THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
RELAXED SYSTEM LAB

Reinforcement Learning Alignment

COMP4901Y

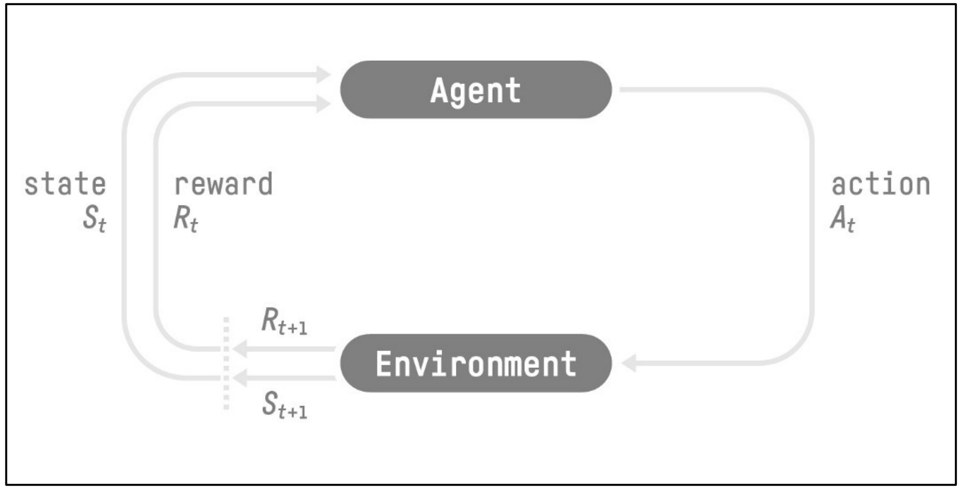
Binhang Yuan

Reinforcement Learning Preliminary

What is Reinforcement Learning?

RELAXED SYSTEM LAB

- Basic idea: Reinforcement Learning (RL) is a framework for solving decision problems by building agents that learn from the environment by interacting with it through trial and error and receiving rewards (positive or negative) as unique feedback.
- RL Framework:
  - **State**: information our agent gets from the environment;
  - **Action**: possible movement our agent can take in an environment.
  - **Reward**: the feedback our agent gets from the environment.



Markov Decision Processes

RELAXED SYSTEM LAB

- An MDP formalizes sequential decision-making under uncertainty. It is defined by a tuple of (S, A, P, R, γ):
  - States S: the set of possible states;
  - Actions A: the set of possible actions;
  - Transition probability $P(s'|s, a)$: transition probability of next state s' given current state s and action a;
  - Reward $R(s, a)$: reward for taking action a in state s;
  - Discount factor $\gamma \in [0,1]$: for future rewards.
- **Agent** environment loop: At each time step, the agent observes state s, takes action a, transitions to s and receives reward $R(s, a)$ from the environment.
  - The process is Markovian: the next state distribution depends only on the current state and action (Markov property).
- **Policy:** A policy π defines the agent's behavior, mapping states to actions. It can be deterministic $\pi(s) = a$ or stochastic $\pi(a|s) = P(a|s)$.

Return and Value Function

RELAXED SYSTEM LAB

- **<u>Return (cumulative future reward)</u>** G_t: represents the cumulative reward an agent expects to receive over time, starting from time step t. It accounts for both immediate and future rewards, incorporating the discount factor γ to prioritize immediate rewards over distant ones:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k}) = \sum_{k=0}^{\infty} \gamma^k R_{t+k}$$

- **<u>State-value function</u>** $V^{\pi}(s)$: The expected return (i.e., cumulative future rewards) when starting from state s and following policy π:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s\right]$$

Return and Value Function

- **Action-State-value function** $Q^{\pi}(s,a)$: The expected return (i.e., cumulative future rewards) after taking action a from state s and following policy π:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^{t} R(s_t,a_t)\,\middle|\, s_0 = s, a_0 = a\right]$$

- Relationship between $V^{\pi}(s)$ and $Q^{\pi}(s,a)$: The state-value function is the expected value of the action-value function over the policy's action distribution:

$$V^{\pi}(s) = \sum_{a \in A} \pi(a|s)\, Q^{\pi}(s,a)$$

Optimal Policy

- The core goal of reinforcement learning (RL) training is to enable an agent to learn an **optimal policy** that *maximizes the expected cumulative reward through interactions with an environment*.
- To obtain such an optimal policy:
  - Value based method;
  - Policy based method;
  - Combine both: actor-critic method.

Value Based Method

- *Value-based methods focus on estimating value functions*: Specifically, the expected return (cumulative future rewards) from states or state-action pairs. The agent derives its policy implicitly by selecting actions that maximize these estimated values.
- Value function estimation:
  - Learn functions $V^{\pi}(s)$ and $Q^{\pi}(s,a)$ to evaluate the desirability of states or actions.
  - $V^{\pi}(s)$ and $Q^{\pi}(s,a)$ can be parameterized by some neural networks.
  - Learning method: Monte Carlo vs Temporal Difference (https://huggingface.co/learn/deep-rl-course/en/unit2/mc-vs-td)
- Policy derivation:
  - The policy is not explicitly learned but is derived by choosing actions that maximize the estimated value, e.g., selecting the action with the highest $Q^{\pi}(s,a)$.

