Reinforcement Learning for LLMs

UNIBA Lecture

Michal Gregor

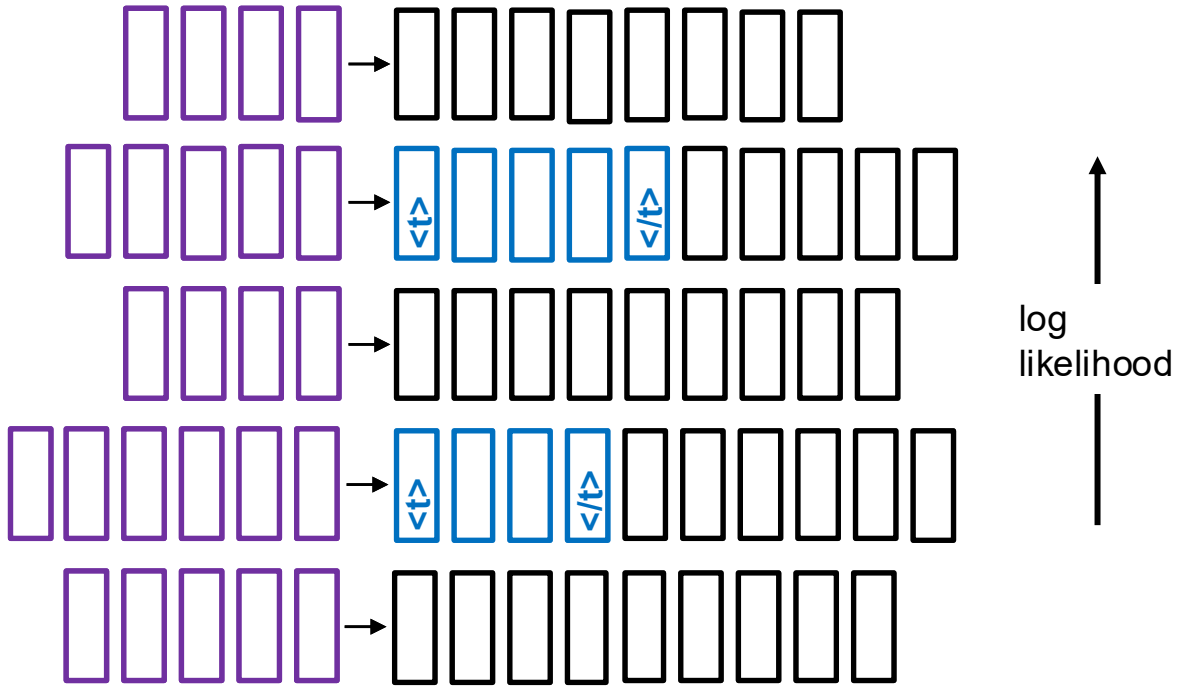
Researcher – Expert

michal.gregor@kinit.sk | www.kinit.sk

https://michalgregor.gitlab.io | LinkedIn

SFT vs. Reinforcement Learning in Post-Training

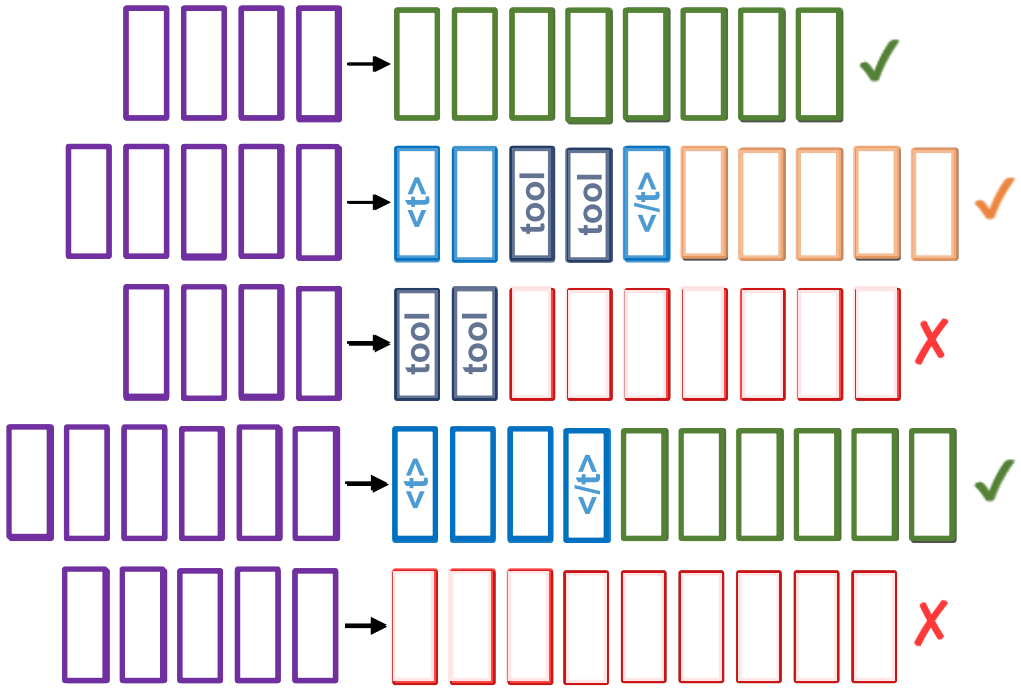
- You get a batch of inputs annotated with desired outputs;
- You increase the log likelihood of everything you uniformly;



Supervised Fine-Tuning (SFT)

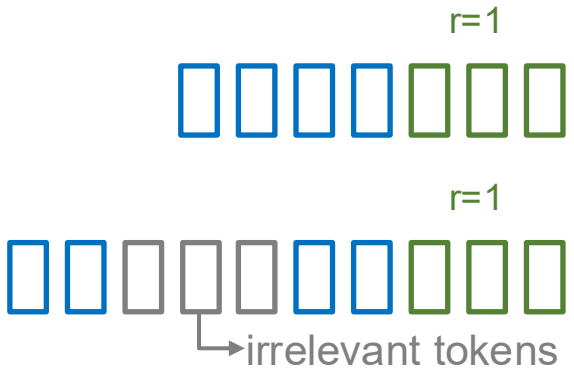
SFT vs. Reinforcement Learning in Post-Training

- You get a batch of inputs;
- You **generate** outputs using your LLM;
- You compute **rewards**;
- **Reinforce** based on advantages;
  - How much > than average;



Reinforcement Learning (RL)

RL: The Rewards Are the Signal

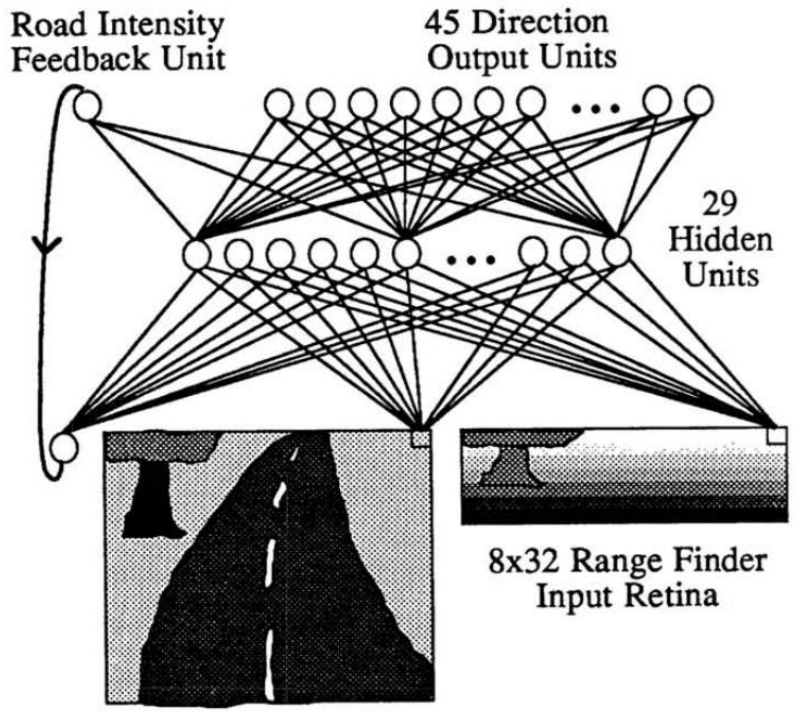


- In RL, the reward gives you a signal;
- You can figure out e.g. that:
  - Some chunks are not task-relevant;
  - Some improve/harm performance;

- Supervised: only on “near-optimal” sequence; no rewards;
  - **Supervised:** The net is **not trained to** produce harmful/irrelevant chunks;
  - **RL:** The net is **trained not to** produce them;

The Pitfalls of Imitation Learning

- The student only as good as the teacher;
- And actually worse:
  - **Compounding errors** can steer the model out of data seen during training;
  - Early example: the ALVINN [1] self-driving experiment, 1989;



[1]

Figure 1: ALVINN Architecture

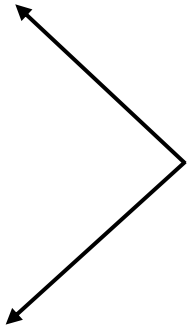
For Truly Competent LLM Agents, RL is a Must

- **SFT:** Supervised Fine-Tuning;
  - Very easy to use and stable;
  - Imitation-based, i.e.:
    - Requires **supervised data**;
    - Performance won't exceed that of the teacher;
  - Encourages hallucination [1,2];
  - Everything needs to be differentiable;

- **RL:** Reinforcement Learning;
  - Can learn new behaviours;
    - Not bounded by the teacher;
  - Does not encourage hallucination;
    - Or at least RLHF does not [2];
  - Things **don't need to be differentiable** = search, tool use, …
  - Typically much more difficult to use and stabilize;
  - Requires a **reward function**!

RL Has Many LLM Applications

- **Alignment:**
  - RLHF: [2009.01325] Learning to summarize from human feedback;
  - InstructGPT: [2203.02155] Training language models to follow instructions with human feedback; ...
- **Search, deep research, tool use, agentic AI, …**
  - Optimizing query generation: [2503.00223v3] DeepRetrieval: Hacking Real Search Engines and Retrievers with Large Language Models via Reinforcement Learning;
  - ReSearch: [2503.19470] ReSearch: Learning to Reason with Search for LLMs via Reinforcement Learning; …
- **Bootstrapping reasoning;**
  - DeepSeek-R1: [2501.12948] DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning;
  - Quiet STaR: [2403.09629] Quiet-STaR: Language Models Can Teach Themselves to Think Before Speaking (arxiv.org); …



building competence

Ways to Get the Rewards

Ways to Get Rewards in RL for LLMs: Overview

- **RLHF:** Reinforcement Learning from Human Feedback [1,2];
  - Collect human preference data;
  - Train a reward model;
  - For additional context: [3,4];
- **RLVR:** Reinforcement Learning with Verifiable Rewards [5];
  - Train on tasks where correctness can be verified automatically;
  - E.g. questions with simple answers;
  - Optionally w/ a model to match free-form answers against reference answers;
- **RPT:** Reinforcement Pretraining;
  - Uses unsupervised data;
  - Rewards based on:
    - Prefix matching [6];
    - ↑ log probability of ground truth [7];
- **GenRMs:** Generative Rewards Models;
  - More widely applicable;
  - Prone to reward hacking, instability;
  - LLM as a judge [8,9]; self-rewarding [10,11];
  - Reflective rewards (reasoning) [12,13];
  - Rubric-based rewards [14,15];
- **Generalist reward functions:** next slide;

An LLM is a Generalist Reward Function

- After standard next-token prediction pretraining (and SFT) [1]:
  - The **logits** of the LLM can be interpreted as a soft Q-function = **values**;
  - **Rewards** can be recovered using an inverse soft Bellmann operator [1];
    - One of the ways to actually get **dense rewards**;
- Learning from these rewards:
  - Induces a **provably superior error bound** compared to the base model;
  - Outperforms LLM-as-Judge **and** explicitly trained reward models;

equivalence to a form of soft **inverse** reinforcement learning!

[1] Li, Y.C., Xu, T., Yu, Y., Zhang, X., Chen, X.H., Ling, Z., Chao, N., Yuan, L. and Zhou, Z.H., 2025. Generalist Reward Models: Found Inside Large Language Models. *arXiv preprint arXiv:2506.23235*.

