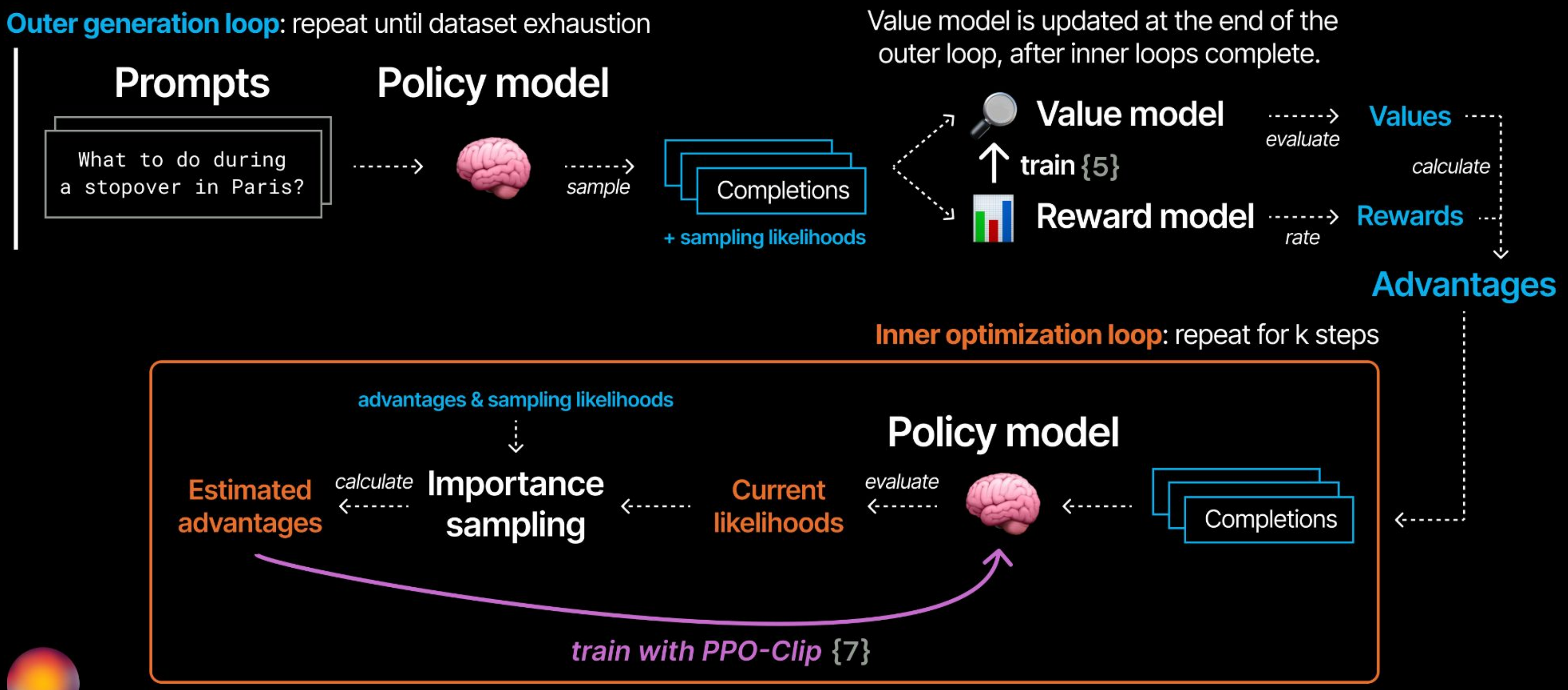
RFT fine tuning

Agenda

- Overview of Proximal Policy Optimization (PPO)
- Introduction to Group Relative Policy Optimization (GPRO)
- Understanding Reinforcement Fine-Tuning (RFT)
- Comparative Analysis of OpenAI’s Reinforcement Fine-Tuning (RFT)
- Look at Group Relative Reinforcement Fine-Tuning (GRFT)

Overview of Proximal Policy Optimization (PPO)

Proximal policy optimization (w/ clipping)



PPO

Probabilities of the next token **with the updated LLM**

$$L^{POLICY} = \min\left(\frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t \mid s_t\right)} \cdot \hat{A}_t, \text{clip}\left(\frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t \mid s_t\right)}, 1-\epsilon, 1+\epsilon\right) \cdot \hat{A}_t\right)$$

