LLM ALIGNMENT

Team 6
Fengyu Gao, Shunqiang Feng, Wei Shen, Zihan Zhao

A Comprehensive Survey of LLM Alignment Techniques: RLHF, RLAIF, PPO, DPO and More

Additional references:

[1] https://anukriti-ranjan.medium.com/preference-tuning-llms-ppo-dpo-grpo-a-simple-guide-135765c87090

[2] https://web.stanford.edu/class/cs224n/slides/cs224n-spr2024-lecture10-prompting-rlhf.pdf

Fengyu Gao (wan6jj)

Aligning Language Models

LMs like GPT-3 are misaligned: they maximize the likelihood of large untrusted datasets.

This leads to:

- Not following the user's instruction
- Making up facts
- Generating harmful/toxic content
- ……

Explain the moon landing to a 6 year old in a few sentences.

GPT-3

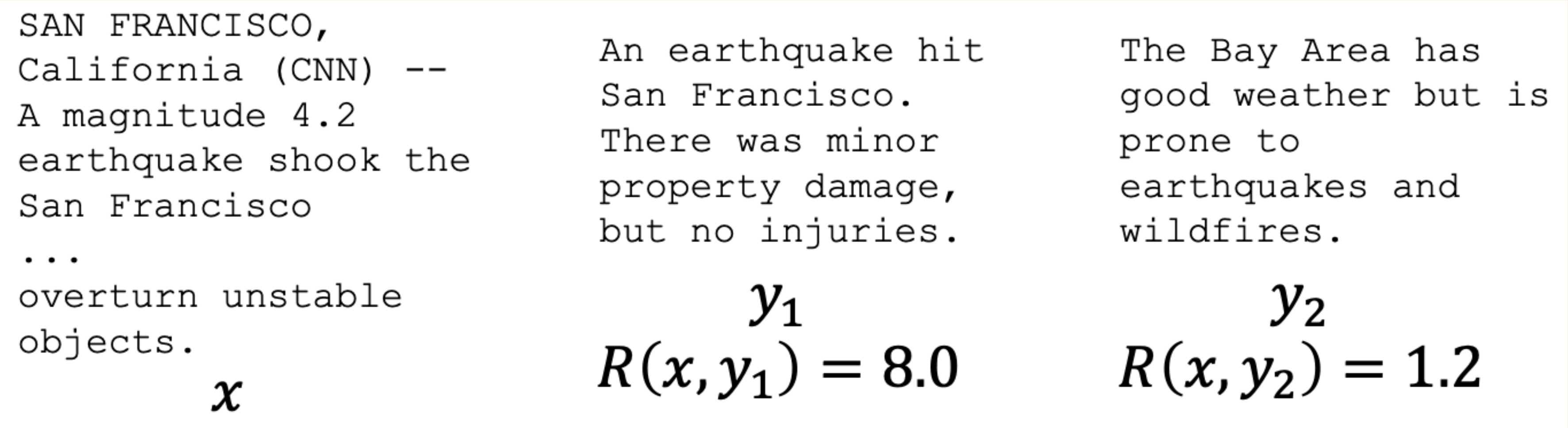
```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

Language models are not *aligned* with user intent.

Aligning LMs with Human Feedback

Suppose we are training a LM for a summarization task.

For a given instruction x and a generated summary y, we assume we can obtain a human reward of that summary: $R(x, y)$ – where higher values indicate better quality.



We want to maximize the expected reward based on this feedback.

A (very!) brief introduction to RL

Reinforcement Learning = Learning by Doing and Getting Feedback

- An agent (LLM) interacts with an environment and learns by trial and error.
- Large Rewards (✅ Correct answer!) encourage desirable outputs.
- Small Rewards (❌ Incorrect response!) discourage undesirable outputs..
- RL algorithms (e.g., PPO, DPO, GRPO) train LLMs to maximize this reward.

How do we get the rewards?

Q1: Human-in-the-loop is expensive!

Solution: Instead of asking humans directly, we train a separate reward model to learn human preferences.

Q2: Human judgments are noisy and miscalibrated!

Solution: Use pairwise comparisons instead of direct ratings.

An earthquake hit San Francisco. There was minor property damage, but no injuries.	>	A 4.2 magnitude earthquake hit San Francisco, resulting in massive damage.

$$L_{\mathrm{RM}}(r_\phi) = -\frac{1}{C_K^2}\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\phi(x,y_w) - r_\phi(x,y_l)\right)\right)\right]$$

y_w: winning sample

y_l: losing sample

y_w should score higher than y_l

RLHF: Optimizing the learned reward model

We have the following:

- A pretrained (possibly instruction-finetuned) LM $\pi_{ref}(y|x)$