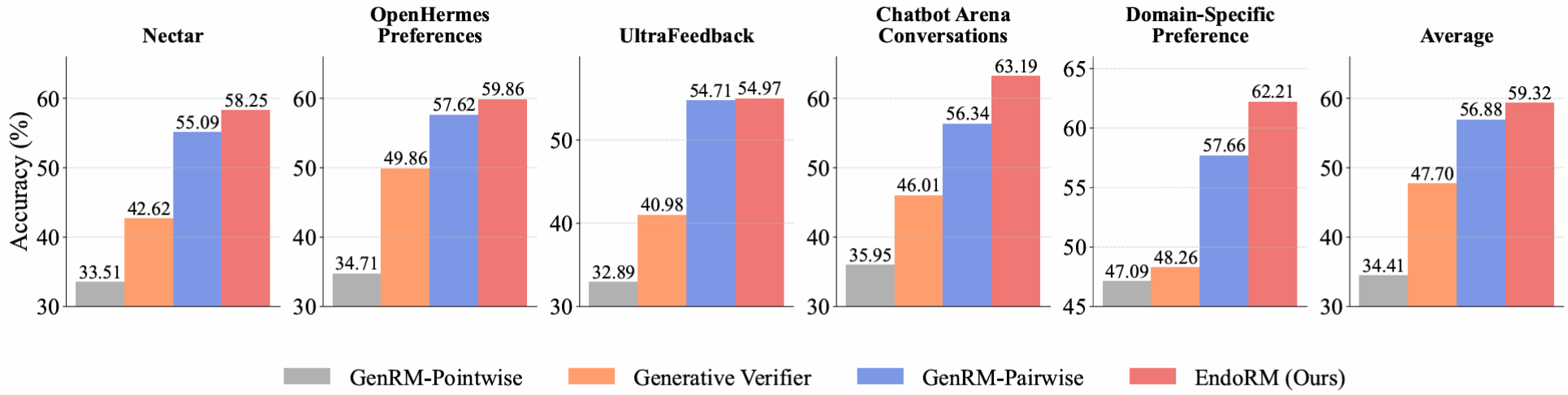


Figure 1: Response classification accuracy on the Multifacted-Bench.

RL: A Quick Recap

(Policy Gradient) RL: Just Weighted MLE 🤷

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\rho^{\pi_\theta}, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\, Q^{\pi_\theta}(s,a)]$$

this is just **MLE**

this is how **good** we think the action is in the long run = its **value**

but how do we get this?

$$\mathcal{L}_{MLE}(\pi_\theta) = \mathbb{E}_{(s,a)\sim\mathcal{D}}[\nabla_\theta \log \pi_\theta(s,a)]$$

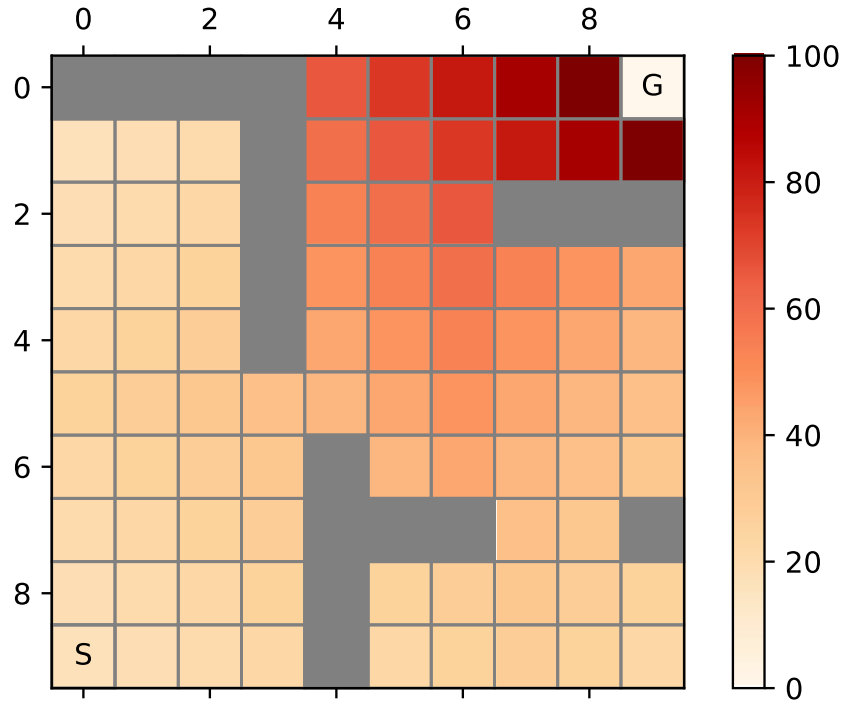
a fixed dataset

What Are Values?

So again, how do we get values?

- The **state-value** function [1]:

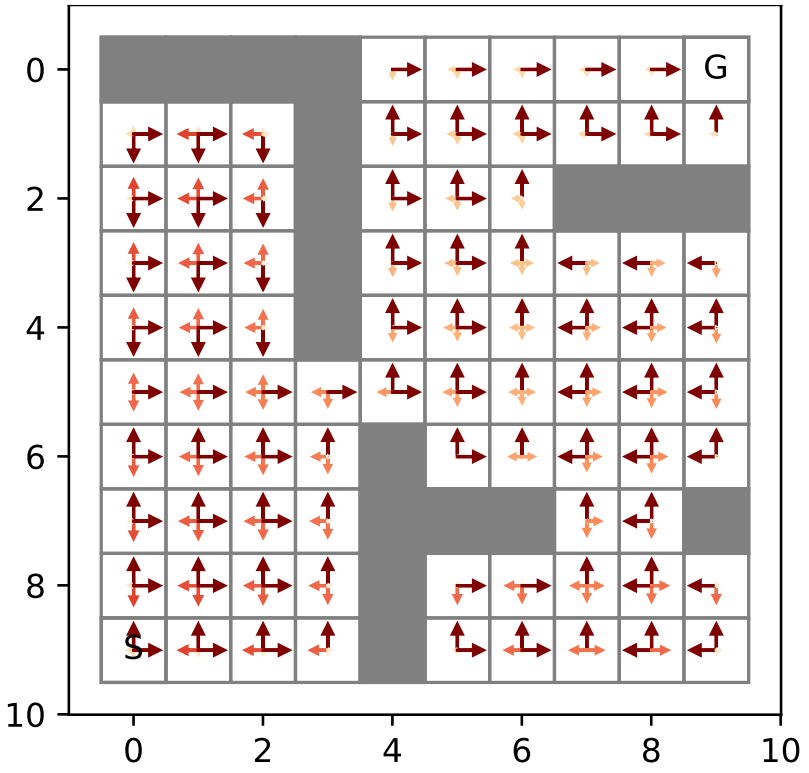
$$V^{\pi}(s) = \mathbb{E}_{\pi}\{R_t | s_t = s\}$$



The state-value function $V(s)$ for a maze (S is the starting point and G is the goal)

- The **action-value** function [1]:

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\{R_t | s_t = s, a_t = a\}$$



The action-value function $Q(s, a)$ for a maze (S: the starting point; G: the goal)

We'll use the **discounted return**:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+1+k}$$

- $\gamma \in [0,1]$: the discount factor;

REINFORCE: Use the Sample Return

- Run the episode until its end and use the **sample return**:

$$Q^{\pi}(s,a) \approx R_t$$

  - I.e. we approximate the expected value using a single sample (or potentially multiple samples) – Monte Carlo estimate;

- **Pro:** very easy to implement;
- **Con:** very **high-variance**;
  - The sample return **accumulates the stochasticity** of all steps in the episode;
  - Think e.g. of all that an LLM can generate after the first token in a sequence;

Actor-Critic Approaches

- **Policy-based** methods:
  - They represent and train **only the policy**;
  - Like REINFORCE; high variance;
- **Actor-critic** methods:
  - Also explicitly represent the **value function**;
  - Typically actually work with **advantages**;
  - Lower variance, but higher bias;
- In standard RL actor-critic is the norm;

Advantage:

- How much better an action is **than average**;

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$$

  - Can be negative = subpar actions actively suppressed;

The Recursive Nature of Returns / Value Functions

- Return R_t can be decomposed into:
  - What happens right now; immediate reward r_{t+1};
  - What happens afterwards: γR_{t+1};
    - Note again the discount factor γ;
    - "A bird in the hand is worth two in the bush";

$$\overbrace{r_{t+1} + \underbrace{\gamma(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} + \cdots)}_{\gamma R_{t+1}}}^{R_t}$$

- And you can do the same with the value function:

$$V(s_t) \approx r_{t+1} + \gamma V^{\pi}(s_{t+1})$$

  - Sample estimate over one particular state transition;
  - Recall: in general, $V(s_t)$ is an expected value;

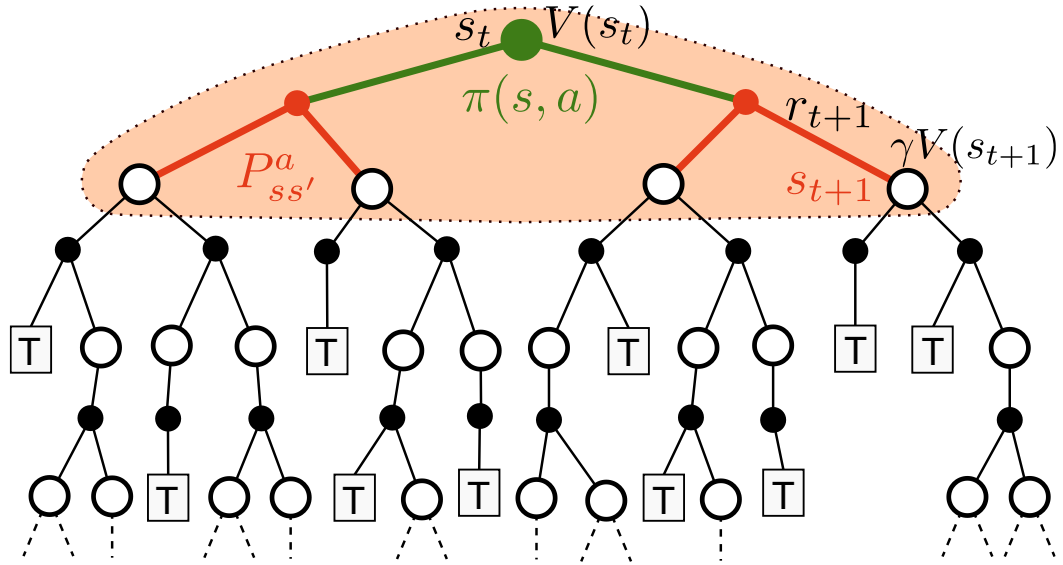
The Need for Bootstrapping

Approximation: $V(s_t) \approx r_{t+1} + \gamma V^{\pi}(s_{t+1})$

- The Bellman expectation equation [1]:

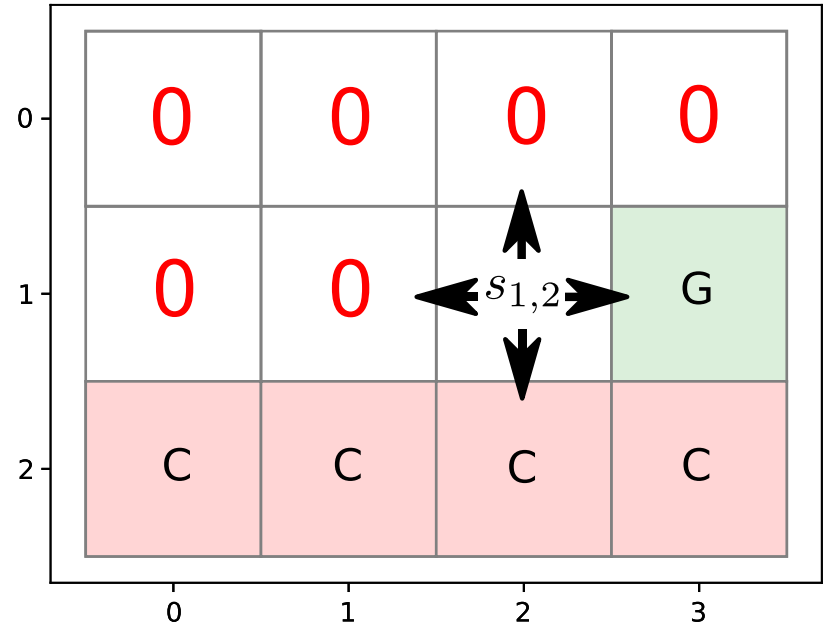
$$V^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(s,a) \sum_{s' \in \mathcal{S}} P^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s') \right]$$

