

PyTorch Conference · Lightning Talk

When Rollout and Training Disagree

Lessons from Building a Mismatch-Free RL Engine

rollout $\log p = -0.694$

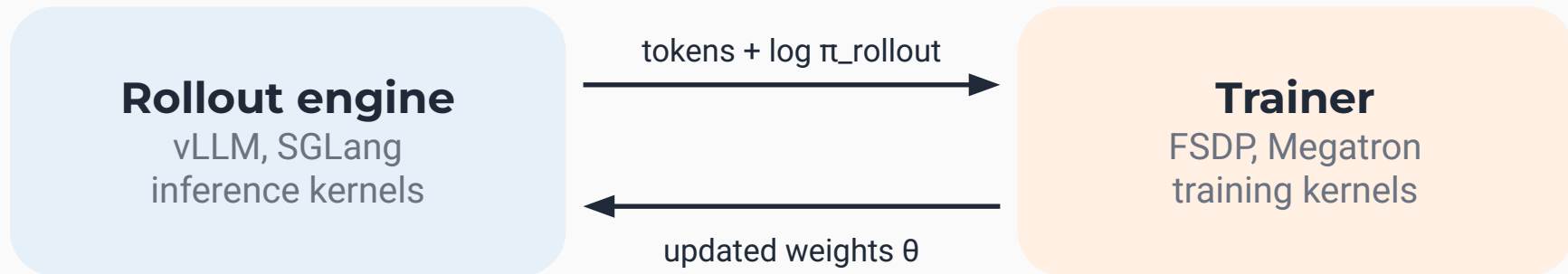
≠

training $\log p = -0.827$

Tianle Zhong (University of Virginia) & **Neiwen Ling** (ByteDance)

This work was done in collaboration with
Yifan Pi, Zijun Wei, Tianshu Yu, Geoffrey Fox, Peng Wu, and Xiao Yu

On-policy RL needs matching probabilities



$$\delta t = \log \pi_{\text{train}}(a_t \mid s_t) - \log \pi_{\text{rollout}}(a_t \mid s_t)$$

Same weights and same tokens, so on-policy RL needs $\delta t = 0$ for every token.

Same tokens, different probabilities

token	The	problem	states	that	there	exist	real	numbers
$\log \pi_{\text{rollout}}$	-0.279	-0.063	-0.314	-0.694	-0.000	-0.030	-0.000	-0.000
$\log \pi_{\text{train}}$	-0.278	-0.063	-0.314	-0.827	-0.000	-0.038	-0.000	-0.000
δt	0.001	0.000	0.000	-0.133	0.000	-0.008	0.000	0.000

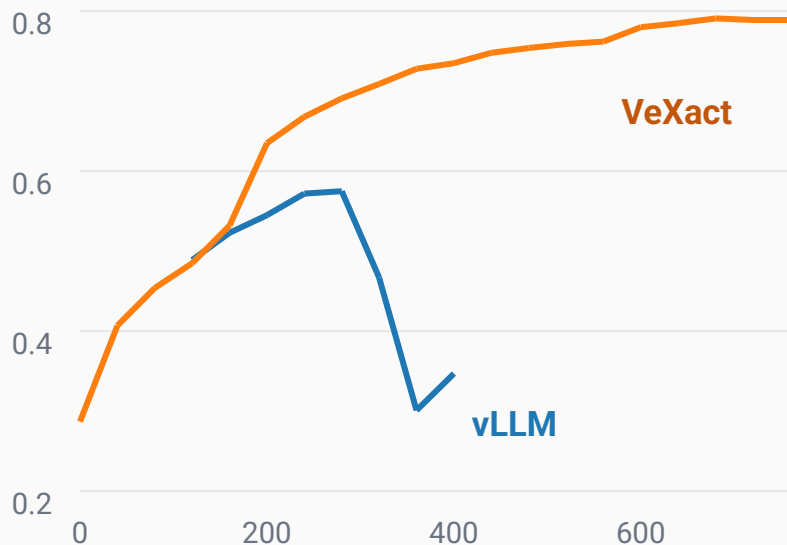
At “that”, the trainer’s top-1 token is “:\n\n”. The two engines would not even write the same sentence. Most positions agree to three decimals; a few do not.

