Post-Training Verifiable Agents

Guest Speaker: Jiantao Jiao

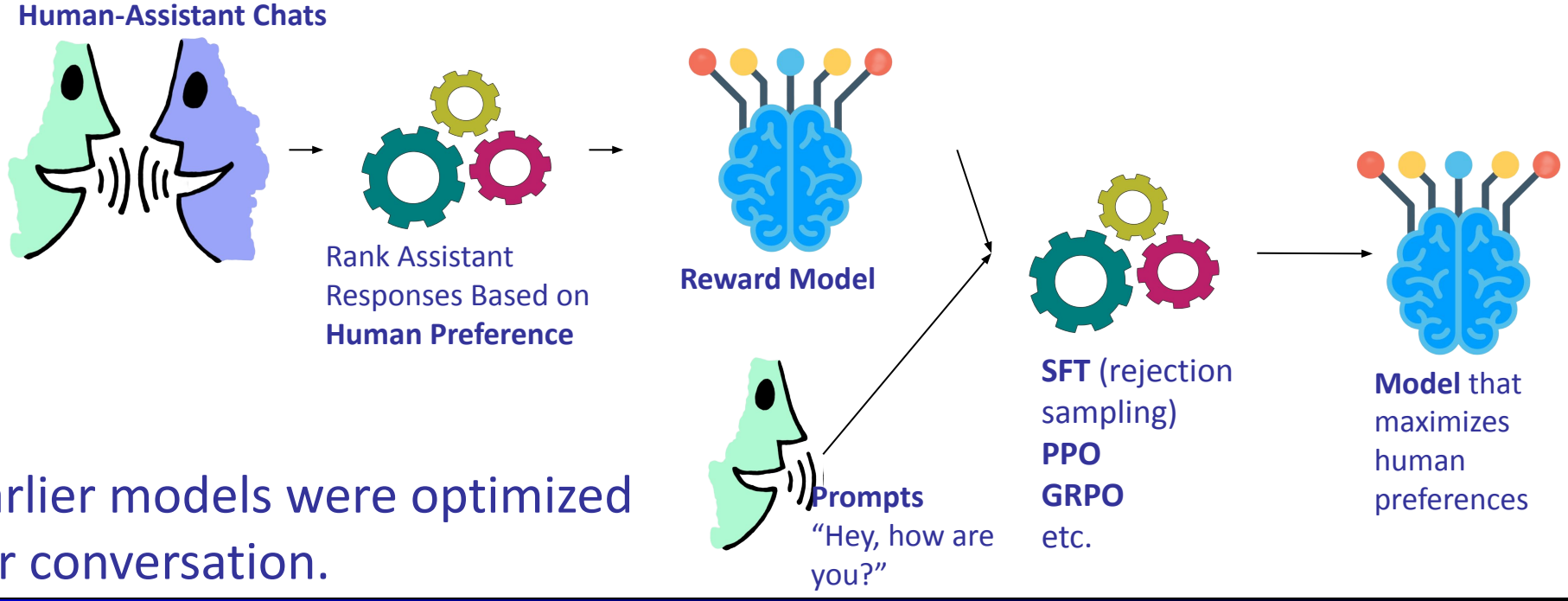
Disclaimer: The views expressed are solely those of the guest and are not representative of their company.

How Do “Agentic” Models Differ From Traditional LLMs?

1. **Earlier Chat Models**: Human Aligned Models, designed to provide interactions that **maximize human preference**.

2. **Agentic Models**: Environment Feedback Aligned Models, designed to provide interactions that **maximize verifiable rewards** (in addition to human preference).

Earlier Models Were Trained To Be Great Chat-Bots



Earlier models were optimized for conversation.

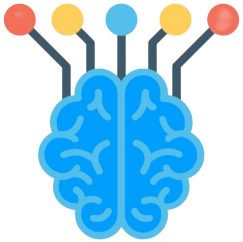
What More Is Necessary?

- Conversation is not verifiable*, there is often no clear single correct answer.* You need to *model preference.*
- Conversation requires interaction with the user only.
- Many tasks have one (or a few) correct answers, and blatantly wrong ones.
- Require interactions with many entities (user and the environment) to understand the full state.

Today's Models Are Trained To Be Great Agents

Prompts
"I want to achieve this"

SFT (rejection sampling)
PPO
GRPO
etc.



Agentic Model that maximizes **correctness** by interacting within sandboxes with tools

Rubrics
Ground Truths
Environment States
Unit Tests

Environment
Tools

What is needed to train the verifiable agentic models?

Step 1: Get good verifiable training data to train these agents.

Step 2: Verifiable agents are intelligent. We need to define what intelligence is.

Step 3: Understand how to feed the verifiable data to the model. The recipes.

Step 1: Getting Good Training Data!

Agentic Models Maximize **Verifiable Reward**

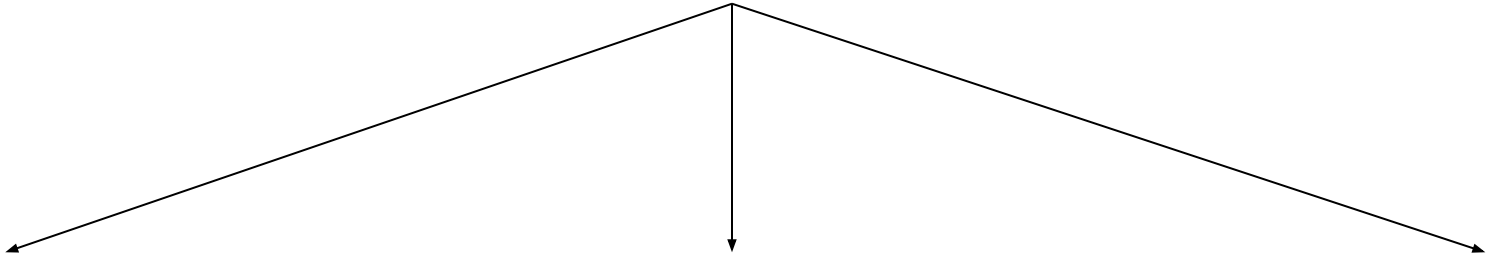
Point 1 – The Environment: These models are trained to **consume** tokens that inform the model about the environment's current state, and the user's intention. An environment is something like a code repository, or a web browser, etc.

Point 2 – Tools: These models are trained to generate tokens that are **consumed by software tools that provide additional relevant information to the model** to help it maximize the reward from the verifier. These software tools can be APIs that provide additional information, or tools that change the environment's state.

Point 3 – Verifier: These models are trained to generate tokens that **maximize the reward from the verifier**. Verifiers are unit tests for code, math checker for proofs, DOM scripts for web tasks, etc.

How Do We Train A Good Agent Model?

First Step: Good Training Data!