  - How to express $V^{\pi}(s)$ in terms of the values of **surrounding states**;



The Bellman backup (inspired by [2])

- But we don't really have values of the surrounding states!
- We need to **bootstrap;**
  - I.e. initialize somehow, update iteratively;
  - Theoretically converges to true values;
  - Biased in practice (due to approximation);



GAE & Bias-Variance Trade-Off

- **PPO** (Proximal Policy Optimization) is a method that uses GAE;
- It uses **importance sampling** and **clipping**; we'll gloss over that;

- Recall the definition of **advantage**:

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$$

- Under some assumptions **temporal difference** δ_t can be taken to estimate it;
  - If we assume V^{π} is already perfect;
  - TD is also used to train value functions; 🤷

immediate reward
sample estimate

$$\delta_t = \underbrace{r_t + \gamma V^{\pi}(s_{t+1})}_{\text{approx. } Q^{\pi}(s,a)} - V^{\pi}(s_t)$$

- Generalized advantage estimation (GAE);
  - λ controls the bias-variance trade-off;
  - $\lambda = 0$: **full bootstrapping**;
    - Only a single transition is sample-estimated; lowest variance, greatest bias;
  - $\lambda = 1$: **full Monte Carlo** (if we have the full episode);
    - The target depends entirely on sample return R_t; not at all on the value function;
    - The value function terms cancel out;

$$\hat{A}_t^{GAE(\lambda)} = \sum_{i=0}^{T} (\gamma\lambda)^i \delta_{t+i}$$

$$= \delta_t + \gamma\lambda\delta_{t+1} + \cdots + (\gamma\lambda)^T \delta_{t+T}$$

RL for LLMs Is Different: Some Challenges

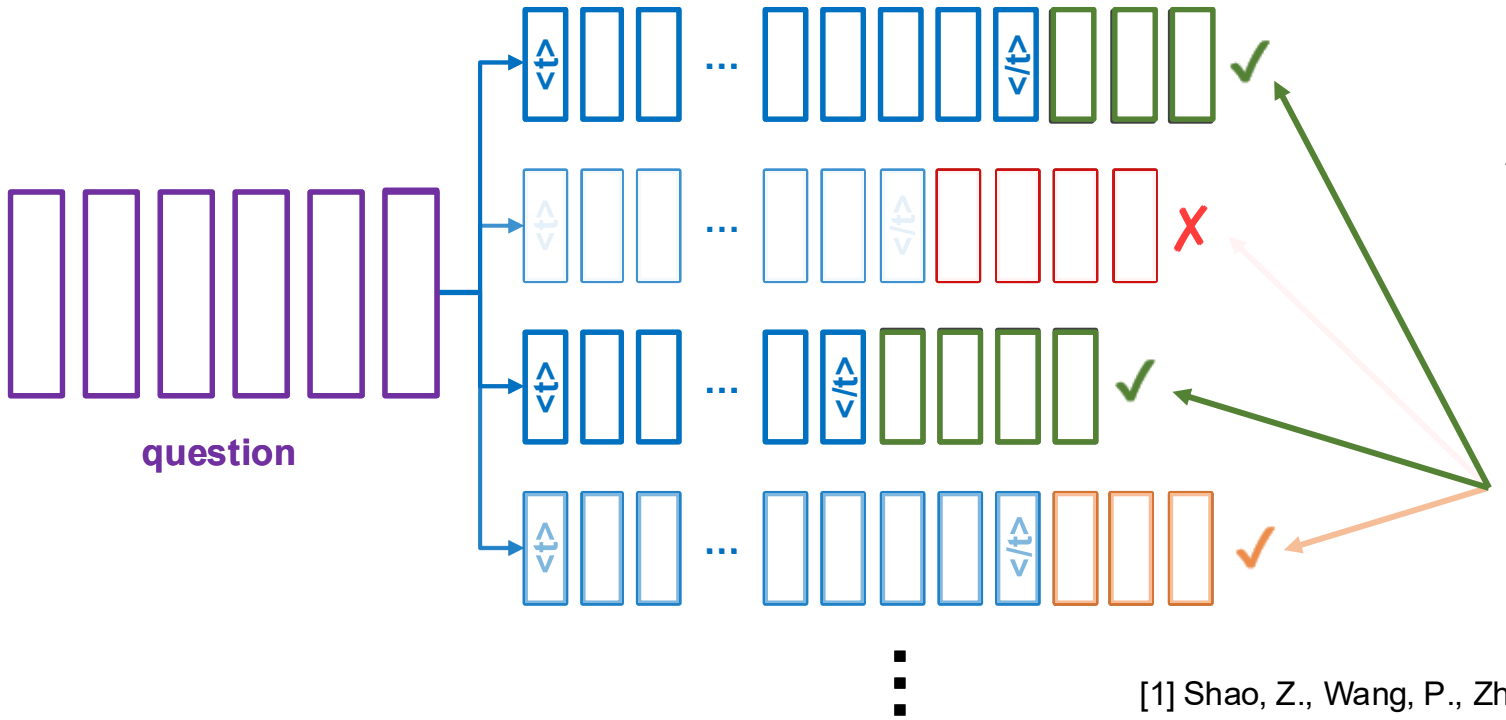
RL for LLMs Is Different: The Challenges

1. **Huge models:** LLMs are huge and expensive to run;
   - Difficult to represent both the **policy** and the **value function**;
   - Methods like SAC need even more networks + target networks;
2. **Huge spaces:** a huge action space and episode length;
   - Very long sequences that cannot be chunked (transformers);
   - A huge range of tokens to choose from at every step;
3. **Exploration:** meaningful exploration is hard to achieve;
   - Think exploring a new proof technique vs. sampling a random token now and then;
   - Very, very complex behaviours required;
4. **Bias:** models are heavily pretrained;
   - Challenge for actor-critic methods – adapting a base model into a value function, etc.
   - Variance seems to be less of an issue; but this may be because we are not good at exploring;
5. **Reward sparsity:** often just an episode-level reward;
   - Collapse on difficult tasks: extreme label imbalance (rewards almost always 0);
   - GRPO groups with 0 variance induce no gradients, …

GRPO-Style Methods: Let's Just Use Rollouts

GRPO: Use Sample Estimates [1,2]

1. Leverage the model's inherent reasoning capabilities; generate N different reasoning chains + answers;
   - Zero-shot CoT prompt: think about the reasoning process in your mind first, then answer…



2. **Reinforce** thoughts based on sample-estimated advantage;
   - Accuracy of answers;
   - Correct use of <think>;
   - Policy-based RL;

$$\hat{A}_i^{GRPO} = \frac{r_i - mean(\{r_1, r_2, \ldots, r_G\})}{std(\{r_1, r_2, \ldots, r_G\})}$$

reinforce thoughts in proportion to how good the answers are

[1] Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y.K., Wu, Y. and Guo, D., 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300.
[2] Daya Guo, et al., 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv preprint arXiv:2501.12948.

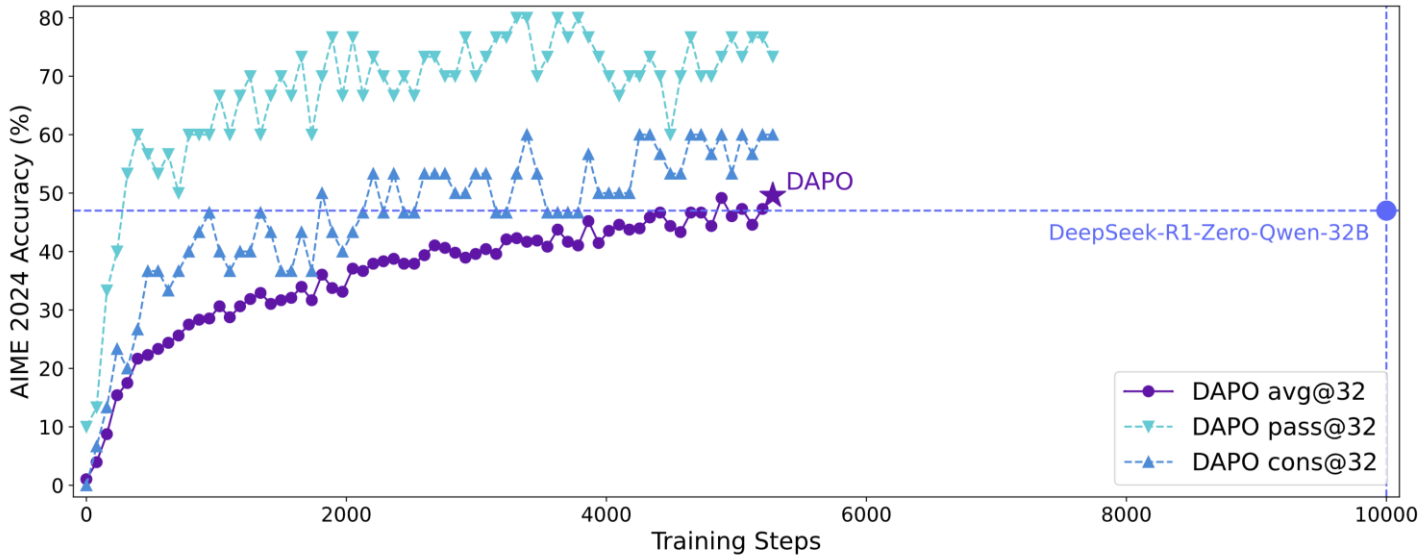
Vanilla GRPO Is Quite Unstable

- In practice, GRPO was found to be quite **unstable**;
- Difficult to reproduce DeepSeek-R1 results, etc.
- There is a number of works that extend it;
  - Improved stability;
  - Faster learning;
  - …
- We'll look at some…

DAPO: Decoupled Clip and Dynamic Sampling Policy Optimization Algorithm

[1] Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Dai, W., Fan, T., Liu, G., Liu, L. and Liu, X., 2025. Dapo: An open-source llm reinforcement learning system at scale. arXiv preprint arXiv:2503.14476.

- DAPO introduces 4 improvements:
  - **Clip-higher**: modify clipping to mitigate entropy collapse;
  - **Dynamic sampling:** oversampling rollouts to ensure non-zero gradients;
  - **Token-level policy gradient loss:** every *token* should have the same weight in the loss function, not every *sequence*;
  - **Overlong reward shaping:** handling rewards for truncated rollouts;

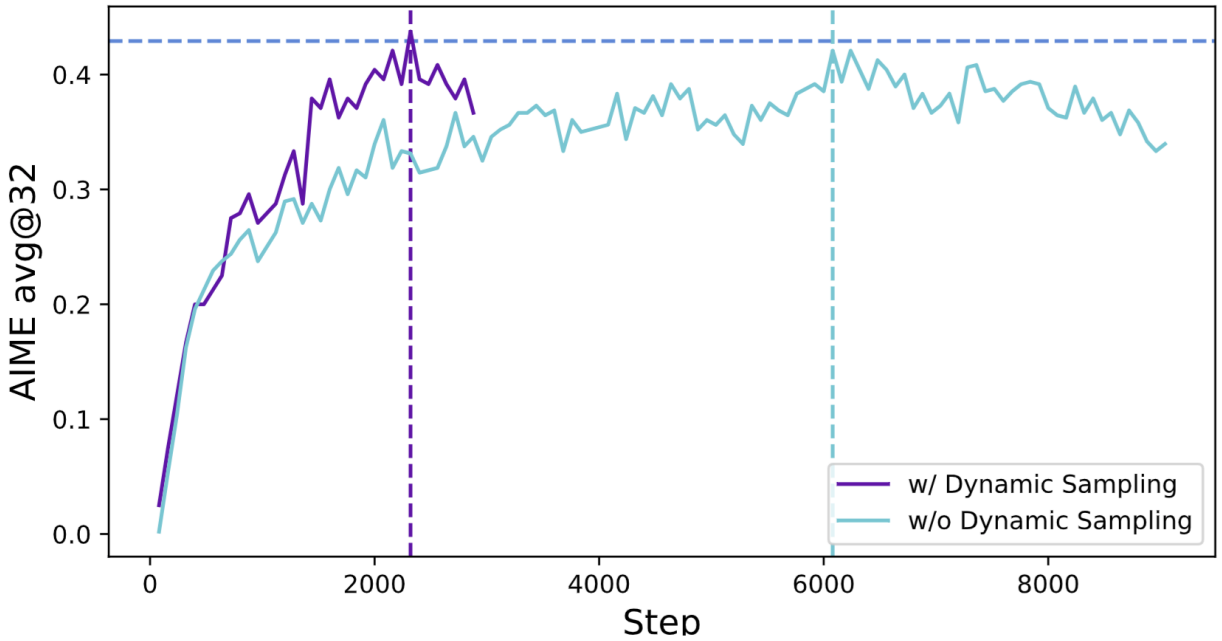


Note: DAPO also drops the KL term from the loss

Figure 1 AIME 2024 scores of **DAPO** on the Qwen2.5-32B base model, outperforming the previous SoTA DeepSeek-R1-Zero-Qwen-32B using 50% training steps. The x-axis represents the gradient update steps.