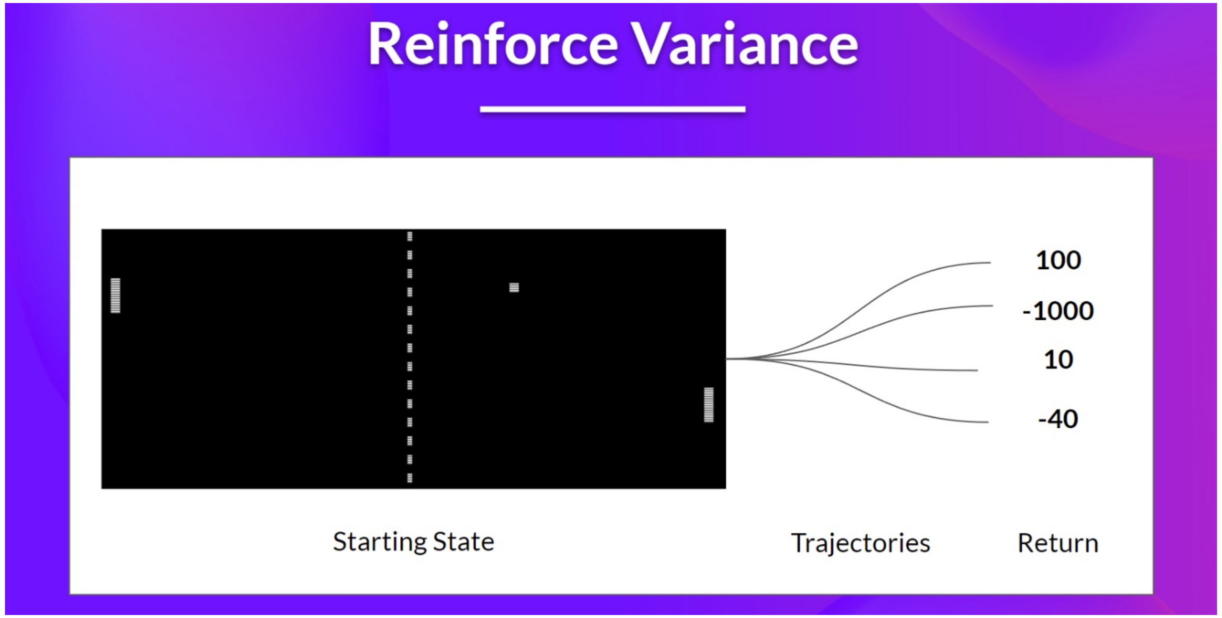
Policy Based Method

RELAXED SYSTEM LAB

- Policy-based methods *directly parameterize and optimize the policy* $\pi(a|s)$, which defines the agent's behavior by *mapping states to action probabilities*.
- Well-suited for environments with continuous or high-dimensional action spaces.
- Policy optimization:
  - The policy is explicitly represented (e.g., by some neural network) and optimized, often using gradient based optimization methods.
  - Training based on policy gradient:
    - Objective function: $J(\theta) = \mathbb{E}_{\tau \sim \pi}[R(\tau)]$;
    - where τ is a sampled trajectory of the policy π;
    - Policy gradient estimation: $\nabla_\theta J(\theta) \approx \sum_{t=0}^{|\tau|} \nabla_\theta \log \pi_\theta (a_t|s_t) R(\tau)$.

Discussion about Policy Gradients

- We collect a trajectory and calculate the discounted return;
- We use this score to increase or decrease the probability of every action taken in that trajectory.
  - If the return is good, all actions will be "reinforced" by increasing their likelihood of being taken.
  - The estimation is unbiased — we use only the true return we obtain.
- Given the stochasticity of the environment (random events during an episode) and stochasticity of the policy:
  - Trajectories can lead to different returns, which can lead to high variance.
  - The same starting state can lead to very different returns.
  - The return starting at the same state can vary significantly across episodes.



Actor-Critic Methods

- Actor-critic methods integrate value-based and policy-based approaches to leverage their respective strengths:
  - **Actor**: Represents the policy $\pi(a|s)$ and is responsible for selecting actions.
  - **Critic**: Estimates the value function $V^{\pi}(s)$ or $Q^{\pi}(s, a)$ to evaluate the actions made by the actor.
- Modify the policy gradients by the **advantage function** $A(s, a)$:
- $A(s, a)$quantifies the relative value of taking action a in state s compared to the average expected value of state s: $A(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$
- Policy gradient estimation changes to $\nabla_\theta J(\theta) \approx \sum_{t=0}^{|\tau|} \nabla_\theta \log \pi_\theta (a_t|s_t) A(s_t, a_t)$.