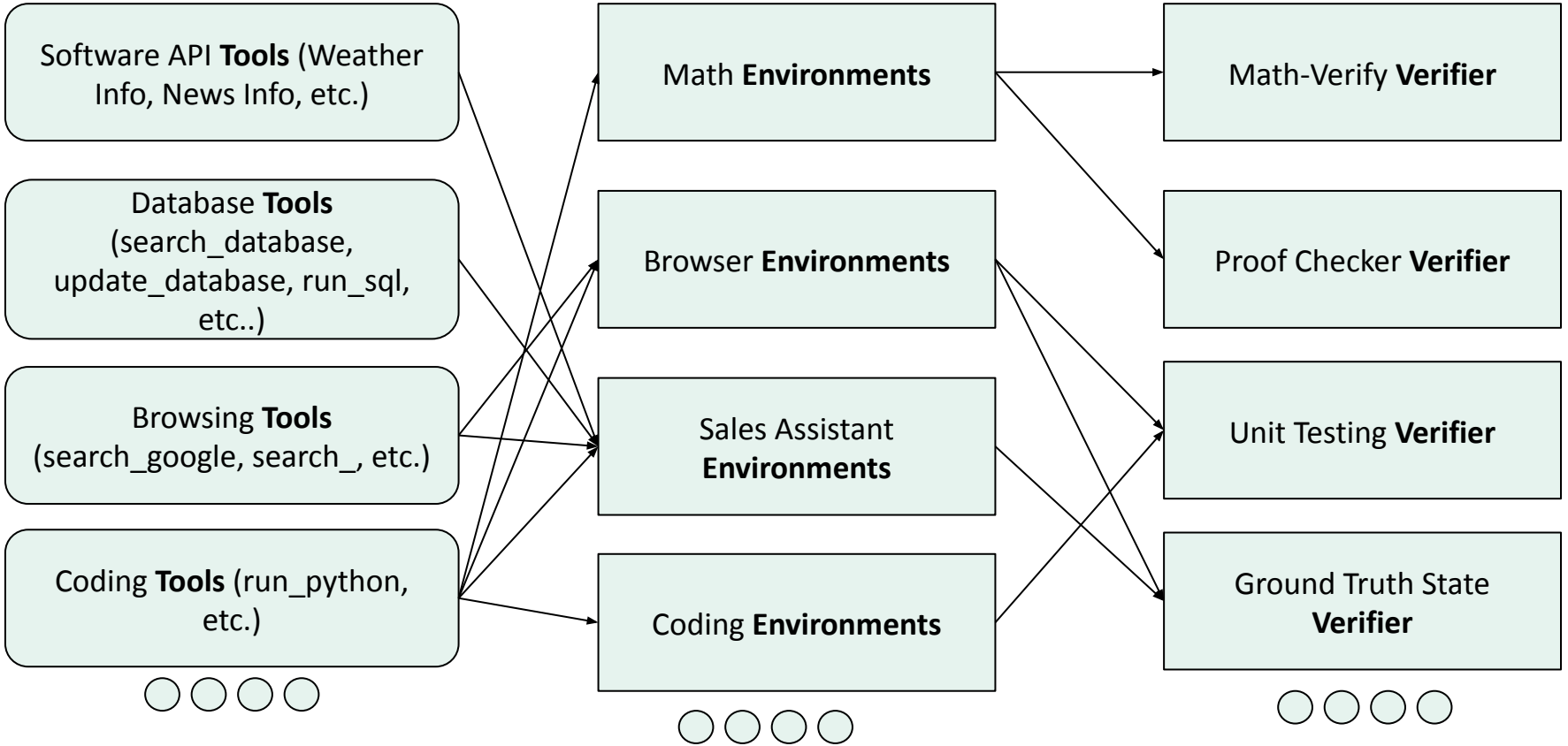
Train To Understand Many Different Environments *(Point 1)*

Train To Use Many Different Kinds of Tools *(Point 2)*

Train To Do Well Under Many Different Verifiers *(Point 3)*

Environment and verifier diversity is critical!

The Many Kinds of Environments, Tools, and Verifiers! We want to train in all



Along With Scale, Verifier Quality Is Also Extremely Important

Verifier Quality: Make sure we have few **False Positives and False Negatives!**

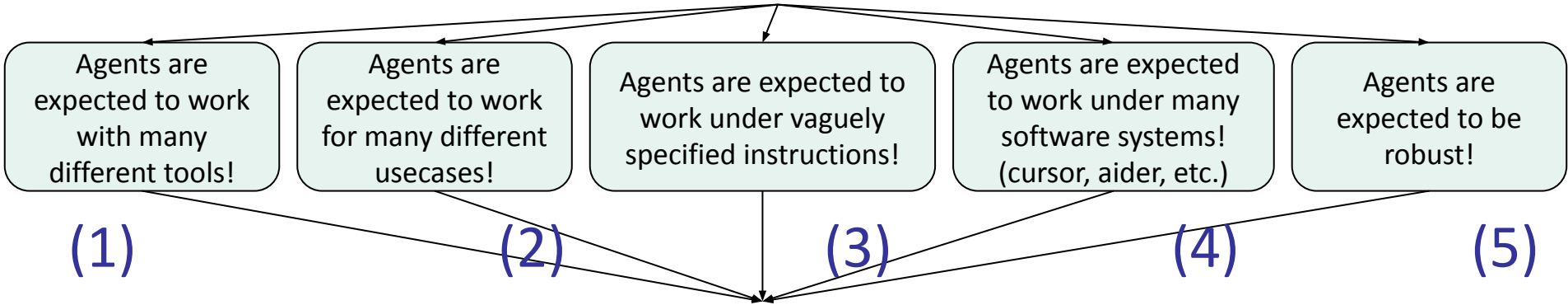
1. If there are several ways to get the right ground truth, you want the verifier to reward them *all.*
    a. "How much change will I get back if I buy a drink for fifty cents and I give a dollar" → ½, 2/4, are all correct, etc.
2. You *don't* want to reward wrong answers.
    a. "How much change will I get back if I buy a drink for fifty cents and I give a dollar? Give me simplest form." → ½, but NOT 2/4.

Verifier Quality Matters – This is very important for difficult prompts!

Step 2: Getting Good Evaluation Data!

Well, we have training data now. But what next?

Build Evaluations! We won't know what works until we know what "works" means!



All of these need to be quantitatively tracked!

How Do We Train A Good Agent Model?
Second Step: Good Evaluations!

Off-The-Shelf Benchmarks (for (1), (2))

Agentic Coding:
- SWEBench
- TerminalBench
- AIDER

ToolCalling
- TauBench
- BFCL
- ComplexFuncBench

Knowledge
- HLE
- BrowserComp

….