DAPO: Dynamic Sampling

- In GRPO-style methods, variance of rewards is important:
  - If all rollouts of a query have the same reward, there are **0 gradients**;
- DAPO:
  - Discard queries with zero-variance rewards;
  - Sample more queries to form a full batch;



[1] Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Dai, W., Fan, T., Liu, G., Liu, L. and Liu, X., 2025. Dapo: An open-source llm reinforcement learning system at scale. arXiv preprint arXiv:2503.14476.

DAPO: Token-Level Policy Gradient Loss

- GRPO: every sequence has the same weight in the loss function;
  - I.e. tokens in long sequences get less weight and are very difficult to correct;

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left(\min\left(r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\Big(r_{i,t}(\theta),1-\varepsilon,1+\varepsilon\Big)\hat{A}_{i,t}\right)-\beta D_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})\right)\right]$$

- DAPO: every token should have the same weight;
  - I.e. DAPO flattens the averaging: sum up first, then divide by the total number of tokens;

$$\mathcal{J}_{\text{DAPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{\sum_{i=1}^{G}|o_i|}\sum_{i=1}^{G}\sum_{t=1}^{|o_i|}\min\left(r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\Big(r_{i,t}(\theta),1-\varepsilon_{\text{low}},1+\varepsilon_{\text{high}}\Big)\hat{A}_{i,t}\right)\right]$$

$$\text{s.t.}\quad 0 < \Big|\{o_i \mid \texttt{is\_equivalent}(a,o_i)\}\Big| < G.$$

[1] Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Dai, W., Fan, T., Liu, G., Liu, L. and Liu, X., 2025. Dapo: An open-source llm reinforcement learning system at scale. arXiv preprint arXiv:2503.14476.

DAPO: Overlong Reward Shaping

- We set a maximum length for rollouts = some are truncated;
  - Truncated rollouts get 0 rewards;
  - Potentially confusing because they can be completely valid!
- DAPO ideas:
  - **Overlong filtering:** i.e. mask out truncated rollouts in the loss; already helps!
  - **Soft overlong punishment:** penalty for overlong responses; scales based on length up to some maximum value;

$$R_{\text{length}}(y) = \begin{cases} 0, & |y| \le L_{\max} - L_{\text{cache}} \\ \frac{(L_{\max} - L_{\text{cache}}) - |y|}{L_{\text{cache}}}, & L_{\max} - L_{\text{cache}} < |y| \le L_{\max} \\ -1, & L_{\max} < |y| \end{cases}$$

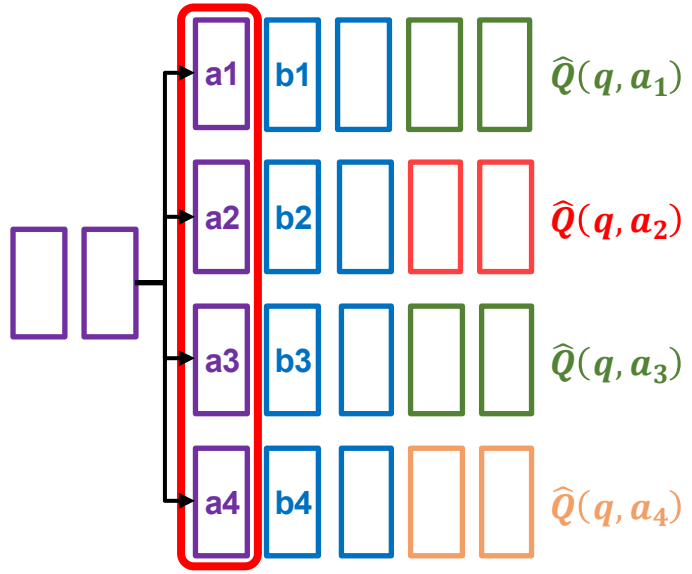
Table 1 Main results of progressive techniques applied to **DAPO**

Model	AIME24$_{\text{avg@32}}$
DeepSeek-R1-Zero-Qwen-32B	47
Naive GRPO	30
+ Overlong Filtering	36
+ Clip-Higher	38
+ Soft Overlong Punishment	41
+ Token-level Loss	42
+ Dynamic Sampling (**DAPO**)	**50**

[1] Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Dai, W., Fan, T., Liu, G., Liu, L. and Liu, X., 2025. Dapo: An open-source llm reinforcement learning system at scale. arXiv preprint arXiv:2503.14476.

GSPO Sets Out to Fix GRPO' Incorrect Monte Carlo Estimates

- GRPO samples n rollouts;
- **1st token:** we can pretend that each rollout gives a Monte Carlo (single sample) estimate of its value;
- **The following tokens: the prefixes don't match!**
  - I.e. what we would really need to do is **pick a prefix** and then **resample from that position**;
  - But this would be **very expensive!**



GSPO: Group Sequence Policy Optimization [1]

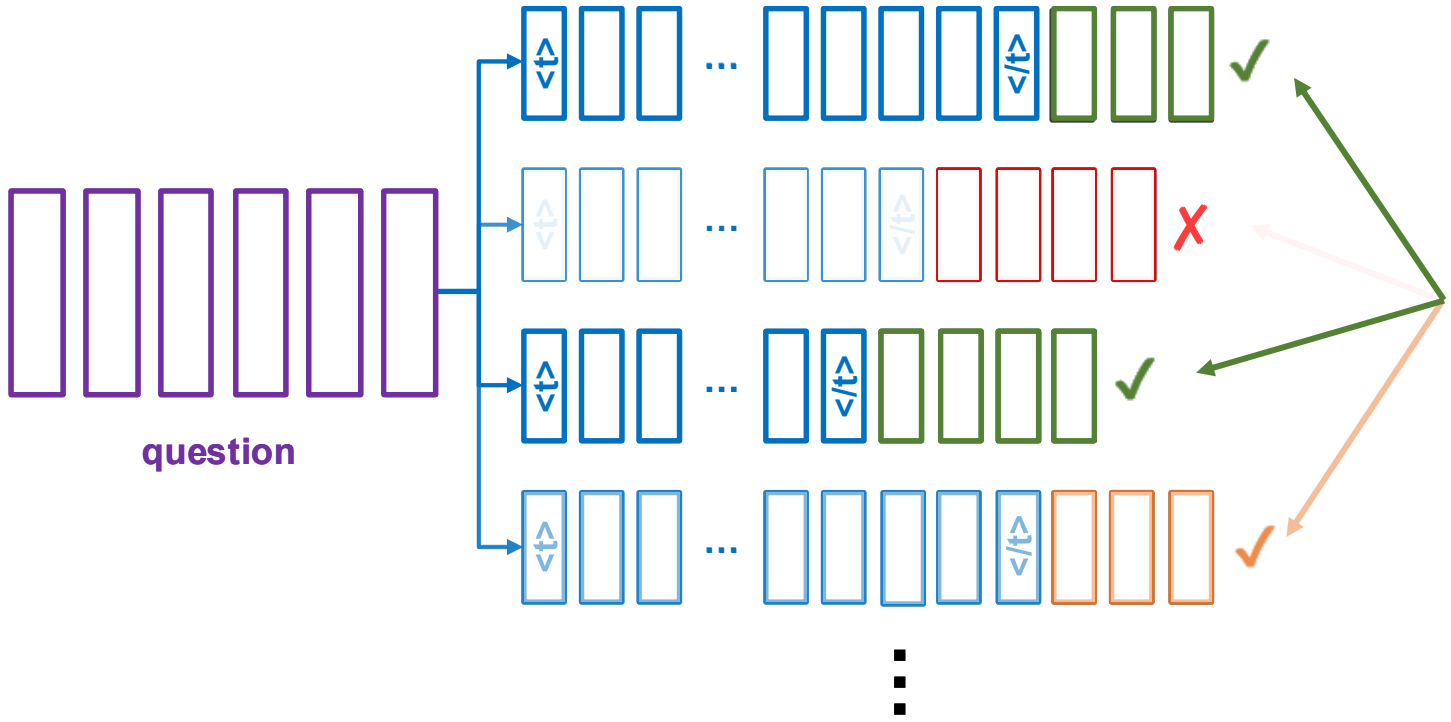
- We can’t get correct **token-level advantages** from GRPO-style rollouts;
  - Without resampling from every position;
- **GSPO:** let’s apply GRPO at the **sequence level**;
  - Work with sequence-level advantages and likelihoods;
  - GRPO: each token in the sequence is reinforced to a different degree;
  - GSPO: all tokens in the sequence are reinforced to the same degree;

[1] Zheng, C., Liu, S., Li, M., Chen, X.H., Yu, B., Gao, C., Dang, K., Liu, Y., Men, R., Yang, A. and Zhou, J., 2025. Group sequence policy optimization. arXiv preprint arXiv:2507.18071.

Do GRPO-Style Methods Really Work?

GRPO: Use Just Sample Estimates [1]

1. Leverage the model's inherent reasoning capabilities; generate N different reasoning chains + answers;
   - Zero-shot CoT prompt: think about the reasoning process in your mind first, then answer…



2. **Reinforce** thoughts based on advantage;
   - Accuracy of answers;
   - Correct use of <think>;
   - Policy-based RL;

reinforce thoughts in proportion to how good the answers are

But [1] finds that this does not work as well with **small LLMs**! Why?

[1] Daya Guo, et al., 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv preprint arXiv:2501.12948.

GRPO Does Not Work As Well for Smaller Models

- Qwen2.5-32B trained with GRPO is mostly on par with a prior 32B model: QwQ-32B-Preview;
- But much **better results can be achieved by distilling** from the large DeepSeek-R1 model;

