Variance Reduction and Adaptive Sampling in Reinforcement Learning with Verifiable Rewards

Tong Zhang

Joint work with Jiarui Yao, Yifan Hao, Hanning Zhang, Hanze Dong, Wei Xiong, Chenlu Ye, Baohao Liao, Xinxing Xu, Christof Monz, Jiang Bian, and Nan Jiang

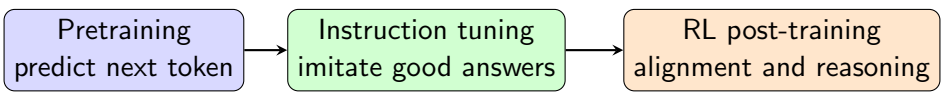
University of Illinois Urbana-Champaign

LLM Training Pipeline

We start with a pretrained base language model that predicts text. Post-training turns it into a useful assistant or reasoning model.

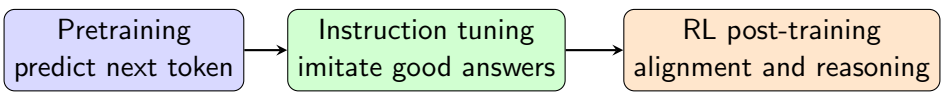
LLM Training Pipeline

We start with a pretrained base language model that predicts text. Post-training turns it into a useful assistant or reasoning model.



LLM Training Pipeline

We start with a pretrained base language model that predicts text. Post-training turns it into a useful assistant or reasoning model.



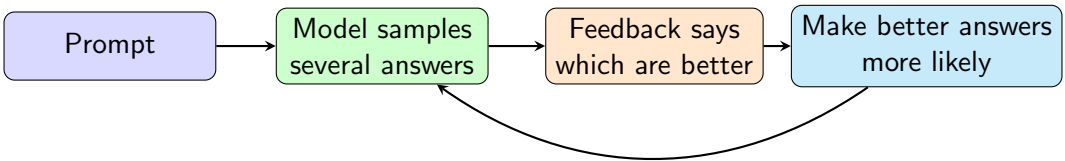
- Pretraining teaches broad language and knowledge.
- Instruction tuning teaches the model to follow prompts.
- RL post-training improves behavior using feedback from humans, AI judges, or verifiers.

Post-Training Reweights Model Behavior

Post-training does not teach the model from scratch. It changes which answers the model is more likely to produce.

Post-Training Reweights Model Behavior

Post-training does not teach the model from scratch. It changes which answers the model is more likely to produce.



- For chat models, feedback often means human preference.
- For reasoning models, feedback can be automatic: did the final answer check out?
- This talk focuses on the automatic-feedback setting.

RLVR: Automatic Feedback for Reasoning

RLVR means reinforcement learning with verifiable rewards. Instead of learning a reward model from human feedback, we use an automatic verifier.

RLVR: Automatic Feedback for Reasoning

RLVR means reinforcement learning with verifiable rewards. Instead of learning a reward model from human feedback, we use an automatic verifier.

$x \sim \mathcal{D}$	prompt or problem,
$a \sim \pi_\theta(\cdot \mid x)$	sampled response: reasoning plus answer,
$r^\star(x, a) \in \{0, 1\}$	verifier reward.

RLVR: Automatic Feedback for Reasoning

RLVR means reinforcement learning with verifiable rewards. Instead of learning a reward model from human feedback, we use an automatic verifier.

$x \sim \mathcal{D}$	prompt or problem,
$a \sim \pi_\theta(\cdot \mid x)$	sampled response: reasoning plus answer,
$r^\star(x, a) \in \{0, 1\}$	verifier reward.

x :	"What is $17 + 28$?"	
a_1 :	"$17 + 28 = 45$, so the answer is 45"	$r^\star(x, a_1) = 1$
a_2 :	"$17 + 28 = 46$, so the answer is 46"	$r^\star(x, a_2) = 0$

RLVR Objective is Average Pass Rate

The training objective is to make verifier-passing responses more likely. Equivalently, maximize the average pass rate across prompts.

