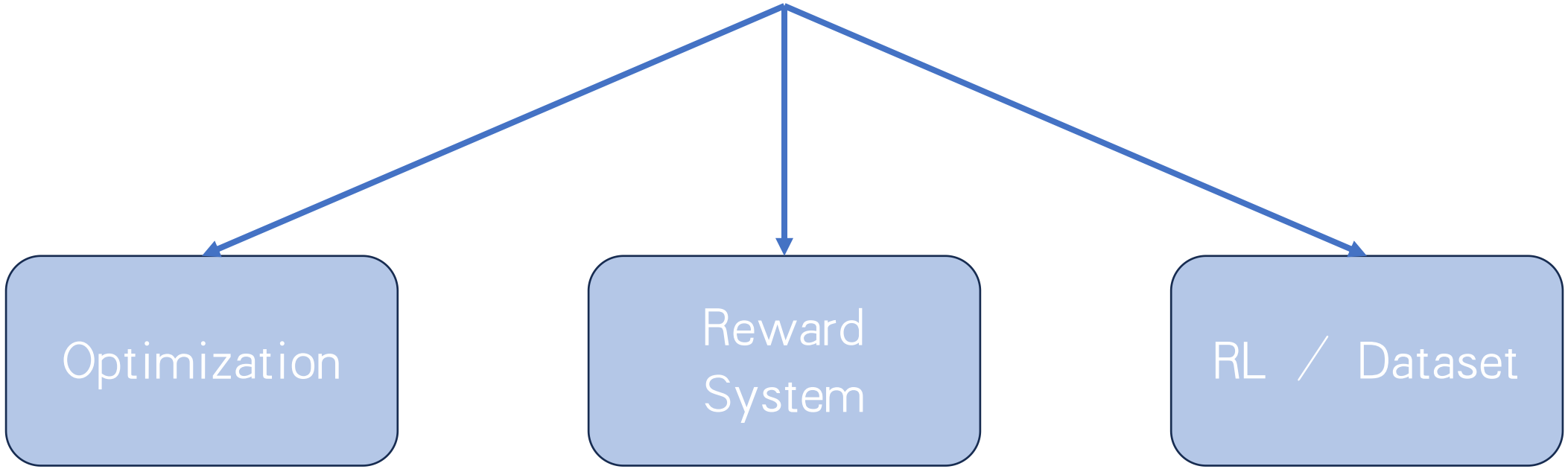
DeepSeek R1 (Zero)



Qingbin Li (感谢Xuerui Su)
2/25/2025

1.1. Contributions

Post-Training: Large-Scale Reinforcement Learning on the Base Model

- We directly apply RL to the base model without relying on supervised fine-tuning (SFT) as a preliminary step. This approach allows the model to explore chain-of-thought (CoT) for solving complex problems, resulting in the development of DeepSeek-R1-Zero. DeepSeek-R1-Zero demonstrates capabilities such as self-verification, reflection, and generating long CoTs, marking a significant milestone for the research community. Notably, it is the first open research to validate that reasoning capabilities of LLMs can be incentivized purely through RL, without the need for SFT. This breakthrough paves the way for future advancements in this area.
- We introduce our pipeline to develop DeepSeek-R1. The pipeline incorporates two RL stages aimed at discovering improved reasoning patterns and aligning with human preferences, as well as two SFT stages that serve as the seed for the model's reasoning and non-reasoning capabilities. We believe the pipeline will benefit the industry by creating better models.

Distillation: Smaller Models Can Be Powerful Too

- We demonstrate that the reasoning patterns of larger models can be distilled into smaller models, resulting in better performance compared to the reasoning patterns discovered through RL on small models. The open source DeepSeek-R1, as well as its API, will benefit the research community to distill better smaller models in the future.
- Using the reasoning data generated by DeepSeek-R1, we fine-tuned several dense models that are widely used in the research community. The evaluation results demonstrate that the distilled smaller dense models perform exceptionally well on benchmarks. DeepSeek-R1-Distill-Qwen-7B achieves 55.5% on AIME 2024, surpassing QwQ-32B-Preview. Additionally, DeepSeek-R1-Distill-Qwen-32B scores 72.6% on AIME 2024, 94.3% on MATH-500, and 57.2% on LiveCodeBench. These results significantly outperform previous open-source models and are comparable to o1-mini. We open-source distilled 1.5B, 7B, 8B, 14B, 32B, and 70B checkpoints based on Qwen2.5 and Llama3 series to the community.

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think> <answer> answer here </answer>. User: prompt. Assistant:

Table 1 | Template for DeepSeek-R1-Zero. prompt will be replaced with the specific reasoning question during training.

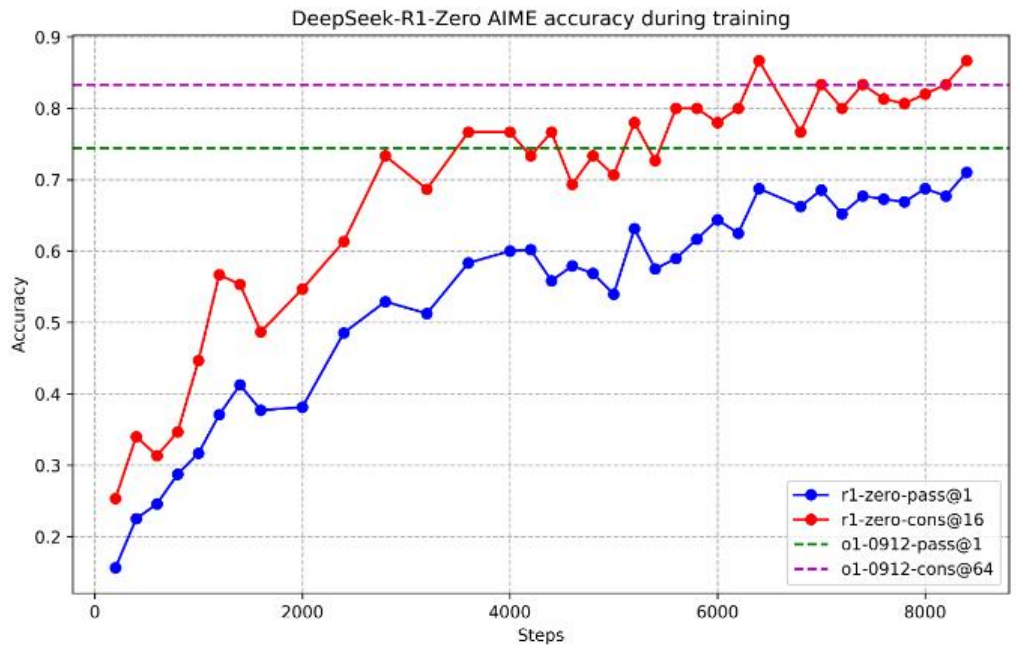


Figure 2 | AIME accuracy of DeepSeek-R1-Zero during training. For each question, we sample 16 responses and calculate the overall average accuracy to ensure a stable evaluation.