Probabilities of the next token **with the initial LLM**

Advantage term

Hyperparameters

$$L^{PPO} = \underbrace{L^{POLICY}}_{\text{Policy loss}} + \underbrace{c_1 L^{VF}}_{\text{Value loss}} + \underbrace{c_2 L^{ENT}}_{\text{Entropy loss}}$$

Prompt

```
This car is
```

Completion

```
This car is
a . . .
```

Value function

$$L^{VF} = \frac{1}{2}\left\| \underbrace{\boxed{V_\theta(s)}}_{\substack{\textbf{Estimated} \\ \text{future total reward}}} - \left(\sum_{t=0}^{T} \gamma^t r_t \mid s_0 = s\right)\right\|_2^2$$

0.34

Prompt

```
This car is
```

Completion

```
This car is
good ...
```

Value function

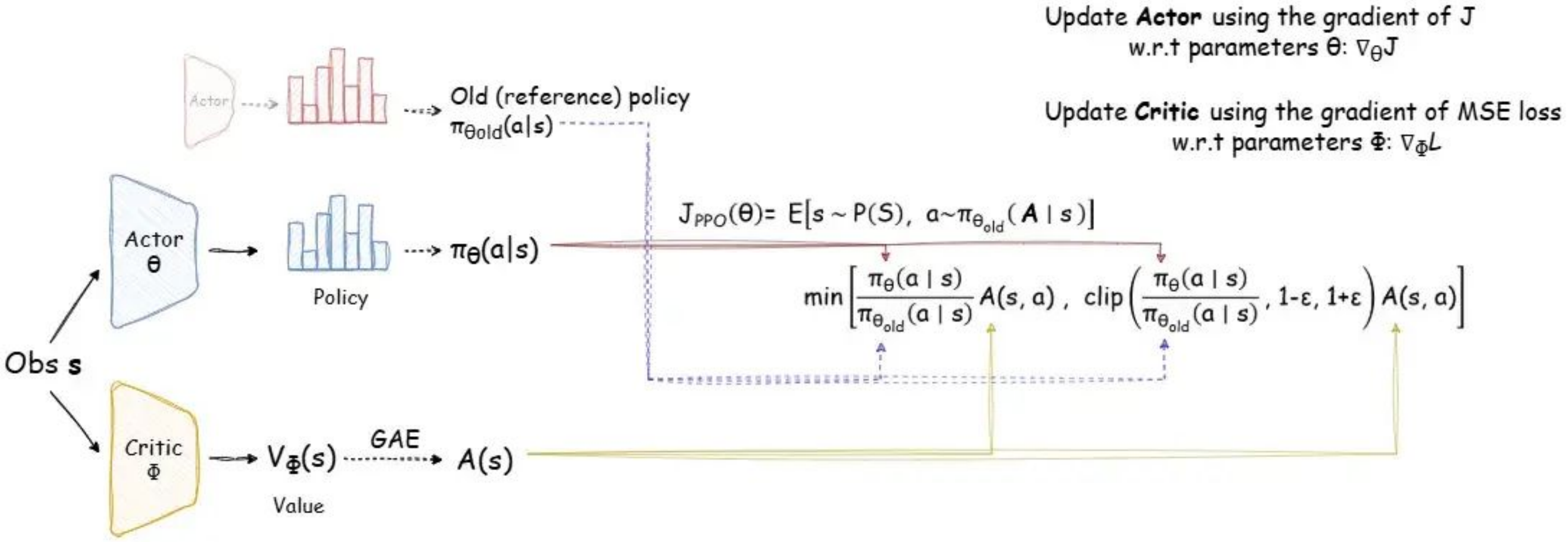
$$L^{VF} = \frac{1}{2}\left\| \underbrace{\boxed{V_\theta(s)}}_{\substack{\textbf{Estimated} \\ \text{future total reward}}} - \left(\sum_{t=0}^{T} \gamma^t r_t \mid s_0 = s\right)\right\|_2^2$$