RLVR Objective is Average Pass Rate

The training objective is to make verifier-passing responses more likely. Equivalently, maximize the average pass rate across prompts.

$$J(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, a\sim\pi_\theta(\cdot|x)}\left[r^\star(x,a)\right] = \mathbb{E}_{x\sim\mathcal{D}}\left[p_\theta(x)\right], \qquad p_\theta(x) = \Pr_{a\sim\pi_\theta(\cdot|x)}[r^\star(x,a)=1].$$

RLVR Objective is Average Pass Rate

The training objective is to make verifier-passing responses more likely. Equivalently, maximize the average pass rate across prompts.

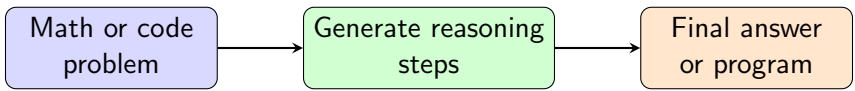
$$J(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, a\sim\pi_\theta(\cdot|x)}\left[r^\star(x,a)\right] = \mathbb{E}_{x\sim\mathcal{D}}\left[p_\theta(x)\right], \qquad p_\theta(x) = \Pr_{a\sim\pi_\theta(\cdot|x)}[r^\star(x,a)=1].$$

For a fixed prompt, we only observe this pass rate through sampled responses:

$$\hat{p}_\theta(x) = \frac{1}{n}\sum_{j=1}^{n} r^\star(x,a_j), \qquad a_j \sim \pi_\theta(\cdot \mid x).$$

Reasoning Models Generate Intermediate Work

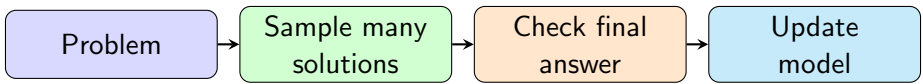
A reasoning model is trained to spend more computation on a problem before giving the final answer.



- It may generate a long chain of thought before the answer.
- We usually cannot directly score every intermediate step.
- But for many tasks, we can check the final result.

Reasoning Training Uses Sampled Solutions

For reasoning tasks, one prompt can produce many candidate solutions. Training uses the candidates that pass an automatic check.



- Math: compare the final answer to a known answer.
- Code: run unit tests.
- Structured tasks: check deterministic constraints.

Policy Gradient Uses Sampled Rewards

For one prompt x, sample responses

$$a_1, \ldots, a_n \sim \pi_\theta(\cdot|x), \qquad r_j = r^\star(x, a_j).$$

Policy Gradient Uses Sampled Rewards

For one prompt x, sample responses

$$a_1, \ldots, a_n \sim \pi_\theta(\cdot|x), \qquad r_j = r^\star(x, a_j).$$

The policy-gradient identity gives

$$\nabla_\theta p_\theta(x) = \mathbb{E}_{a\sim\pi_\theta(\cdot|x)} \left[r^\star(x, a) \nabla_\theta \log \pi_\theta(a|x) \right].$$

Policy Gradient Uses Sampled Rewards

For one prompt x, sample responses

$$a_1, \ldots, a_n \sim \pi_\theta(\cdot|x), \qquad r_j = r^\star(x, a_j).$$

The policy-gradient identity gives

$$\nabla_\theta p_\theta(x) = \mathbb{E}_{a\sim\pi_\theta(\cdot|x)} \left[r^\star(x, a) \nabla_\theta \log \pi_\theta(a|x) \right].$$

In practice we use a baseline $b(x)$ to reduce variance:

$$\hat{g}(x) = \frac{1}{n} \sum_{j=1}^{n} \big(r_j - b(x)\big) \nabla_\theta \log \pi_\theta(a_j|x).$$

The sampling question is: which prompts should receive more draws n before we form this estimator?