RELAXED SYSTEM LAB

RL Alignment for Language Model

Overview

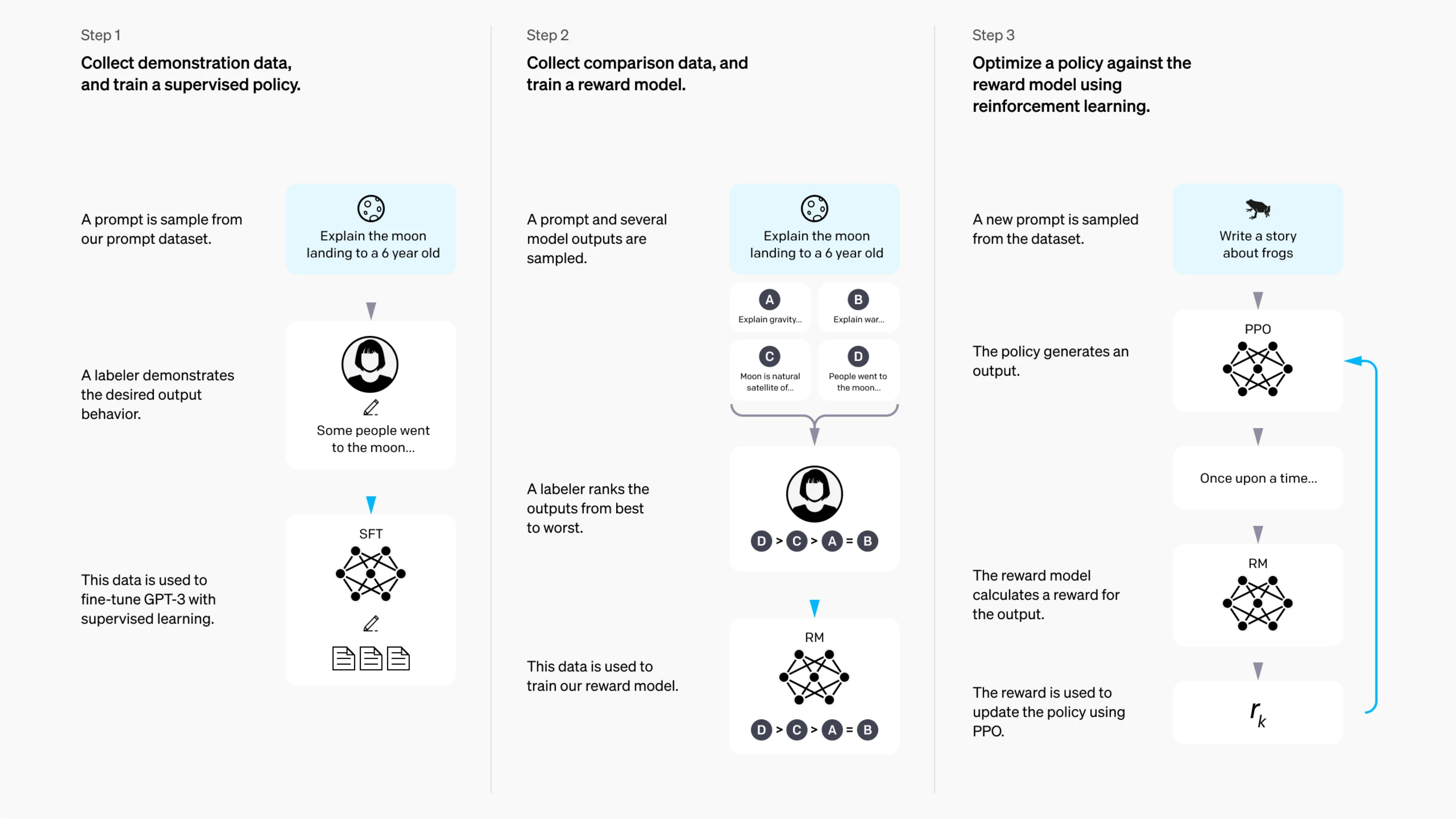
- Under the context of LLM alignment by RL:
  - **State**: The state corresponds to the input prompt provided to the language model. In RL terms, this is the observation that informs the agent's next action.
  - **Action**: The action is the response generated by the LLM in reaction to the input prompt. Each token produced can be considered an individual action, making the entire response a sequence of actions. This sequential generation aligns with the token-by-token decision-making process in reinforcement learning.
  - **Policy**: The policy in this context is the LLM itself, which maps input prompts (states) to probability distributions over possible next tokens (actions).
  - **Reward**: The reward is a scalar value representing the quality of the generated response, which guides the policy updates, encouraging the model to produce outputs that align with human expectations.

More on Rewards

- Two main categories of rewards: model-based reward v.s. rule-based reward.
- Model-based reward:
    - Model-based rewards utilize learned models—often neural networks—to predict the quality of a model's output. These models are trained on datasets reflecting some desired behaviors.
    - E.g., the reward model that learns human preferences in RLHF.
- Rule-based reward:
    - Rule-based rewards are determined using explicit, predefined rules or heuristics that assess the correctness or quality of outputs.
    - E.g., mathematical correctness verifier, program execution results (a set of unit tests associated with the problem).