1.23

Value loss

$$\boxed{L^{VF}} = \frac{1}{2}\left\| \underbrace{V_\theta(s)}_{\substack{\textbf{Estimated} \\ \text{future total reward}}} - \underbrace{\left(\sum_{t=0}^{T} \gamma^t r_t \mid s_0 = s\right)}_{\substack{\textbf{Known} \\ \text{future total reward}}}\right\|_2^2$$

1.23 1.87


Update Actor using the gradient of J w.r.t parameters θ: $\nabla_\theta J$
Update Critic using the gradient of MSE loss w.r.t parameters Φ: $\nabla_\Phi L$
Actor
Old (reference) policy
$\pi_{\theta old}(a|s)$
Actor
θ
Policy
$\pi_\theta(a|s)$
$J_{PPO}(\theta) = E[s \sim P(S),\ a \sim \pi_{\theta_{old}}(\mathbf{A} \mid s)]$
$\min\left[\frac{\pi_\theta(a \mid s)}{\pi_{\theta_{old}}(a \mid s)} A(s,a),\ \mathrm{clip}\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_{old}}(a \mid s)}, 1-\varepsilon, 1+\varepsilon\right) A(s,a)\right]$
Obs s
Critic
Φ
$V_\Phi(s)$
GAE
A(s)
Value

Introduction to Group Relative Policy Optimization (GPRO)