- A reward model $r_\phi(x, y)$ that produces scalar rewards for LM outputs, trained on a dataset of human comparisons

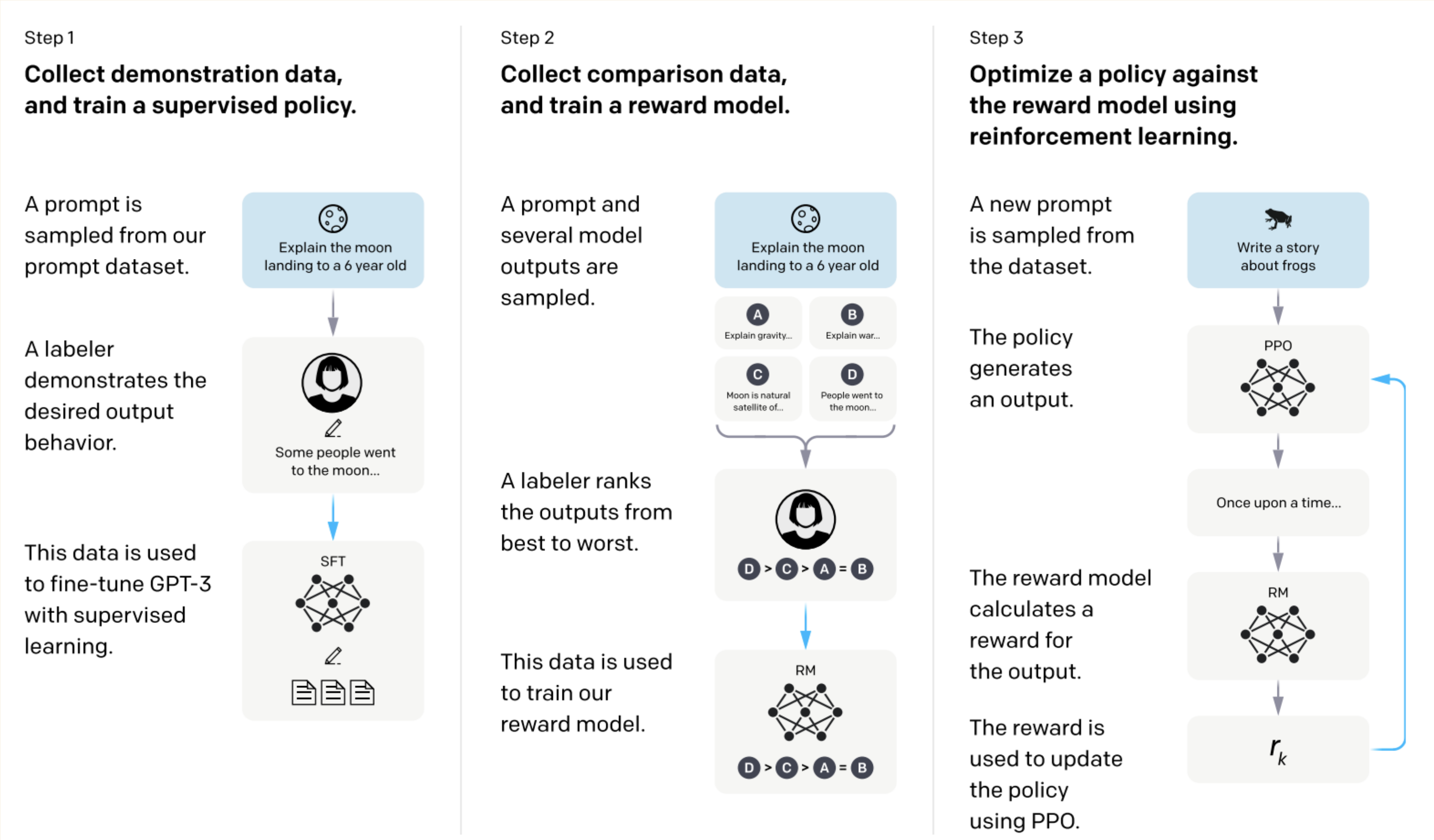
Now to do RLHF:

$$\pi_\theta^*(y|x) = \max_{\pi_\theta} \mathbb{E}_{x\sim D}\left[\mathbb{E}_{y\sim\pi_\theta(y|x)} r_\phi(x, y) - \beta D_{\mathrm{KL}}(\pi_\theta(y|x) || \pi_{\mathrm{ref}}(y|x))\right]$$

Maximizing rewards

Minimizing divergence between current policy and reference policy

High-Level Overview: RLHF Pipeline



supervised fine-tuning/instruction tuning -> reward modeling -> policy optimization

Can we simplify RLHF? Towards DPO

Direct Preference Optimization (DPO): directly optimizes policy based on human preference data using a clever loss function.

Recall our objective in RLHF:

$$\pi_\theta^*(y|x) = \max_{\pi_\theta} \mathbb{E}_{x\sim D}\left[\mathbb{E}_{y\sim\pi_\theta(y|x)} r_\phi(x,y) - \beta D_{\mathrm{KL}}(\pi_\theta(y|x)||\pi_{\mathrm{ref}}(y|x))\right]$$

There is a closed form solution to this:

$$\pi_\theta(y|x) = \frac{1}{Z(x)}\pi_{\mathrm{ref}}(y|x)e^{\left(\frac{1}{\beta}r_\theta(x,y)\right)}$$

Rearrange the terms:

$$r_\theta(x,y) = \beta\log\left(\frac{\pi_\theta(y|x)}{\pi_{\mathrm{ref}}(y|x)}\right) + \beta\log Z(x)$$

Reward model can be written in terms of policy!

Can we simplify RLHF? Towards DPO

Direct Preference Optimization (DPO): directly optimizes policy based on human preference data using a clever loss function.

Recall, how we fit the reward model in RLHF:

$$L_{\mathrm{RM}}(r_\phi) = -\frac{1}{C_K^2}\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\phi(x,y_w) - r_\phi(x,y_l)\right)\right)\right]$$

Notice that we only need the difference between the rewards. Simplify for rewards:

$$r_\theta(x,y_w) - r_\theta(x,y_l) = \beta\left[\log\left(\frac{\pi_\theta(y_w|x)}{\pi_{\mathrm{ref}}(y_w|x)}\right) - \log\left(\frac{\pi_\theta(y_l|x)}{\pi_{\mathrm{ref}}(y_l|x)}\right)\right]$$

The final DPO loss function is:

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

We have a classification loss function that connects **preference data** to **LM parameters** directly!

Summary (RLHF and DPO)

- Our goal is to optimize for Human Preferences
  - Instead of humans writing the answers or giving uncalibrated scores, we get humans to rank different LM generated answers.
- RLHF
  - Step 1: Supervise fine-tuning on a labeled dataset
  - Step 2: Train an explicit reward model on comparison data to predict a score for a completion
