Reinforcement Learning Gyms

Kefan Song

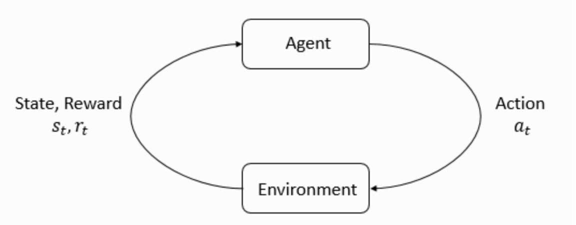
Department of Computer Science
University of Virginia

Overview

- Motivation: Why Reinforcement Learning Gyms
- Physical RL Gyms
  - Atari (discrete control)
  - MuJoCo (continuous control)
  - Isaac Lab (embodied & sim-to-real)
- RL for Large Language Models
  - RL from Human Feedback
    - OpenRLHF
    - TRL
  - RL from Verifiable Rewards
    - Verl
    - Tinker
    - SimpleGRPO

RL Gyms are Simulated Environments of the Real World

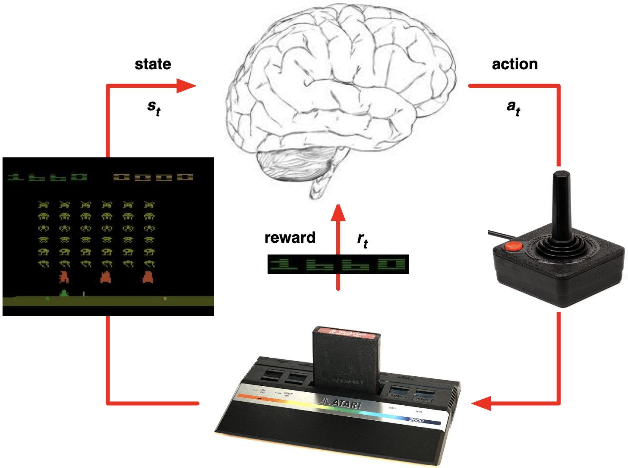
- Recall that Reinforcement Learning involves learning by trial and error.
- It requires collecting trajectories of states, actions, and rewards from the environment.
- AlphaGo Zero required **4.9M self-play games** (Silver et al. 2017).
- Such massive trial-and-error is inefficient and unsafe in the real world.
- RL researchers started with games...



RL Env: Atari

- DeepMind aims to develop a general-purpose learning algorithm for Artificial General Intelligence.
- This requires a single algorithm that can solve diverse tasks at or above human level.
- Atari provides such a testbed (Bellemare et al. 2013): it contains 472 diverse video games, each requiring a different strategy.
- Deep Q-Network (DQN) was developed to solve Atari.
  - End-to-end learning: input raw video frames, output discrete control actions.
  - Achieved human-level or better performance in **29 of 49** benchmarked games (Mnih et al. 2015).

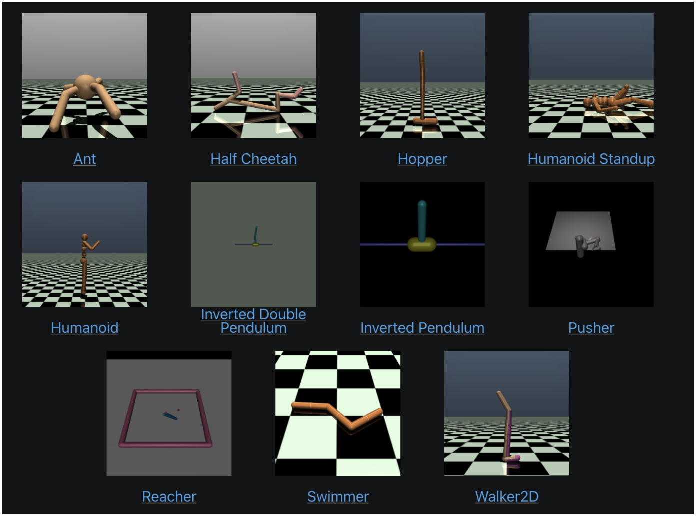
RL Env: Atari



RL Env: Mujoco

- Real-world control is continuous: you can move or rotate your arm by any arbitrary degree.
- In contrast, Atari's action space is discrete, e.g., left or right for Pong.
- **Mujoco** is proposed as a testbed for RL algorithms on continuous control problems (Todorov, Erez, and Tassa 2012).
- Trying to solve Mujoco led to algorithms like TRPO and PPO.
- PPO later becomes the de facto RL algorithm in many settings, including post-training LLMs.