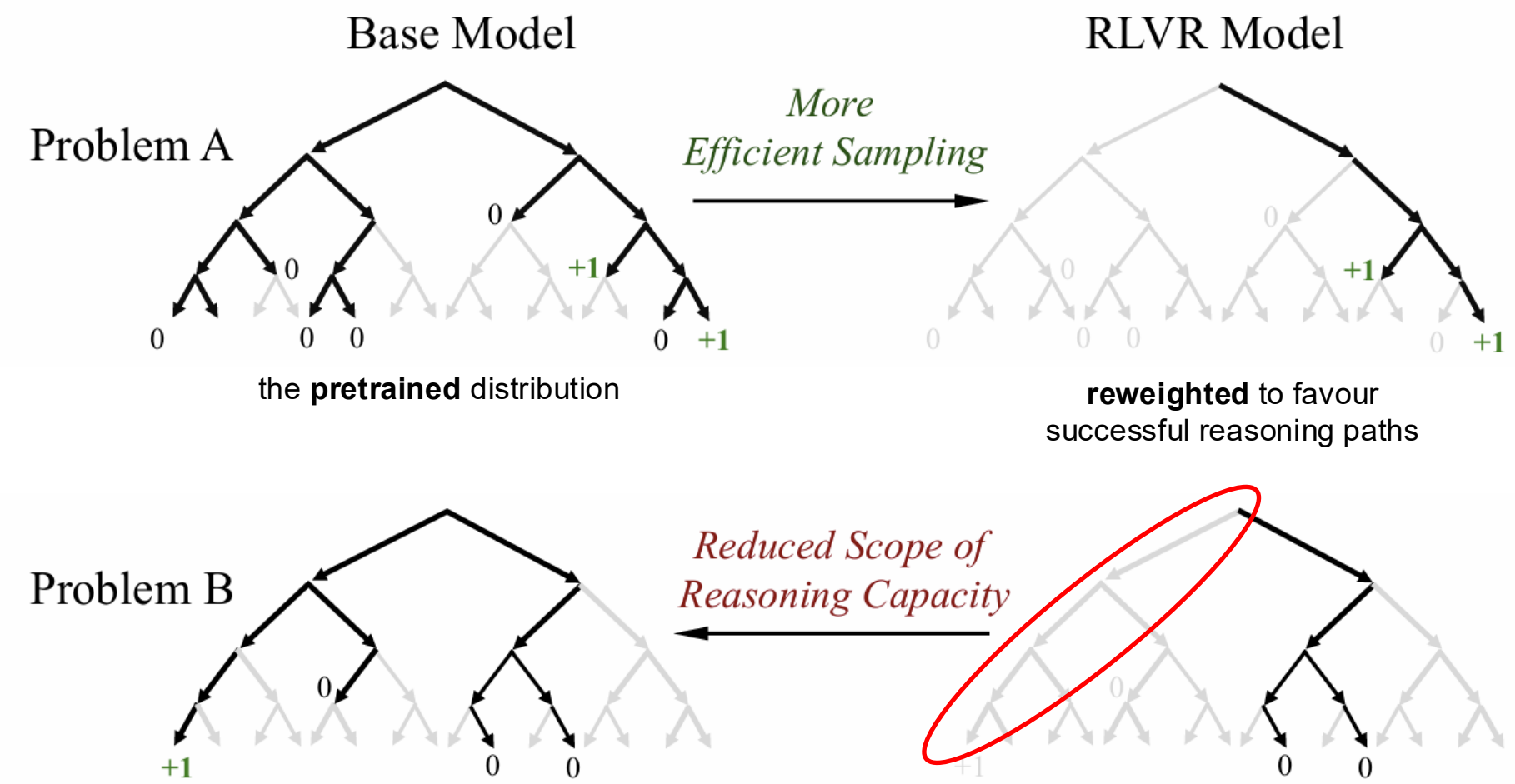
Table 16 | Comparison of distilled and RL Models on Reasoning-Related Benchmarks.

Model	AIME 2024		MATH	GPQA Diamond	LiveCode Bench
	pass@1	cons@64	pass@1	pass@1	pass@1
QwQ-32B-Preview	50.0	60.0	90.6	54.5	41.9
Qwen2.5-32B-Zero	47.0	60.0	91.6	55.0	40.2
DeepSeek-R1-Distill-Qwen-32B	**72.6**	**83.3**	**94.3**	**62.1**	**57.2**

So why does GRPO not work that well for smaller models?

[1] Daya Guo, et al., 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv preprint arXiv:2501.12948.

Because It Does Not Really Work with Large Models Either...



- [1] compares RL against allowing a base model $k \gg 1$ tries;
- RL adds **no new** reasoning paths – just **reweights** the pretrained distro;
- # of reasoning paths actually **decreases**!!! 🤦

[1] [2504.13837] Does Reinforcement Learning Really Incentivize Reasoning Capacity in LLMs Beyond the Base Model?

RL: Reweighting or Sharpening?

- According to some results, RL may not even be reweighting – more or less just sharpening [1,2];
  - Plot: **perplexity** of responses of the RL model keeps decreasing **under the base** model!
- We are just **further ↑ the probability** of trajectories already highly probable under the base model;
  - Seems counter-intuitive, but note that we typically **sample greedily** so even with $\tau = 0$ we do not sample the most probable trajectory!
  - In this view, RL kind of **enables sampling it greedily**!

[2]

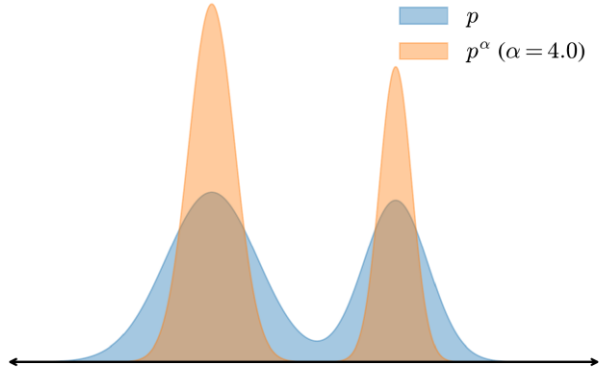


Figure 2: **A toy example of distribution sharpening.** Here p is a mixture of Gaussians, which we plot against p^{α} ($\alpha = 4.0$).

[1]

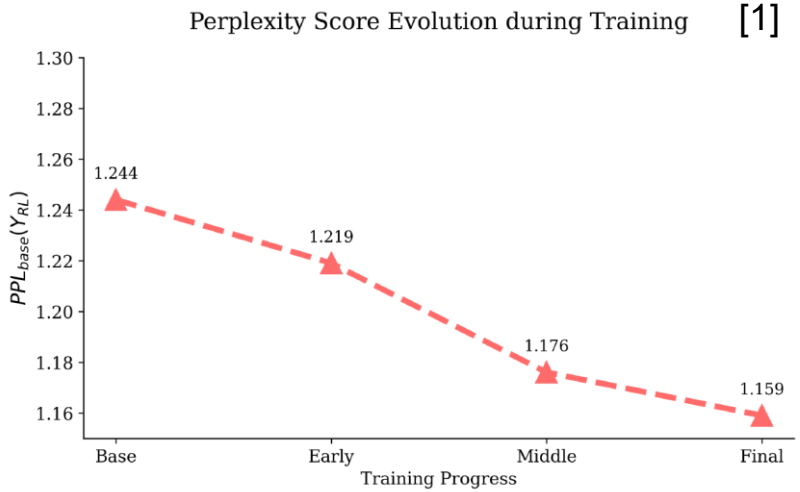


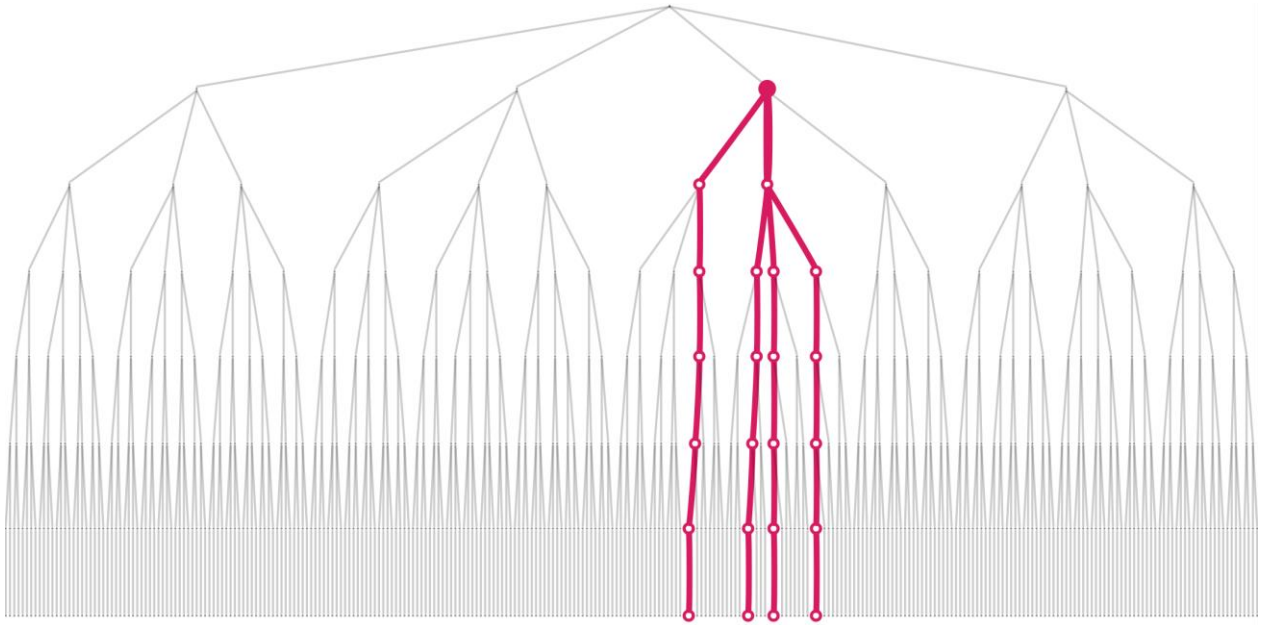
Figure 15: Perplexity Evolution during RL Training.

[1] [2504.13837] Does Reinforcement Learning Really Incentivize Reasoning Capacity in LLMs Beyond the Base Model?
[2] [2510.14901] Reasoning with Sampling: Your Base Model is Smarter Than You Think

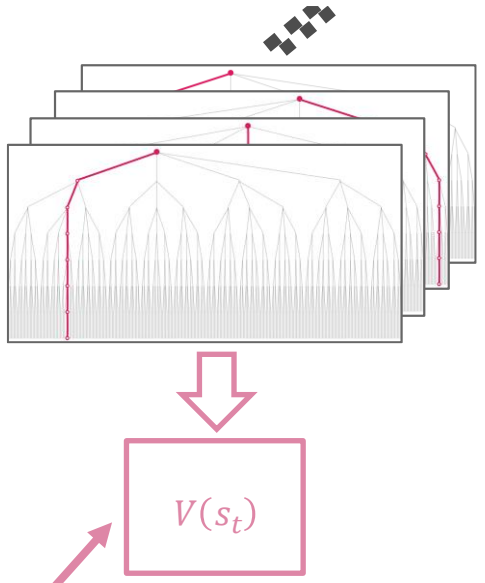
Policy Improvement Operators Are Crucial

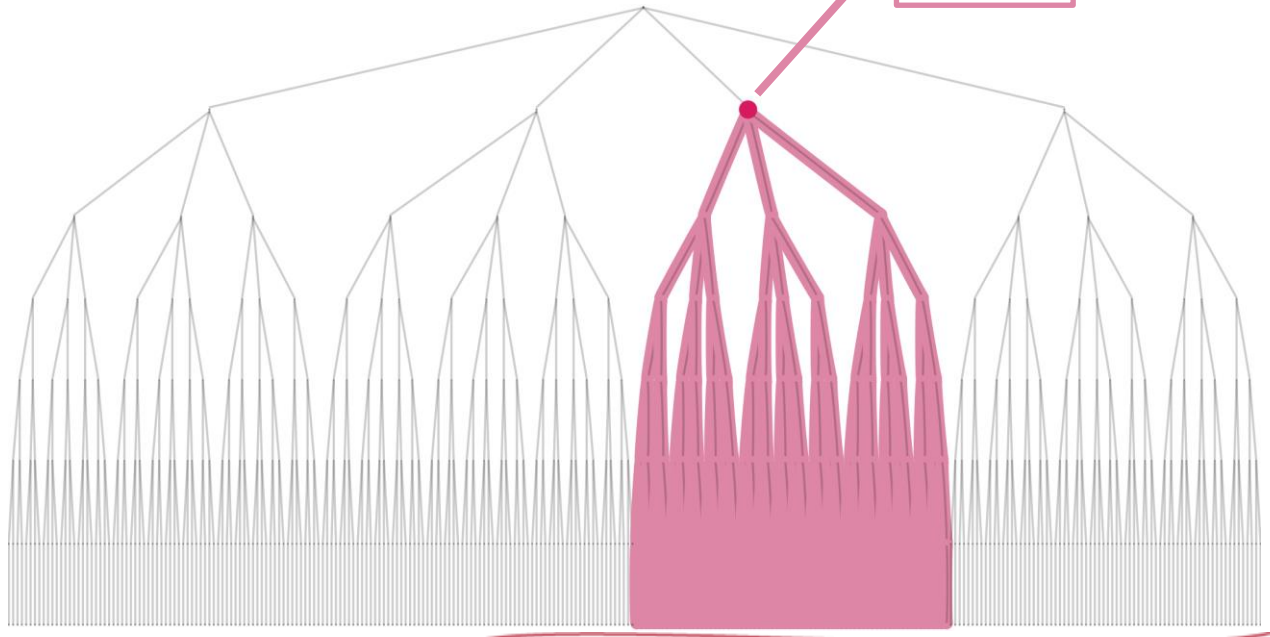
GRPO: Very Weak Policy Evaluation

- **GRPO:** sample-estimate of advantage [1];
  - (Should be) very **high-variance**;
    - Especially for early tokens;
  - No value function = **no generalization**;
  - Also wrong estimate, corrected in GSPO [2];