OpenAI InstructGPT First Introduces RLHF

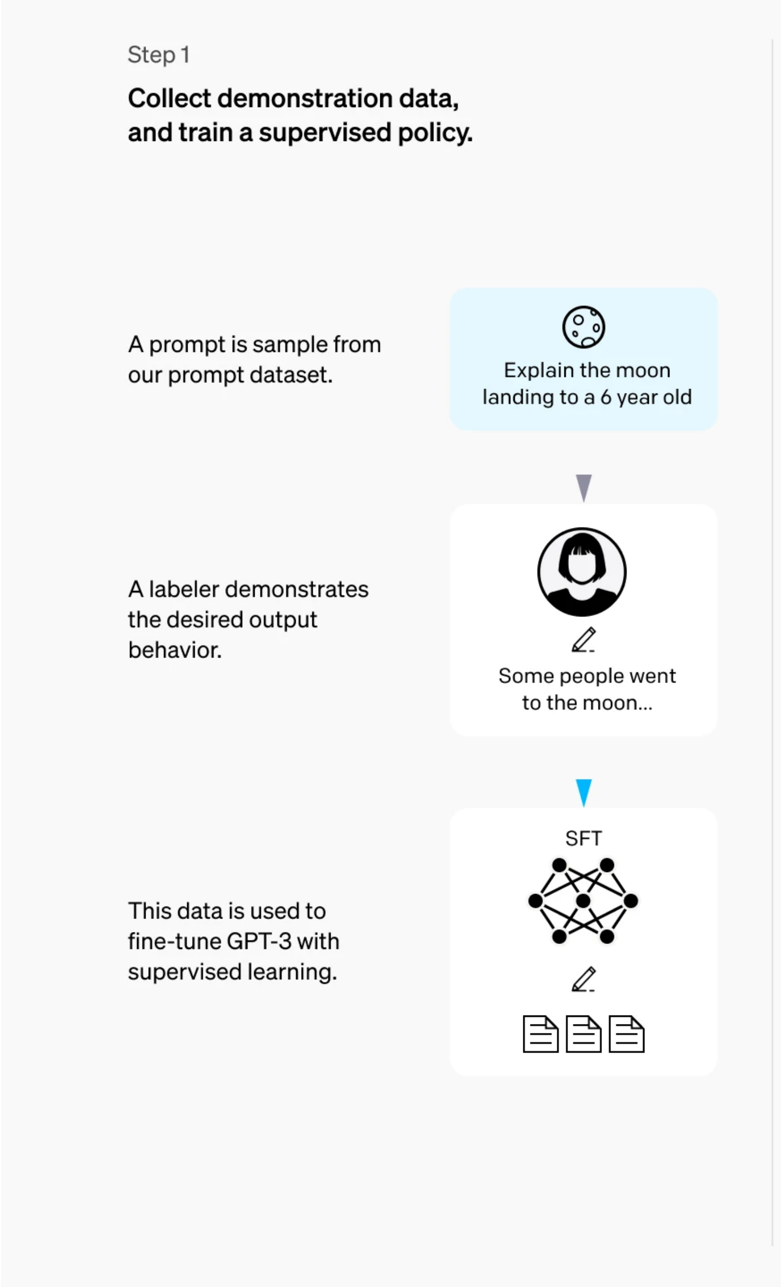
RELAXED SYSTEM LAB



https://openai.com/index/instruction-following/

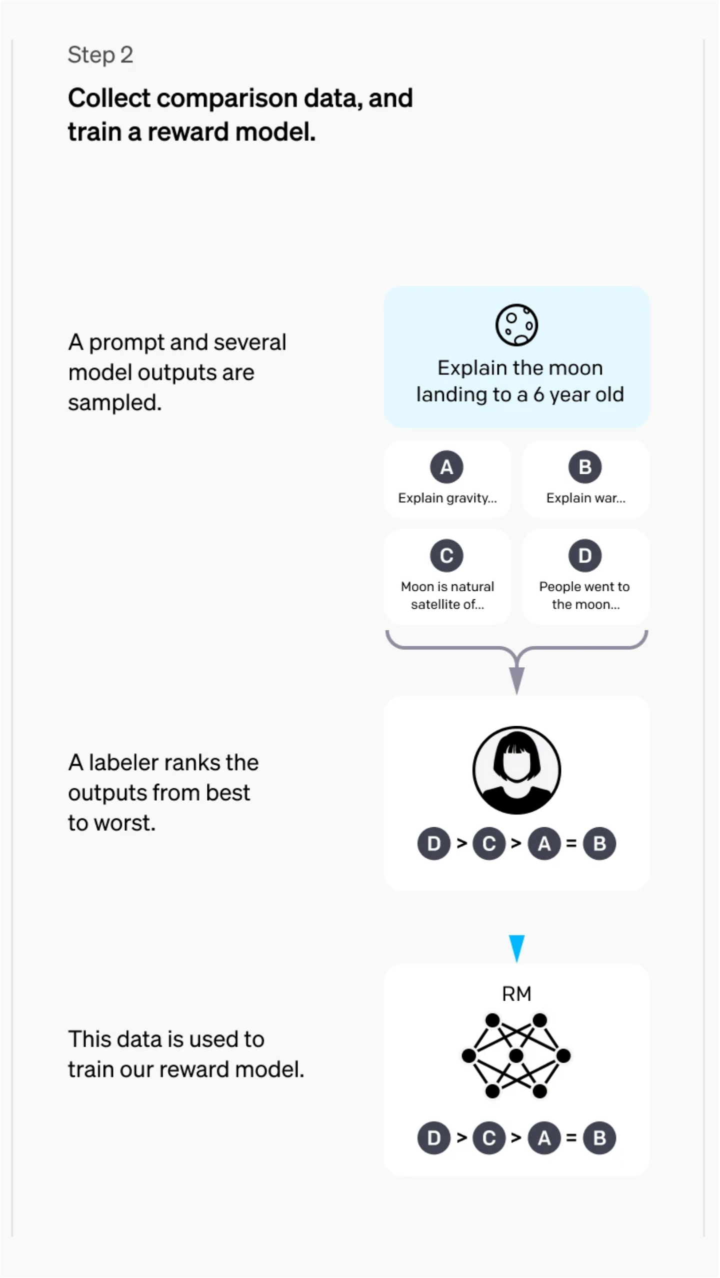
RLHF Step 1

- **<u>Supervised Fine-Tuning (SFT):</u>**
  - In this initial phase, human labelers provide demonstrations of desired model behavior by writing ideal responses to a variety of prompts.
  - These demonstrations are used to fine-tune a pre-trained language model, teaching it to generate outputs that align more closely with human expectations.