~1.0

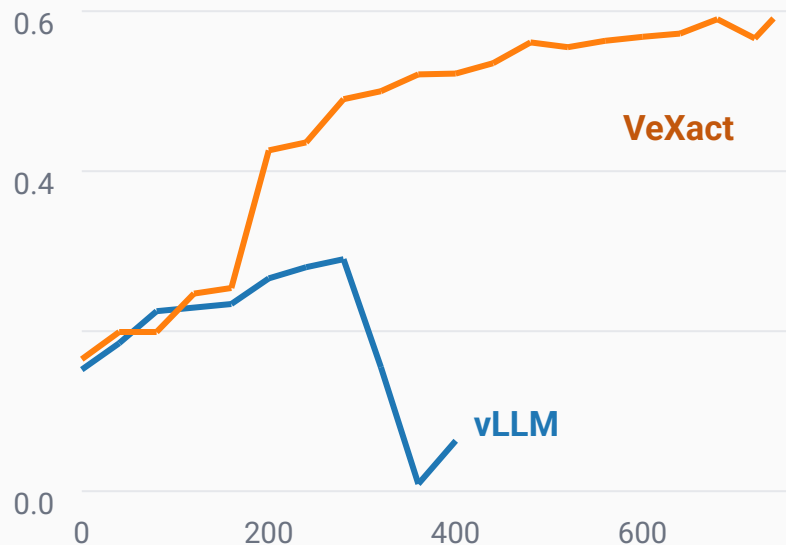
max $|\delta t|$ in a training batch, while the batch mean stays below 0.07

Lesson: Mismatch can collapse RL training

Qwen3-30B-A3B (MoE) Training reward

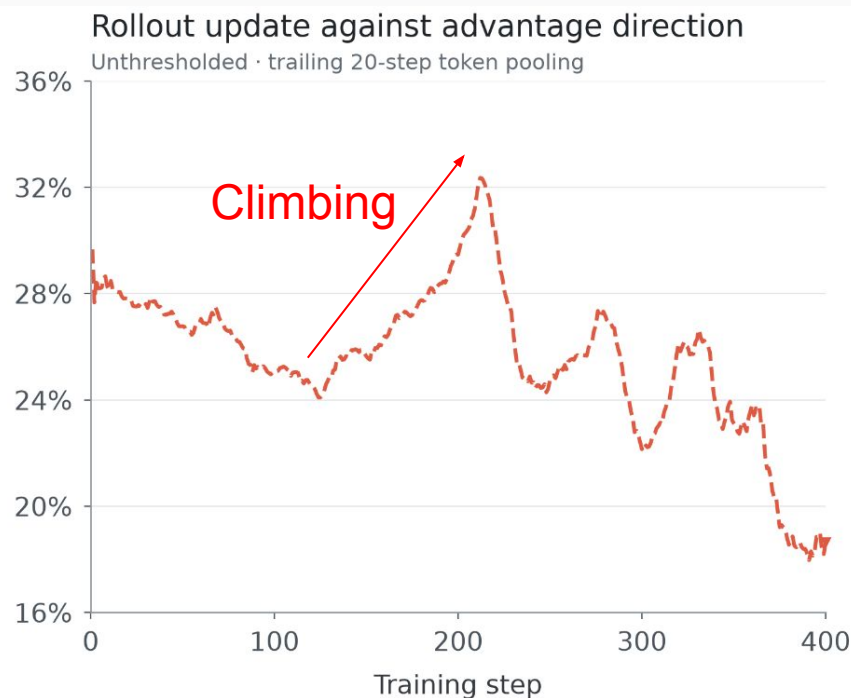
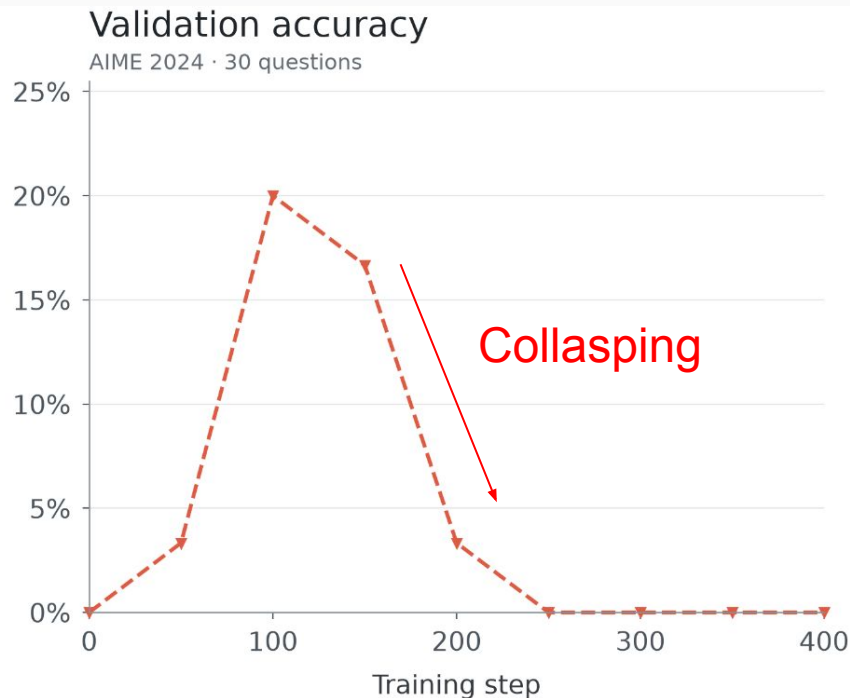


