward means better performance.

5.3 Experimental details

For baseline, we used text-only LLMs (GPT-3.5-turbo, Mistral-7B-Instruct-v0.2), multimodal model (GPT-4o, LLaVA) and PPO with CNN to get lower bound and upper bound for our experiments. The experimental pipeline, depicted in the accompanying diagram, utilizes an Observer class to interpret the game state from RGB pixel data to determine the relative positions and actions of game characters. The Observer processes frames to extract position data using color detection, which informs the models about each character's location relative to their opponent.

5.3.1 Text Only Models

For the text-only models, prompts are constructed based on the game's current state, including character and enemy positions, action history, and potential moves. These models can only generate responses based on textual descriptions of the game environment without direct visual inputs. To resolve this, the Observer class plays a critical role here, converting visual data from the game into a structured text format that allows the models "see" the current game dynamics. The original position detection logic in our experiments, adapted from LLM Colosseum (9), utilized a single RGB pixel to represent the character. This approach, while straightforward, proved insufficient due to the game's dynamic background, which sometimes closely matched the color of the character's pixel color, leading to frequent wrong position detection. To address these challenges, we developed a more robust method of detecting character positions, focusing on localized regions of the frame and improving accuracy through centroid calculations. The enhanced method involves analyzing a specified portion of the frame from the game, reducing interference from unrelated background colors, and utilizing a centroid calculation to better represent the character's position.

5.3.2 Multimodal Models

In contrast to the text-only approach, our multimodal models, which include **GPT-4o** and **LLaVA**, are designed to utilize both textual descriptions and direct frames from the game. This model synthesizes information from different modalities, thereby enhancing the accuracy of AI agent's decision-making. The direct visual inputs provide context that is inherently hard to capture using text-only descriptions, such as spatial dynamics and nuanced movements from opponents, which are critical for strategic gameplay in complex scenarios like Street Fighter.

5.3.3 Finetune Models with Reinforcement Learning

To optimize the performance of **LLaVA** and **Mistral** models specifically for our gaming environment, we use the `TRL PPOTrainer` library, which focus on training LLM with reinforcement learning. We

configured the training with single batch size and a learning rate of 1.41×10^{-5}, a total of 100,000 training iterations corresponding to individual game steps.

The training process leverages Nvidia's A6000 GPUs with 48GB of memory, utilizing **bfloat16** precision to maximize computational efficiency. The use of **LoRA** is to reduce the hardware constraints without significant loss of learning capacity.

Two distinct strategies were tested for the reinforcement learning phase:

- **Immediate Reward Update:** A gradient step was taken at every frame based on the immediate reward of the current frame's action.
- **Accumulated Reward Update:** Rewards were accumulated throughout a game round, with a single update step taken post-round based on the total accumulated reward.

Our experiments indicated that the immediate reward update method resulted in a higher winning rate, due to the more responsive adaptation to the game dynamics.

An additional strategy was employed to encourage aggressive gameplay strategies: during the first 10,000 iterations, a small reward bias was introduced for actions classified as attacks (punching or kicking), effectively motivating the models to engage more frequently in combat actions, which are essential for scoring in the game.

Implementation and Integration The integration of the PPOTrainer with our game environment required careful synchronization of the model outputs with the game engine inputs. The action commands generated by the models were directly translated into game controls using a mapping system that converts high-level actions (e.g., 'move left', 'jump right') into specific game engine commands.

This setup not only allowed for real-time model training and evaluation but also facilitated detailed analysis of model performance across different training conditions and configurations.

5.4 Results

Below are the winning rate and average reward of each model measured by playing 30 rounds of game.

Model	Winning Rate	Average Reward
Random Action	0	-0.0539
GPT-3.5	0	0.02
GPT-4o	0	0.074
Mistral	0	-0.061
LLaVA	0	-0.035
CNN RL	0.77	0.4825
LLaVA PPO	0.13	-0.038

6 Analysis

The results confirm the superior performance of the CNN trained with Reinforcement Learning (RL), which significantly outperforms other models. This was anticipated due to the model's direct training on the game environment, allowing it to adapt more effectively to its dynamics. Notably, the LLaVA model fine-tuned using our PPO-based training pipeline demonstrated improved performance over their untrained counterparts, validating the efficacy of our training approach.

A critical insight from our experiments relates to the reward function employed. While direct game-derived rewards (positive for hitting the enemy and negative for being hit) seem intuitively correct, they do not consistently correlate with the winning rate. This discrepancy suggests that the reward signals may not fully capture long-term strategic gameplay elements that contribute to winning. For instance, a model could achieve a high score by frequently hitting the opponent but still lose if those hits are not strategically significant. We observed this trend in both of baseline models and finetuned LLM models.

The improved performance of fine-tuned LLMs over their vanilla baselines suggests that while the general knowledge encoded in LLMs provides a foundation, domain-specific adaptations are crucial. However, the substantial performance gap between these models and the RL-trained CNN indicates that current fine-tuning approaches may not yet fully leverage the LLMs' potential in this gaming context. We also experimented training Mistral with PPO, but the final evaluation result indicated several shortcomings of using text-based LLMs in dynamic gaming environment. The challenge of using text-based Large Language Models in action-intensive games like Street Fighter can be attributed to as belows:

1. Loss of Critical Game Information due to Mismatch Between Modalities

Mistral like LLMs are designed to process and generate text, which lacks the spatial and contextual understanding of real-world environment. Game like Street Fighter II is deigned to be fast-paced, involving constant real-time decision making, which depends hevaily on visual information and action timing. LLMs cannot capture these nuances such as recognizing spatial relations, enemy actions through only text descriptions. This is obvious with Mistral baseline, where the model has higher probability of predicting 'move left' or 'move right' because the distance between enemy and character cannot be captured precisely.