  - Step 3: Optimize the LM to maximize the predicted score (under KL-constraint)
  - Very effective when tuned well, computationally expensive
- DPO
  - Optimize LM parameters directly on preference data by solving a binary classification problem
  - Simple and effective, similar properties to RLHF

Research directions of LLM alignment

- Reward model
- Feedback
- RL policy
- Optimization

Reward model

- **Explicit Reward Model vs. Implicit Reward Model**
  - e.g., RLHF vs. DPO
- **Pointwise Reward Model vs. Preferencewise Model**
  - $R(x, y)$ vs. prob. that the desired response is preferred over the undesired one
- **Response-Level Reward vs. Token-Level Reward**
  - Assign a single score to the entire response vs. provide feedback at each token
- **Negative Preference Optimization**
  - Use only prompts and undesired responses from RLHF datasets, generating desired responses with LLMs instead of relying on human-labeled preferred responses

Feedback

- **Preference Feedback vs. Binary Feedback**
  - Rank responses vs. simple positive or negative signal without ranking
- **Pairwise Feedback vs. Listwise Feedback**
  - Compare two responses vs. rank multiple responses together
- **Human Feedback vs. AI Feedback**
  - Real user preferences vs. LLM-generated evaluations

RL

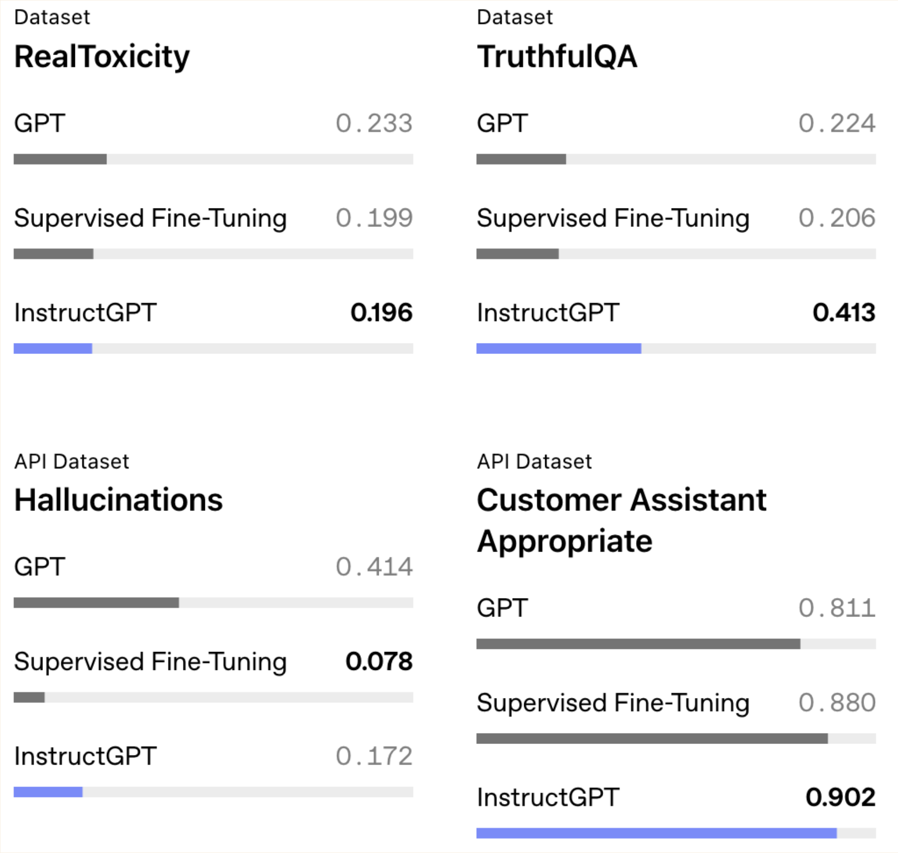
- **Reference-Based RL vs. Reference-Free RL**
  - Minimize divergence from a reference policy vs. remove reference policy (e.g. SimPO)
- **Length-Control RL**
  - Standard RL ignores response length. Length-control RL adjusts rewards to prevent verbosity bias in LLM-generated responses. E.g., R-DPO and SimPO.
- **Different Divergences in RL**
  - KL divergence, f-divergence, ……
- **On-policy or Off-policy Learning**
  - Generate responses using the latest policy vs. reuse past responses

Optimization

- **Iterative/Online Preference Optimization vs. Non-Iterative/Offline Preference Optimization**
    - Continuously update alignment with new data vs. align models using a fixed dataset
- **Separating SFT and Alignment vs. Merging SFT and Alignment**
    - Newer approaches integrate SFT and alignment into a single process, e.g., ORPO, PAFT.

InstructGPT: scaling up RLHF to 30k tasks!

Evaluate InstructGPT from three perspectives: Helpful, Honest, and Harms.



InstructGPT: scaling up RLHF to 30k tasks!

Prompt

Explain the moon landing to a 6 year old in a few sentences.

Completion

GPT-3