RL Env: Mujoco



RL Env: ISSAC Lab

- Mujoco comes with simplified robots (HalfCheetah, 2D Walker).
- One can customize the .xml file to design their own robot in Mujoco.
- Yet, there is a notorious gap between simulation and reality in robotics.
- Issac Lab is designed to simulate real-world scenarios (NVIDIA 2024).

Starting Point to Understanding RL Code Implementation: CleanRL

- RL code has more complexity than supervised learning.
- Single File High Performance Implementation of popular RL algorithms (S. Huang et al. 2022).
- For example, the ppo_atari.py only contains 340 lines of code.

CleanRL: Training on (Atari & MuJoCo & IssacGym)

- Train DQN on Breakout (Atari):

```
python cleanrl/dqn_atari.py --env-id BreakoutNoFrameskip-v4
--capture-video
```

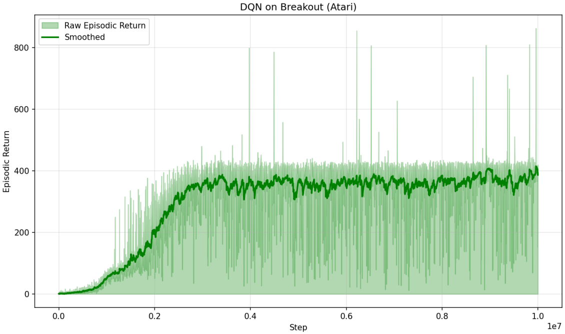
- Train PPO on HalfCheetah (MuJoCo):

```
MUJOCO_GL=egl python cleanrl/ppo_continuous_action.py --env-id
HalfCheetah-v4 --capture-video
```

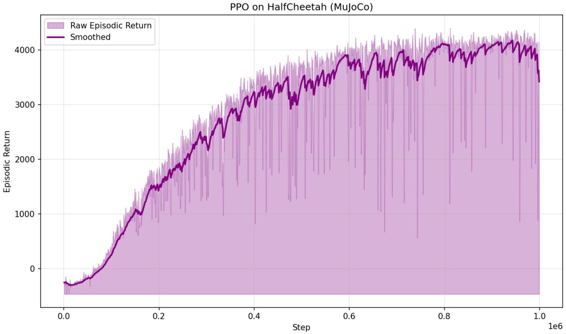
- Train PPO on a quadruped robot (IsaacGym):

```
python
cleanrl/ppo_continuous_action_isaacgym/ppo_continuous_action_isaacgym.py
--env-id Anymal
```

CleanRL Training Results

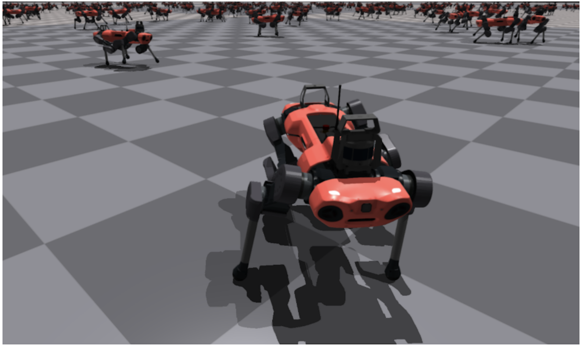


DQN on Breakout (Atari)

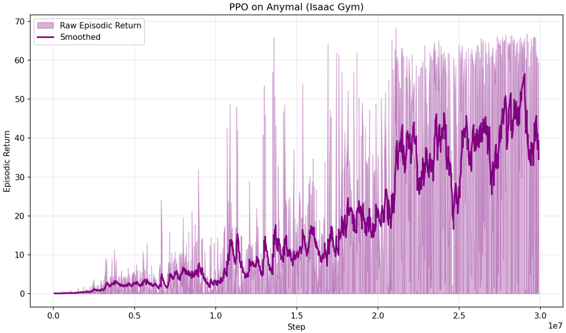


PPO on HalfCheetah (MuJoCo)