The Statistical Bottleneck

The policy-gradient signal is estimated from sampled responses:

$$g(x, a) = (r^{\star}(x, a) - b(x)) \nabla_{\theta} \log \pi_{\theta}(a|x).$$

The Statistical Bottleneck

The policy-gradient signal is estimated from sampled responses:

$$g(x, a) = (r^\star(x, a) - b(x)) \nabla_\theta \log \pi_\theta(a|x).$$

With a fixed small group size n:

- hard prompts often produce all-zero rewards;
- easy prompts often produce all-one rewards;
- then the sampled group carries little or no contrastive signal.

The Statistical Bottleneck

The policy-gradient signal is estimated from sampled responses:

$$g(x, a) = (r^\star(x, a) - b(x)) \nabla_\theta \log \pi_\theta(a|x).$$

With a fixed small group size n:

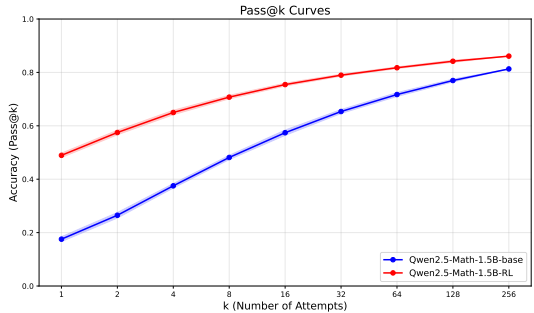
- hard prompts often produce all-zero rewards;
- easy prompts often produce all-one rewards;
- then the sampled group carries little or no contrastive signal.

If $p_\theta(x) = 0.01$ and $n = 8$, then

$$\Pr[\text{all samples wrong}] = (1 - p)^n = 0.99^8 \approx 92\%.$$

The prompt may be learnable, but the observed group carries no gradient.

Signal Loss is Often Undersampling



- Larger k reveals many prompts are not impossible.
- Small fixed groups hide valid learning signals.
- Uniform large-n sampling recovers signal, but spends too much inference on every prompt.

Inference Budget Allocation

How should we allocate a fixed inference budget across prompts?

For m prompts $x_1, \ldots, x_m$, uniform sampling answers:

$$n_1 = n_2 = \cdots = n_m.$$

Variance-aware sampling asks instead:

$$\min_{\{n_i\}_{i=1}^m} \mathbb{V}[\widehat{\nabla J}] \quad \text{s.t.} \quad \sum_{i=1}^m n_i = N.$$

This is the common statistical perspective behind both papers.

Reasoning Training as Latent-Variable Learning

J. Yao, Y. Hao, H. Zhang, H. Dong, W. Xiong, N. Jiang, and T. Zhang (2025). "Optimizing Chain-of-Thought Reasoners via Gradient Variance Minimization in Rejection Sampling and RL". *Neurips*

Statistical view: given prompt x, a sampled solution is $a = (a_{\text{think}}, a_{\text{ans}})$. We observe the final answer a_{ans}, while the thinking tokens a_{think} are latent.

$$\mathcal{L}(\theta) = -\mathbb{E}_{x\sim d_0} \log \mathbb{P}_\theta(a_{\text{ans}}|x) = -\mathbb{E}_{x\sim d_0} \log \sum_{a_{\text{think}}} \mathbb{P}_\theta(a_{\text{think}}|x)\, \mathbb{P}_\theta(a_{\text{ans}}|x, a_{\text{think}}).$$

- The true objective is answer likelihood, not likelihood of an arbitrary sampled trace.
- Introducing latent a_{think} decomposes generation into thinking tokens and final answer tokens.
- EM replaces the intractable marginal likelihood with an upper bound defined by a posterior $Q_i(a_{\text{think}})$.