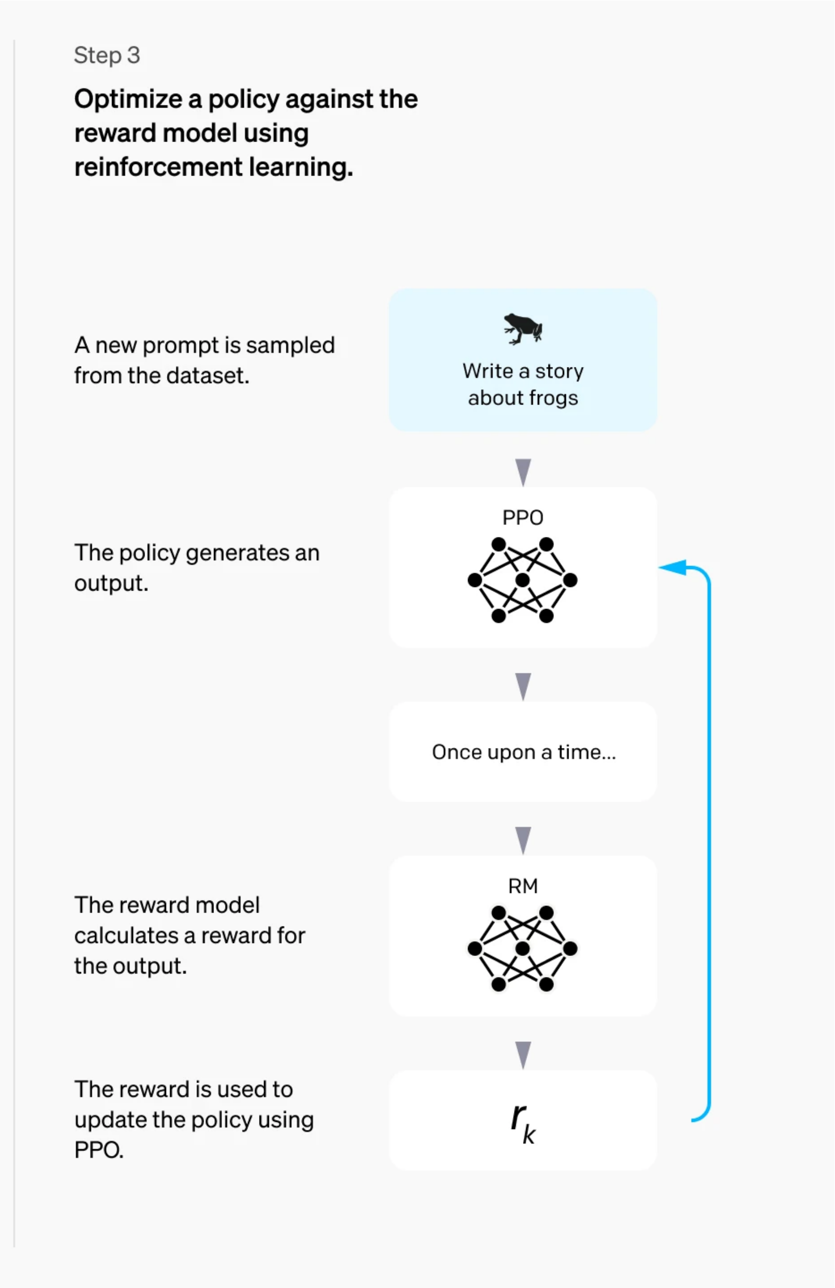
RLHF Step 2

- <u>**Reward Model Training**</u>:
  - The fine-tuned model generates multiple responses to a set of prompts.
  - Human labelers then rank these responses based on their quality.
  - These rankings are used to train a reward model that can predict the quality of responses, effectively learning to assess how well a response aligns with human preferences.



RLHF Step 3

- **<u>Reinforcement Learning</u>:**
  - In the final stage, the model is further fine-tuned using RL (PPO algorithm), guided by the reward model developed in the previous step.
  - This process encourages the model to produce outputs that maximize the predicted reward, leading to responses that better adhere to human instructions and values.