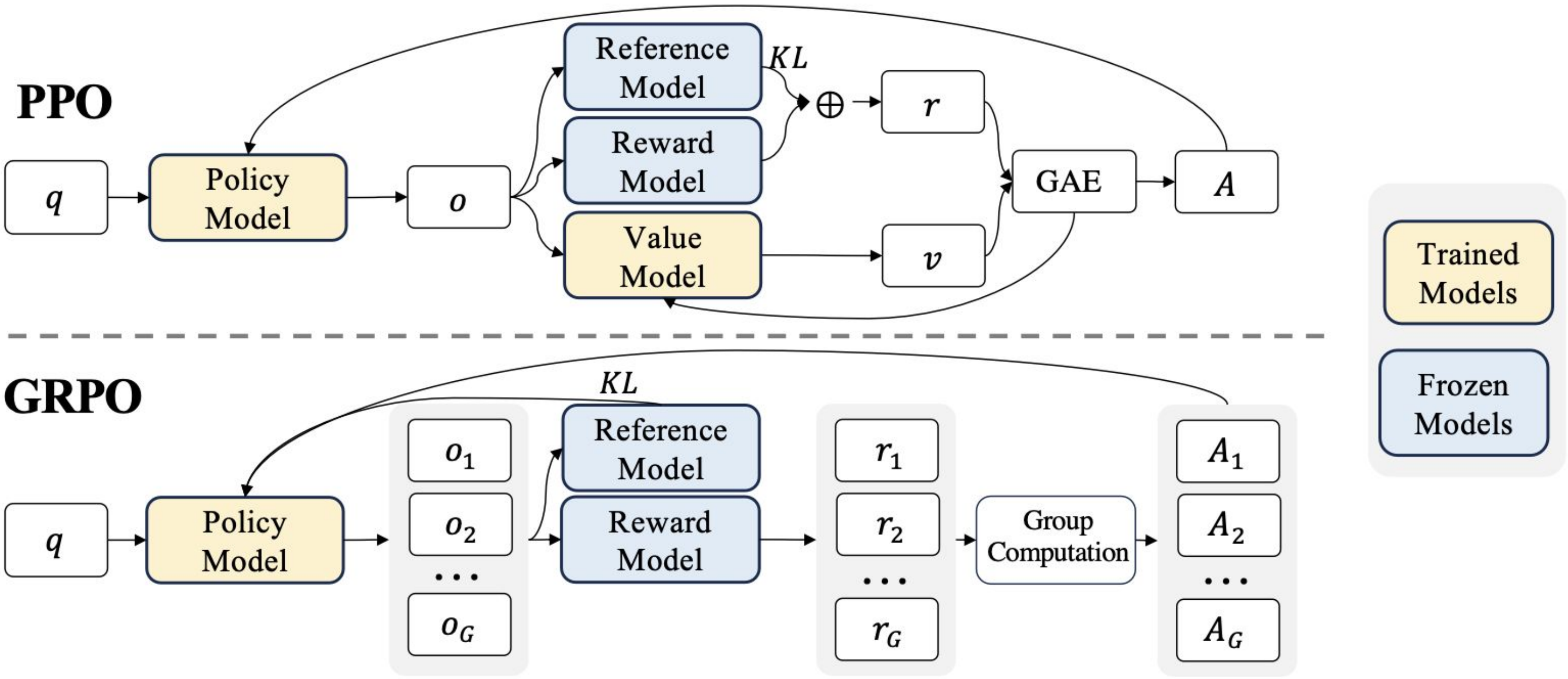


Figure 4 | Demonstration of PPO and our GRPO. GRPO foregoes the value model, instead estimating the baseline from group scores, significantly reducing training resources.

```python
def generate(model, input_ids, max_new_tokens=20, temperature=1.0, top_k=50, eos_token_id=None, pad_token_id=None,):
    model.eval()
    generated = input_ids.clone()

    for _ in range(max_new_tokens):
        # Get logits from model
        outputs = model.forward(input_ids=generated)
        logits = outputs.logits  # shape: (batch_size, seq_len, vocab_size)

        # Take the logits for the last token
        next_token_logits = logits[:, -1, :] / temperature  # shape: (batch_size, vocab_size)

        # Top-k filtering
        if top_k is not None:
            top_k_values, top_k_indices = torch.topk(next_token_logits, top_k)
            probs = torch.zeros_like(next_token_logits).scatter(1, top_k_indices, F.softmax(top_k_values, dim=-1))
        else:
            probs = F.softmax(next_token_logits, dim=-1)

        # Sample from the distribution WITH RANDOMNESS
        next_token = torch.multinomial(probs, num_samples=1)  # shape: (batch_size, 1)

        # Append to sequence
        generated = torch.cat((generated, next_token), dim=1)

        # Stop if EOS is generated
        if eos_token_id is not None:
            if (next_token == eos_token_id).all():
                break

    return generated
```

GRPO was designed to address the limitations of applying traditional PPO to language model fine-tuning. Its primary innovation is the use of **relative advantage estimation** derived from a group of on-policy samples rather than training a separate value network.

Traditional advantage estimation in RL is defined as:

$$A_t = R_t + \gamma V(s_{t+1}) - V(s_t)$$

or, in its Generalized Advantage Estimation (GAE) form, it seeks to predict the absolute “goodness” of actions relative to a learned value function. However, GRPO introduces a **relative advantage** concept. Rather than relying on an external value network, GRPO compares the rewards of multiple responses generated for the same prompt.

Consider a given query (or state) q at policy parameters θ_{old}. We sample a group of G responses:

$$o_1, o_2, \ldots, o_G$$

Each response o_i is assigned a scalar reward R_i (which may come from a learned reward model, rule-based metric, or human feedback). We then compute the baseline reward for the prompt as:

$$\bar{R} = \frac{1}{G} \sum_{i=1}^{G} R_i$$

Failure Points of GRPO

Failure Mode	Scenario	Reason
Response Length Bias	Logic/Reasoning problems with long CoT	Normalization under-penalizes longer incorrect responses, as noted in Evolution of Policy Optimization.
Question-Level Difficulty Bias	Mixed-difficulty datasets, e.g., math benchmarks	Normalization by std(r) skews focus to easy/hard questions, per Evolution of Policy Optimization.
Inadequate Data Coverage	Domains underrepresented in data, e.g., geometry in math	Lack of examples limits group-based advantage, from DeepSeekMath Paper.
Limited Generalization	Out-of-distribution prompts, e.g., novel problem types	Focus on in-distribution tasks, inferred from DeepSeekMath Paper.
Insufficient Diversity in Outputs	Tasks needing varied responses, e.g., creative generation	Low diversity reduces advantage signal, inferred from Predibase GRPO.

DAPO: Addressing Length Bias and KL Constraints

The DAPO paper highlights the limitations of the GRPO algorithm's sample-level loss in long-CoT scenarios, where longer responses are under-penalized, leading to poorer quality outputs. The proposed solution is a token-level normalization, which better handles longer sequences by assigning more balanced rewards to individual tokens, regardless of response length:

$$\mathcal{L}_{\text{DAPO}}(\theta) = -\frac{1}{\sum_{i=1}^{G} |o_i|} \sum_{i=1}^{G} \sum_{t=1}^{|o_i|} l_{i,t},$$

Additionally, the DAPO paper proposed the following innovations:

1. **Raise the Ceiling: Clip-Higher**
   DAPO introduces asymmetric clipping ranges (ϵ_low) and (ϵ_high) to address entropy collapse. With traditional symmetric clipping ($\epsilon = 0.2$), high-probability tokens can easily be reinforced, but low-probability "exploration tokens" struggle to increase significantly. By using a higher upper bound, DAPO enables better exploration while maintaining training stability.