Agent Harnesses (for (4))

SWEBench:
1. OpenHands
2. MiniSWE (bash only)
3. OpenSWE
4. Aider-SWEBench

HLE:
1. With Search Harness
2. Without Search

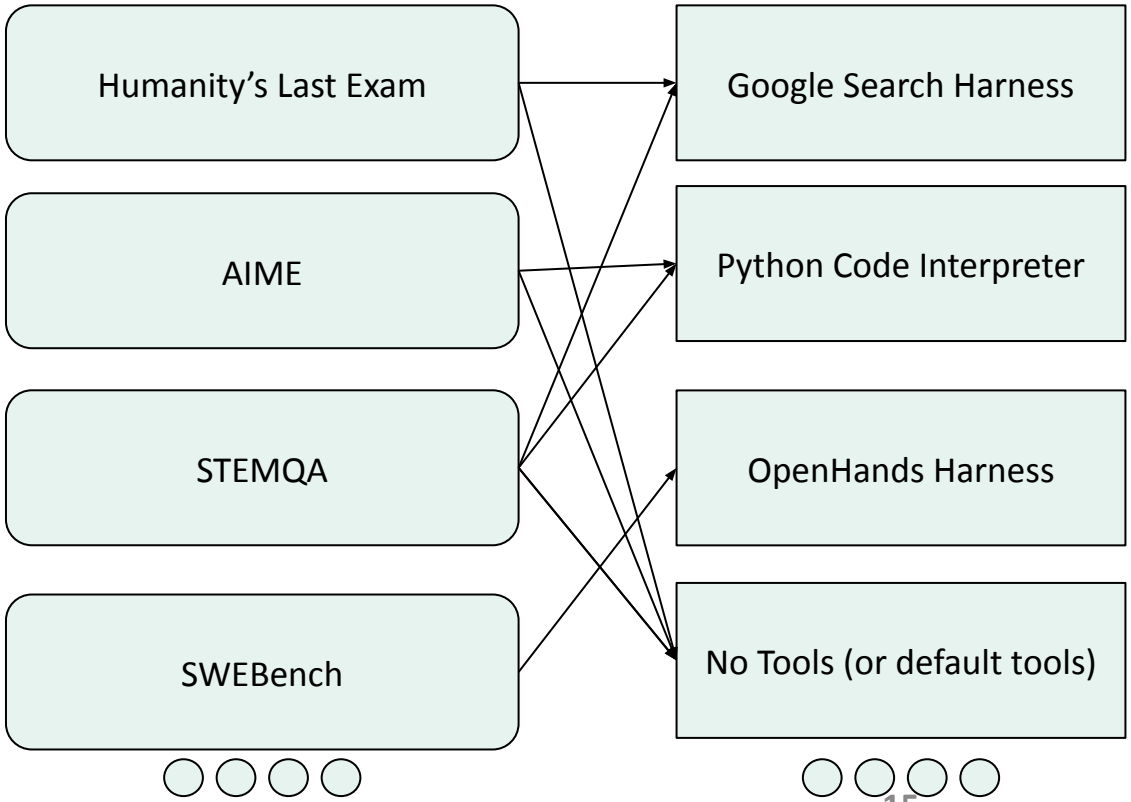
….

Focused "Unit-Test" Evaluations (for (3), (5))

1. Structured Output Adherence (structEval, etc.)
2. Instruction Following, w/ Conditional Instructions (IFBench, etc.)
3. Long Trajectories (NexusBench, etc.)

….

Holistic Measurements



Measure many tasks.
Measure many tasks under harnesses.

Keep the definition of intelligence holistic.

Good Evaluations Need To Account For ALL of This!

We should evaluate on:

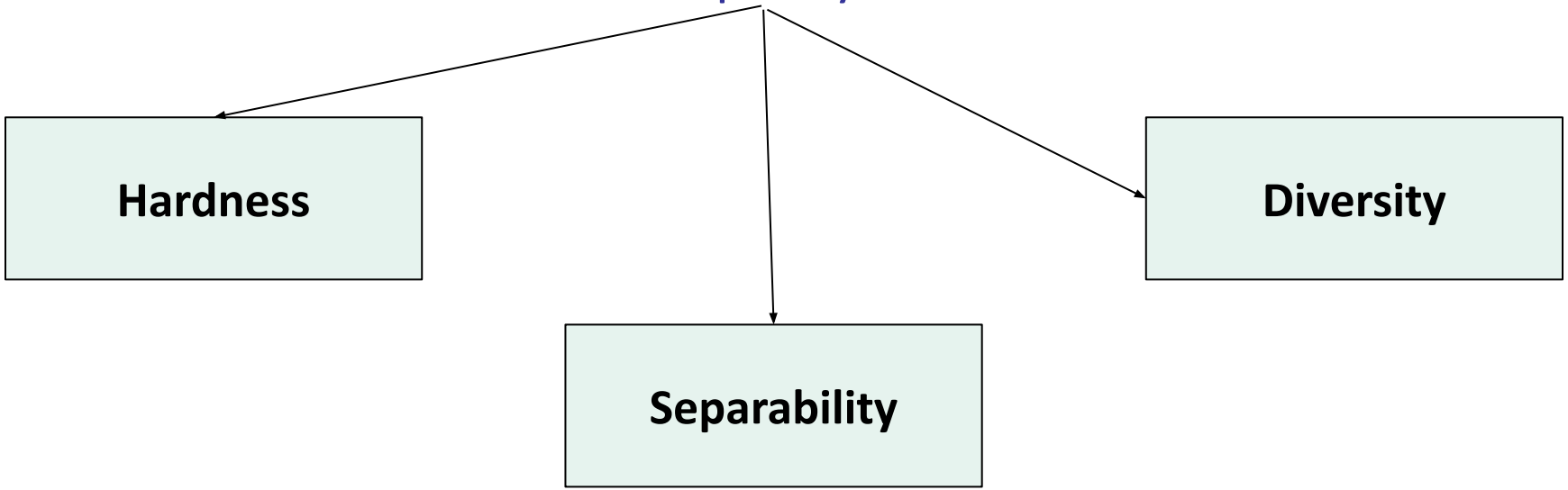
1. Many **tasks and verifiers**
2. Many different ways of letting the agent explore in the benchmarks' sandbox by seeing how the **accuracy changes as we swap out the harnesses**
3. **Many tools within each task** so that the agent understands **how to use tools in different** action spaces

When we choose a benchmark suite, we are **defining** what intelligence is. We **MUST** make sure it is a holistic and good measure!

Okay, we just curate a bunch of different tasks and harnesses to evaluate, is that enough? How do we even define if they're good measures of intelligence?