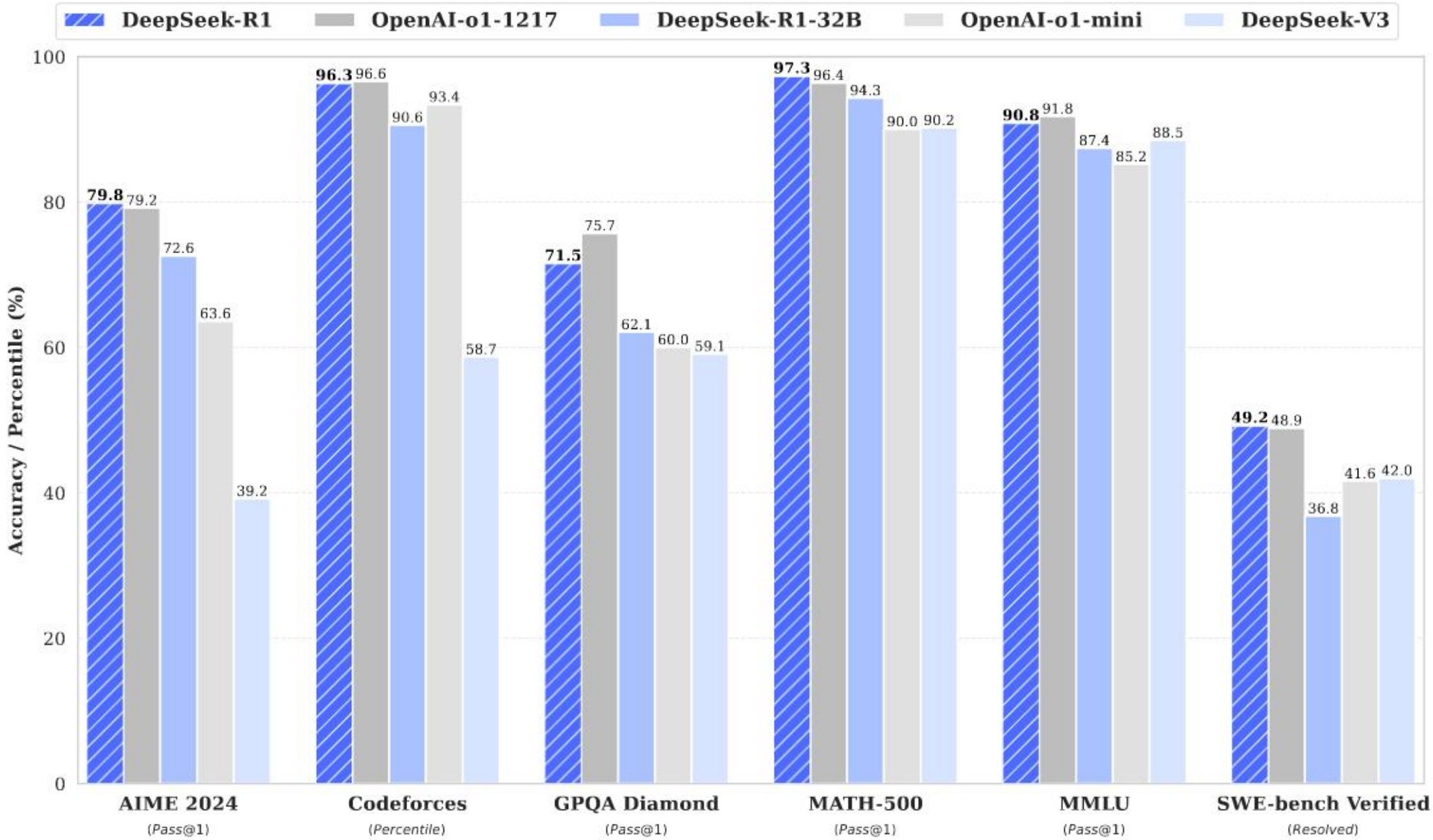
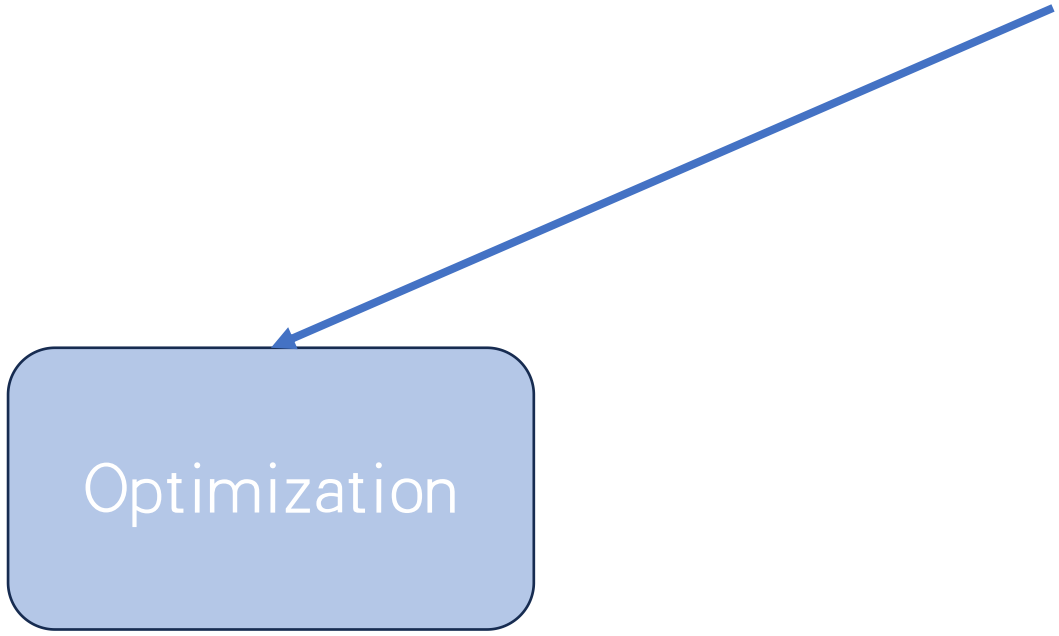


Figure 1 | Benchmark performance of DeepSeek-R1.

DeepSeek R1 Zero



Group Relative Policy Optimization In order to save the training costs of RL, we adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which foregoes the critic model that is typically the same size as the policy model, and estimates the baseline from group scores instead. Specifically, for each question q, GRPO samples a group of outputs $\{o_1, o_2, \cdots, o_G\}$ from the old policy $\pi_{\theta_{old}}$ and then optimizes the policy model π_θ by maximizing the following objective:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1-\varepsilon, 1+\varepsilon\right)A_i\right) - \beta\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right)\right), \tag{1}$$

$$\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right) = \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - 1, \tag{2}$$

where ε and β are hyper-parameters, and A_i is the advantage, computed using a group of rewards $\{r_1, r_2, \ldots, r_G\}$ corresponding to the outputs within each group:

$$A_i = \frac{r_i - mean(\{r_1, r_2, \cdots, r_G\})}{std(\{r_1, r_2, \cdots, r_G\})}. \tag{3}$$

$\frac{1}{G}\sum_{i=1}^{G}$ Ensemble

$\text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1-\varepsilon, 1+\varepsilon\right)$ PPO—Clipping

$\beta\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right)$ KL penalty(Not PPO)