```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

InstructGPT

```
People went to the moon, and they took pictures of what they saw, and sent them back to the earth so we could all see them.
```

Wei Shen (zyy5hb)

OpenRLHF: An Easy-to-use, Scalable and High-performance RLHF Framework

PPO

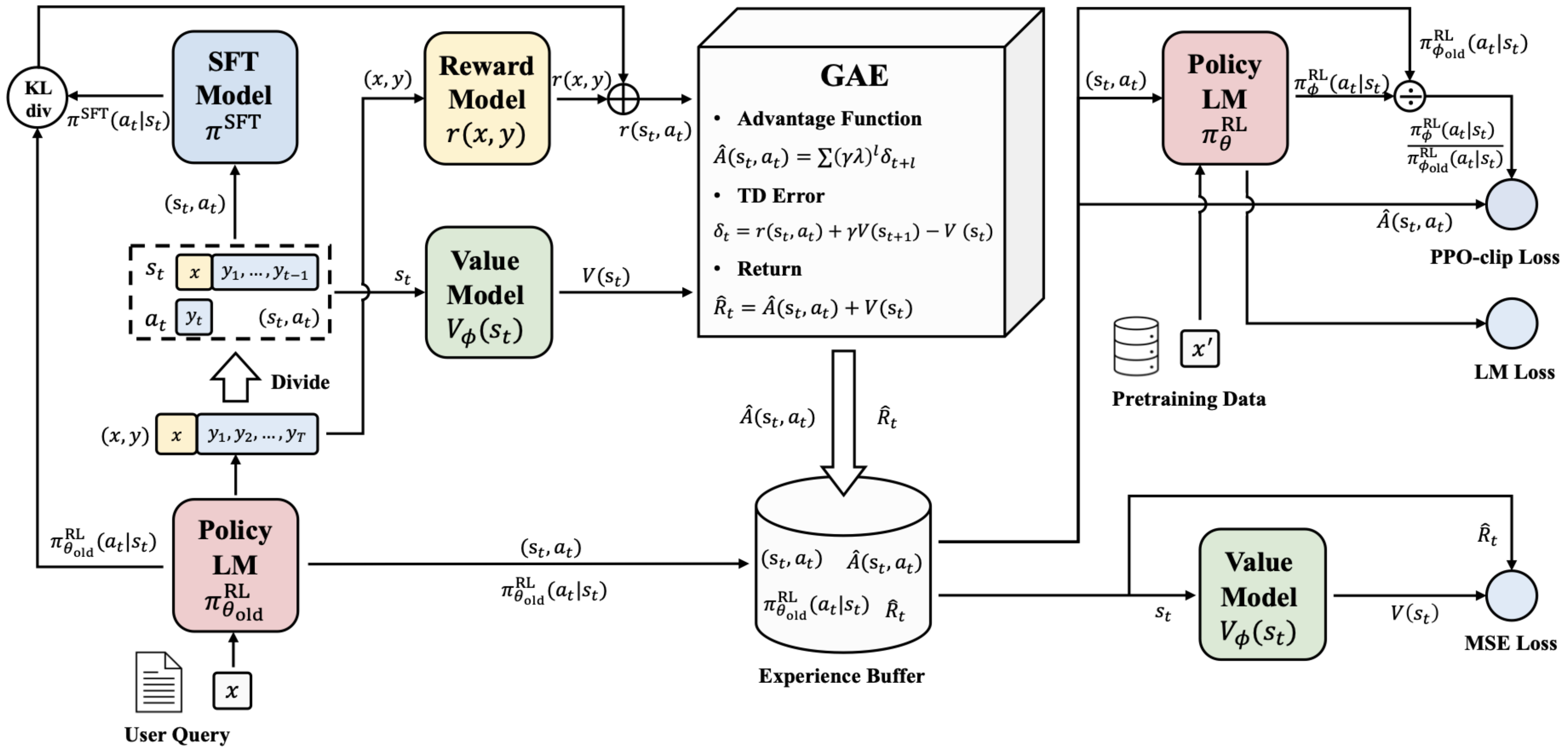


Figure 1: PPO workflow, depicting the sequential steps in the algorithm's execution. The process begins with sampling from the environment, followed by the application of GAE for improved advantage approximation. The diagram then illustrates the computation of various loss functions employed in PPO, signifying the iterative nature of the learning process and the policy updates derived from these losses.