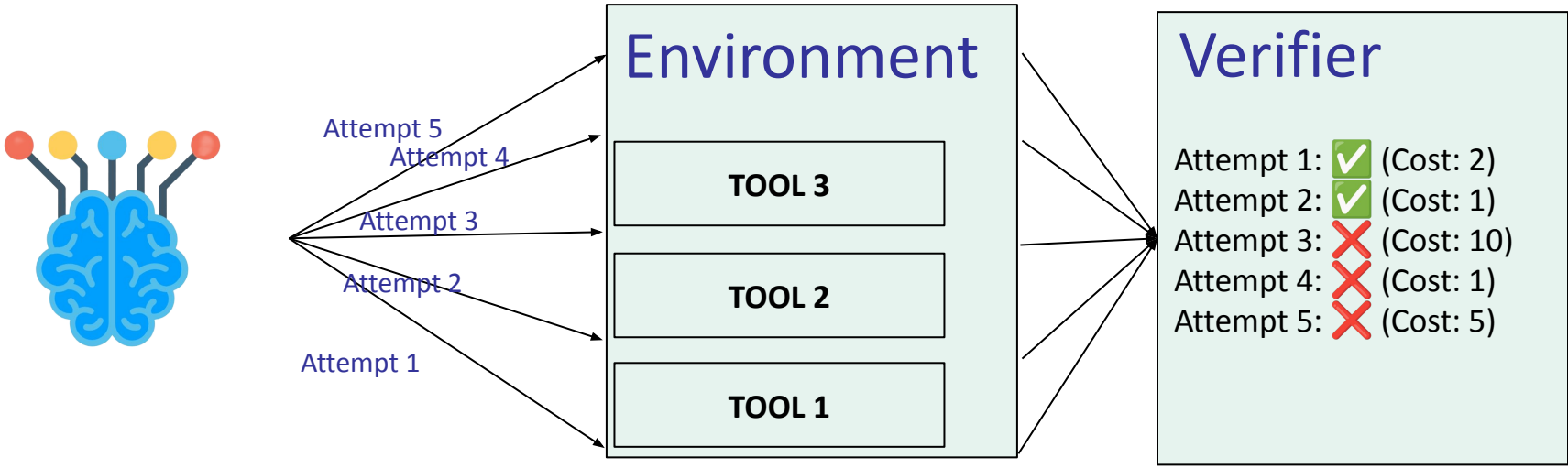
How Do We Quantify Benchmark Quality?

Beyond just evaluating on different verifiers and harnesses, how do we measure quality of the tasks?



Step 3: Training Well!

But What Does “Training” an Agent Mean?



You want the Agent LLM to attempt different actions in your environment and see what gets it to the final answer the best. Whichever gets you the best results, you want the model to **reinforce and repeat**.

1. You want each attempt to be very different from each other since you want the model to reinforce or discourage many different *styles of* attempts since that generalizes the best.
2. But each attempt **costs compute to sample and process!** You **don’t want very silly attempts** that has no change of even remotely being correct.

But What Does “Training” an Agent Mean?

Training an Agent means **maximizing** the **model’s ability** to use **tools** to achieve **correctness**!

You want to train to Imitate first and then to Explore:

1. SFT – **Imitate** demonstration trajectories that achieve correctness. Goal here is to **minimize** any silly or non-meaningful attempts.
2. RL – Let the model **explore** and **reinforce** trajectories that achieve correctness and **discourage** ones that do not. Goal here is to reinforce diverse, great attempts from the current model.

Why BOTH SFT and RL?

SFT Stage

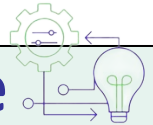
Maximize likelihood of sampling the demonstration trajectories and minimize likelihood of sampling ***anything else (like silly attempts)***

SFT is just there so that we can sample meaningful trajectories during RL with tractable compute!

1. Demonstration samples should be diverse
2. **SFT should be light**

We want to ensure **we don't affect diversity of responses** during RL by over SFT'ing on non-diverse examples!

RL Stage



Rollout Stage

Sample meaningful but very different actions for the task, environment, and tools.

We have limited compute here, so we want to ensure we get as **diverse answers as possible** that are **as meaningful as possible.**

Reward and Advantage Computation Stage

Figure out how correct each action sequence is, and how rare and useful certain actions are!

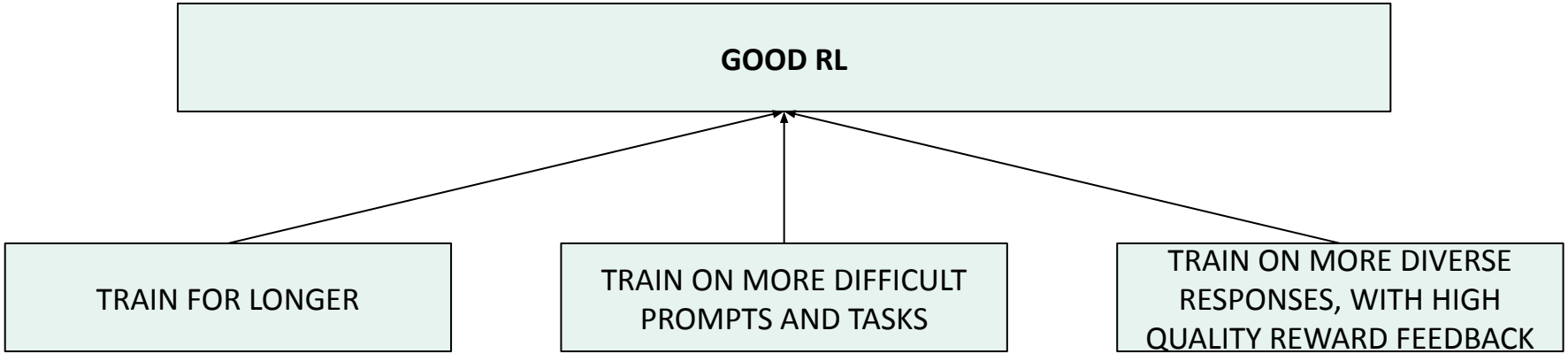
Backprop. Stage

Reinforce good actions and discourage bad actions!

SFT is just there to discourage meaningless and low-quality answers.

RL is there to truly reinforce intelligence.

Let's Dive Into RL!



Core Idea: Maintain **entropy balance**, controlled **model verbosity increase**

Core Idea: Train on tasks and prompts that are related to the task at hand, but that have **low (but NOT zero) chances of success** for the current model.

Core Idea: See as many different, meaningful answers per prompt as possible.

The core of good RL algorithms focus on these three pillars!

“Train Longer”

GOAL: *BE ABLE TO* TRAIN FOR LONGER

Without intervention, e.g., entropy or KL regularization, policy entropy is ***traded for reward predictably***

https://arxiv.org/pdf/2505.22617 [Entropy Mechanism of RL]

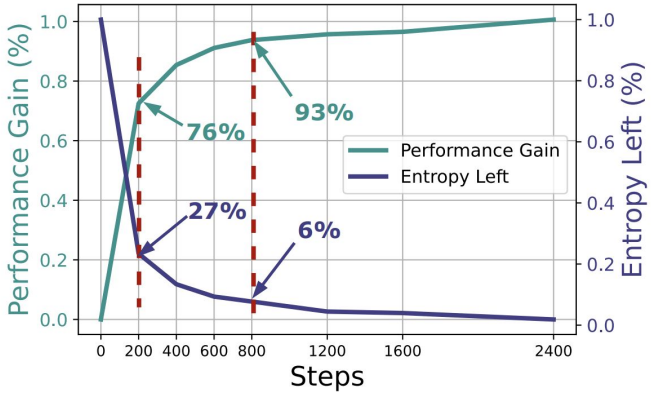


Figure 2: Avg. entropy consumption/performance gain (%) in 11 RL runs with different models.