Same Goal: For Stability

Group Relative Policy Optimization In order to save the training costs of RL, we adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which foregoes the critic model that is typically the same size as the policy model, and estimates the baseline from group scores instead. Specifically, for each question q, GRPO samples a group of outputs $\{o_1, o_2, \cdots, o_G\}$ from the old policy $\pi_{\theta_{old}}$ and then optimizes the policy model π_θ by maximizing the following objective:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1-\varepsilon, 1+\varepsilon\right)A_i\right) - \beta\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right)\right), \tag{1}$$

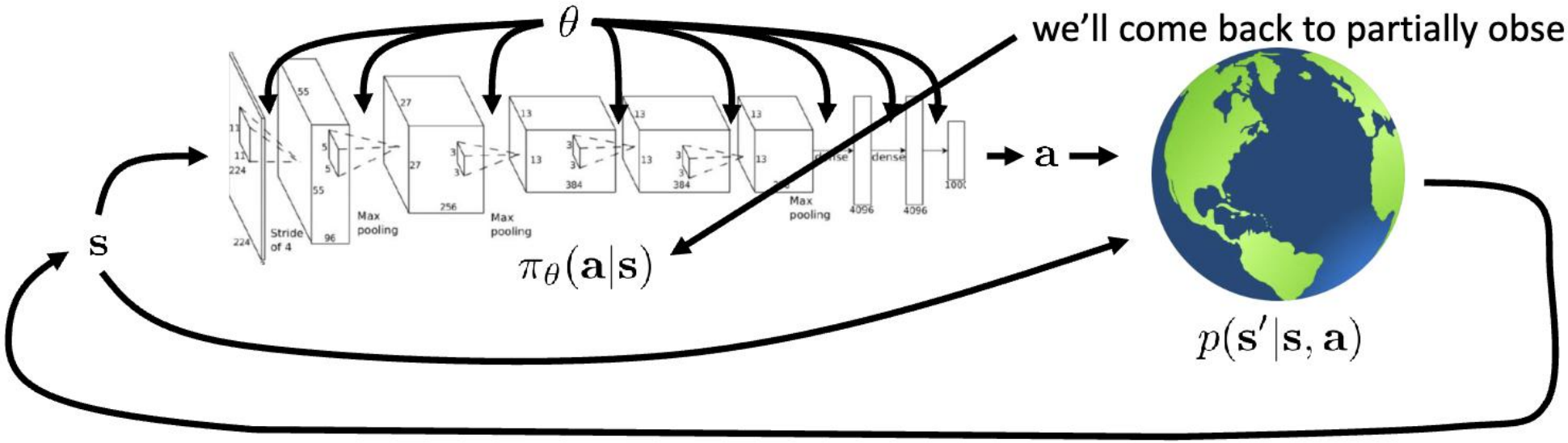
$$\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right) = \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - 1, \tag{2}$$

where ε and β are hyper-parameters, and A_i is the advantage, computed using a group of rewards $\{r_1, r_2, \ldots, r_G\}$ corresponding to the outputs within each group:

$$A_i = \frac{r_i - mean(\{r_1, r_2, \cdots, r_G\})}{std(\{r_1, r_2, \cdots, r_G\})}. \tag{3}$$

The only interesting quantity that's guaranteed to have zero expectation is $\frac{p(x)}{q(x)} - 1 = r - 1$. So for any λ, the expression $-\log r + \lambda(r-1)$ is an unbiased estimator of $\mathrm{KL}[q, p]$.

The goal of reinforcement learning



$$\underbrace{p_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)}_{p_\theta(\tau)} = p(\mathbf{s}_1) \prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

$$\theta^\star = \arg\max_\theta E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

Definition: Q-function

$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t, \mathbf{a}_t\right]$: total reward from taking $\mathbf{a}_t$ in $\mathbf{s}_t$

Definition: value function

$V^\pi(\mathbf{s}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t\right]$: total reward from $\mathbf{s}_t$

$V^\pi(\mathbf{s}_t) = E_{\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)}[Q^\pi(\mathbf{s}_t, \mathbf{a}_t)]$

$E_{\mathbf{s}_1 \sim p(\mathbf{s}_1)}[V^\pi(\mathbf{s}_1)]$ is the RL objective!

prediction of future reward. The future reward, also known as **return**, is a total sum of discounted rewards going forward. Let's compute the return G_t starting from time t:

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

The **state-value** of a state s is the expected return if we are in this state at time t, $S_t = s$:

$$V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

Similarly, we define the **action-value** ("Q-value"; Q as "Quality" I believe?) of a state-action pair as:

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$$

Additionally, since we follow the target policy π, we can make use of the probility distribution over possible actions and the Q-values to recover the state-value:

$$V_\pi(s) = \sum_{a \in \mathcal{A}} Q_\pi(s, a) \pi(a|s)$$

The difference between action-value and state-value is the action **advantage** function ("A-value"):

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$$

Policy gradient as policy iteration

$$J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\sum_t \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

$$\text{claim: } J(\theta') - J(\theta) = E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_t \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right]$$

$$\begin{aligned}
J(\theta') - J(\theta) &= J(\theta') - E_{\mathbf{s}_0 \sim p(\mathbf{s}_0)} \left[V^{\pi_\theta}(\mathbf{s}_0) \right] \\
&= J(\theta') - E_{\tau \sim p_{\theta'}(\tau)} \left[V^{\pi_\theta}(\mathbf{s}_0) \right] \\
&= J(\theta') - E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t V^{\pi_\theta}(\mathbf{s}_t) - \sum_{t=1}^{\infty} \gamma^t V^{\pi_\theta}(\mathbf{s}_t) \right] \\
&= J(\theta') + E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t (\gamma V^{\pi_\theta}(\mathbf{s}_{t+1}) - V^{\pi_\theta}(\mathbf{s}_t)) \right] \\
&= E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right] + E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t (\gamma V^{\pi_\theta}(\mathbf{s}_{t+1}) - V^{\pi_\theta}(\mathbf{s}_t)) \right] \\
&= E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t (r(\mathbf{s}_t, \mathbf{a}_t) + \gamma V^{\pi_\theta}(\mathbf{s}_{t+1}) - V^{\pi_\theta}(\mathbf{s}_t)) \right] \\
&= E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right]
\end{aligned}$$

Policy gradient as policy iteration

$$J(\theta') - J(\theta) = E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_t \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right]$$

expectation under $\pi_{\theta'}$

advantage under π_θ

importance sampling

$$\begin{aligned} E_{x \sim p(x)}[f(x)] &= \int p(x) f(x) dx \\ &= \int \frac{q(x)}{q(x)} p(x) f(x) dx \\ &= \int q(x) \frac{p(x)}{q(x)} f(x) dx \\ &= E_{x \sim q(x)} \left[\frac{p(x)}{q(x)} f(x) \right] \end{aligned}$$

$$E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_t \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] = \sum_t E_{\mathbf{s}_t \sim p_{\theta'}(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)} \left[\gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right]$$

$$= \sum_t E_{\mathbf{s}_t \sim p_{\theta'}(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right]$$

is it OK to use $p_\theta(\mathbf{s}_t)$ instead?