- Explicit **value function**;
  - Much **lower variance**;
  - Introduces **bias**;
  - Has **generalization!**

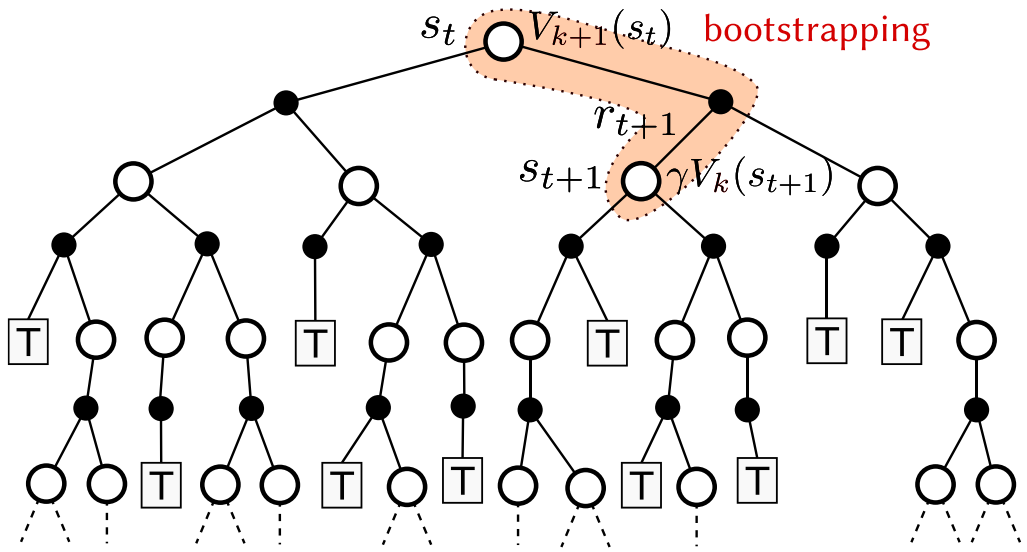




- Variance not currently a huge issue in practice: multiple rollouts and sequence-level advantages help;
- But this could be an artifact of insufficient exploration;

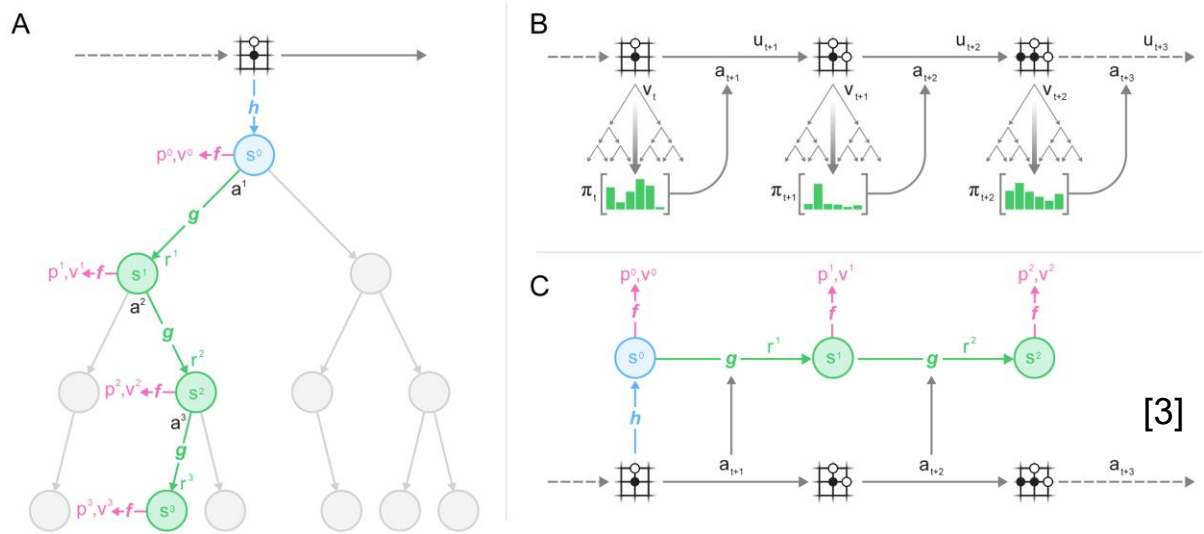
Critic-Based vs. Search-Based

- PPO/DQN: **temporal differences** based;
  - Observe a transition = new information;
  - Back up the **value** of the next state + immediate reward (or several steps);



The TD backup (inspired by [2])

- **Alpha/MuZero:** value function + search [3];
  - Explicit value function = **generalization**;
  - **Search** = policy improvement more or less guaranteed;



[1] SUTTON, R.S. - BARTO, A.G. *Reinforcement Learning: An Introduction*. [s.l.]: The MIT Press, 1998. ISBN 0262193981.
[2] SILVER, D. Lecture 4: Model-Free Prediction. 2015. [cit. 2019-05-11]. URL: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/MC-TD.pdf

[1] Daya Guo, et al., 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv preprint arXiv:2501.12948.
[2] Zheng, C., Liu, S., Li, M., Chen, X.H., Yu, B., Gao, C., Dang, K., Liu, Y., Men, R., Yang, A. and Zhou, J., 2025. Group sequence policy optimization. arXiv preprint arXiv:2507.18071.
[3] Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T. and Lillicrap, T., 2020. Mastering atari, go, chess and shogi by planning with a learned model. Nature, 588(7839), pp.604-609.

Policy Improvement Recap

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\rho^{\pi_\theta}, a\sim\pi_\theta}[\nabla_\theta \ln \pi_\theta(s,a)\, Q^{\pi_\theta}(s,a)]$$

- **Uniform (SFT style): uniform** weights: $Q^{\pi_\theta}(s,a) = 1\ \forall s,a$;
  - Push everything up equally – **imitation learning** from experts;
- **Sample-based (GRPO/GSPO style):** sample estimate based on N rollouts;
  - No critic = high-variance & **no generalization**;
- **Critic-based (PPO style):** advantages $\hat{A}_t^{GAE(\lambda)}$ computed using an explicit **critic**;
  - Explicit value function $V(s_t)$ to act as **critic**;
- **Search-based (AlphaGo style):**
  - Explicit **value function** + **search** for (more or less) guaranteed policy improvement;
- **Reflective (GEPA style):** reasoning about outcomes (we'll come back to this);

We Need Different Kinds of Memory

Types of Memory in Current LLMs

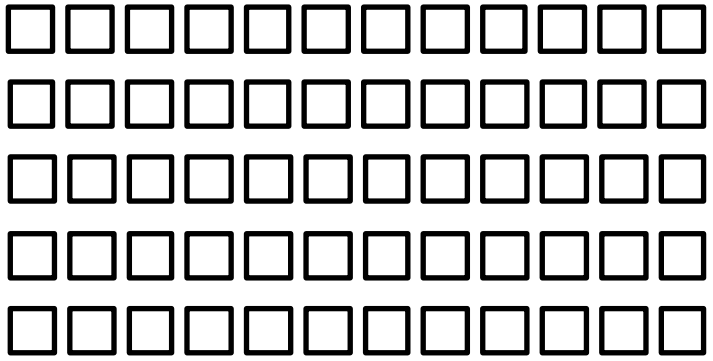
- **The context:** instance-specific; directly accessible;
- **Supporting documents:** instance specific; RAG, etc.
- **Weights:** long-term memory; knowledge & skills;


```
{  "timestamp_created": "2025-04-14T12:30:00Z",
  "text": "User likes color blue.",
  "source": "user_message",
  "last_updated": "2025-04-14T12:30:00Z“ }
```

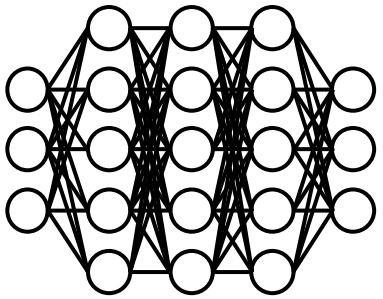

- **Explicit tool-based memory:**
  - E.g. short excerpts in a JSON-like format;
  - LLM decides what to store & calls a tool;
  - Relevant memories are automatically injected into the context;

the context

supporting documents, …



weights

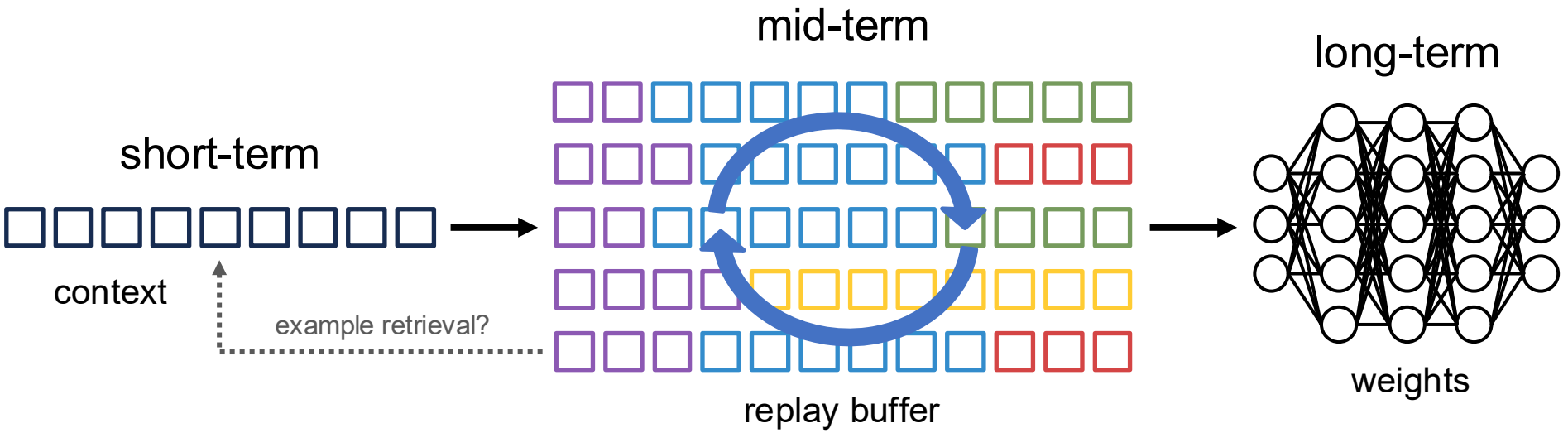


Effective Use of Different Kinds of Memory

- Storing updates directly in the weights is **very inefficient**;
  - Gradient updates are very slow!