AIME 2024 (mean@32)



Only the rollout engine differs: vLLM's AIME score fell from 0.29 to 0.07 after step 280.

Insight: Why behavior log-probs fail



---vLLM behavior log-probs + PPO Clip

VeXact Design

Support dense/MoE models with full/GQA/SWA/MLA attentions

Verified on H100/B200 GPUs with reasonable throughput

Inference

VeXact



Training

FSDP / VeOmni



Shared Hugging Face model + batch-invariant kernels

Fixed tiling and reduction order across batch size

Same weights + tokens = identical token log-probs

Lightweight, auditable code (~8.4K lines of Python)

VeXact keeps rollout-correction KL at zero

Rollout-correction K3 KL (log scale)



Rollout-correction KL: vLLM rises sharply; VeXact stays at zero throughout the run.

Positive KL: log scale; zero shown separately. Curve traced from the supplied figure (approximate).

For TIM algorithm researchers

A zero-mismatch reference for rollout correction

Compare your correction against VeXact

Same model, data and RL objective

Learning curve

Does your correction recover the TIM-free learning curve?

Correction bias

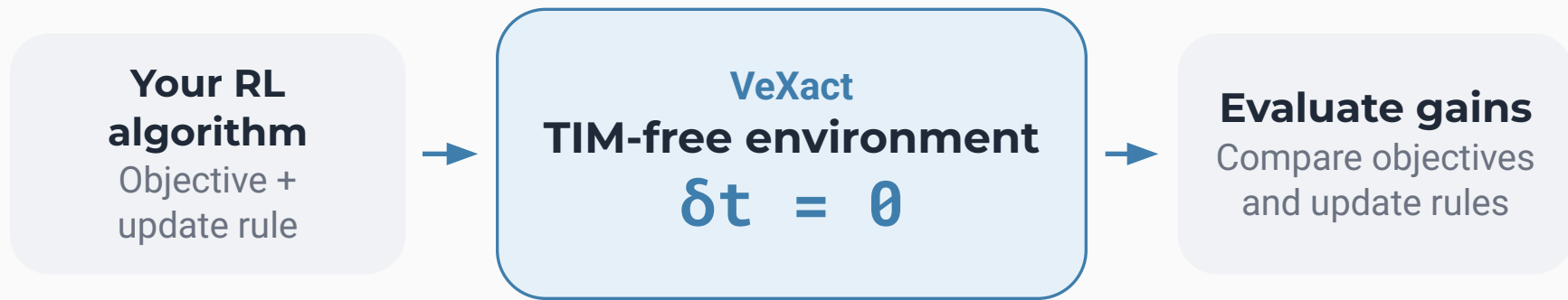
What bias does clipping or filtering introduce?

Separate mismatch mitigation from optimization effects

For general RL algorithm researchers

Curious about the performance limits of your RL algorithm?

Try your algorithm in a **TIM-free environment**



Test algorithmic gains before adding engine mismatch

Getting started with VeXact

Start with an end-to-end verl recipe

```
uv sync --extra gpu --extra verl --extra veomni  
bash examples/getting_started/run_qwen3_1b7.sh
```

Verify alignment

Use the verification examples to check token log-probs.

Run your method

Plug in your RL objective or rollout correction.

Recipes and supported configurations: [GitHub README](#)

Try VeXact

A zero-mismatch reference for RL research

Code

Zero-mismatch rollout for verl
FSDP and VeOmni support

github.com/verl-project/vexact

Paper

Diagnosing Training-Inference Mismatch
in LLM Reinforcement Learning via a
Zero-Mismatch Reference

arxiv.org/abs/2605.14220

Neiwen Ling · ByteDance
Tianle Zhong · The University of Virginia