CleanRL Training Results



ANYmal training in Isaac Gym



PPO on ANYmal (IssacGYM)

RL for Post-Training LLMs

- From LLM perspective, RL is the last stage of fine-tuning:
  - Stage 1: Pre-training with self-supervised learning on next token prediction on large corpus of text.
  - Stage 2: Supervised fine-tuning on human-annotated responses.
  - Stage 3a: Reinforcement Learning from Human Feedback **(RLHF)** to further improve helpfulness, and reduce harmfulness.
  - Stage 3b: Reinforcement Fine-tuning with Verifiable Reward **(RLVR)** on Math and Coding to Incentivize Reasoning.

RL for Post-Training LLMs

- From RL Perspective, natural language tasks fill a missing piece in existing RL Gyms
- Atari/Go/DOTA covers action space defined in human-designed games.
- Mujoco/IssacLab covers continous control action space, robotic movement in the real-world.
- Missing out an essential action space for human: **Language**.

Reinforcement Learning from Human Feedback (RLHF)

- Stage 1: SFT.
- Stage 2: Reward Modeling.
- Stage 3: RL post-training.

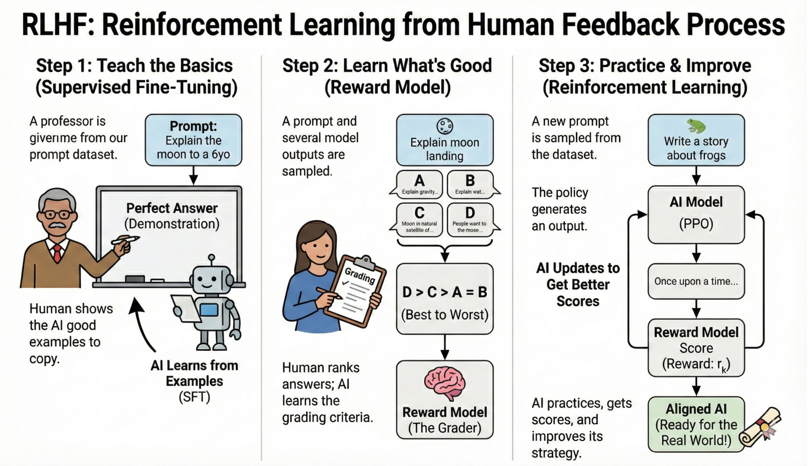


Diagram of the three stages of RLHF

OpenRLHF

- Provides a stable and scalable RLHF implementation (Hu et al. 2024).
- Includes a built-in Reward Modeling pipeline
  - Originally designed for modeling human preference over responses
  - Can be adapted to other tasks
  - Prepare your customized instruction, accepted response, and rejected response, and use `openrlhf/datasets/reward_dataset.py` to process the dataset.
- Typically requires substantial GPU memory
  - Tip: use a remote reward model to reduce GPU usage
  - Or use DPO for further reduction (with some performance trade-offs)

OpenRLHF provides many example training scripts (.sh)

Training DPO in Practice

- **Hands-on Practice:** Run this Google Colab Notebook[1].
  Note: Requires a GPU (e.g., A100 via Colab Pro) for efficient training.
- **Model (SFT Base):** We use the TRL pipeline to fine-tune Zephyr-7B-sft (Tunstall et al. 2023), a Mistral-7B derivative already fine-tuned on the UltraChat dataset.
- **Offline DPO Method:** Unlike online RL (e.g., PPO), DPO is **offline** and learns from a static dataset without generating new samples during training.
- **Dataset Construction:** We utilize the **UltraFeedback** dataset (Cui et al. 2024) (60k+ prompts, GPT-4 labeled). To create the preference pairs (y_w, y_l) required for DPO:
  - **Chosen (y_w):** The response with the highest overall score.
  - **Rejected (y_l):** A randomly selected remaining response.