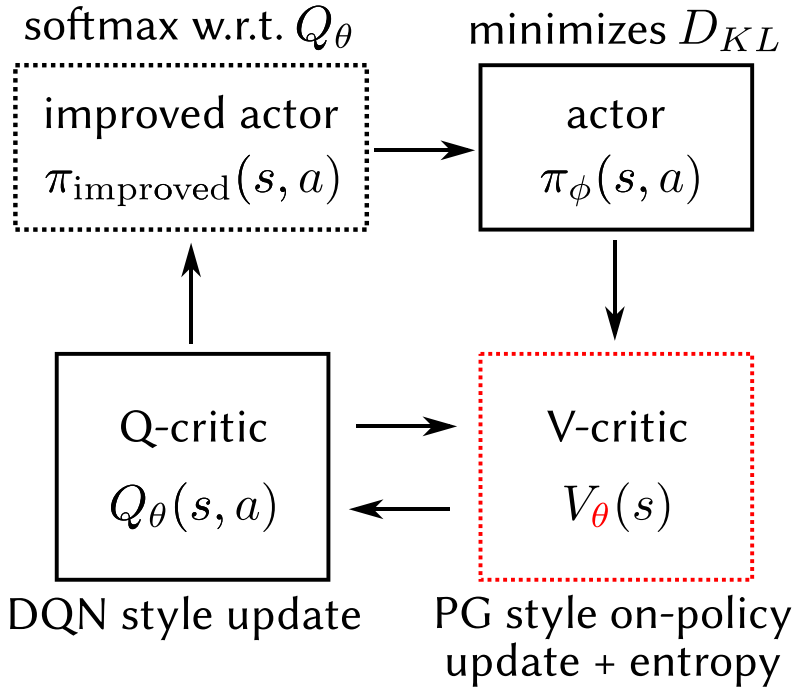
eventually, all types of memory should ideally be neural, of course

- Reflective feedback could have the form of self-prompts just like in GEPA;
  - Update the **prompt** & use it to process the next batch (super quick adaptation);
  - Replay buffer can be used for **instance-based retrieval** if we want;
  - Skills gradually distilled into weights through **experience replay** – analogies in the brain;



Replay Buffer = We Need Off-Policy Methods

- Methods like GRPO and PPO are **on-policy – can't** use a **replay buffer**!
  - I.e. also not very **sample-efficient**;
- We should ideally be using an **off-policy** method – something like SAC (Soft Actor Critic) [1];
  - In standard RL, SAC is a very sample-efficient and quite stable method;
  - Difficult to apply to LLMs with the N different networks that one needs to hold in memory and run;



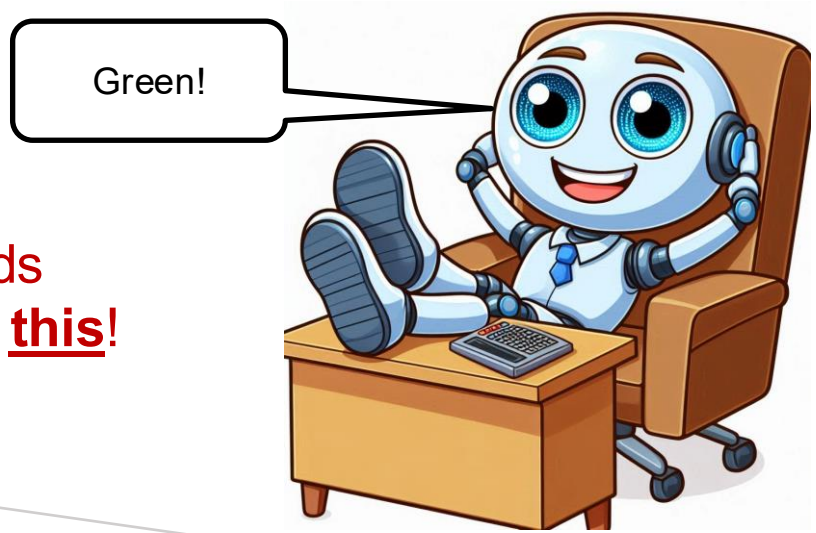
[1] HAARNOJA, T. et al. Learning to walk via deep reinforcement learning. In *arXiv preprint arXiv:1812.11103*. 2018.

Bootstrapping Reasoning

Bootstrapping Reasoning

- Leverage the model's inherent reasoning capabilities;
- **Reinforce** reasoning traces that lead to **positive outcomes** on a task;
- Gradually improve the ability to reason;

Question: What is your favorite color?



This probably needs **less** compute than **this**!

Question: Please, find the proof for the following lemma from the Soft Actor Critic paper:

Soft Policy Improvement

Let $\pi_{old} \in \Pi$ *and let* π_{new} *be the optimizer of the minimization problem defined in Eq. (4).*

Then:

$Q^{\pi_{new}}(\boldsymbol{s}_t, \boldsymbol{a}_t) \geq Q^{\pi_{old}}(\boldsymbol{s}_t, \boldsymbol{a}_t)$

for all $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$ *with* $|\mathcal{A}| < \infty$.

KINIT

Pharaoh Example: Without Reasoning

Qwen3-8B

Q: Who ruled Egypt in 1328 BC?

A: In 1328 BC, Egypt was ruled by Pharaoh Horemheb. Horemheb was the last Pharaoh of the 18th Dynasty and reigned from approximately 1320 to 1292 BC. His reign marked the end marked the end of the New Kingdom's

[...]

So, in 1328 BC, the ruler of Egypt was Pharaoh Horemheb.

Wikimedia

Pharaoh Example: With Reasoning

Qwen3-8B

Q: Who ruled Egypt in 1328 BC?

<thought>Okay, so I need to figure out who ruled Egypt in 1328 BC.

[...]

Akhenaten: 1353–1336 BC
Smenkhkare: possibly 1336–1334 BC
Tutankhamun: 1332–1323 BC
Ay: 1323–1318 BC

[...]

So, if Tutankhamun's reign is from 1332 to 1323 BC, then 1328 BC is during his reign.

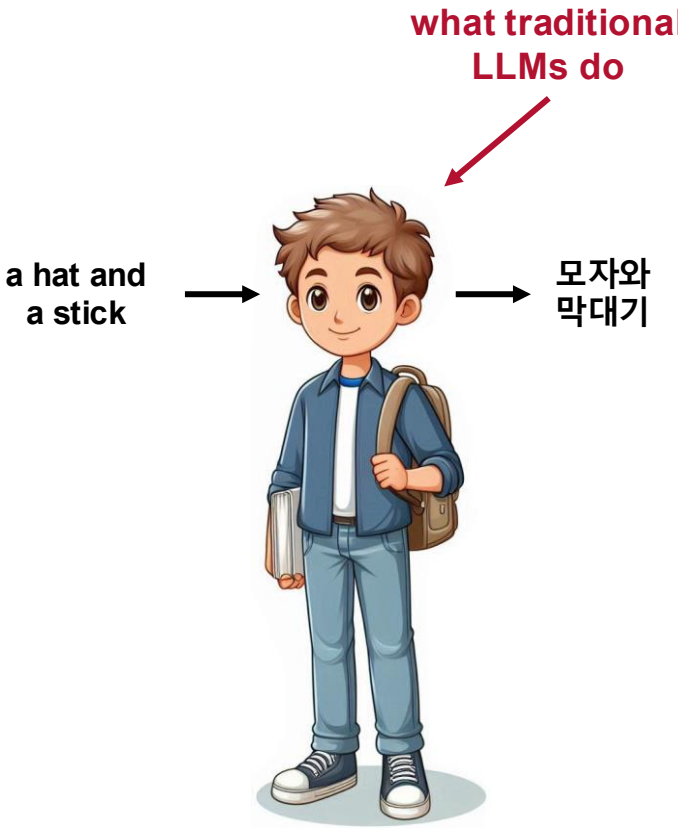
[...]**</thought>**[...]

A: [...] Tutankhamun was the pharaoh ruling Egypt in 1328 BC.

Wikimedia

Reasoning-Enabled Learning

- To accommodate really low-resource languages:
  - Do we need to make models learn **more actively**?
  - Inspirations from human learning & **reasoning** models?



System 1

Passive approach:

- Observe huge amounts of samples from distro;
- Inductively learn the rules;
- Purely “intelligence-from-data”;

Reasoning-Enabled Learning

what reasoning LLMs could do

- To accommodate really low-resource languages:
  - Do we need to make models learn **more actively**?
  - Inspirations from human learning & **reasoning** models?

Active approach:

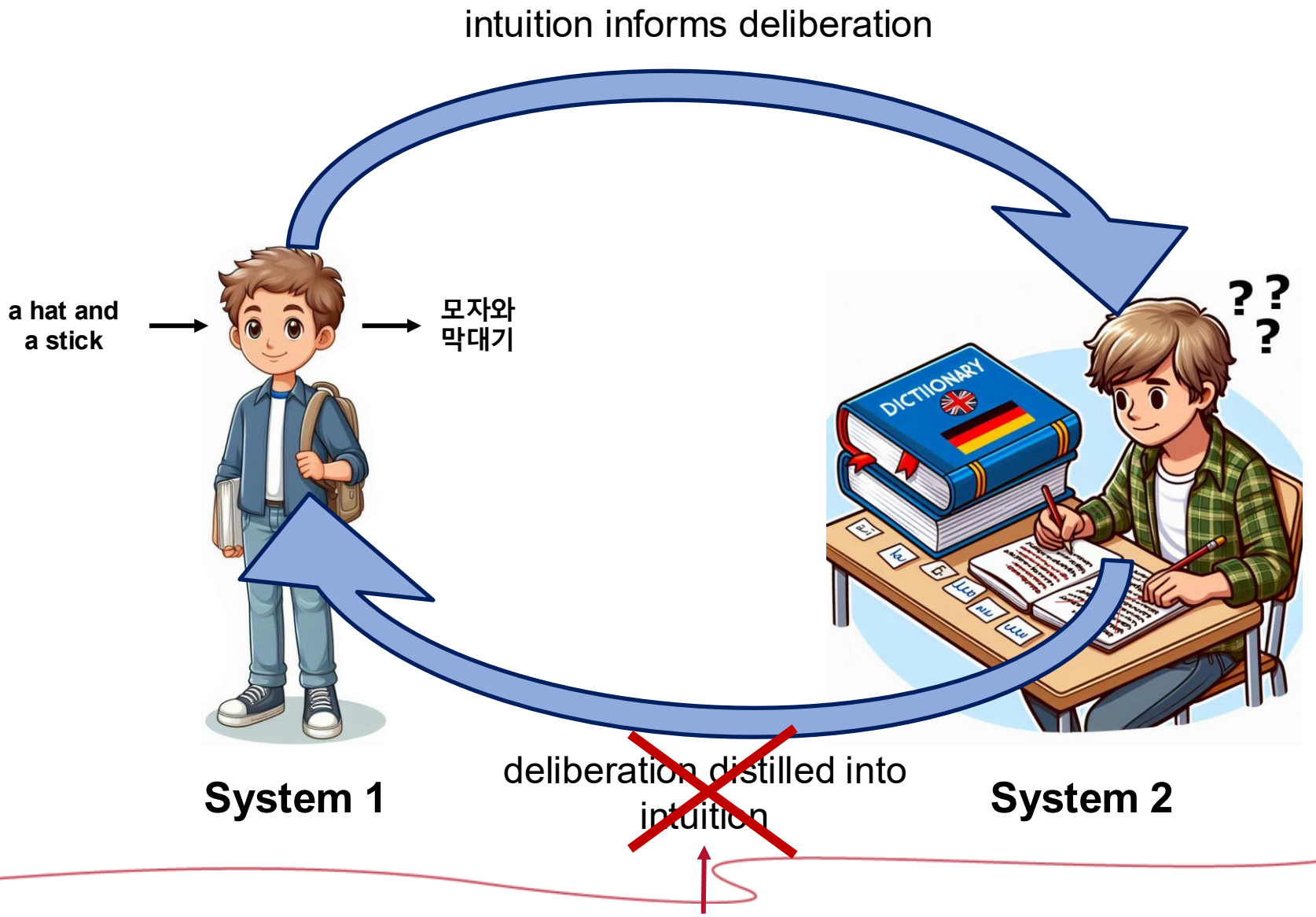
- Dictionary, grammatical rules, examples, etc.
- **Reason to construct sentences**, get feedback, improve;

System 2

Note that even very proficient language users still do this sometimes: look words up, figure out what tense to apply, etc.

Crucial: Interplay of Systems I & 2!