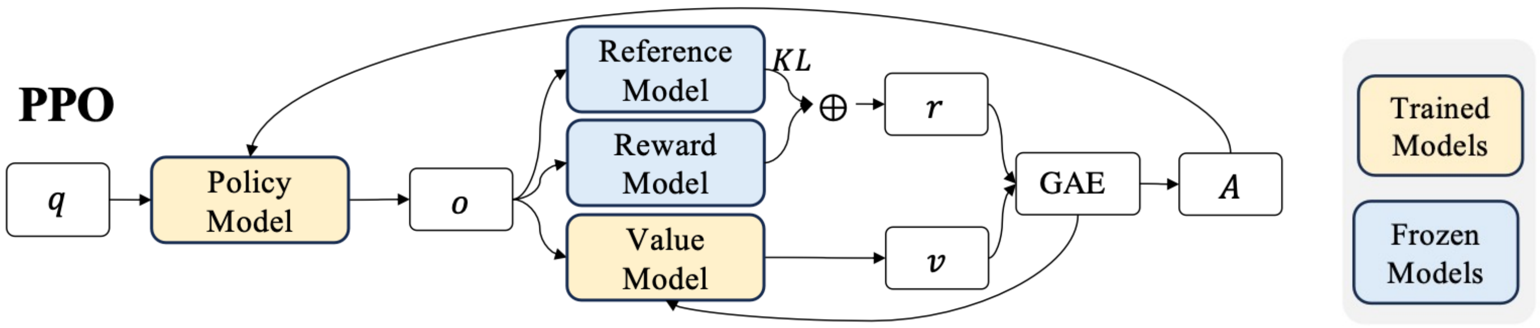
PPO Algorithm

RELAXED SYSTEM LAB

- Four different model works jointly in PPO:
- **Actor Model (Policy Model)** π_θ:
  - Generates responses to prompts;
  - The model to be aligned as the output of RLHF;
  - Parameter θ is updated during PPO;
- **Reward Model** r:
  - Provides scalar rewards indicating the quality of responses.
  - Trained separately using human-labeled data.
  - Remains fixed during PPO training. (so we ignore the parameter notation here).
- **Reference Model** π_{ref}:
  - Acts as a baseline to prevent the actor model from deviating excessively from its original behavior;
  - Used to compute the Kullback-Leibler (KL) divergence penalty, discouraging the actor from straying too far from its initial policy.

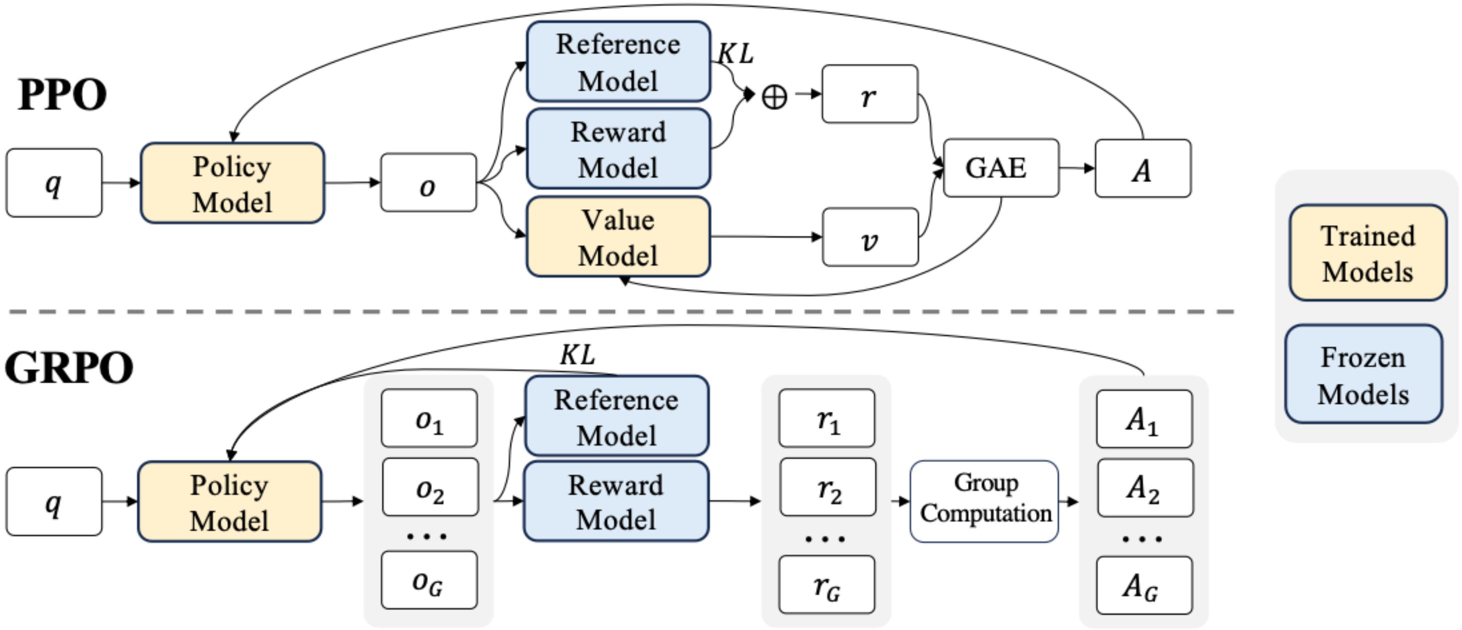
PPO Algorithm

- <u>**Critic Model (Value Function)**</u> V_{ϑ}:
  - Estimates the expected future reward (value) of a given state or prompt.
  - Reduce variance in policy updates.
  - Parameter ϑ is updated during PPO training.