Ignoring distribution mismatch?

$$\sum_t E_{\mathbf{s}_t \sim p_{\theta'}(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right] \overset{?}{\approx} \underbrace{\sum_t E_{\mathbf{s}_t \sim p_\theta(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right]}_{\bar{A}(\theta')}$$

why do we want this to be true?

$$J(\theta') - J(\theta) \approx \bar{A}(\theta') \quad \Rightarrow \quad \theta' \leftarrow \arg\max_{\theta'} \bar{A}(\theta)$$

2. Use $\hat{A}^\pi(\mathbf{s}_t, \mathbf{a}_t)$ to get *improved* policy π'

is it true? and when?

Claim: $p_\theta(\mathbf{s}_t)$ is *close* to $p_{\theta'}(\mathbf{s}_t)$ when π_θ is *close* to $\pi_{\theta'}$

$$\theta' \leftarrow \arg\max_{\theta'} \sum_t E_{\mathbf{s}_t \sim p_\theta(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right]$$

$$\text{such that } D_{\mathrm{KL}}(\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) \| \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)) \leq \epsilon$$

for small enough ϵ, this is guaranteed to improve $J(\theta') - J(\theta)$

$$E_{\tau\sim p_{\theta'}(\tau)}\left[\sum_t \gamma^t A^{\pi_\theta}(\mathbf{s}_t,\mathbf{a}_t)\right] = \sum_t E_{\mathbf{s}_t\sim p_{\theta'}(\mathbf{s}_t)}\left[E_{\mathbf{a}_t\sim\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}\left[\gamma^t A^{\pi_\theta}(\mathbf{s}_t,\mathbf{a}_t)\right]\right]$$

$$= \sum_t E_{\mathbf{s}_t\sim p_{\theta'}(\mathbf{s}_t)}\left[E_{\mathbf{a}_t\sim\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}\left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}\gamma^t A^{\pi_\theta}(\mathbf{s}_t,\mathbf{a}_t)\right]\right]$$

is it OK to use $p_\theta(\mathbf{s}_t)$ instead?

$$\theta' \leftarrow \arg\max_{\theta'} \sum_t E_{\mathbf{s}_t\sim p_\theta(\mathbf{s}_t)}\left[E_{\mathbf{a}_t\sim\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}\left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}\gamma^t A^{\pi_\theta}(\mathbf{s}_t,\mathbf{a}_t)\right]\right]$$

such that $D_{\mathrm{KL}}(\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)\|\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)) \leq \epsilon$

for small enough ϵ, this is guaranteed to improve $J(\theta') - J(\theta)$

$$E_{\tau\sim p_{\theta'}(\tau)}\left[\sum_t \gamma^t A^{\pi_\theta}(\mathbf{s}_t,\mathbf{a}_t)\right] = \sum_t E_{\mathbf{s}_t\sim p_{\theta'}(\mathbf{s}_t)}\left[E_{\mathbf{a}_t\sim\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}\left[\gamma^t A^{\pi_\theta}(\mathbf{s}_t,\mathbf{a}_t)\right]\right]$$

$$= \sum_t E_{\mathbf{s}_t\sim p_{\theta'}(\mathbf{s}_t)}\left[E_{\mathbf{a}_t\sim\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}\left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}\gamma^t A^{\pi_\theta}(\mathbf{s}_t,\mathbf{a}_t)\right]\right]$$

is it OK to use $p_\theta(\mathbf{s}_t)$ instead?

$$\theta' \leftarrow \arg\max_{\theta'} \sum_t E_{\mathbf{s}_t\sim p_\theta(\mathbf{s}_t)}\left[E_{\mathbf{a}_t\sim\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}\left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}\gamma^t A^{\pi_\theta}(\mathbf{s}_t,\mathbf{a}_t)\right]\right]$$

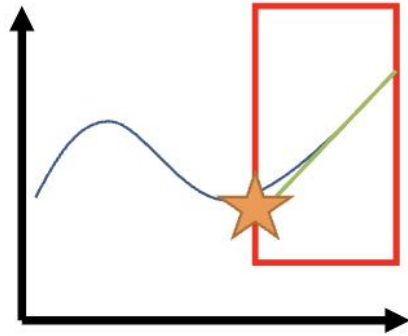
such that $D_{\mathrm{KL}}(\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)\|\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)) \leq \epsilon$

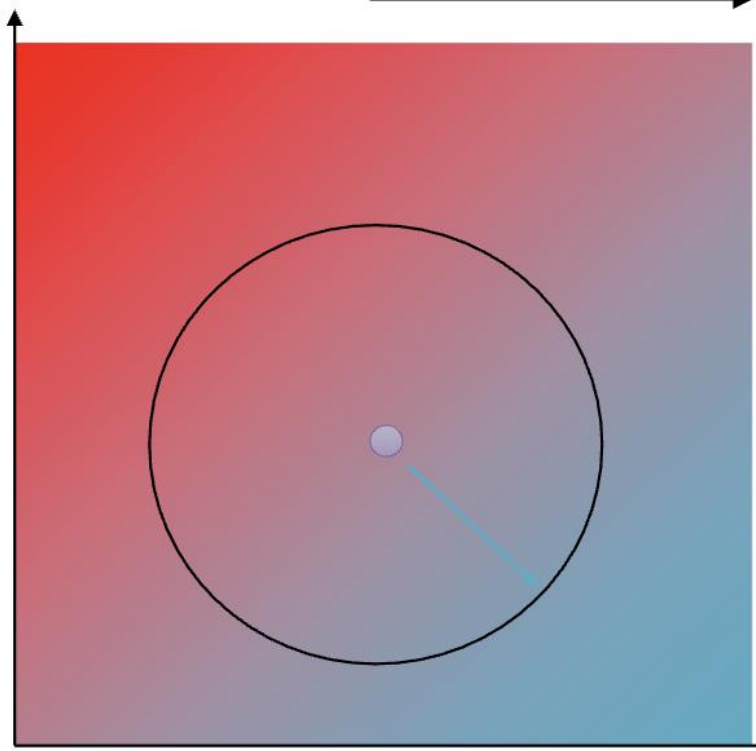
for small enough ϵ, this is guaranteed to improve $J(\theta') - J(\theta)$

Can we just use the gradient then?

$$\theta' \leftarrow \arg\max_{\theta'} \nabla_\theta J(\theta)^T (\theta' - \theta)$$

$$\text{such that } D_{\mathrm{KL}}(\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) \| \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)) \leq \epsilon$$





$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \qquad\qquad \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$$

some parameters change probabilities a lot more than others!

Claim: gradient ascent does this:

$$\theta' \leftarrow \arg\max_{\theta'} \nabla_\theta J(\theta)^T (\theta' - \theta)$$

$$\text{such that } \|\theta - \theta'\|^2 \leq \epsilon$$

$$\theta' = \theta + \sqrt{\frac{\epsilon}{\|\nabla_\theta J(\theta)\|^2}} \nabla_\theta J(\theta)$$

4 Adaptive KL Penalty Coefficient

Another approach, which can be used as an alternative to the clipped surrogate objective, or in addition to it, is to use a penalty on KL divergence, and to adapt the penalty coefficient so that we achieve some target value of the KL divergence d_{targ} each policy update. In our experiments, we found that the KL penalty performed worse than the clipped surrogate objective, however, we've included it here because it's an important baseline.