EM Selects Verifier-Passing Reasoning Traces

At iteration t, the E-step targets the posterior over thinking traces that explain the known correct answer $a_{i,\text{ans}}$.

$$Q_i^t(a_{\text{think}}) = \mathbb{P}(a_{\text{think}}|x_i, a_{i,\text{ans}}, \theta_{t-1}) = \frac{\mathbb{P}_{\theta_{t-1}}(a_{\text{think}}|x_i)\,\mathbb{P}_{\theta_{t-1}}(a_{i,\text{ans}}|x_i, a_{\text{think}})}{\sum_{a'_{\text{think}}} \mathbb{P}_{\theta_{t-1}}(a'_{\text{think}}|x_i)\,\mathbb{P}_{\theta_{t-1}}(a_{i,\text{ans}}|x_i, a'_{\text{think}})}.$$

$$\mathcal{J}_{Q^t}(\theta) = -\frac{1}{m}\sum_{i=1}^{m} \mathbb{E}_{a_{\text{think}} \sim Q_i^t} \log \mathbb{P}_\theta(a_{\text{think}}, a_{i,\text{ans}}|x_i).$$

- Exact Q_i^t is intractable because it sums over all rationales.
- Rejection sampling approximates it by keeping generations with the correct final answer.
- DIBS changes the sampling budget per prompt used in this approximation.

Optimal Allocation

For prompt-answer pair $(x_i, a_{i,\mathsf{ans}})$ at iteration t, define

$$p_i^t = \mathbb{E}_{a_{\mathsf{think}} \sim \mathbb{P}_{\theta_{t-1}}(\cdot|x_i)} \mathbb{P}_{\theta_{t-1}}(a_{i,\mathsf{ans}}|x_i, a_{\mathsf{think}}), \qquad G_i^2 = \mathbb{E}_{a_{\mathsf{think}} \sim Q_i^t} \left\| \nabla_\theta \log \mathbb{P}_\theta(a_{\mathsf{think}}, a_{i,\mathsf{ans}}|x_i) \right\|^2 .$$

Optimal Allocation

For prompt-answer pair $(x_i, a_{i,\mathsf{ans}})$ at iteration t, define

$$p_i^t = \mathbb{E}_{a_{\mathsf{think}} \sim \mathbb{P}_{\theta_{t-1}}(\cdot|x_i)} \mathbb{P}_{\theta_{t-1}}(a_{i,\mathsf{ans}}|x_i, a_{\mathsf{think}}), \qquad G_i^2 = \mathbb{E}_{a_{\mathsf{think}} \sim Q_i^t} \left\| \nabla_\theta \log \mathbb{P}_\theta(a_{\mathsf{think}}, a_{i,\mathsf{ans}}|x_i) \right\|^2 .$$

With fixed inference budget $\sum_{i=1}^m n_i^t = N$, minimize the variance bound

$$\min_{\{n_i^t\}_{i=1}^m} \sum_{i=1}^m \frac{G_i^2}{p_i^t n_i^t} \qquad \text{s.t.} \quad \sum_{i=1}^m n_i^t = N.$$

Optimal Allocation

For prompt-answer pair $(x_i, a_{i,\text{ans}})$ at iteration t, define

$$p_i^t = \mathbb{E}_{a_{\text{think}} \sim \mathbb{P}_{\theta_{t-1}}(\cdot|x_i)} \mathbb{P}_{\theta_{t-1}}(a_{i,\text{ans}}|x_i, a_{\text{think}}), \qquad G_i^2 = \mathbb{E}_{a_{\text{think}} \sim Q_i^t} \left\| \nabla_\theta \log \mathbb{P}_\theta(a_{\text{think}}, a_{i,\text{ans}}|x_i) \right\|^2 .$$

With fixed inference budget $\sum_{i=1}^m n_i^t = N$, minimize the variance bound

$$\min_{\{n_i^t\}_{i=1}^m} \sum_{i=1}^m \frac{G_i^2}{p_i^t n_i^t} \qquad \text{s.t.} \quad \sum_{i=1}^m n_i^t = N.$$