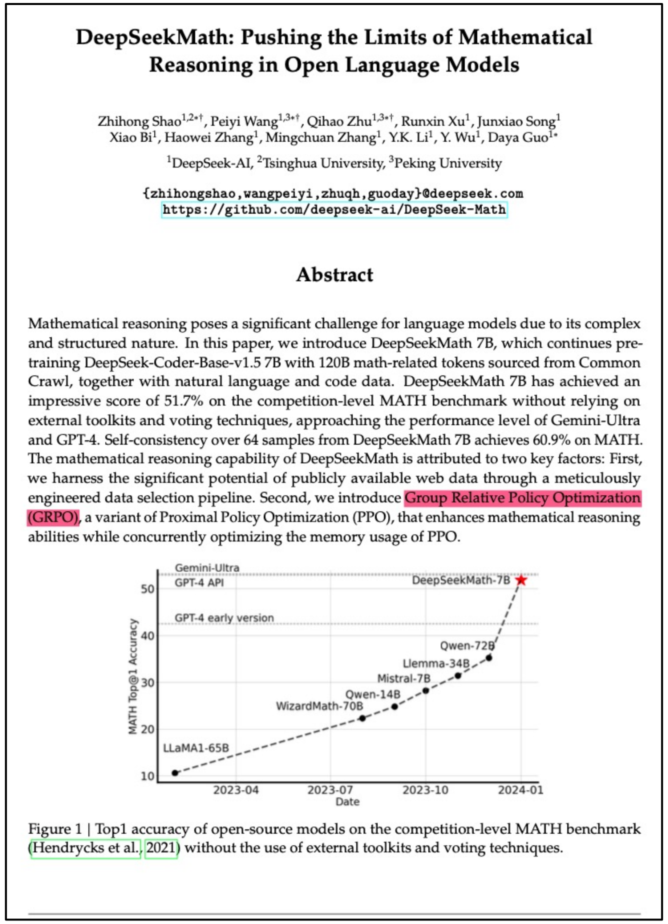


- q is the input question prompt; o is the output generated by the policy model;
- r is the adjusted reward with KL divergence;
- v is the value estimated by the value model;
- A is the advantage, which is computed by applying Generalized Advantage Estimation (GAE): https://arxiv.org/abs/1506.02438

GRPO



- Value function employed in PPO is typically another model of comparable size as the policy model, which brings a substantial memory and computational burden.
- Group Relative Policy Optimization (GRPO) uses the average reward of multiple sampled outputs, produced in response to the same question, as the estimation to replace the value model.



DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models

Zhihong Shao[1,2*†], Peiyi Wang[1,3*†], Qihao Zhu[1,3*†], Runxin Xu[1], Junxiao Song[1]
Xiao Bi[1], Haowei Zhang[1], Mingchuan Zhang[1], Y.K. Li[1], Y. Wu[1], Daya Guo[1*]

[1]DeepSeek-AI, [2]Tsinghua University, [3]Peking University

{zhihongshao,wangpeiyi,zhuqh,guoday}@deepseek.com
https://github.com/deepseek-ai/DeepSeek-Math

Abstract

Mathematical reasoning poses a significant challenge for language models due to its complex and structured nature. In this paper, we introduce DeepSeekMath 7B, which continues pre-training DeepSeek-Coder-Base-v1.5 7B with 120B math-related tokens sourced from Common Crawl, together with natural language and code data. DeepSeekMath 7B has achieved an impressive score of 51.7% on the competition-level MATH benchmark without relying on external toolkits and voting techniques, approaching the performance level of Gemini-Ultra and GPT-4. Self-consistency over 64 samples from DeepSeekMath 7B achieves 60.9% on MATH. The mathematical reasoning capability of DeepSeekMath is attributed to two key factors: First, we harness the significant potential of publicly available web data through a meticulously engineered data selection pipeline. Second, we introduce Group Relative Policy Optimization (GRPO), a variant of Proximal Policy Optimization (PPO), that enhances mathematical reasoning abilities while concurrently optimizing the memory usage of PPO.

Figure 1 | Top1 accuracy of open-source models on the competition-level MATH benchmark (Hendrycks et al., 2021) without the use of external toolkits and voting techniques.

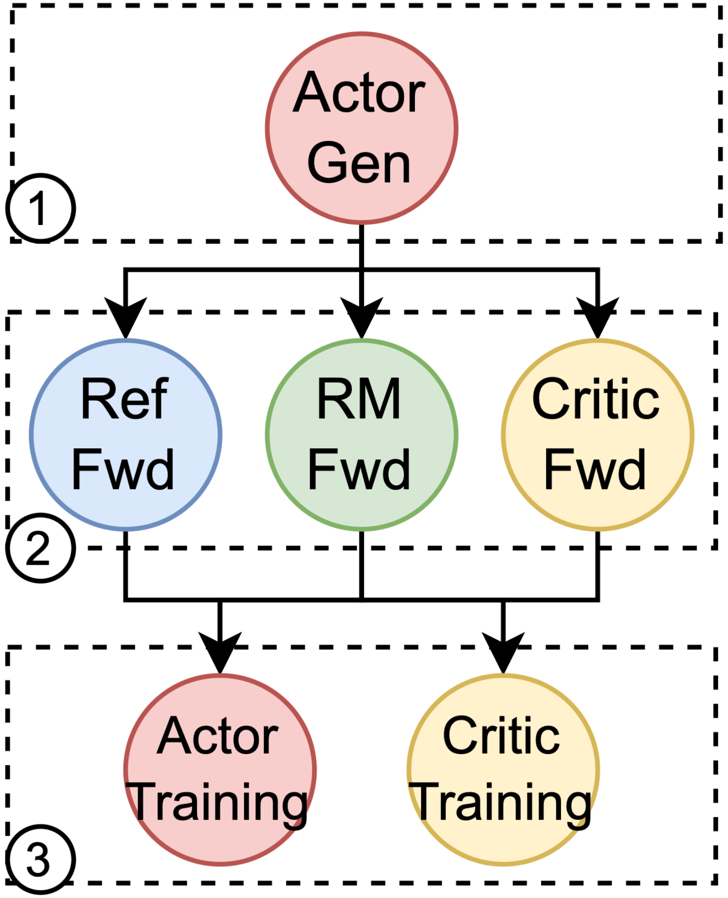
RELAXED SYSTEM LAB

System Challenges for RL Alignment

The RL Alignment Workflow is Unique

RELAXED SYSTEM LAB

- **<u>Stage 1 Generation</u>**: The actor produces responses from a batch of prompts using generative inference.
- **<u>Stage 2 Preparation</u>**: Using prompts and generated responses, the critic computes their values, the reference policy computes their reference log probabilities, and the reward model computes their rewards, all via a single pass of forward computation of the respective model.
- **<u>Stage 3 Training</u>**: The actor and the critic are updated via gradient based optimization, using the batch of data produced by previous stages and the loss function.