In the simplest instantiation of this algorithm, we perform the following steps in each policy update:

- Using several epochs of minibatch SGD, optimize the KL-penalized objective

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t - \beta \, \text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)] \right] \tag{8}$$

- Compute $d = \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]]$
  - If $d < d_{\text{targ}}/1.5$, $\beta \leftarrow \beta/2$
  - If $d > d_{\text{targ}} \times 1.5$, $\beta \leftarrow \beta \times 2$

3 Clipped Surrogate Objective

Let $r_t(\theta)$ denote the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$, so $r(\theta_{\text{old}}) = 1$. TRPO maximizes a "surrogate" objective

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]. \tag{6}$$

The superscript CPI refers to conservative policy iteration [KL02], where this objective was proposed. Without a constraint, maximization of L^{CPI} would lead to an excessively large policy update; hence, we now consider how to modify the objective, to penalize changes to the policy that move $r_t(\theta)$ away from 1.

The main objective we propose is the following:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right] \tag{7}$$

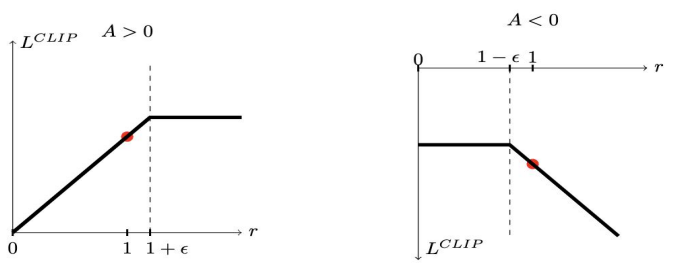


Figure 1: Plots showing one term (i.e., a single timestep) of the surrogate function L^{CLIP} as a function of the probability ratio r, for positive advantages (left) and negative advantages (right). The red circle on each plot shows the starting point for the optimization, i.e., $r = 1$. Note that L^{CLIP} sums many of these terms.

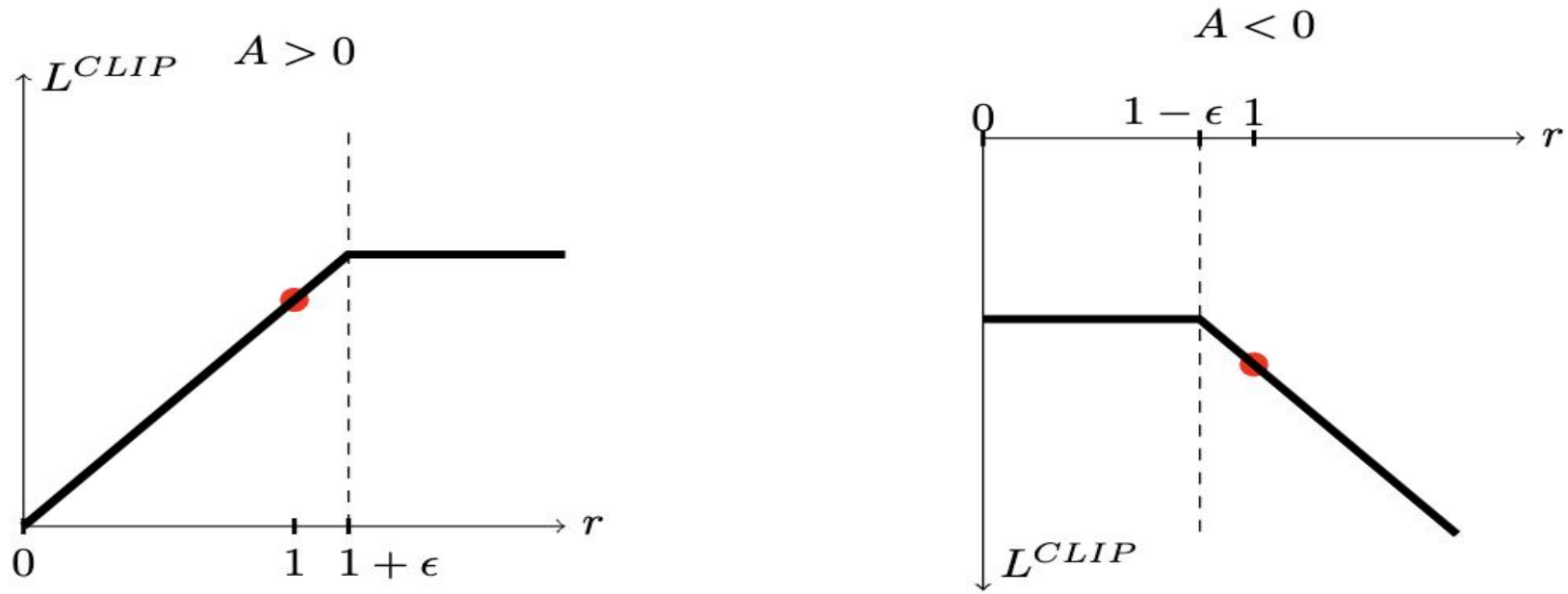


Figure 1: Plots showing one term (i.e., a single timestep) of the surrogate function L^{CLIP} as a function of the probability ratio r, for positive advantages (left) and negative advantages (right). The red circle on each plot shows the starting point for the optimization, i.e., $r = 1$. Note that L^{CLIP} sums many of these terms.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right] \tag{7}$$