SFT Model: Supervised FineTuning Model; GAE: Generalized Advantage Estimation

https://arxiv.org/pdf/2307.04964

Background

- **Problem:** Scaling RLHF training to larger models requires **efficiently allocating** at least **four component models (actor (policy model), critic(value model), reward, reference)** across multiple GPUs due to the memory limit of each accelerator.
- **Existing libraries:**
- Ray is a **distributed execution framework** that provides powerful scheduling and scaling capabilities for parallel and distributed computing workloads.
- vLLM is a fast and easy-to-use library for **LLM inference and serving.** It delivers state-of-the-art serving throughput through efficient management of attention key and value memory with **PagedAttention**, continuous batching of incoming requests, and fast model execution with CUDA graph.
- DeepSpeed is an **optimization library** designed to enhance the efficiency of **large-scale** deep-learning models.

Scheduling Optimization

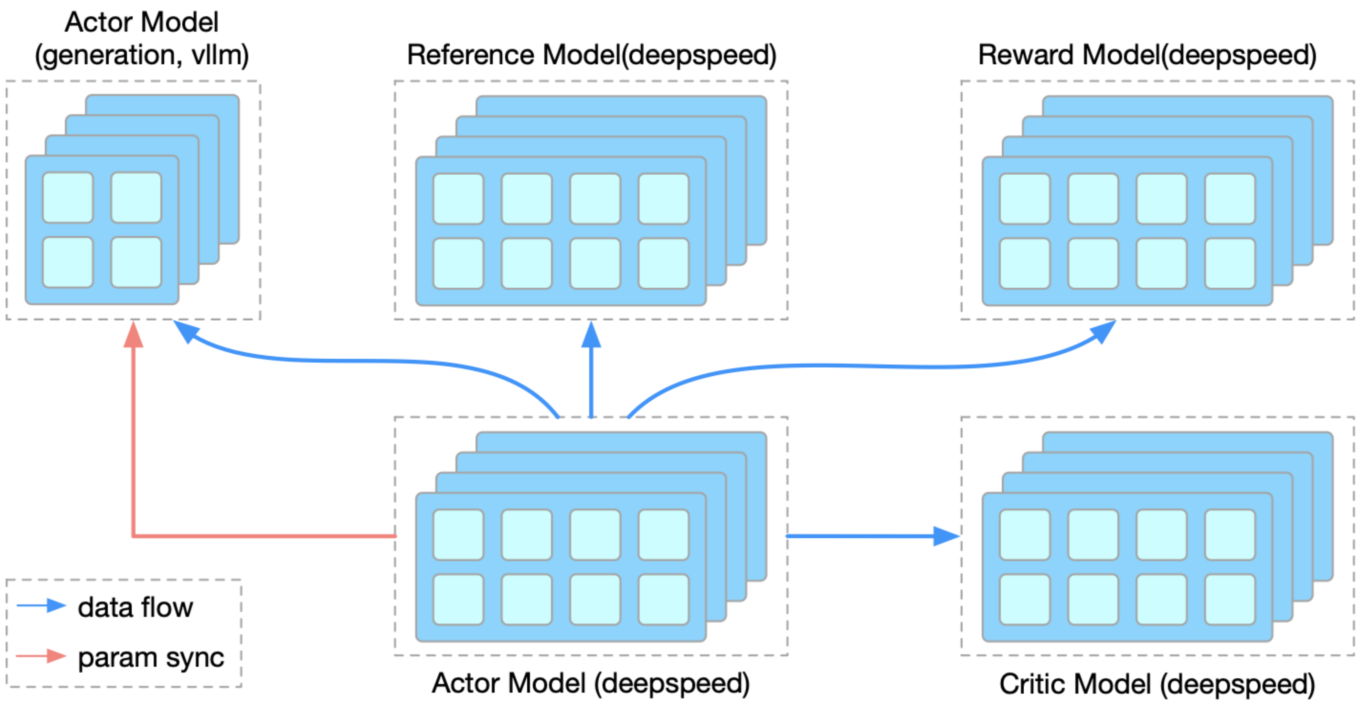


Figure 1: Ray Architecture of OpenRLHF. The four models in RLHF are distributed across different GPUs by Ray, which can also be freely merged or offloaded to save GPUs. The vLLM is used to accelerate actor generation. OpenRLHF synchronizes the weights of the ZeRO engine to the vLLM engine using the NVIDIA Collective Communications Library (NCCL).