2. **The More the Merrier: Dynamic Sampling**
   As training progresses, more prompts achieve perfect accuracy, leading to zero advantage and thus no gradient signal. DAPO addresses this by intelligently filtering the training batch, over-sampling to ensure all prompts have accuracies between 0 and 1. This maintains consistent learning signals throughout training, improving sample efficiency without sacrificing performance.

. . .

Comparative Analysis of OpenAI’s Reinforcement Fine-Tuning (RFT)

Score Model Grader

A ScoreModelGrader object that uses a model to assign a score to the input.

`input` array

The input text. This may include template strings.

Show properties

`model` string

The model to use for the evaluation.

`name` string

The name of the grader.

`range` array

The range of the score. Defaults to `[0, 1]`.

`sampling_params` object

The sampling parameters for the model.

`type` string

The object type, which is always `score_model`.

OBJECT Score Model Grader

```
{
    "type": "score_model",
    "name": "Example score model grader",
    "input": [
        {
            "role": "user",
            "content": (
                "Score how close the reference answer i
                " Return just a floating point score\n\
                " Reference answer: {{item.label}}\n\n"
                " Model answer: {{sample.output_text}}"
            ),
        }
    ],
    "model": "gpt-4o-2024-08-06",
    "sampling_params": {
        "temperature": 1,
        "top_p": 1,
        "seed": 42,
    },
}
```

Multi Grader

A MultiGrader object combines the output of multiple graders to produce a single score.

`calculate_output` string
A formula to calculate the output based on grader results.

`graders` object

`name` string
The name of the grader.

`type` string
The object type, which is always `multi`.

OBJECT Multi Grader

```
"type": "multi",
"name": "example multi grader",
"graders": [
  {
    "type": "text_similarity",
    "name": "example text similarity grader",
    "input": "The graded text",
    "reference": "The reference text",
    "evaluation_metric": "fuzzy_match"
  },
  {
    "type": "string_check",
    "name": "Example string check grader",
    "input": "{{sample.output_text}}",
    "reference": "{{item.label}}",
    "operation": "eq"
  }
],
"calculate_output": "0.5 * text_similarity_score +  0.5 * s
```

Understanding Reinforcement Fine-Tuning (RFT)

Question:
Sarah had $50. She bought a book for $15 and then a toy for $10. After that, she earned $25 from a part-time job. How much money does Sarah have now?

<think>
Sarah had $50 initially

She bought a book for 15 dollars, leaving her with 50 - 15 = 35 dollars

The book was about mermaids and knights …

She then bought a toy for 10 dollars, leaving her with 35 - 10 = 25 dollars.

Finally, she earned 25 dollars from a part-time job, bringing her total to 25 + 25 = 40 dollars but a then she got another $10

<think / >

Sarah ends up with $50

Question:
Sarah had $50. She bought a book for $15 and then a toy for $10. After that, she earned $25 from a part-time job. How much money does Sarah have now?

<think>
Sarah had $50 initially

She bought a book for 15 dollars, leaving her with 50 - 15 = 35 dollars

Question:
Sarah had $50. She bought a book for $15 and then a toy for $10. After that, she earned $25 from a part-time job. How much money does Sarah have now?

<think>
Sarah had $50 initially

She bought a book for 15 dollars, leaving her with 50 - 15 = 35 dollars

The book was about mermaids and knights …

Question:
Sarah had $50. She bought a book for $15 and then a toy for $10. After that, she earned $25 from a part-time job. How much money does Sarah have now?

<think>
Sarah had $50 initially

She bought a book for 15 dollars, leaving her with 50 - 15 = 35 dollars

The book was about mermaids and knights …

She then bought a toy for 10 dollars, leaving her with 35 - 10 = 25 dollars.