[1] `https://colab.research.google.com/drive/1i7aDdW-RHNNPgCCHW9Ky858riqYbrN1R?usp=sharing`

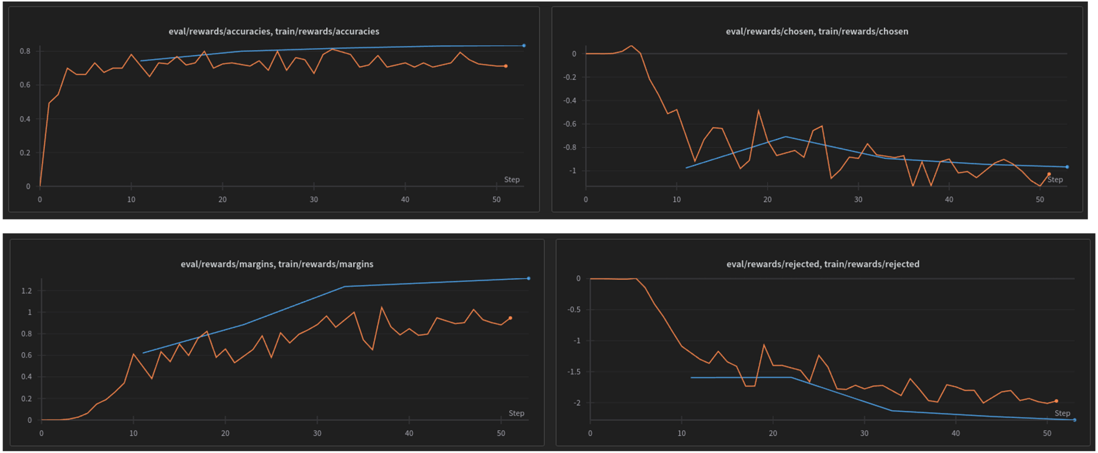


Figure: Examplar Training Curve of DPO[1].

[1] Image borrowed from the tutorial at https://www.youtube.com/watch?v=QXVCqtAZAn4.

Reinforcement Learning from Verifiable Reward

- RL from Human Feedback works well for tasks requiring subjective evaluations, which is great for conversational assistants (ChatGPT).
- However, many hard real-world tasks require objective evaluation.
- Examples: olympia math, coding, and websearch for an answer.
- Idea: use Rule-Based Reward as in Atari, Mujoco and Go,
- Specifically, we
  - (a) Verify the correctness of the model's answer against ground truth.
  - (b) Verify the format <think></think> <answer></answer> to require the LLM produce a chain-of-thought reasoning before the final answer.

Understanding RLVR Training: SimpleGRPO

Task Setup: Training on the GSM8k dataset (8k problems).

Format Reward

```python
def reward_format(item, answer):
    pattern = r"^<think>.*?</think>[\n ]*<answer>.*?</answer>$"
    think_count = answer.count("<think>") + answer.count("</think>")
    answer_count = answer.count("<answer>") + answer.count("</answer>")
    return 1.25 if re.match(pattern, answer, re.DOTALL | re.VERBOSE) and think_count==2 and answer_count==2 else -1
```

Correctness Reward

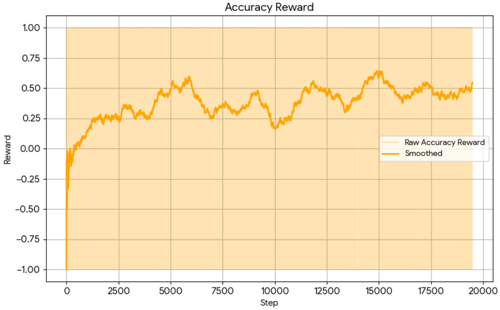
simple_GRPO / grpo_vllm_one.py

Code Blame 271 lines (242 loc) · 12.7 KB

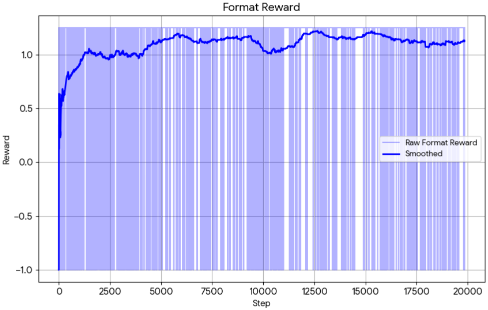
```python
def gen_worker(Q, physics_device):
    def gen_answers(prompts):

        from math_verify import parse, verify, ExprExtractionConfig
        def reward_correct(item, answer):
            pattern = r'\d+\.\d+|\d+/\d+|\d+'
            nums = re.findall(pattern, answer)
            if len(nums) == 0: return -1.0
            lastnum = nums[-1]
            ans = parse(lastnum, extraction_config=[ExprExtractionConfig()])
            ground_truth = parse(item["A"], extraction_config=[ExprExtractionConfig()])
            return 1 if verify(ans, ground_truth) else -1
```