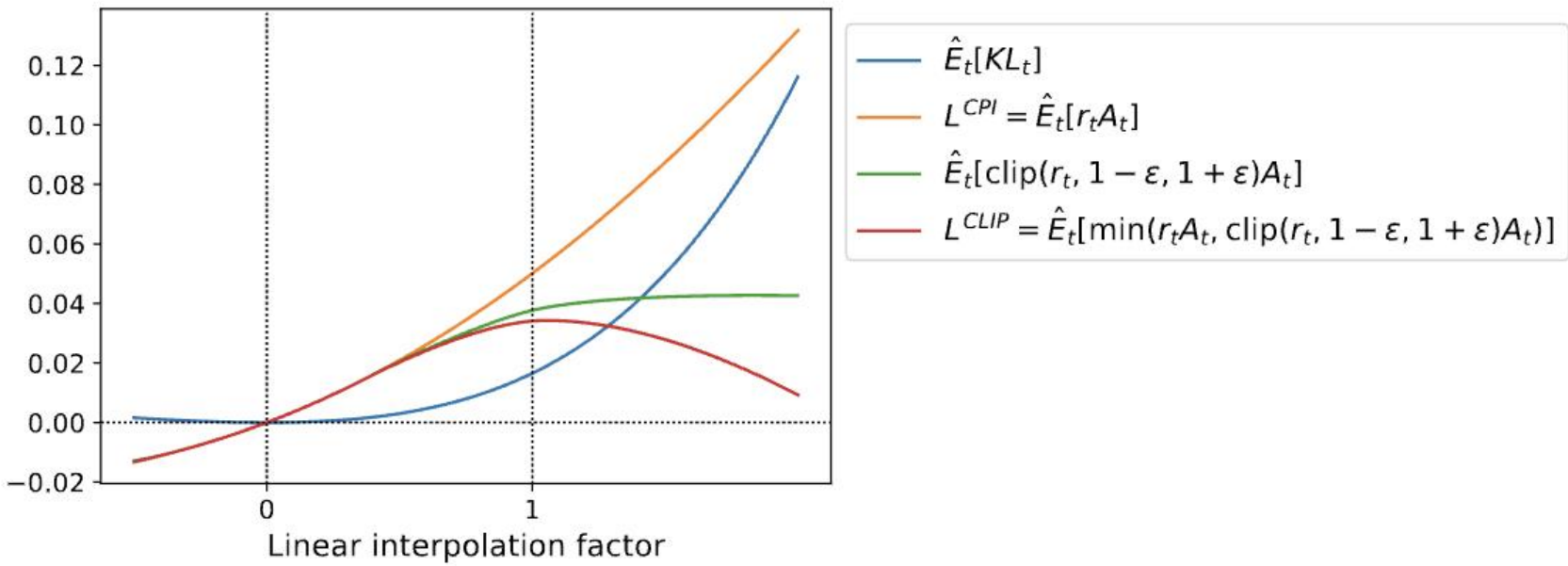


Figure 2: Surrogate objectives, as we interpolate between the initial policy parameter θ_{old}, and the updated policy parameter, which we compute after one iteration of PPO. The updated policy has a KL divergence of about 0.02 from the initial policy, and this is the point at which L^{CLIP} is maximal. This plot corresponds to the first policy update on the Hopper-v1 problem, using hyperparameters provided in Section 6.1.

DPO

Policy gradient as policy iteration

$$J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\sum_t \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

$$J(\theta') - J(\theta) = J(\theta') - E_{\mathbf{s}_0 \sim p(\mathbf{s}_0)} \left[V^{\pi_\theta}(\mathbf{s}_0) \right]$$
$$= J(\theta') - E_{\tau \sim p_{\theta'}(\tau)} \left[V^{\pi_\theta}(\mathbf{s}_0) \right]$$

$$\text{claim: } J(\theta') - J(\theta) = E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_t \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right]$$

RL Fine-Tuning Phase: During the RL phase, the learned reward function is used to provide feedback to the language model. Following prior works [17, 18], the optimization is formulated as

$$\max_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(y|x)} \left[r_\phi(x, y) \right] - \beta \mathbb{D}_{\text{KL}} \left[\pi_\theta(y \mid x) \,||\, \pi_{\text{ref}}(y \mid x) \right], \tag{3}$$

Deriving the DPO objective. We start with the same RL objective as prior work, Eq. 3, under a general reward function r. Following prior work [31, 30, 19, 15], it is straightforward to show that the optimal solution to the KL-constrained reward maximization objective in Eq. 3 takes the form:

$$\pi_r(y \mid x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y) \right), \tag{4}$$

where $Z(x) = \sum_y \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y) \right)$ is the partition function. See Appendix A.1 for a complete derivation. Even if we use the MLE estimate r_ϕ of the ground-truth reward function r^*, it is still expensive to estimate the partition function $Z(x)$ [19, 15], which makes this representation hard to utilize in practice. However, we can rearrange Eq. 4 to express the reward function in terms of its corresponding optimal policy π_r, the reference policy π_{ref}, and the unknown partition function $Z(\cdot)$. Specifically, we first take the logarithm of both sides of Eq. 4 and then with some algebra we obtain:

$$r(x, y) = \beta \log \frac{\pi_r(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x). \tag{5}$$

DPO

We can apply this reparameterization to the ground-truth reward r^* and corresponding optimal model π^*. Fortunately, the Bradley-Terry model depends only on the difference of rewards between two completions, i.e., $p^*(y_1 \succ y_2 \mid x) = \sigma(r^*(x, y_1) - r^*(x, y_2))$. Substituting the reparameterization in Eq. 5 for $r^*(x, y)$ into the preference model Eq. 1, the partition function cancels, and we can express the human preference probability in terms of only the optimal policy π^* and reference policy π_{ref}. Thus, the optimal RLHF policy π^* under the Bradley-Terry model satisfies the preference model:

$$p^*(y_1 \succ y_2 \mid x) = \frac{1}{1 + \exp\left(\beta \log \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)} - \beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)} \tag{6}$$

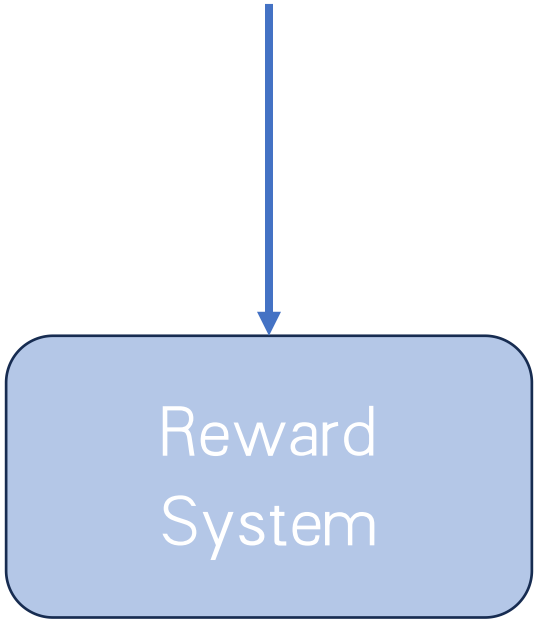
The derivation is in Appendix A.2. While Eq. 6 uses the Bradley-Terry model, we can similarly derive expressions under the more general Plackett-Luce models [32, 23], shown in Appendix A.3.

Now that we have the probability of human preference data in terms of the optimal policy rather than the reward model, we can formulate a maximum likelihood objective for a parametrized policy π_θ. Analogous to the reward modeling approach (i.e. Eq. 2), our policy objective becomes:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right]. \tag{7}$$

This way, we fit an implicit reward using an alternative parameterization, whose optimal policy is simply π_θ. Moreover, since our procedure is equivalent to fitting a reparametrized Bradley-Terry

DeepSeek R1 Zero



Aha Moment

Question: If $a > 1$, then the sum of the real solutions of $\sqrt{a - \sqrt{a+x}} = x$ is equal to

Response: <think>
To solve the equation $\sqrt{a - \sqrt{a+x}} = x$, let's start by squaring both $\cdots$
$\left(\sqrt{a - \sqrt{a+x}}\right)^2 = x^2 \implies a - \sqrt{a+x} = x^2.$
Rearrange to isolate the inner square root term:
$(a - x^2)^2 = a + x \implies a^2 - 2ax^2 + (x^2)^2 = a + x \implies x^4 - 2ax^2 - x + (a^2 - a) = 0$
...
Wait, wait. Wait. That's an aha moment I can flag here.
Let's reevaluate this step-by-step to identify if the correct sum can be $\cdots$
We started with the equation:
$\sqrt{a - \sqrt{a+x}} = x$
First, let's square both sides:
$a - \sqrt{a+x} = x^2 \implies \sqrt{a+x} = a - x^2$
Next, I could square both sides again, treating the equation: $\cdots$
...