Question:
Sarah had $50. She bought a book for $15 and then a toy for $10. After that, she earned $25 from a part-time job. How much money does Sarah have now?

<think>
Sarah had $50 initially

She bought a book for 15 dollars, leaving her with 50 - 15 = 35 dollars

The book was about mermaids and knights …

She then bought a toy for 10 dollars, leaving her with 35 - 10 = 25 dollars.

Finally, she earned 25 dollars from a part-time job, bringing her total to 25 + 25 = 40 dollars but a then she got another $10

Question:
Sarah had $50. She bought a book for $15 and then a toy for $10. After that, she earned $25 from a part-time job. How much money does Sarah have now?

<think>
Sarah had $50 initially

She bought a book for 15 dollars, leaving her with 50 - 15 = 35 dollars

The book was about mermaids and knights …

She then bought a toy for 10 dollars, leaving her with 35 - 10 = 25 dollars.

Finally, she earned 25 dollars from a part-time job, bringing her total to 25 + 25 = 40 dollars but a then she got another $10

<think / >

Sarah ends up with $50

Reward Processing

Let there be M total segments indexed by $k = 1, 2, \ldots, M$. Define:

- S_k: Thought score at segment k
- P_k: Progress score at segment k
- G: Overall critic score

First, normalize the thought score using z-score normalization:

$$\mu_S = \frac{1}{M}\sum_{k=1}^{M} S_k, \quad \sigma_S = \sqrt{\frac{1}{M}\sum_{k=1}^{M}(S_k - \mu_S)^2 + \epsilon_{std}}$$

$$\hat{S}_k = \frac{S_k - \mu_S}{\sigma_S}$$

Apply asymmetric scaling:

$$\hat{S}_k \leftarrow \begin{cases} \alpha \cdot \hat{S}_k, & \text{if } \hat{S}_k \geq 0 \\ \delta \cdot \hat{S}_k, & \text{if } \hat{S}_k < 0 \end{cases}$$

Define the moving average of past progress over window W:

$$\bar{P}_k = \begin{cases} P_k, & k = 1 \\ \frac{1}{\min(W, k-1)} \sum_{j=\max(1,k-W)}^{k-1} P_j, & k > 1 \end{cases}$$

The progress delta:

$$\Delta P_k = P_k - \bar{P}_k$$

Define the base progress effect:

$$E_k = \begin{cases} \sqrt{\Delta P_k} \cdot \frac{P_k}{\kappa^+}, & \Delta P_k > 0 \\ -\log(1 + |\Delta P_k|) \cdot \frac{B - P_k}{\kappa^-}, & \Delta P_k < 0 \\ 0, & \Delta P_k = 0 \end{cases}$$

Define the progress multiplier:

$$m_k = \frac{S_k}{\lambda}$$

Then the full progress effect is:

$$Q_k = E_k \cdot m_k$$

Finally, the combined segment score is:

$$C_k = G + \hat{S}_k + Q_k$$

For normalization of combined scores (if required):

$$\mu_C = \frac{1}{M}\sum_{k=1}^{M} C_k, \quad \sigma_C = \sqrt{\frac{1}{M}\sum_{k=1}^{M}(C_k - \mu_C)^2 + \epsilon_{std}}$$

Symbol	Meaning
G	Overall critic score
S_k	Thought score at segment k
P_k	Progress score at segment k
μ_S , σ_S	Mean and standard deviation of $S_1, \ldots, S_M$
ϵ_{std}	Small constant for numerical stability
α	Boost factor for positive normalized scores
δ	Dampening factor for negative normalized scores
W	Window size for computing moving average of past progress
κ^+ , κ^-	Divisors for scaling positive and negative progress
B	Base offset applied when progress is negative
λ	Divisor for progress multiplier
μ_C , σ_C	Mean and standard deviation of combined scores $C_1, \ldots, C_M$
ρ	Scaling factor for the final answer
α_1	Multiplier to cap the minimum segment score
α_2	Additive offset to cap the maximum segment score
γ	Discount factor for future rewards
ϵ	PPO clipping threshold for policy updates
c_v	Maximum allowed change in value predictions (clipping range)
c_{vf}	Coefficient for the value-loss term
β	Coefficient for the KL-divergence penalty
Y_k	TD target (reward + discounted next-state value)
R_k	Reward assigned to segment k (already defined in text, not table)
s_k, s_{k+1}	State before generating segment k or $k+1$
$V_\theta(s_k)$	Value prediction from current model θ at state s_k
$V_{\text{rollout}}(s_k)$	Value prediction from rollout policy at state s_k
$t_{k,j}$	Token j in segment k
$s_{k,j}$	State before generating token $t_{k,j}$
π_θ	Current model's policy (probability distribution over tokens)
π_{rollout}	Rollout model's policy
π_{ref}	Reference policy (used for KL regularization)
N_k	Number of tokens in segment k
D_{KL}	KL divergence between two distributions