Performance Optimization

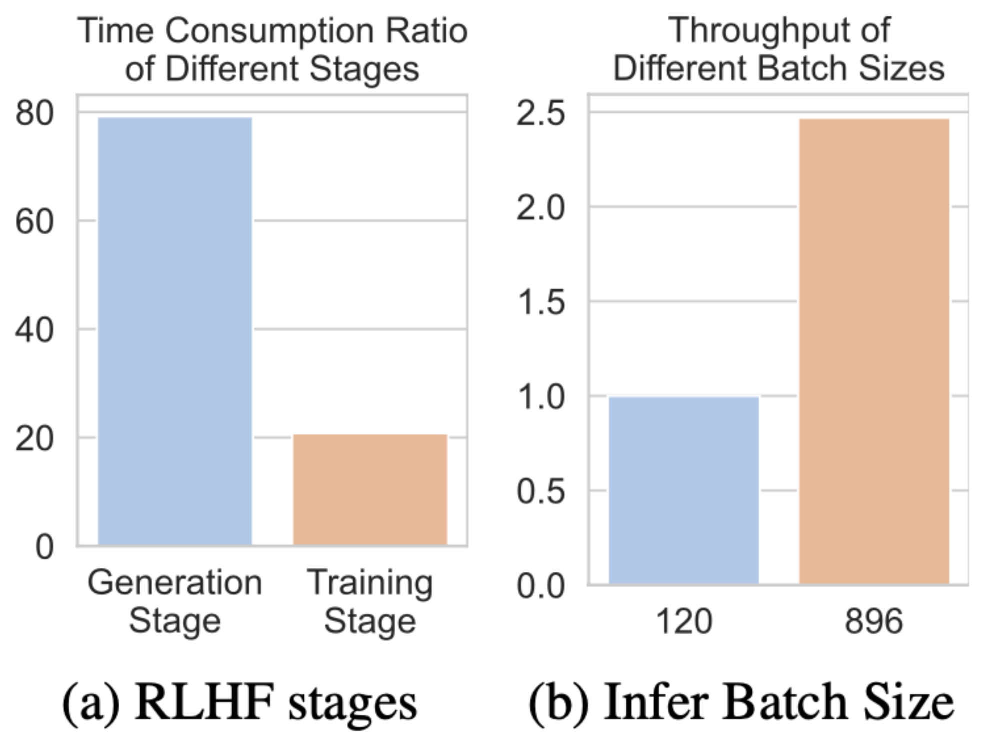


Figure 4: Performance Profiling using LLaMA2 7B and NVIDIA A100.

- **The primary bottleneck** is at the **PPO sample generation** stage which takes up 80% of overall training time.
- Figure 4b shows that the **larger inference batch size** can significantly improve the generation throughput.
- OpenRLHF **distributes** the four models across multiple GPUs using Ray, effectively **increasing the batch size**.

Additional improvements:

- Offloading Adam optimizer states to the CPU frees up GPU memory, allowing for **larger batch sizes** during generation
- Employing **Flash Attention 2** accelerates Transformer model training.
- **Remove redundant padding** from training samples using PyTorch tensor slicing.

PPO Implementation Tricks

- Predict reward only on the end-of-text token of the sequence.
- Use token-level reinforcement learning for language models.
- Use Kullback–Leibler (KL) divergence loss term in PPO.
- Use pre-trained loss term in PPO, tuned based on a relative scale of the policy loss.
- Apply reward normalization for training stability.
- Apply distributed advantage normalization with global statistics.
- Use the Linear Warmup Cosine Annealing learning rate scheduler.
- Initialize the Critic with the weights of the reward model.
- Use a lower learning rate for the Actor while the Critic has a higher learning rate.
- Freeze the weights of the Actor in the initial learning stage for better initialization of the Critic.
- Use GAE (Generalized Advantage Estimation).

Ease of Use

For user-friendliness, OpenRLHF provides **one-click trainable scripts** for supported algorithms, fully compatible with the Hugging Face library for specifying model and dataset names or paths.

```
pip install openrlhf[vllm]

ray start --head --node-ip-address 0.0.0.0
ray job submit -- python3 openrlhf.cli.train_ppo_ray \
    --ref_num_gpus_per_node 4 \                         # Number of GPUs for Ref model
    --reward_num_gpus_per_node 4 \                      # Number of GPUs for RM
    --critic_num_gpus_per_node 4 \                      # Number of GPUs for Critic
    --actor_num_gpus_per_node 4 \                       # Number of GPUs for Actor
    --vllm_num_engines 4 \                              # Number of vLLM engines
    --vllm_tensor_parallel_size 2 \                     # vLLM Tensor Parallel Size
    --colocate_actor_ref \                              # Colocate Actor and Ref
    --colocate_critic_reward \                          # Colocate Critic and RM
    --ref_reward_offload \                              # Offload Ref and RM
    --pretrain {HF Model name or path after SFT} \
    --reward_pretrain {HF Reward model name or path} \
    --zero_stage 3 \                                    # DeepSpeed ZeRO stage
    --bf16 \                                            # Enable BF16
    --init_kl_coef 0.01 \                               # KL penalty coefficient
    --prompt_data {HF Prompt dataset name or path} \
    --input_key {Prompt dataset input key}
    --apply_chat_template \                             # Apply HF tokenizer template
    --normalize_reward \                                # Enable Reward Normalization
    --adam_offload \                                    # Offload Adam Optimizer
    --flash_attn \                                      # Enable Flash Attention
    --save_path {Model output path}
```

Listing 1: PPO startup method based on Deepspeed and Ray

Supported Algorithms

- Supervised FineTuning
- Reward Model Training
- Proximal Policy Optimization (PPO)
- Direct Preference Optimization (DPO)
- Kahneman-Tversky Optimization (KTO)
- Iterative Direct Preference Optimization (Iterative DPO)
- Rejection Sampling Finetuning (RS)
- Conditional Supervised Finetuning

Group Relative Policy Optimization (GRPO)

Ref: https://medium.com/@sahin.samia/the-math-behind-deepseek-a-deep-dive-into-group-relative-policy-optimization-grpo-8a75007491ba

What is GRPO?