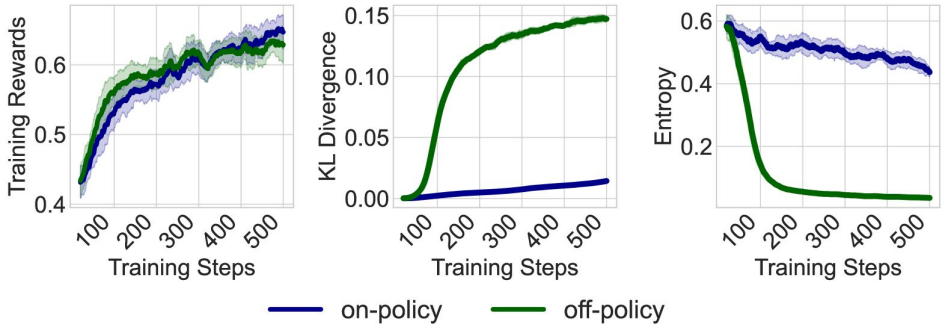
What is the right intervention to have a better trade-off between entropy and reward curve?

“Train Longer”

Question: What is the right intervention for us to be able to trade-off less aggressively and be able to train longer?

Reduce Biased Updates

Don’t do off-policy



https://arxiv.org/html/2505.23585v1

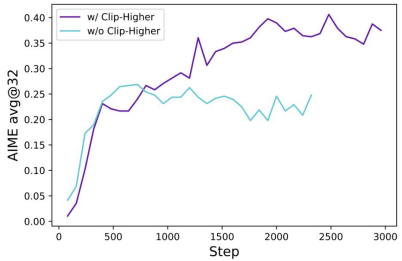
On Policy RL with Optimal Reward Baseline

Balance the Update

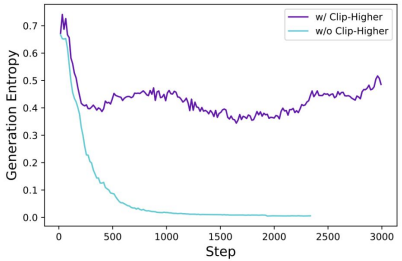
Clip correctly.

https://arxiv.org/pdf/2503.14476

DAPO



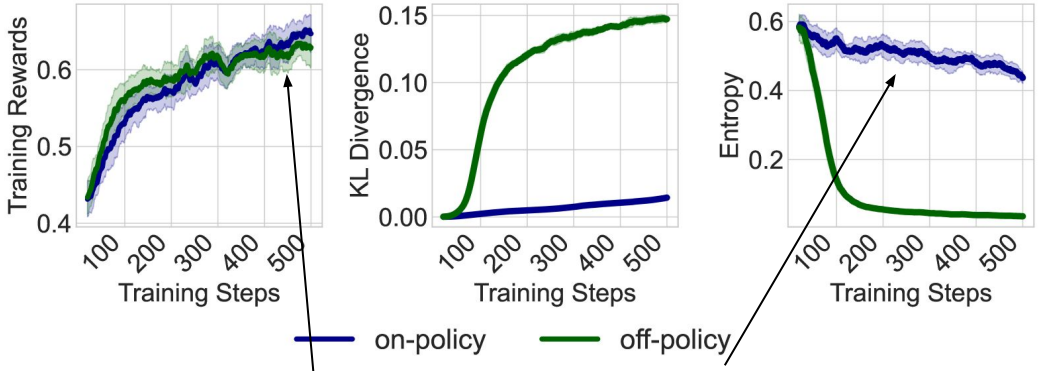
(a) Accuracies on AIME.



(b) Entropy of actor model.

Reduce Biased Updates

https://arxiv.org/html/2505.23585v1

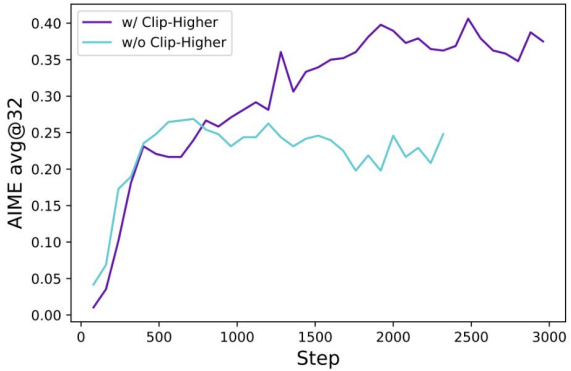


Better Entropy Control

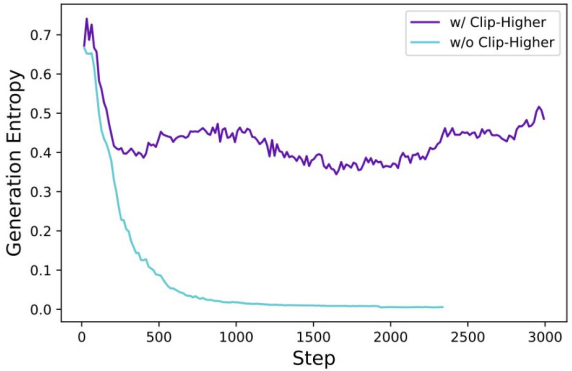
Better generalization

Dataset	Method	pass@1	pass@2
MATH-500	Off-Policy	92.97	95.60
	On-Policy	**93.90**	**96.21**
AIME 2024	Off-Policy	53.50	65.62
	On-Policy	**55.42**	**66.60**
AIME 2025	Off-Policy	36.21	44.05
	On-Policy	**38.37**	**46.58**

Balance The Update Strength



(a) Accuracies on AIME.



(b) Entropy of actor model.

ε_high allows us to leave more room for the increase of low-probability tokens.

https://arxiv.org/pdf/2503.14476

$$\mathcal{J}_{\text{DAPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{\sum_{i=1}^{G}|o_i|} \sum_{i=1}^{G}\sum_{t=1}^{|o_i|} \min\left(r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}}\right)\hat{A}_{i,t} \right) \right] \tag{10}$$

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

“Train Longer”

Question: What is the right intervention for us to be able to trade-off less aggressively and be able to train longer?

Reduce Biased Updates

Reduce Biased Updates

https://arxiv.org/pdf/2505.22312
Skywork Open Reasoner 1

Encourage Entropy Directly
Add a loss

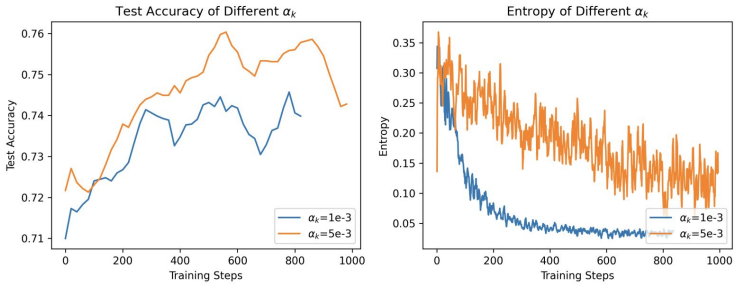


Figure 13: Preliminary experiments on mitigating entropy collapse by introducing entropy loss. We tested two different coefficients α_k = 1e-3 and 5e-3, and found that the entropy loss with the higher coefficient α_k, i.e., 5e-3, more effectively prevents entropy collapse and achieves higher test performance. **Left**: Accuracy curves on test benchmarks during RL training. **Right**: Entropy of generated responses during RL training.