Table 3 | An interesting "aha moment" of an intermediate version of DeepSeek-R1-Zero. The model learns to rethink using an anthropomorphic tone. This is also an aha moment for us, allowing us to witness the power and beauty of reinforcement learning.

Response Length：

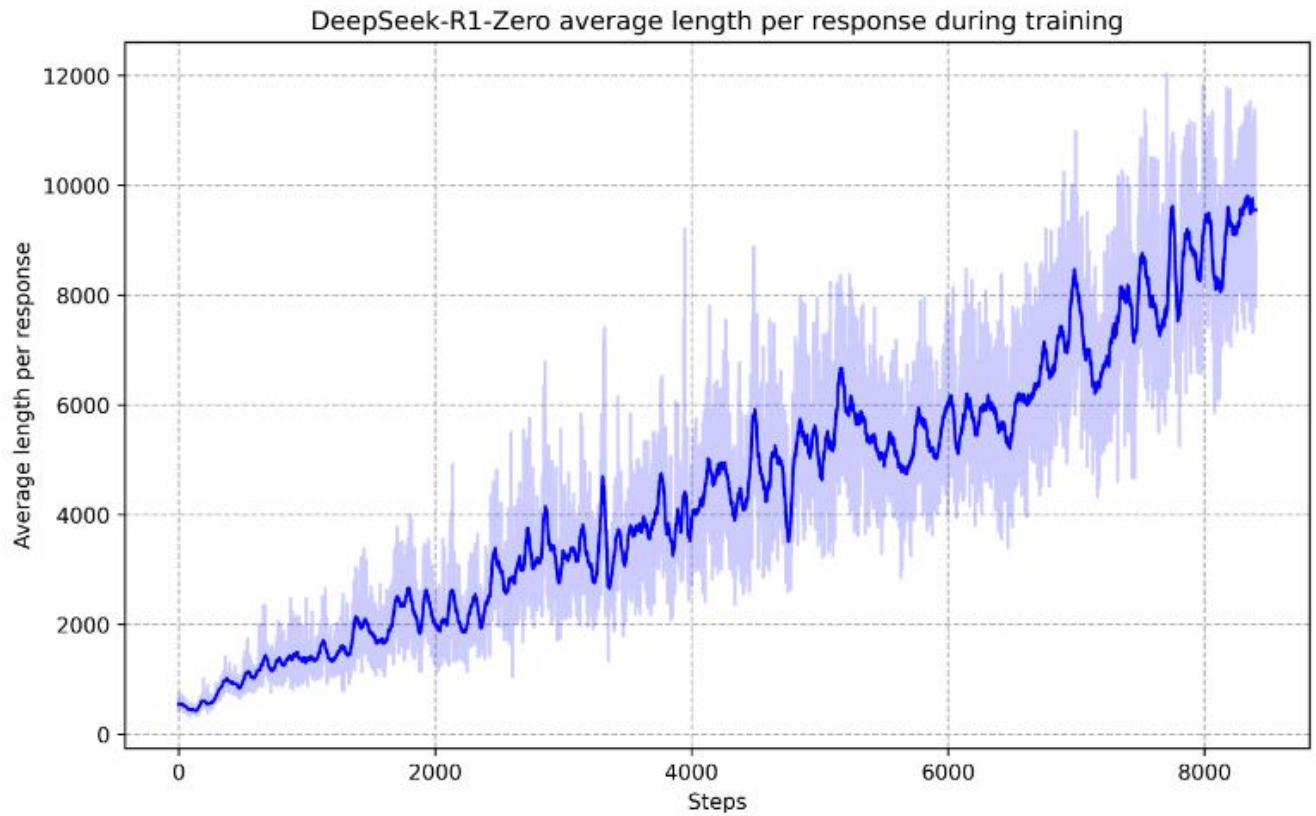


Figure 3 | The average response length of DeepSeek-R1-Zero on the training set during the RL process. DeepSeek-R1-Zero naturally learns to solve reasoning tasks with more thinking time.

2.2.2. *Reward Modeling*

The reward is the source of the training signal, which decides the optimization direction of RL. To train DeepSeek-R1-Zero, we adopt a rule-based reward system that mainly consists of two types of rewards:

- **Accuracy rewards**: The accuracy reward model evaluates whether the response is correct. For example, in the case of math problems with deterministic results, the model is required to provide the final answer in a specified format (e.g., within a box), enabling reliable rule-based verification of correctness. Similarly, for LeetCode problems, a compiler can be used to generate feedback based on predefined test cases.
- **Format rewards**: In addition to the accuracy reward model, we employ a format reward model that enforces the model to put its thinking process between '<think>' and '</think>' tags.

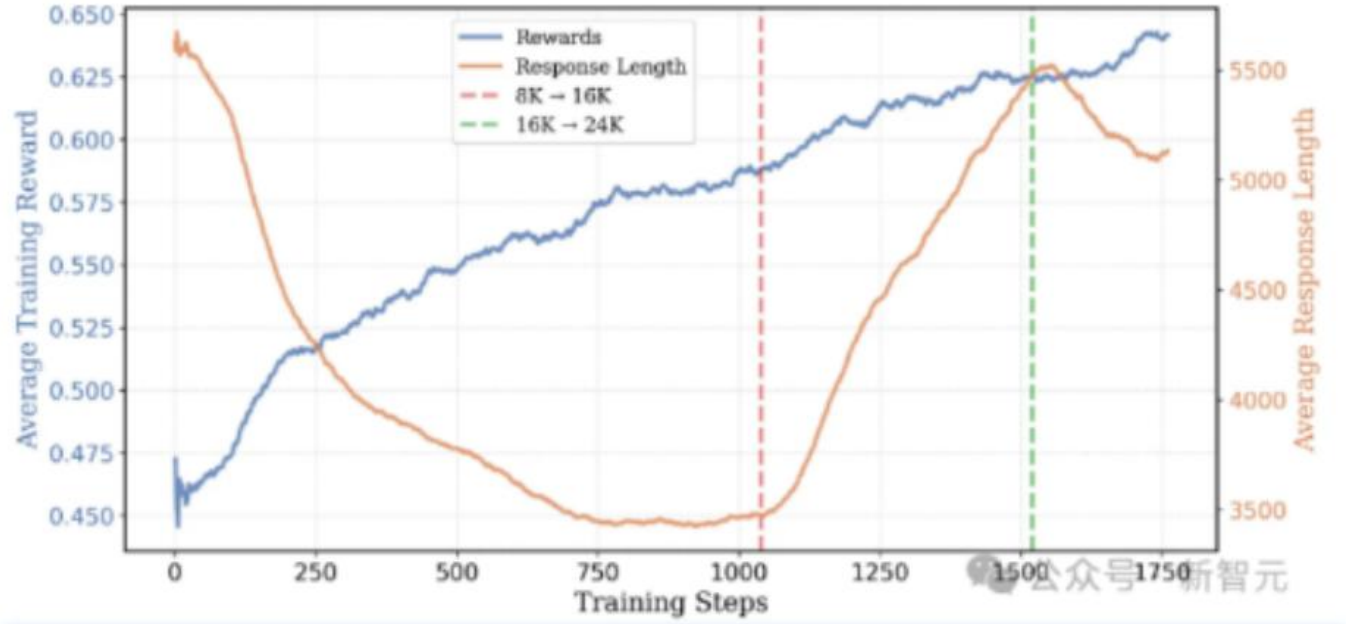
Current types of reward mechanisms：
1.Model—centric：ORM，PRM，Implicit—reward(ORM2PRM)
2.Data—centric：**outcome—reward**，Self—evaluation，**format—reward**，KL reward (current model and base model)，Response Length reward，**Language consistency rewards (DeepSeek R1)**