- Group Relative Policy Optimization (GRPO) is a **reinforcement learning** (RL) algorithm specifically designed to enhance reasoning capabilities in Large Language Models (LLMs). Unlike traditional RL methods, which rely heavily on external evaluators (critics) to guide learning, GRPO optimizes the model by evaluating **groups of responses** relative to one another. This approach enables more **efficient** training, making GRPO ideal for reasoning tasks that require complex problem-solving and long chains of thought.
- Proposed and used in DeepSeek R1

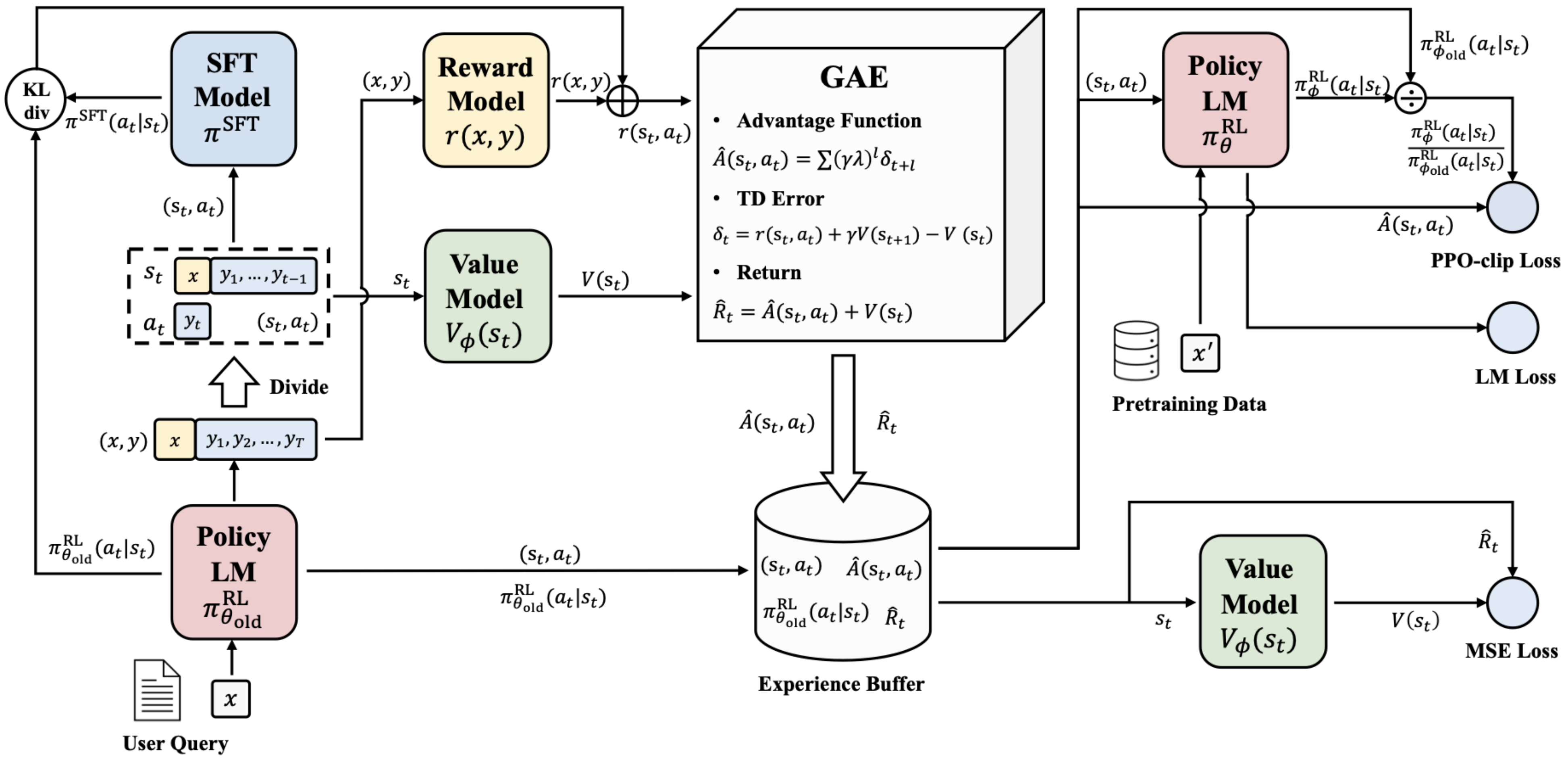


Figure 1: PPO workflow, depicting the sequential steps in the algorithm's execution. The process begins with sampling from the environment, followed by the application of GAE for improved advantage approximation. The diagram then illustrates the computation of various loss functions employed in PPO, signifying the iterative nature of the learning process and the policy updates derived from these losses.

https://arxiv.org/pdf/2307.04964

Why GRPO

- **Challenges** of Traditional RL methods like Proximal Policy Optimization (PPO)
- **Dependency on a Critic Model**:
  - PPO requires a separate critic model to estimate the value of each response, which doubles memory and computational requirements.
- **High Computational Cost**:
  - RL pipelines often demand significant computational resources to evaluate and optimize responses iteratively.
- **Scalability Issues**:
  - Absolute reward evaluations struggle with diverse tasks, making it hard to generalize across reasoning domains.

Why GRPO

- How GRPO **Addresses** These Challenges of PPO
- **Critic-Free Optimization**:
  - GRPO removes the need for a critic model by comparing responses within a group, significantly reducing computational overhead.
- **Relative Evaluation**:
  - Instead of relying on an external evaluator, GRPO uses group dynamics to assess how well a response performs relative to others in the same batch.
- **Efficient Training**:
  - By focusing on group-based advantages, GRPO simplifies the reward estimation process, making it faster and more scalable for large models.