Chain-of-Thought RFT Objective

For each segment $k = 1, \dots, M+1$, let:

- s_k: the state before generating the k-th chain-of-thought CL_k or the final answer.
- R_k: reward assigned to segment k

Define the temporal-difference target (return) and advantage:

$$Y_k = R_k + \gamma\, V_{\text{rollout}}(s_{k+1}), \quad A_k = Y_k - V_\theta(s_k).$$

Probability ratio:

$$\rho_k(\theta) = \exp\left(\sum_{j=1}^{N_k} \left[\log \pi_\theta(t_{k,j} \mid s_{k,j}) - \log \pi_{\text{rollout}}(t_{k,j} \mid s_{k,j})\right]\right).$$

Clipped policy loss:

$$L_k^{\text{CLIP}} = \max\bigl(-A_k\, \rho_k(\theta), -A_k\, \text{clip}(\rho_k(\theta),\, 1-\epsilon,\, 1+\epsilon)\bigr).$$

Value loss:

Let

$$V_k^{\text{clip}} = V_{\text{rollout}}(s_k) + \text{clip}\bigl(V_\theta(s_k) - V_{\text{rollout}}(s_k),\, -c_v,\, c_v\bigr).$$

Then

$$L_k^{\text{VF}} = \frac{1}{2}\begin{cases} \max\bigl((V_\theta(s_k) - Y_k)^2, (V_k^{\text{clip}} - Y_k)^2\bigr) & \text{if } c_v > 0, \\ (V_\theta(s_k) - Y_k)^2 & \text{if } c_v \le 0. \end{cases}$$

KL penalty:

$$L_k^{\text{KL}} = \sum_{j=1}^{N_k} D_{\text{KL}}\bigl(\pi_{\text{rollout}}(\cdot \mid s_{k,j}) \,\|\, \pi_{\text{ref}}(\cdot \mid s_{k,j})\bigr).$$

Total segment loss:

$$L_k(\theta) = L_k^{\text{CLIP}} + c_{vf}\, L_k^{\text{VF}} + \beta\, L_k^{\text{KL}}.$$

Overall objective (minimize expected sum of segment losses):

$$\mathcal{L}_{\text{RFT}}(\theta) = \mathbb{E}_{\tau \sim \pi_{\text{rollout}}}\left[\sum_{k=1}^{M+1} L_k(\theta)\right].$$

Value loss:

Let

$$V_k^{\text{clip}} = V_{\text{rollout}}(s_k) + \text{clip}\big(V_\theta(s_k) - V_{\text{rollout}}(s_k),\, -c_v,\, c_v\big).$$

Then

$$L_k^{\text{VF}} = \frac{1}{2}\begin{cases} \max\big((V_\theta(s_k) - Y_k)^2, (V_k^{\text{clip}} - Y_k)^2\big) & \text{if } c_v > 0, \\ (V_\theta(s_k) - Y_k)^2 & \text{if } c_v \le 0. \end{cases}$$

KL penalty:

$$L_k^{\text{KL}} = \sum_{j=1}^{N_k} D_{\text{KL}}\big(\pi_{\text{rollout}}(\cdot \mid s_{k,j}) \,\|\, \pi_{\text{ref}}(\cdot \mid s_{k,j})\big).$$

Total segment loss:

$$L_k(\theta) = L_k^{\text{CLIP}} + c_{vf}\, L_k^{\text{VF}} + \beta\, L_k^{\text{KL}}.$$