Solving the constrained allocation problem gives the rule

$$\boxed{n_i^t \propto \frac{G_i}{\sqrt{p_i^t + \alpha/(p_i^t)^{\beta-1}}}}$$

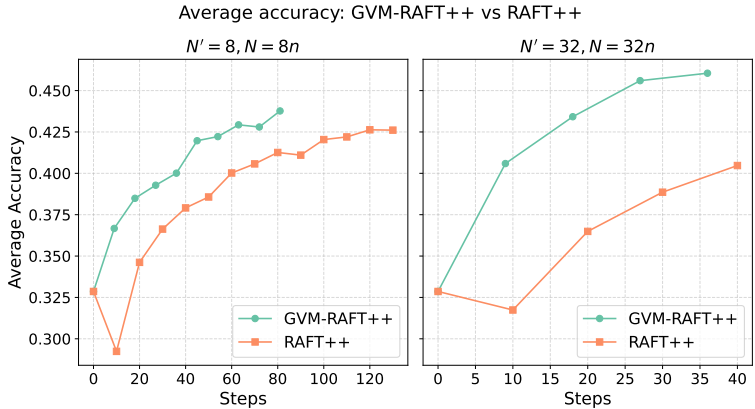
Interpreting the Allocation Rule

$$n_i^t \propto \frac{G_i}{\sqrt{p_i^t + \alpha/(p_i^t)^{\beta-1}}}$$

Component	Effect on n_i^t	Interpretation
Low p_i^t	more budget	hard but solvable prompts need more draws to observe accepted traces
Large G_i	more budget	accepted traces have larger gradient impact
Extremely low p_i^t	less budget after regularization	avoid spending all samples on nearly impossible prompts

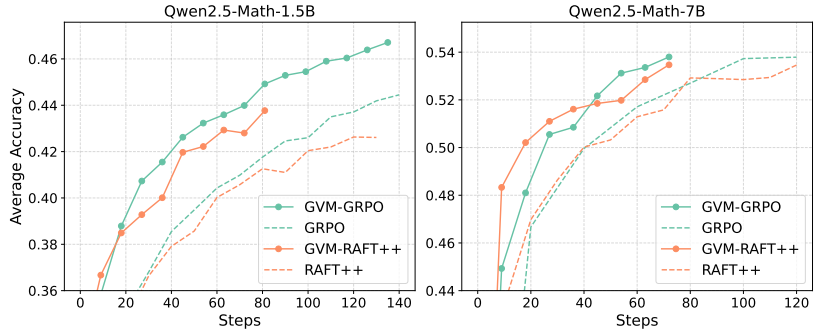
DIBS is not just "sample hard prompts more." It samples where accepted reasoning traces are both hard to find and useful for the update.

DIBS Results: Faster Learning



- On Qwen2.5-Math-1.5B, DIBS-RAFT++ improves average accuracy across Math500, Minerva Math, and OlympiadBench.
- Reported speedups are about $2\times$ to $4\times$ in update steps, depending on the sampling budget.

DIBS Transfers Beyond RAFT



- The derivation starts from EM / rejection sampling.
- Empirically, the same budget rebalancing also helps GRPO.
- This motivates a broader online RL formulation.

Adaptive Sampling for Online Reinforce

W. Xiong, C. Ye, B. Liao, H. Dong, X. Xu, C. Monz, J. Bian, N. Jiang, and T. Zhang (2025). *Reinforce-Ada: An Adaptive Sampling Framework under Non-linear RL Objectives*.arXiv:arXiv: 2510.04996 (cs.LG)

REINFORCE-ADA asks how to make adaptive sampling native to online Reinforce-style RL, rather than only an offline RAFT allocation step.

Adaptive Sampling for Online Reinforce

W. Xiong, C. Ye, B. Liao, H. Dong, X. Xu, C. Monz, J. Bian, N. Jiang, and T. Zhang (2025). *Reinforce-Ada: An Adaptive Sampling Framework under Non-linear RL Objectives*.arXiv:arXiv: 2510.04996 (cs.LG)

REINFORCE-ADA asks how to make adaptive sampling native to online Reinforce-style RL, rather than only an offline RAFT allocation step.

Question: is the log pass-rate objective the right objective, or just a useful way to reveal how sampling should change?

$$J_{\log}(\theta) = \mathbb{E}_x[\log p_\theta(x)], \qquad \nabla J_{\log}(\theta) = \mathbb{E}_x\left[\frac{1}{p_\theta(x)} \nabla p_\theta(x)\right].$$

- The $1/p_\theta(x)$ factor gives larger marginal value to hard prompts.
- This is useful for signal recovery, but can over-focus on very low-pass rate prompts if implemented too aggressively.
- REINFORCE-ADA studies adaptive sampling as a practical implementation of this hard-prompt weighting.