Key Idea of GRPO: relative evaluation

- For each input query, the model generates a **group** of potential responses.
- These responses are scored based on how they **compare to others in the group**, rather than being evaluated in isolation.
- The advantage of a response reflects how much better or worse it is relative to the group's average performance.

Understanding the GRPO Objective Function

The GRPO Objective Function

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)}\left[\frac{1}{G}\sum_{i=1}^{G}\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}A_i,\text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)},1-\epsilon,1+\epsilon\right)A_i\right)-\beta D_{KL}(\pi_\theta||\pi_{\text{ref}})\right]$$

This might look daunting at first, but each component plays a critical role in stabilizing learning and improving performance.

1. **Expected Value**:
   - $\mathbb{E}_{q\sim P(Q)}$: The expectation is over all input queries q, drawn from the training dataset $P(Q)$.
   - $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(O|q)$: For each query, a group of responses $\{o_i\}_{i=1}^{G}$ is sampled from the old policy $\pi_{\theta_{\text{old}}}$.

Understanding the GRPO Objective Function

The GRPO Objective Function

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)}\left[\frac{1}{G}\sum_{i=1}^{G}\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon\right)A_i\right) - \beta D_{KL}(\pi_\theta||\pi_{\text{ref}})\right]$$

This might look daunting at first, but each component plays a critical role in stabilizing learning and improving performance.

2. **Policy Ratio:**

- $\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}$: The ratio between the probability of generating a response o_i under the new policy π_θ versus the old policy $\pi_{\theta_{\text{old}}}$.
- This ratio indicates how the new policy differs from the old one for a given response.

Understanding the GRPO Objective Function

The GRPO Objective Function

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)}\left[\frac{1}{G}\sum_{i=1}^{G}\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon\right)A_i\right) - \beta D_{KL}(\pi_\theta||\pi_{\text{ref}})\right]$$

This might look daunting at first, but each component plays a critical role in stabilizing learning and improving performance.

3. **Advantage Estimate** (A_i):
   - A_i: The advantage of a response o_i, which reflects how much better or worse it is compared to others in the group.
   - Computed as:

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \ldots, r_G\})}{\text{std}(\{r_1, r_2, \ldots, r_G\})}$$

   Here:

   - r_i: Reward assigned to response o_i.
   - $\text{mean}(\{r_1, r_2, \ldots, r_G\})$: The average reward for the group.
   - $\text{std}(\{r_1, r_2, \ldots, r_G\})$: The standard deviation of rewards within the group.

Understanding the GRPO Objective Function

The GRPO Objective Function

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)}\left[\frac{1}{G}\sum_{i=1}^{G}\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon\right)A_i\right) - \beta D_{KL}(\pi_\theta||\pi_{\text{ref}})\right]$$

This might look daunting at first, but each component plays a critical role in stabilizing learning and improving performance.

Reward Modeling in DeepSeek R1-Zero: **rule-based reward system**

- **Accuracy rewards:** The accuracy reward model evaluates whether the response is correct.
- **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.

We **do not** apply the outcome or process **neural reward model** in developing DeepSeek-R1-Zero, because we find that the neural reward model may suffer from reward hacking in the large-scale reinforcement learning process, and retraining the reward model needs additional training resources and it complicates the whole training pipeline.

https://arxiv.org/pdf/2501.12948

Understanding the GRPO Objective Function

The GRPO Objective Function

$$J_{\mathrm{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\mathrm{old}}}(O|q)}\left[\frac{1}{G}\sum_{i=1}^{G}\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\mathrm{old}}}(o_i|q)}A_i, \mathrm{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\mathrm{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon\right)A_i\right) - \beta D_{KL}(\pi_\theta||\pi_{\mathrm{ref}})\right]$$

This might look daunting at first, but each component plays a critical role in stabilizing learning and improving performance.

4. **Clipping for Stability**:
   - $\mathrm{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\mathrm{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon\right)$: Limits the policy ratio to a range $[1-\epsilon, 1+\epsilon]$ to prevent overly large updates.
   - This stabilizes learning and avoids drastic changes to the policy.
5. **KL Divergence Penalty**:
   - $-\beta D_{KL}(\pi_\theta||\pi_{\mathrm{ref}})$: Regularizes the new policy π_θ by penalizing its divergence from a reference policy π_{ref}.
   - Ensures that the new policy doesn't deviate too much, maintaining consistency.
6. **Averaging Across the Group**:
   - $\frac{1}{G}\sum_{i=1}^{G}$: The objective is averaged across the group of responses, ensuring fair evaluation.

Understanding the GRPO Objective Function

The GRPO Objective Function

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)}\left[\frac{1}{G}\sum_{i=1}^{G}\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon\right)A_i\right) - \beta D_{KL}(\pi_\theta||\pi_{\text{ref}})\right]$$

This might look daunting at first, but each component plays a critical role in stabilizing learning and improving performance.

1. **Generate a group of responses** for a query.
2. **Calculate rewards** for each response based on predefined criteria (e.g., accuracy, format).
3. **Compare responses within the group** to calculate their relative advantage (AiA_iAi).
4. **Update the policy** to favor responses with higher advantages, ensuring stability with clipping.
5. **Regularize the updates** to prevent the model from drifting too far from its baseline.

Thank you!