- Humans combine systems I & II for effective learning;
  - High-level reasoning;
  - & internalizing into an intuitive skill!
- Motivation:
  - **Starting from "rules"** & internalizing them vs. figuring everything out from scratch;
  - Building up an **implicit curriculum** (motivation from distillation curricula);
  - RL, not imitation learning: works around **compounding errors**;



Distillation into Intuitive Skills: A Few Notes

Explicit Distillation — very practical

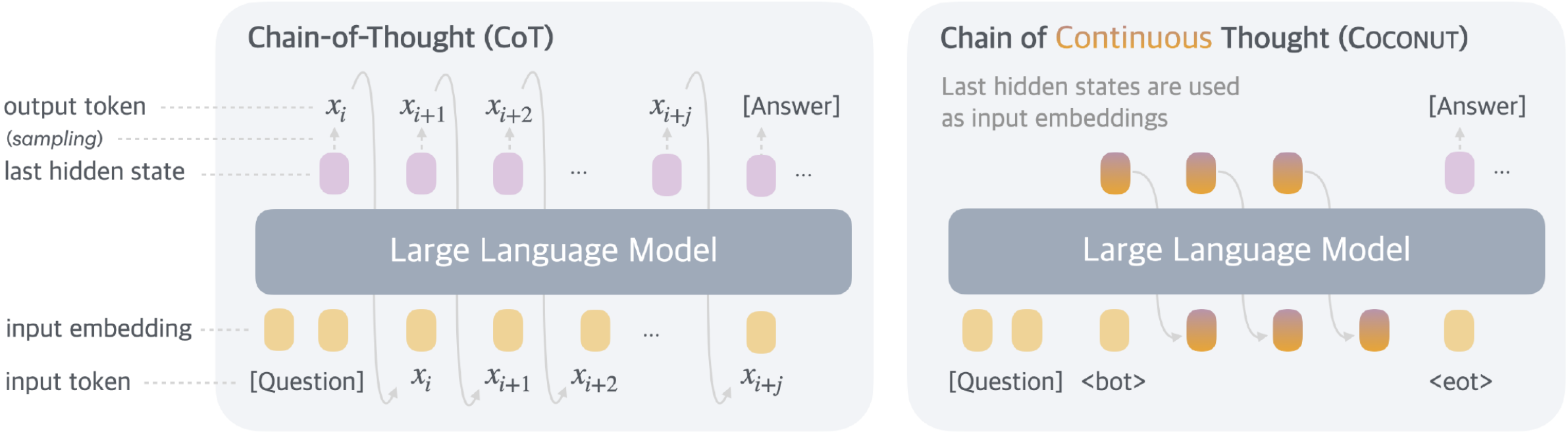
- Train on **question -> answer**, w/o reasoning path;
- Likely to work quickly;
  - But system 1 and system 2 need to be **balanced explicitly**;
- Need to **manually identify** distillable sub-problems;
  - Not easy – a difficult skill to train explicitly – very long horizon;

Implicit Distillation — more general

- Simply run RL to improve reasoning;
- This should gradually build up **useful representations**, etc.;
  - These should gradually make it much easier to learn heuristics = intuition;
- The model might **discover** that it can now solve the problem **intuitively**;
  - Needs **strong exploration** – e.g. the maximum entropy framework?
  - Could be encouraged by **sampling** a **reasoning budget** & rewarding for keeping within it;

CoT Reasoning vs. Soft Reasoning

- Pass a raw embedding around instead of going through actual discrete tokens [1];
  - Or use a weighted sum of discrete token embeddings [2] – (kind of) works without training + usable for models with different input / output embeddings;
- Paper [3] shows that this way the LLM can explore multiple search frontiers in parallel;



[1] Hao, S. et al., 2024. Training large language models to reason in a continuous latent space. arXiv preprint arXiv:2412.06769.
[2] Zhang, Z., et al., 2025. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. *arXiv preprint arXiv:2505.15778*.
[3] Zhu, H. et al., 2025. Reasoning by Superposition: A Theoretical Perspective on Chain of Continuous Thought. arXiv preprint arXiv:2505.12514.

PPO-Style Methods and Their Challenges

VC-PPO [1]: Addressing Bias in Long-CoT Training

- Core issue for PPO on long sequences: **huge amount of bias**;
  - **Pre-training bias:** value function (VF) initialized from policy / reward model;
  - **In-training bias:** bootstrapping in GAE; decay of the terminal reward over seq. length;

- Solution:
  - **Pretrain VF** on rollouts from the base policy;
  - **Decoupled** λs:
    - $\lambda = 1$ for VF;
    - $\lambda < 1$ for the policy;

Model	AIME 2024 pass@1	AIME 2024 pass@32	GPQA pass@1	CodeForces pass@1
GRPO	38.9	70.0	**49.4**	12.6
VC-PPO	**48.8**	**73.3**	48.8	**12.8**

Table 1 Comparison between VC-PPO and GRPO in 16K context length.

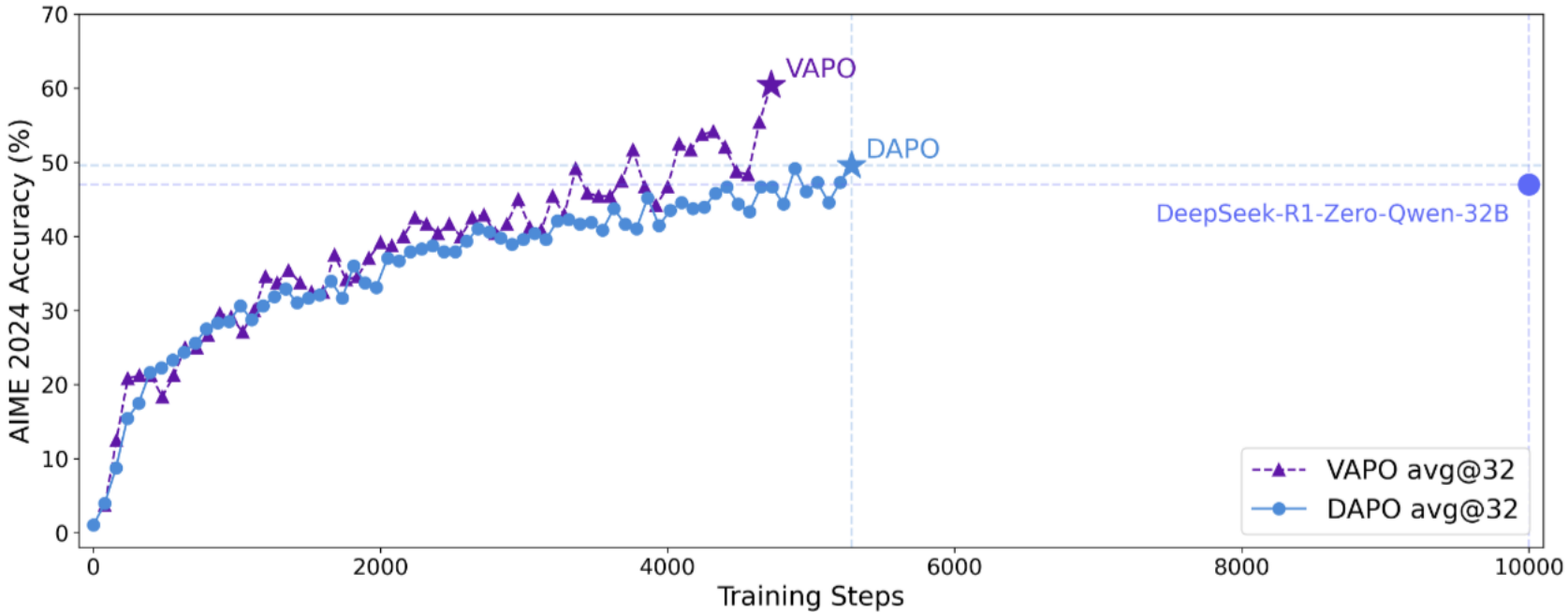
→ **slightly more biased, but stable gradients are beneficial for the policy**

[1] Yuan, Y., Yue, Y., Zhu, R., Fan, T. and Yan, L., 2025. What's Behind PPO's Collapse in Long-CoT? Value Optimization Holds the Secret. *arXiv preprint arXiv:2503.01491*.

VAPO: Use Length-Adaptive λ

- **Key observation:** GAE bias-variance trade-off depends on sequence length;
  - **Short sequences** are **variance-dominant**: the terminal reward can still propagate;
  - **Long sequences** are **bias-dominant**: bootstrapping is much more prominent;
- **Idea:** length-adaptive GAE: $\lambda = 1 - {}^{1}/_{\alpha l}$
- Additional **NLL loss** over samples with positive rewards;

a bit of a hack to be honest 🤦



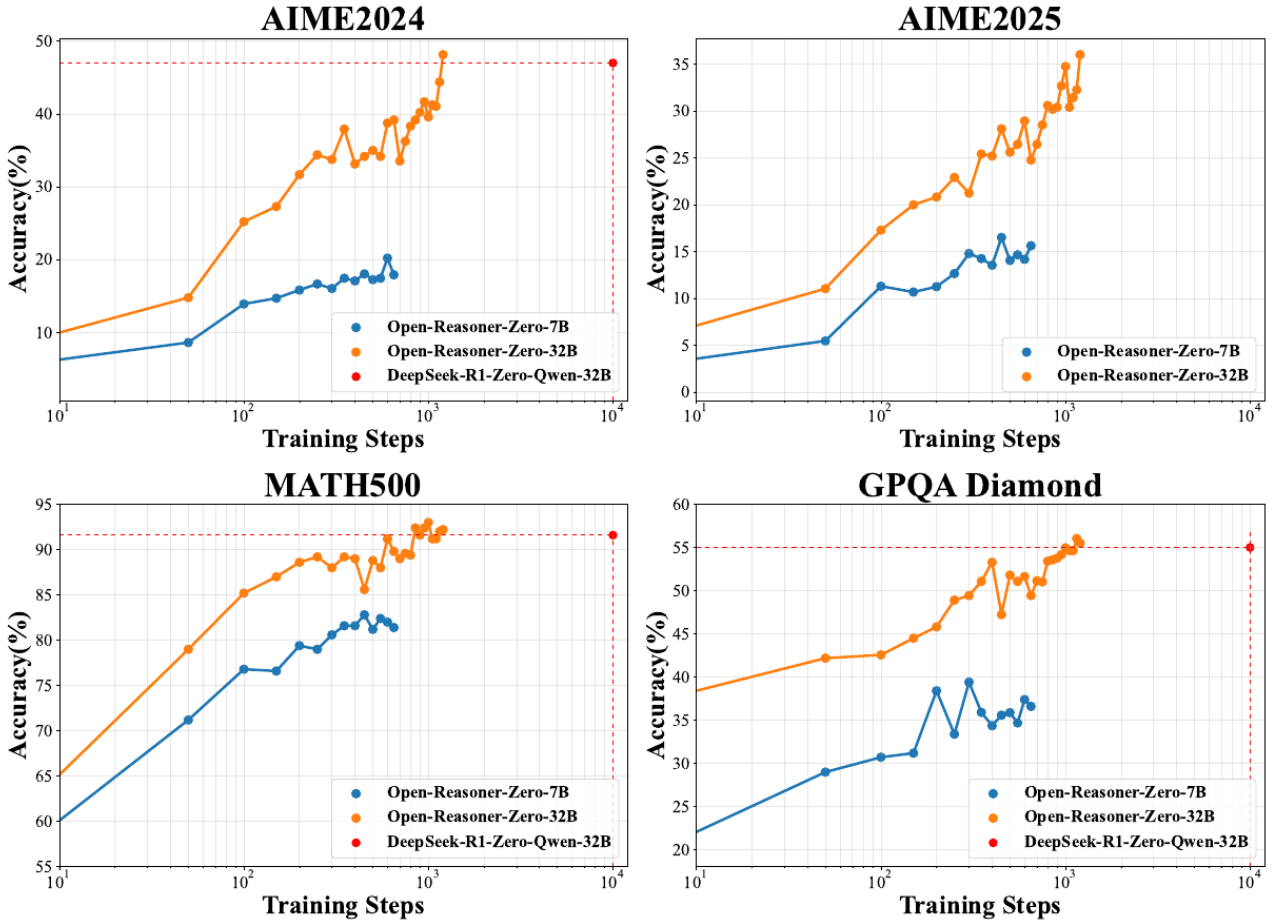
They also use:

- Value-Pretraining from VC-PPO;
- Clip higher from DAPO;
- "Every token has the same weight from" DAPO;
- Group sampling (GRPO);

Open-Reasoner-Zero (ORZ) [1]

a straight-forward application of PPO can be stable and performant

- Simple vanilla PPO with:
  - $\lambda = 1; \gamma = 1;$
  - They do multiple rollouts like GRPO (better exploration);
  - No KL penalty;
- VF initialized from the policy + value head; no param sharing;
- Outperforms the GRPO-trained DeepSeek-R1-Zero-Qwen-32B;
  - With **1/10 training steps**;



[1] Hu, J., Zhang, Y., Han, Q., Jiang, D., Zhang, X. and Shum, H.Y., 2025. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. arXiv preprint arXiv:2503.24290.

Thank you for your attention!

KINIT

Kempelen Institute
of Intelligent Technologies

Nivy Tower
Mlynské Nivy II. 18890/5
821 09 Bratislava
Slovakia

www.kinit.sk

Michal Gregor

Researcher – Expert
michal.gregor@kinit.sk | www.kinit.sk
https://michalgregor.gitlab.io | LinkedIn