Non-Linear Objectives Choose Prompt Weights

Standard RLVR objective:

$$J(\theta) = \mathbb{E}_x[p_\theta(x)].$$

It values one point of improvement on an easy prompt the same as one point on a difficult prompt.

Non-Linear Objectives Choose Prompt Weights

Standard RLVR objective:

$$J(\theta) = \mathbb{E}_x[p_\theta(x)].$$

It values one point of improvement on an easy prompt the same as one point on a difficult prompt.

Consider instead

$$J_f(\theta) = \mathbb{E}_x[f(p_\theta(x))],$$

where f is increasing and concave.

Non-Linear Objectives Choose Prompt Weights

Standard RLVR objective:

$$J(\theta) = \mathbb{E}_x[p_\theta(x)].$$

It values one point of improvement on an easy prompt the same as one point on a difficult prompt.

Consider instead

$$J_f(\theta) = \mathbb{E}_x[f(p_\theta(x))],$$

where f is increasing and concave.

Then

$$\nabla J_f(\theta) = \mathbb{E}_x\left[f'(p_\theta(x))\,\nabla p_\theta(x)\right].$$

The choice of f is a design choice. Concave choices favor hard prompts because $f'(p)$ is larger when the pass rate p is small.

GRPO Matches an Arcsine Objective

GRPO is a common online RL baseline for reasoning models, introduced in **Shao et al.**, *DeepSeekMath*, arXiv 2024.

For binary rewards, write $p = p_\theta(x)$ and approximate the group mean and standard deviation by their population values:

$$A_{\mathrm{GRPO}} \approx \frac{r - p}{\sqrt{p(1-p)}}.$$

GRPO Matches an Arcsine Objective

GRPO is a common online RL baseline for reasoning models, introduced in **Shao et al.**, *DeepSeekMath*, arXiv 2024.

For binary rewards, write $p = p_\theta(x)$ and approximate the group mean and standard deviation by their population values:

$$A_{\mathrm{GRPO}} \approx \frac{r - p}{\sqrt{p(1-p)}}.$$

This is the policy-gradient weight induced by

$$f(p) = 2 \arcsin \sqrt{p}, \qquad f'(p) = \frac{1}{\sqrt{p(1-p)}}.$$

- GRPO normalizes rewards by within-group standard deviation.
- The arcsine view explains this as a non-linear objective over pass rate.
- REINFORCE-ADA keeps the update form but changes how the training group is sampled.

REINFORCE-ADA Allocates Samples by Pass Rate

For a non-linear objective $J_f = \mathbb{E}_x[f(p_\theta(x))]$, the Reinforce estimator uses prompt weight $w_i = f'(p_i)$:

$$\widehat{g} = \frac{1}{B} \sum_{i=1}^{B} \frac{w_i}{n_i} \sum_{j=1}^{n_i} \nabla_\theta \log \pi_\theta(a_{ij}|x_i) \, (r_{ij} - p_i).$$

REINFORCE-ADA Allocates Samples by Pass Rate

For a non-linear objective $J_f = \mathbb{E}_x[f(p_\theta(x))]$, the Reinforce estimator uses prompt weight $w_i = f'(p_i)$:

$$\widehat{g} = \frac{1}{B} \sum_{i=1}^{B} \frac{w_i}{n_i} \sum_{j=1}^{n_i} \nabla_\theta \log \pi_\theta(a_{ij}|x_i)\,(r_{ij} - p_i).$$

With fixed inference budget $\sum_i n_i = N$, variance minimization gives

$$n_i^\star \propto |f'(p_i)|\,\sigma_g(x_i), \qquad \sigma_g^2(x_i) \propto p_i(1-p_i).$$