Rule based RL

1. Rule Based Rewards for Language Model Safety https://cdn.openai.com/rule-based-rewards-for-language-model-safety.pdf
2. **DeepSeek-V2** https://arxiv.org/abs/2405.04434
3. DeepSeek V3
4. DeepSeek R1
5. …

$$\text{len\_reward(i)} = \begin{cases} \lambda & \text{If } r(x, y_i, y^*) = 1 \\ \min(0, \lambda) & \text{If } r(x, y_i, y^*) = 0 \end{cases}, \quad \text{where } \lambda = 0.5 - \frac{\text{len}(i) - \text{min\_len}}{\text{max\_len} - \text{min\_len}}.$$

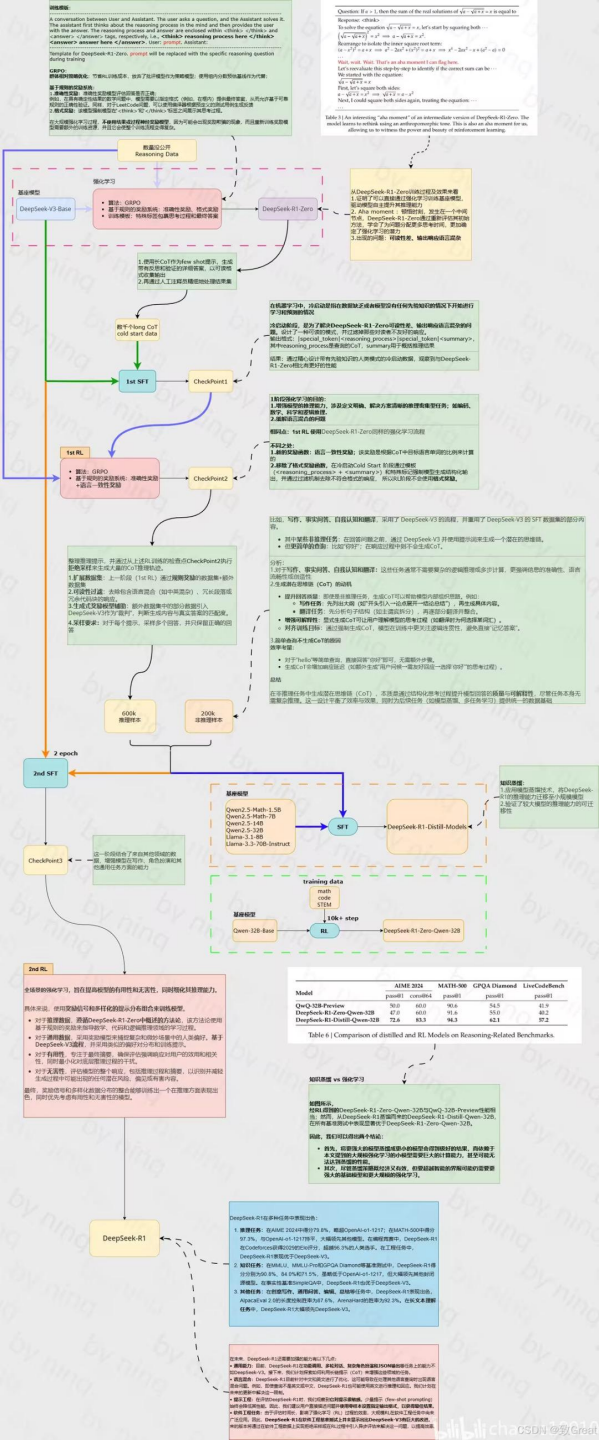
In essence, we promote shorter responses and penalize longer responses among correct ones, while explicitly penalizing long responses with incorrect answers. This length-based reward is then added to the original reward with a weighting parameter.



Current types of reward mechanisms：

1.Model—centric：ORM，PRM，Implicit—reward(ORM2PRM)

2.Data—centric：**outcome—reward**，Self—evaluation，**format—reward**，KL reward (current model and base model)，Response Length reward，**Language consistency rewards (DeepSeek R1)**



DeepSeek R1

RL / Dataset

DeepSeek R1

DeepSeek R1 training steps：Cold Start (SFT) -> RL stage 1 with Reasoning-oriented RL -> Rejection Sampling and SFT

->Reinforcement Learning for all Scenarios

我们构建并收集少量的长 CoT 数据，以作为初始强化学习参与者对模型进行微调：使用长 CoT 的少样本提示作为示例，直接提示模型通过反思和验证生成详细答案，以可读格式收集 DeepSeek-R1Zero 输出，并通过人工注释者的后处理完善结果

我们将推理任务的准确率和语言一致性的奖励直接相加，形成最终的奖励。然后我们对微调后的模型进行 RL 训练，直到它在推理任务上实现收敛。

•推理数据：我们通过从上述强化学习训练的checkpoint进行拒绝抽样来整理推理提示并生成推理轨迹。在上一阶段，我们**仅包含**可以使用基于规则的奖励进行评估的数据。但是，在此阶段，我们通过**合并其他数据来扩展数据集**，其中一些数据使用**reward model**，即将基本事实和模型预测输入 DeepSeek-V3进行判断。此外，由于模型输出有时混乱且难以阅读，我们过滤掉了混合语言、长段落和代码块的思路链。对于每个提示，我们会抽取多个响应并仅保留正确的响应。总共，我们收集了大约 600k 个与推理相关的训练样本。

•非推理数据：对于非推理数据（例如写作、事实问答、自我认知和翻译），我们采用 DeepSeek-V3 pipeline并重用 DeepSeek-V3 的 SFT 数据集的部分内容。对于某些非推理任务，我们会调用 DeepSeek-V3 生成潜在思路链，然后再

Discussion

DeepSeek R1 Zero：

DeepSeek—R1—Zero struggles with challenges like **poor readability**, and **language mixing**.

Distillation：

1. distilling more powerful models into smaller ones yields excellent results.
2. while distillation strategies are both economical and effective, advancing beyond the boundaries of intelligence may still require more powerful base models and larger scale reinforcement learning.

MCTS：

while MCTS can improve performance during inference when paired with a pre—trained value model, iteratively boosting model performance through self—search remains a significant challenge.

PRM：

1. challenging to **explicitly define a fine—grain step in general reasoning**.
2. **determining whether the current intermediate step is correct** is a challenging task (Automated annotation using models may not yield satisfactory results, while manual annotation is not conducive to scaling up).
3. **reward hacking** complicates the whole training pipeline.

RL：

1. long evaluation times.
2. Reward Rule Design.