Encourage Entropy Directly

https://arxiv.org/pdf/2505.22312

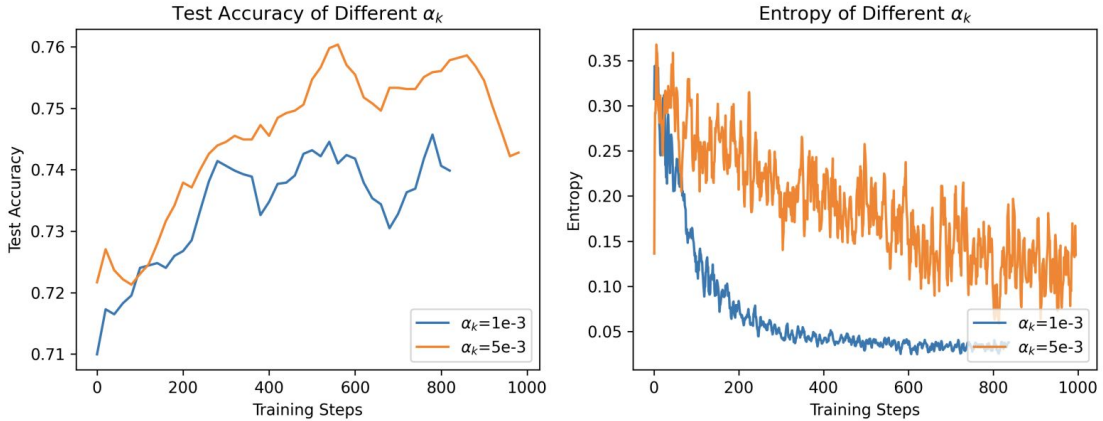


Figure 13: Preliminary experiments on mitigating entropy collapse by introducing entropy loss. We tested two different coefficients α_k = 1e-3 and 5e-3, and found that the entropy loss with the higher coefficient α_k, i.e., 5e-3, more effectively prevents entropy collapse and achieves higher test performance. **Left**: Accuracy curves on test benchmarks during RL training. **Right**: Entropy of generated responses during RL training.

$$\mathcal{L}^{\text{MAGIC}}(\theta) = -\frac{1}{T_k} \sum_{i \in \tilde{\mathcal{T}}_k} \sum_{j=1}^{M} \left\{ \sum_{t=0}^{|y_{ij}|-1} \min\left\{ \rho_{ij}^t(\theta) A_{ij}^t, \text{clip}\left(\rho_{ij}^t(\theta), 1-\varepsilon, 1+\varepsilon\right) A_{ij}^t \right\} + \alpha_k \mathbb{H}_{ij}^t(\theta) \right\}, \quad (3.1)$$

“Train On More Difficult Prompts and Tasks”

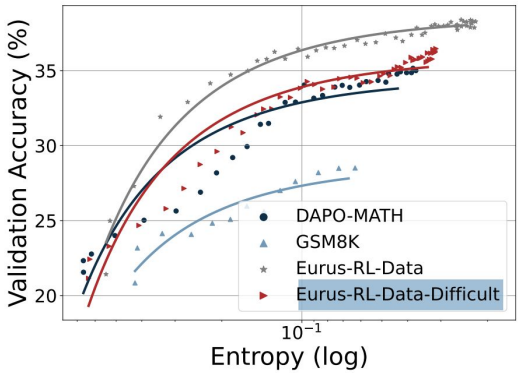
GOAL: TRAIN ON MORE *RELEVANT AND MEANINGFULLY* DIFFICULT PROMPTS AND TASKS

We can’t just dump hard prompts into the model and expect it to work. We need to ensure *harder prompts still improve the model by ensuring the model’s confidence is meaningfully correlated with the reward!*

*Proxy: Cov(pi_theta(y|x) * A(y,x), log_pi_theta(y|x))*

https://arxiv.org/pdf/2505.22617

Entropy Mechanism



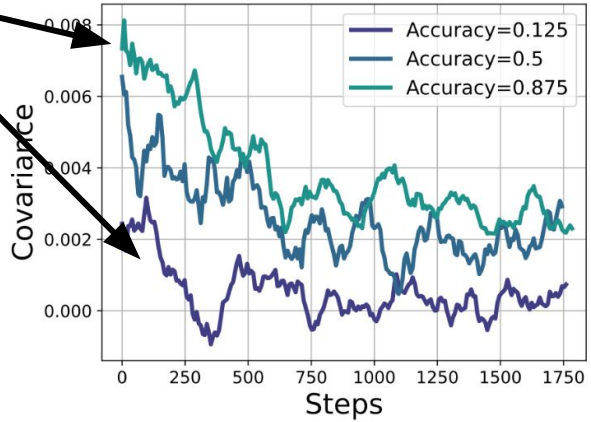


Figure 13: Training Qwen2.5-7B with different data.

“Train On More Difficult Prompts and Tasks”

https://arxiv.org/pdf/2505.22617

You can’t train on anything too easy.

Nor can you blindly throw in super hard prompts.

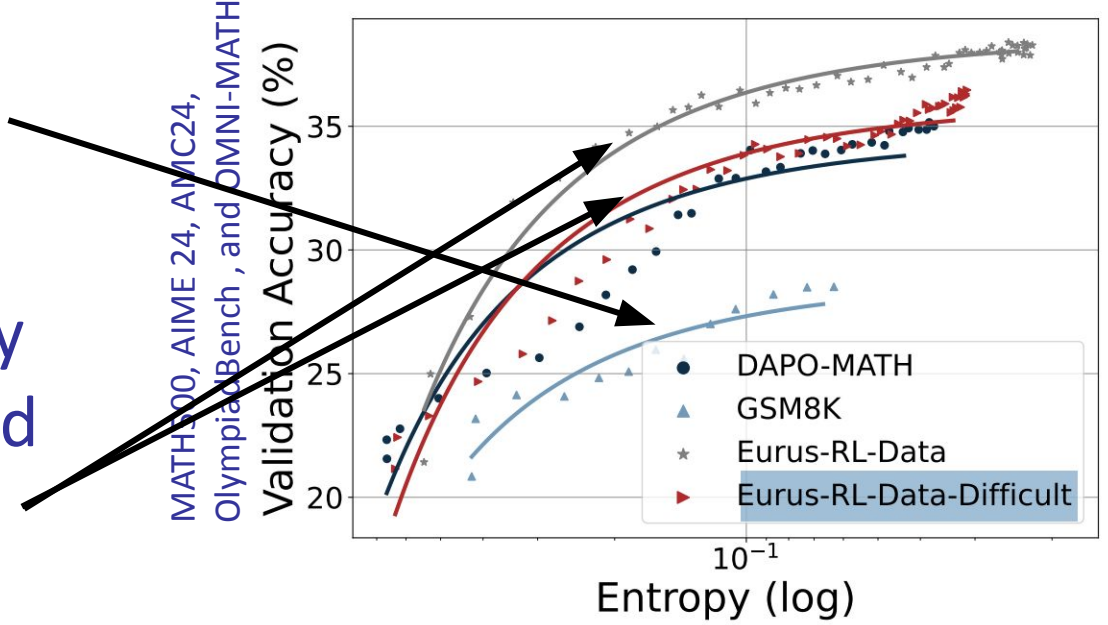


Figure 13: Training Qwen2.5-7B with different data.