REINFORCE-ADA Allocates Samples by Pass Rate

For a non-linear objective $J_f = \mathbb{E}_x[f(p_\theta(x))]$, the Reinforce estimator uses prompt weight $w_i = f'(p_i)$:

$$\widehat{g} = \frac{1}{B} \sum_{i=1}^{B} \frac{w_i}{n_i} \sum_{j=1}^{n_i} \nabla_\theta \log \pi_\theta(a_{ij}|x_i)\,(r_{ij} - p_i).$$

With fixed inference budget $\sum_i n_i = N$, variance minimization gives

$$n_i^\star \propto |f'(p_i)|\,\sigma_g(x_i), \qquad \sigma_g^2(x_i) \propto p_i(1-p_i).$$

Objective $f(p)$	Weight $f'(p)$	Optimal allocation $n_i^\star$
p	1	$\propto \sqrt{p_i(1-p_i)}$
$\log p$	$1/p_i$	$\propto \sqrt{(1-p_i)/p_i}$
$2\arcsin\sqrt{p}$	$1/\sqrt{p_i(1-p_i)}$	$\propto 1$
$\sqrt{p}$	$1/(2\sqrt{p_i})$	$\propto \sqrt{1-p_i}$

Log is most aggressive for hard prompts; arcsine/GRPO normalizes this variance.

REINFORCE-ADA: Nonlinear Objectives, Practical Sampling

REINFORCE-ADA is a framework for nonlinear pass-rate objectives:

$$J_f(\theta) = \mathbb{E}_x[f(p_\theta(x))], \qquad \nabla J_f(\theta) = \mathbb{E}_x\left[f'(p_\theta(x))\,\nabla p_\theta(x)\right].$$

The choice of f determines the prompt-level weighting.

REINFORCE-ADA: Nonlinear Objectives, Practical Sampling

REINFORCE-ADA is a framework for nonlinear pass-rate objectives:

$$J_f(\theta) = \mathbb{E}_x[f(p_\theta(x))], \qquad \nabla J_f(\theta) = \mathbb{E}_x\left[f'(p_\theta(x))\,\nabla p_\theta(x)\right].$$

The choice of f determines the prompt-level weighting.

Adaptive sampling

$$n_i^\star \propto |f'(p_i)|\sqrt{p_i(1-p_i)}$$

implements part of the weight by spending more rollouts where a fixed small group is unlikely to reveal signal.

sample until an acceptance rule is met.

Static training batch

- Positive rule: stop after enough accepted responses.
- Balanced rule: stop after enough correct and incorrect responses.
- Downsample the response pool to a fixed update group.

REINFORCE-ADA: Nonlinear Objectives, Practical Sampling

REINFORCE-ADA is a framework for nonlinear pass-rate objectives:

$$J_f(\theta) = \mathbb{E}_x[f(p_\theta(x))], \qquad \nabla J_f(\theta) = \mathbb{E}_x\left[f'(p_\theta(x))\,\nabla p_\theta(x)\right].$$

The choice of f determines the prompt-level weighting.

Adaptive sampling

$$n_i^\star \propto |f'(p_i)|\sqrt{p_i(1-p_i)}$$

implements part of the weight by spending more rollouts where a fixed small group is unlikely to reveal signal.

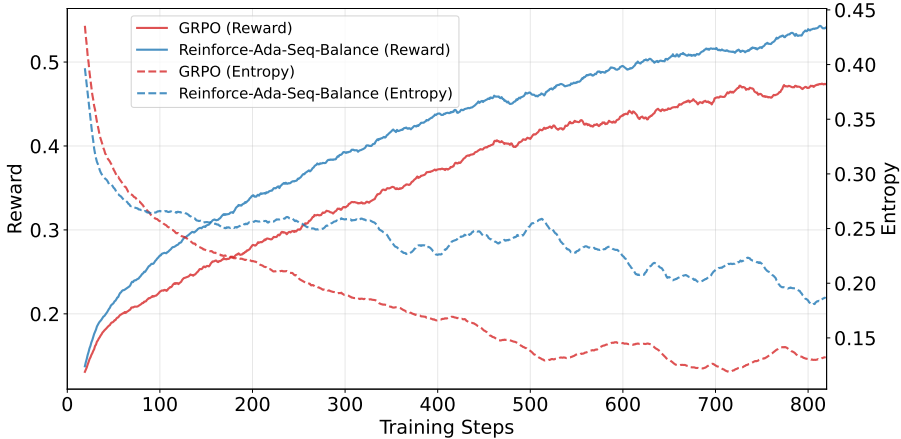
sample until an acceptance rule is met.