Overall objective (minimize expected sum of segment losses):

$$\mathcal{L}_{\text{RFT}}(\theta) = \mathbb{E}_{\tau \sim \pi_{\text{rollout}}}\Big[\sum_{k=1}^{M+1} L_k(\theta)\Big].$$

for epoch = 1 to E **do**

  for each batch $B \subset D$ **do**

    Generate trajectory under π_{ref}

    Sample CoT output $a = \{\mathrm{CL}_1, \ldots, \mathrm{CL}_M\}$ from $\pi_{\text{ref}}(\cdot|B)$

    Critic evaluation

    Get G, (S_k, P_k) from critic on a

    Compute segment rewards

    For $k = 1, \ldots, M$, compute R_k

      compute R_{M+1} for answer

      Optionally whiten $\{R_k\}_{k=1}^{M+1}$

    Precompute rollout values & log-probs

    For $k = 1, \ldots, M+1$:

      compute $V_{\text{ref}}(s_k), \log \pi_{\text{ref}}(a_k|s_k)$

    Policy update under π_θ

    For $k = 1, \ldots, M+1$:

      Compute TD target $Y_k = R_k + \gamma V_{\text{ref}}(s_{k+1})$

      Compute advantage $A_k = Y_k - V_\theta(s_k)$

      Compute $\rho_k = \pi_\theta(a_k|s_k)/\pi_{\text{ref}}(a_k|s_k)$

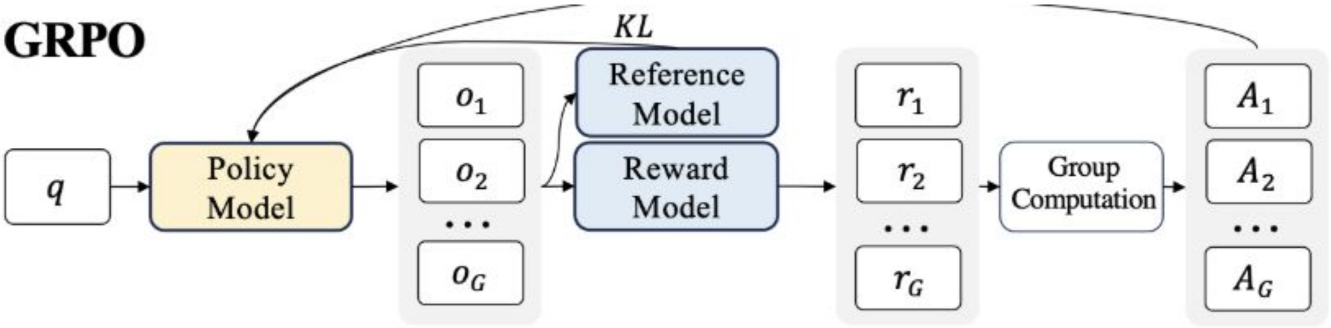
      Compute L_k^{CLIP}, L_k^{VF}, L_k^{KL}

    $L_{\text{total}} = \sum_{k=1}^{M+1}[L_k^{\text{CLIP}} + c_{vf}L_k^{\text{VF}} + \beta L_k^{\text{KL}}]$

    Update $\theta \leftarrow \theta - \eta\nabla_\theta L_{\text{total}}$

    Update $\pi_{\text{ref}} \leftarrow \pi_\theta$ (optional, periodic)

Reinforcement Fine-Tuning v2 (GRFT)



Question:
Sarah had $50. She bought a book for $15 and then a toy for $10. After that, she earned $25 from a part-time job. How much money does Sarah have now?

<think>
Sarah had $50 initially

She bought a book for 15 dollars, leaving her with 50 - 15 = 35 dollars

Question:
Sarah had $50. She bought a book for $15 and then a toy for $10. After that, she earned $25 from a part-time job. How much money does Sarah have now?

<think>
Sarah had $50 initially

She bought a book for 15 dollars, leaving her with 50 - 15 = 35 dollars

The book was about mermaids and knights …

Question:
Sarah had $50. She bought a book for $15 and then a toy for $10. After that, she earned $25 from a part-time job. How much money does Sarah have now?

<think>
Sarah had $50 initially

She bought a book for 15 dollars, leaving her with 50 - 15 = 35 dollars

The book was about mermaids and knights …

She then bought a toy for 10 dollars, leaving her with 35 - 10 = 25 dollars.