“Train On More Difficult Prompts and Tasks”

What Interventions Can We Do?

Try to Improve Learning Signal

https://arxiv.org/pdf/2504.09696

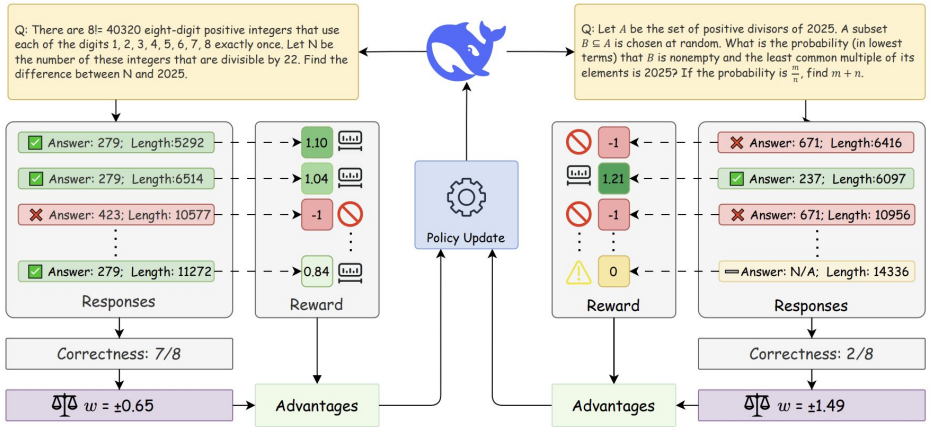


Figure 1: The GRPO-LEAD framework assigns length-regularized positive rewards to correct answers and explicit penalties to incorrect ones. A difficulty-based weight w used for advantage reweighting is determined from the empirical correctness of responses for each question. This weight then scales the advantages derived from each question, prioritizing harder questions over easier ones during the policy update to foster robust reasoning.

Improve Learning Signal: Reward Harder Prompts More

Ablation Setting	AIME24			AIME25		
	Cons@32	Pass@1	Len_{avg}	Cons@32	Pass@1	Len_{avg}
Deepseek-7B	0.767	0.431	6,990	0.467	0.292	7,113
GRPO + len. reward	0.767	0.438	**5,275**	0.533	0.308	**5,210**
+ adv. reweighting	0.767	0.458	5,323	**0.567**	0.325	5,437
+ explicit penalty	**0.800**	**0.470**	6,104	**0.567**	**0.345**	6,308

https://arxiv.org/pdf/2504.09696

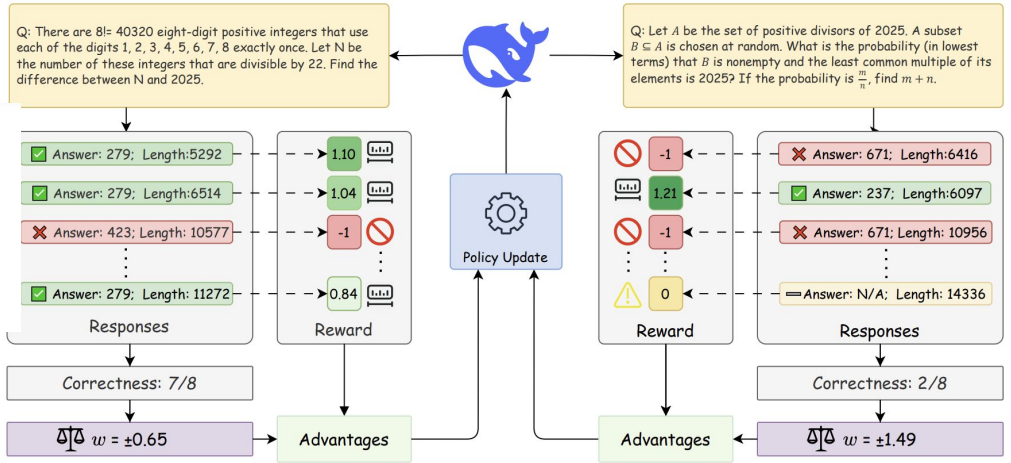


Figure 1: The GRPO-LEAD framework assigns length-regularized positive rewards to correct answers and explicit penalties to incorrect ones. A difficulty-based weight w used for advantage reweighting is determined from the empirical correctness of responses for each question. This weight then scales the advantages derived from each question, prioritizing harder questions over easier ones during the policy update to foster robust reasoning.

“Sample Better”

GOAL: *Sample better so we can* TRAIN ON MORE DIVERSE RESPONSES, WITH HIGH QUALITY REWARD FEEDBACK

We want to scale compute to ensure we get good responses.

Scale-Up Compute Per Answer (Parallel Reasoning)

Model	Comp-Math-24-25			
	AIME24	AIME25	HMMT-24-25	All
DeepSeek-R1-0528	85.9 (93.3)	80.5 (86.7)	67.5 (82.1)	71.2 (84.0)
+ Self GenSelect@32	93.3	90.0	85.7	87.1
QwQ-32B	78.1 (86.7)	67.2 (76.7)	56.1 (64.3)	60.0 (68.4)
+ Self GenSelect@32	90.0	73.3	70.4	73.0

https://arxiv.org/pdf/2507.17797 GenSelect

https://arxiv.org/pdf/2508.15260
DeepConf

Beam Searching Over Reasoning

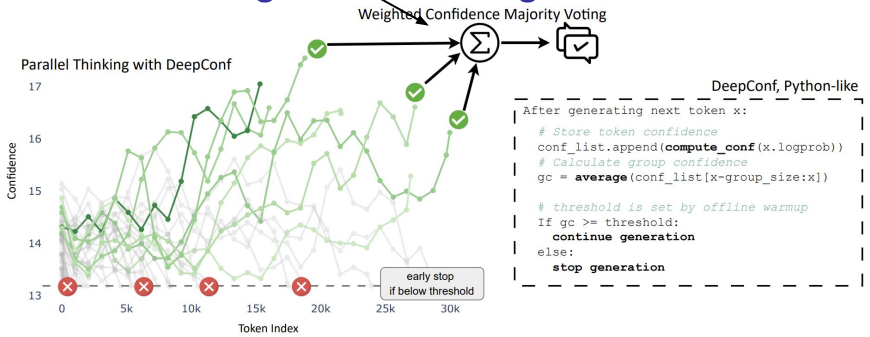


Figure 1: Up: DeepConf on AIME 2025. Down: Parallel thinking using DeepConf.