Unique Challenges

RELAXED SYSTEM LAB

- ***Heterogeneous model workloads***:
  - The actor, critic, reference and reward models in RLHF may execute training, inference or generation at different stages, with different memory footprint and computation demand.
- ***Unbalanced computation between actor training and generation***:
  - Actor training is computation bounded, which requires a carefully tuned configuration of large-degree parallelism;
  - The same configuration might make the inference efficiency decreases;
- ***Diverse model placement requirements***:
  - Strategic device placement of models in the RLHF dataflow is necessary, according to computation workloads and data dependencies of the models.

Still Open Questions

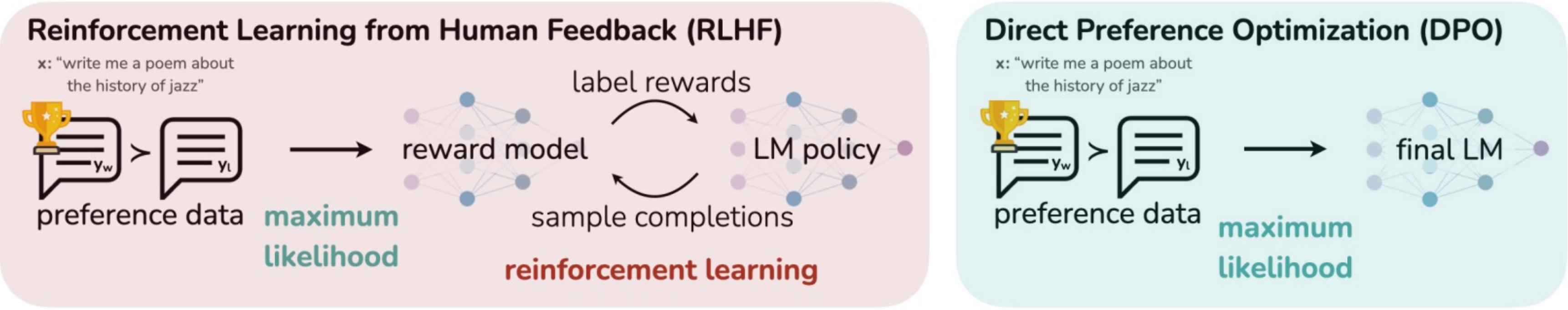
- Some existing systems:
  - https://github.com/volcengine/verl
  - https://github.com/inclusionAI/AReaL
  - https://github.com/OpenRLHF/OpenRLHF
  - https://github.com/huggingface/trl
  - …

Other Relevant Alignment Methods

Rejection Sampling Fine-tuning

- **Standard Supervised Fine-tuning (SFT)**: Standard maximization of the autoregressive LM probability for the (question, output) pairs;
- **Rejection Sampling Fine-tuning (RFT)**: First, samples multiple outputs from the supervised fine-tuned LLMs for each question, and then fine-tunes LLMs on the sampled outputs with the correct answer.
- **Online Rejection Sampling Fine-tuning**: The outputs of Online RFT are sampled from the real-time policy model, rather than from the previous SFT model.

DPO



RELAXED SYSTEM LAB

- **<u>Direct Preference Optimization (DPO)</u>**: align language models with human preferences without employing traditional reinforcement learning techniques.
  - Instead of learning from scalar reward signals, DPO learns directly from pairwise preference data, optimizing the model to prefer outputs that align with human judgments.
  - No explicit reward Model: No separate reward model in RL-based methods.
- Given a dataset of human preference comparisons $\mathrm{D} = \{(x, y_w, y_l)\}$: x is the input question prompt; y_w is the preferred answer by user; y_l is the less preferred answer by user.
- DPO constructs a loss that is analogous to a logistic regression: it treats the pair (x, y_w, y_l) as a positive vs negative example:

$$L_{DPO}(\theta) = -\mathbb{E}_{(x,y_l,y_w)\sim D}\left[\log\sigma\left(\beta\log\left(\frac{\pi_\theta(y_w|x)}{\pi_{\mathrm{ref}}(y_w|x)}\right) - \beta\log\left(\frac{\pi_\theta(y_l|x)}{\pi_{\mathrm{ref}}(y_l|x)}\right)\right)\right]$$

  - σ is the logistic function; β is some coefficient.

Reference

- https://cs229.stanford.edu/main_notes.pdf (Chapter 15,17)
- https://huggingface.co/learn/deep-rl-course/unit0/introduction
- https://openai.com/index/instruction-following/
- https://icml.cc/media/icml-2023/Slides/21554.pdf
- https://huggingface.co/blog/rlhf
- https://arxiv.org/abs/1707.06347
- https://arxiv.org/abs/2402.03300
- https://arxiv.org/abs/2409.19256
- https://arxiv.org/abs/2305.18290