Static training batch

- Positive rule: stop after enough accepted responses.
- Balanced rule: stop after enough correct and incorrect responses.
- Downsample the response pool to a fixed update group.

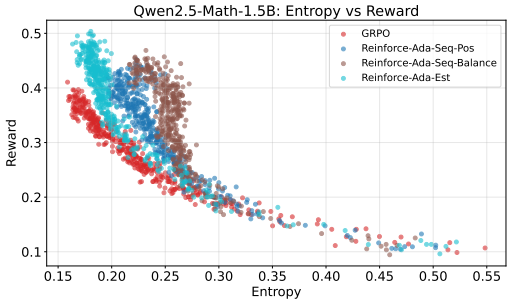
For example, sampling until one accepted response gives $\mathbb{E}[N_i] \approx 1/p_i$. This realizes hard-prompt weighting through generation while keeping the backward batch size controlled.

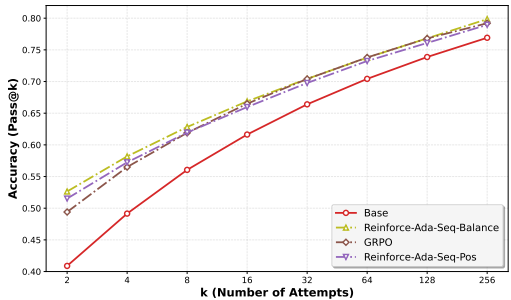
REINFORCE-ADA Experiments



- Adaptive sampling improves training reward without changing the core RL update.

Entropy and Pass@k





- Adaptive sampling shifts the reward–entropy frontier outward.
- Better top-1 quality does not require immediate entropy collapse.

Summary

- RLVR for reasoning is a Monte Carlo estimation problem with sparse binary rewards.

Summary

- RLVR for reasoning is a Monte Carlo estimation problem with sparse binary rewards.
- Uniform sampling wastes compute on easy prompts and misses signal on hard prompts.

Summary

- RLVR for reasoning is a Monte Carlo estimation problem with sparse binary rewards.
- Uniform sampling wastes compute on easy prompts and misses signal on hard prompts.
- DIBS reduces gradient variance by allocating inference budget using acceptance rates and gradient norms using latent variable maximum-likelihood.

Summary

- RLVR for reasoning is a Monte Carlo estimation problem with sparse binary rewards.
- Uniform sampling wastes compute on easy prompts and misses signal on hard prompts.
- DIBS reduces gradient variance by allocating inference budget using acceptance rates and gradient norms using latent variable maximum-likelihood.
- REINFORCE-ADA turns the same idea into an online RL data-generation primitive for general nonlinear functions of pass rate.

Summary

- RLVR for reasoning is a Monte Carlo estimation problem with sparse binary rewards.
- Uniform sampling wastes compute on easy prompts and misses signal on hard prompts.
- DIBS reduces gradient variance by allocating inference budget using acceptance rates and gradient norms using latent variable maximum-likelihood.
- REINFORCE-ADA turns the same idea into an online RL data-generation primitive for general nonlinear functions of pass rate.
- Adaptive sampling gives stronger reasoning models under a fixed or moderately increased compute budget.

Summary

- RLVR for reasoning is a Monte Carlo estimation problem with sparse binary rewards.
- Uniform sampling wastes compute on easy prompts and misses signal on hard prompts.
- DIBS reduces gradient variance by allocating inference budget using acceptance rates and gradient norms using latent variable maximum-likelihood.
- REINFORCE-ADA turns the same idea into an online RL data-generation primitive for general nonlinear functions of pass rate.
- Adaptive sampling gives stronger reasoning models under a fixed or moderately increased compute budget.

Questions?