Besides, converting game states into text results in significant loss of detail. For example, subtle movements or changes in enemy animation could signal an incoming attack or could signal a jump to the other side of the character's position, information that is crucial but may not be immediately captured through text. When the LLM predicts actions based on this incomplete and imprecise description, its ability to respond accurately is not guaranteed. This conversion process creates a lag between observing the state and actual acting, even though we constraint the frames that both characters could see and use when predicting the next move.

2. Complexity of Action Mapping

Action games like Street Fighter II require not just individual actions but sequences of actions at precise timing to make effective moves. Mistral like LLM models, despite being fine-tuned using PPO, may still struggle with learning effective action mapping purely from text. This is because the reward mechanism in reinforcement learning is typically clearer and more immediate when based directly on visual inputs rather than derived textual descriptions.

Considering these findings, redesigning the reward structure could be a promising direction to enhance the training, potentially leading to better alignment with actual winning conditions. Further experiment direction, such as tuning reward magnitudes or adding additional strategic elements into the reward calculation, may yield more consistent improvements in model performance.

7 Conclusion

Our investigation highlights the effectiveness of CNN models trained with RL, which substantially outperformed all other models in a straightforward gaming environment like Street Fighter II. Although LLMs fine-tuned with PPO showed significant improvements over non-fine-tuned versions, they still lagged behind the CNN model. This result underscores the limited utility of general knowledge encoded in LLMs when applied directly to simple, action-based game environments.

Despite the observed limitations, we believe that LLMs could exhibit greater potential in more complex or realistic game environments where decision-making involves multiple layers of strategy and long-term planning. In such scenarios, the comprehensive world knowledge and contextual understanding provided by LLMs might prove more advantageous.

Due to resource constraints, our training was not as extensive as desired. Future work could explore the impact of longer training durations, higher batch sizes, and more sophisticated reward mechanisms on the performance of fine-tuned LLMs. Adjusting these parameters could help in more closely aligning the learning process with the strategic requirements of the game, potentially bridging the gap between LLMs and RL-trained models.

In conclusion, while vanilla RL approaches currently dominate in simple gaming scenarios, the evolving capabilities of LLMs present a promising frontier for research in more complex environments. Further exploration and optimization of training paradigms and reward systems are essential to fully realize the potential of LLMs in gaming and beyond.

8 Ethics Statement

Integrating pre-trained Language Models (LLMs) in game-playing environments may lead to various ethical challenges and societal risks. First, there is the concern of reinforcement of biases. LLMs may contain societal biases, which usually come from its training dataset. In a gaming context, this could lead to biased decision-making strategies. Second, there is the potential for increasing addiction to games. Many people give up a game when they feel it becomes boring or too easy. However, with this LLM game agent, it might be always creating enough challenge to players, it may even react in a way that maximizes the addictiveness. Third, LLMs may have unfair advantages in player-vs-player scenarios, which will destroy one of the most important spirit of games: fairness.

For addressing bias, regular auditing of model decisions and outcomes can be established. This should involve diverse testing groups to ensure a wide range of human perspectives. To tackle the risk of enhancing addictive behaviors, design principles can be adopted that prioritize ethical engagement, such as limiting play time or incorporating features that promote healthy gaming habits. As for fairness, specific softwares could be developed to detect potential gaming LLMs that running in the background.

References

[1] OpenAI, "GPT-4o: model that can reason across audio, vision, and text in real time" Available online: `https://openai.com/index/hello-gpt-4o/` Accessed on: June 7, 2024.

[2] Haotian, L., Chunyuan, L., Qingyang, W., Yong, J. L., Visual Instruction Tuning. *arXiv preprint arXiv:2304.08485*.

[3] Carta, T., Romac, C., Wolf, T., Ahn, S., Chevalier-Boisvert, M., Dai, B., Towers, M., de Lazcano, R., Willems, L., Lahlou, S., Pal, S., Castro, P. S., Terry, J. (2023). Grounding Large Language Models in Interactive Environments with Online Reinforcement Learning. *arXiv preprint arXiv:2302.02662*.

[4] Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., & Wei, J. (2022). Scaling Instruction-Finetuned Language Models. *arXiv preprint arXiv:2210.11416*.

[5] Du, Y., Watkins, O., Wang, Z., Colas, C., Darrell, T., Abbeel, P., Gupta, A., & Andreas, J. (2023). Guiding Pretraining in Reinforcement Learning with Large Language Models. *arXiv preprint arXiv:2302.06692*.

[6] Zhou, Y., Zanette, A., Pan, J., Levine, S., & Kumar, A. (2024). ArCHer: Training Language Model Agents via Hierarchical Multi-Turn RL. *arXiv preprint arXiv:2402.19446*.

[7] StreetFighterAI (2023). Trained a RL agent to beat final boss in Street Fighter II. Retrieved from `https://github.com/linyiLYi/street-fighter-ai/tree/master`

[8] Mistral 7B, "The best 7B model to date, Apache 2.0: `https://mistral.ai/news/announcing-mistral-7b/` Accessed on: June 7, 2024.

[9] DIAMBRA. (2023). LLM Colosseum Documentation. Retrieved from `https://docs.diambra.ai/projects/llmcolosseum/`

[10] SuperCombo Wiki. (2023). Street Fighter 2: Champion Edition/Ryu. Retrieved from `https://wiki.supercombo.gg/w/Street_Fighter_2:_Champion_Edition/Ryu`