RLVR Training Framework: SimpleGRPO

- Training Qwen-1.5B Model on GSM8k dataset



Accuracy Reward Curve



Format Reward Curve

List of RL-Fine-tuning Frameworks

- **TRL**: One of the earliest RL-finetuning libraries from Hugging Face (Werra et al. 2020).
- **SimpleGRPO**: Designed for clarity and ease of understanding (Liang et al. 2025).
- **Llama-Factory**: Capable of fine-tuning trillion-parameter models (e.g., Kimi-k2) using ktransformers with 2 Nvidia 4090 GPUs (Zheng et al. 2024).
- **Verl**: A scalable, production-ready framework supporting multi-turn tool-calling (Dou et al. 2024).
- **Tinker**: A recent framework (by Thinking Machines) focused on stable and robust PPO fine-tuning (Lab 2025).

A Natural Next Question

- Great RL Gyms often inspire great RL algorithms.
  - DQN was created to solve Atari.
  - PPO was created to solve MuJoCo.
- Since then, PPO (and its variants like GRPO) has been applied to many other domains (reasoning, real-world humanoid robots, etc.).
- Yet current models remain far from the original AGI ambition (even optimistic estimates place it 5–10 years away).
- We still rely on RL algorithms that are nearly a decade old!

What is missing in these RL Gyms?

Thanks!

References I

- Bellemare, Marc G et al. (2013). "The arcade learning environment: An evaluation platform for general agents". In: *Journal of Artificial Intelligence Research* 47, pp. 253–279.
- Cui, Ganqu et al. (2024). "ULTRAFEEDBACK: Boosting Language Models with Scaled AI Feedback". In: *Forty-first International Conference on Machine Learning*.
- Dou, Guangyu et al. (2024). "HybridFlow: A Flexible and Efficient RLHF Framework". In: *arXiv preprint arXiv:2409.19256*. Underlying framework for Verl.
- Hu, Jian et al. (2024). "OpenRLHF: An Easy-to-use, Scalable and High-performance RLHF Framework". In: *arXiv preprint arXiv:2405.11143*.
- Huang, Shengyi et al. (2022). "CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms". In: *Journal of Machine Learning Research* 23.274, pp. 1–18. URL: http://jmlr.org/papers/v23/21-1342.html.
- Lab, Thinking Machines (Oct. 2025). *Tinker: A New Era of AI Model Fine-Tuning*. https://thinkingmachines.ai/blog/announcing-tinker/. API and Cookbook for stable PPO fine-tuning.

References II

- Liang, Jiaqing et al. (2025). *KW-R1: A Simple Implementation of the GRPO Algorithm (SimpleGRPO)*. https://github.com/lsdefine/simple_GRPO.
- Mnih, Volodymyr et al. (2015). "Human-level control through deep reinforcement learning". In: *Nature* 518, pp. 529–533.
- NVIDIA (2024). *Isaac Lab: A Unified Framework for Robot Learning*. https://developer.nvidia.com/isaac-lab.
- Silver, David et al. (2017). "Mastering the game of Go without human knowledge". In: *Nature* 550, pp. 354–359.
- Todorov, Emanuel, Tom Erez, and Yuval Tassa (2012). "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 5026–5033.
- Tunstall, Lewis et al. (2023). "Zephyr: Direct distillation of lm alignment". In: *arXiv preprint arXiv:2310.16944*.

References III

Werra, Leandro von et al. (2020). *TRL: Transformer Reinforcement Learning*. https://github.com/huggingface/trl.

Zheng, Yaowei et al. (2024). “LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models”. In: *arXiv preprint arXiv:2403.13372*.