GenSelect

Model	Comp-Math-24-25			
	AIME24	AIME25	HMMT-24-25	All
DeepSeek-R1-0528	85.9 (93.3)	80.5 (86.7)	67.5 (82.1)	71.2 (84.0)
+ Self GenSelect@32	93.3	90.0	85.7	87.1
QwQ-32B	78.1 (86.7)	67.2 (76.7)	56.1 (64.3)	60.0 (68.4)
+ Self GenSelect@32	90.0	73.3	70.4	73.0

We throw all the sampled answers and pick the best one

GenSelect Prompt

```
You will be given a challenging math problem followed by {num_solutions} solutions.
Your task is to systematically analyze these solutions to identify the most
mathematically sound approach.

Input Format:
Problem: A complex mathematical word problem at advanced high school or college
level
Solutions: Detailed solutions indexed 0-{max_idx}, each concluding with an answer in
 \boxed{{}} notation

YOUR TASK

Problem: {problem}

Solutions:
{solutions}

Evaluation Process:

1. Initial Screening
- Group solutions by their final answers
- Identify and explain mathematical contradictions between different answers
- Eliminate solutions with clear mathematical errors

2. Detailed Analysis
For remaining solutions, evaluate:
- Mathematical precision and accuracy
- Logical progression of steps
- Completeness of mathematical reasoning
- Proper use of mathematical notation, including \boxed{{}}
- Handling of edge cases or special conditions
- For solutions containing and addressing errors, evaluate the error identification
and correction methodology.

3. Solution Comparison
Compare viable solutions based on:
- Efficiency of approach
- Clarity of mathematical reasoning
- Sophistication of method
- Robustness of solution (works for all cases)

Your response should include:
1. Brief analysis of conflicting answers
2. Detailed evaluation of mathematically sound solutions
3. Justification for eliminating incorrect solutions
4. Clear explanation for selecting the best approach

End your evaluation with exactly:
Judgment: [IDX]
where IDX is the index 0-{max_idx} of the best solution.
```

DeepConf

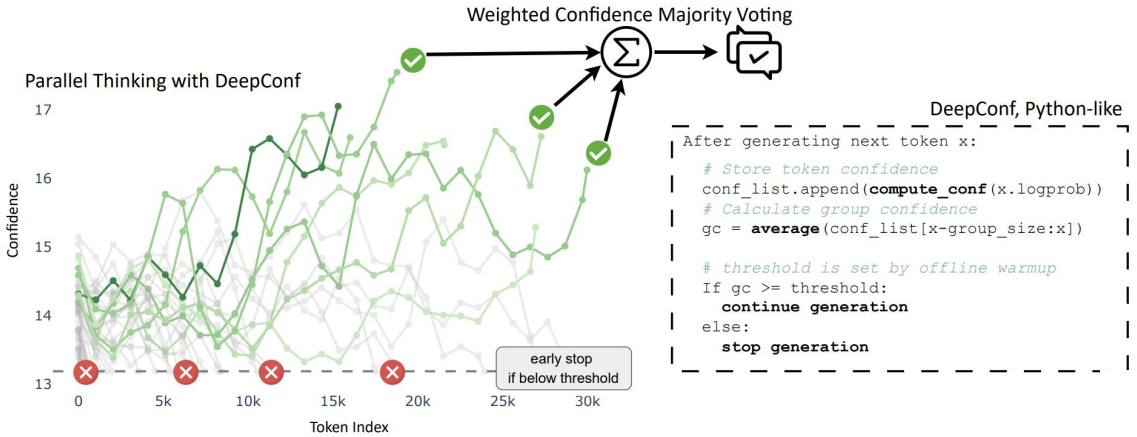
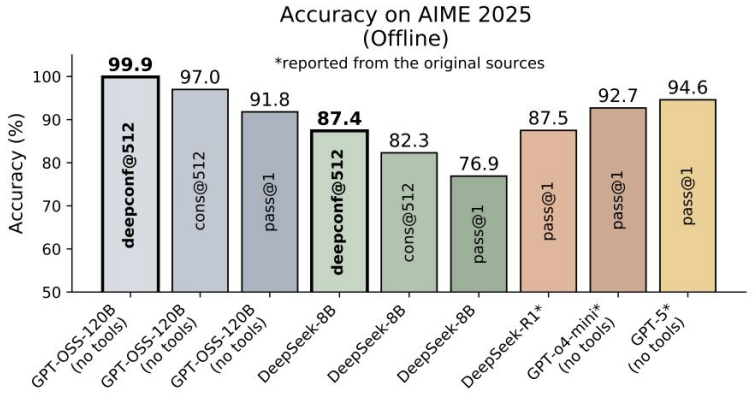


Figure 1: Up: DeepConf on AIME 2025. Down: Parallel thinking using DeepConf.

Only do majority voting on the sampled answers that remain above a threshold of confidence



Putting it all together: How Do We Train A Good Agent Model?

Third Step: Good Recipes!

SFT Data

Expert Demonstrations:

- Human Curated
- Synthetic (Strong Teacher Models)

Self-Distilled

- Rejection Sampling (but from verifier feedback)

RL Recipes

Train Longer: *Encourage Stability*

- Entropy Balance (SkyWorks, etc.)
- Reward, Advantage Shaping (DAPO, DrGRPO, etc.)

Sample Better: *Encourage Better Exploration*

- Better, Diverse Trajectories by Sampling Better (DeepConf., ParaThinker, etc.)

Train Harder: *Train on Harder Tasks*

- Harder Inverted Formulations (DuPO, etc.)
- Harder Prompts by Pass@K -> Pass@1 reduction (GenSelect, etc.)

Step 4: How Do We Actually Do This?

What Comes Next?

STEP 1: Have a ***collection*** where we curate environments and verifiers to scale up intelligence.

STEP 2: Figure out the *best* ***definition*** of what intelligence is.

STEP 3: Release ***algorithms*** to train longer, on harder prompts, and with diverse responses.

What Comes Next?

STEP 1: Have a ***collection*** where we curate environments and verifiers to scale up intelligence.

1. Crowd-source Environments/Evals/Recipes:
   a. From individual contributors
   b. From companies
   c. From researchers
   d. From …..

Diverse, high quality signals to scale up RL in a single place!

What Comes Next?

STEP 1: Have a ***collection*** where we curate environments and verifiers to scale up intelligence.

STEP 2: Figure out the *best* ***definition*** of what intelligence is.

1. Analyze the strengths and weaknesses of benchmarks

2. Include it in the collection so everyone can use it during RL scale up to ensure we're measuring intelligence holistically.

What Comes Next?

STEP 1: Have a ***collection*** where we curate environments and verifiers to scale up intelligence.

STEP 2: Figure out the *best* ***definition*** of what intelligence is.

STEP 3: Release ***algorithms*** to train longer, on harder prompts, and with diverse responses.

Create New Algorithms That Are Stable, Meaningful, and result in Diverse Responses. Add it into the collection so everyone can use it for all the environments we're curating.

Open Questions

Humans engage in many different environments, focus on some lessons more than others, learn from both exploring by themselves and from teachers, and are very often evaluated for intelligence for various tasks. How do we mimic this?

1. Much like humans learn from so many environments, we should have our models do the same. What is the right design for this collection of open source environments, evaluations, and algorithms?

2. Much like humans focus on some lessons more than others, we should do the same. What is the right algorithm that allow us to do so without hurting stability?

3. Much like humans learn from both their own exploration and from teachers, we should do the same. What is the right balance? Can we do both at the same time?

4. Much like humans are evaluated for their intelligence, we should do the same. How can we compare different definitions of intelligence?