Absolute Zero: Reinforced Self-play Reasoning with Zero Data

Eric Kim and Ayush Pai

Georgia Tech

Problem Setting

- LLM improvement through Reinforcement with Verifiable Rewards (RLVR)

- But RLVR depend on expertly curated datasets
  - Question-Answer (QA) pairs
  - Reasoning traces

- **Can a model learn reasoning using only RLVR, without any external data?**

Why this Problem Matters

- **Long term scalability:**
  - Strong reasoning models require more more high quality data
- **Human curation does not scale:**
  - Creating expert-level tasks is expensive and slow
- **Potential capability bottleneck:**
  - Human knowledge/imagination may bound model improvement
  - Problem if AI passes human intelligence in the future

Georgia Tech

Prior Work

- **Self-play paradigm dates back to early 2000s:**
  - Schmidhuber proposed two agent method for theoretically unlimited progress
  - But early models were too weak + no reliable verifier

- **AlphaGo Zero (2017)**
  - Model plays earlier versions in unsupervised setting
  - Success in self-play when environment has verifiable rewards

Prior Work

- **Genius (2025)**
  - Unsupervised self-training from unlabeled queries
  - But relies on external task distribution (human-provided queries)

- **“Zero” RLVR (DeepSeek-AI 2025)**
  - No cold-start distillation data (no human or AI generated reasoning traces)
  - RLVR directly on base model with task rewards
  - But depend on expert QA pairs

Background

- **Supervised Fine-Tuning (SFT)**
  - Dataset: $\mathcal{D} = \{(x, c^\star, y^\star)\}$
  - Provided by experts: x: query, c^* : reference chain of thought (CoT), y^*: correct answer
  - Model trained to imitate expert-labeled distribution by minimizing:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(x,c^\star,y^\star)\sim\mathcal{D}} \log \pi_\theta \left(c^\star, y^\star \mid x\right).$$

  - Limitation: poor scalability of expert data

Georgia Tech

Background

- **Reinforcement Learning with Verifiable Rewards (RLVR)**
  - Dataset: $\mathcal{D} = \{(x, y^\star)\}$
  - c^* not needed: model generates own CoT
  - Calculates verifiable reward based on correct answer $r(y, y^\star)$
  - Policy π_θ trained to maximize expected reward:

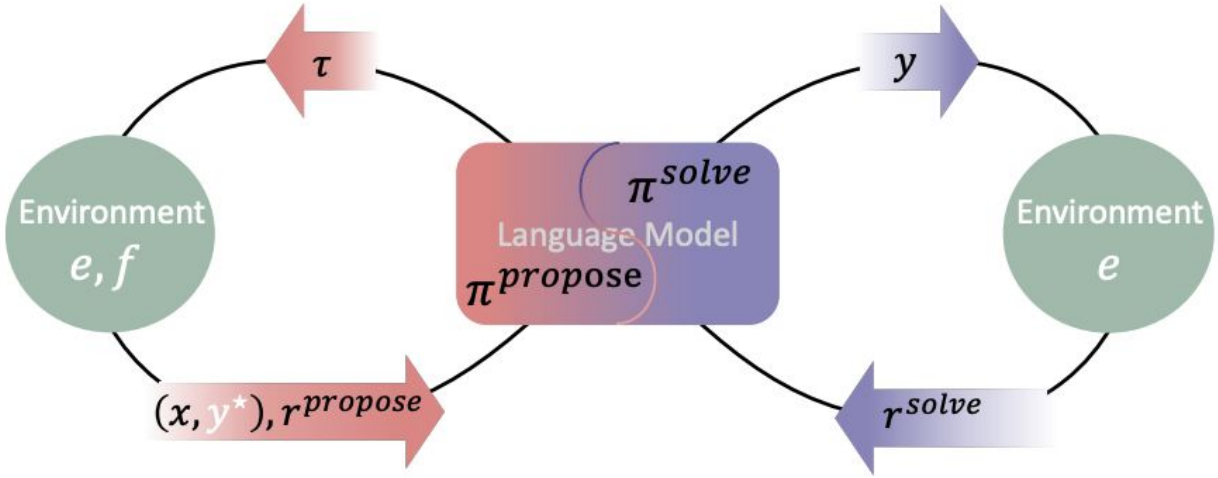
$$J_{\text{RLVR}}(\theta) = \mathbb{E}_{(x,y^\star)\sim\mathcal{D},\ (c,y)\sim\pi_\theta(\cdot\,|x)}\left[r(y, y^\star)\right]$$

  - Limitation: still rely on expert labeled query x and answer y^*

Georgia Tech

Absolute Zero Paradigm

- **Big Idea: model suggests tasks, solves them, and learns**
- Policy π_θ (LM) works as both proposer $\pi_\theta^{\text{propose}}$ and solver $\pi_\theta^{\text{solver}}$



Georgia Tech

Absolute Zero Paradigm

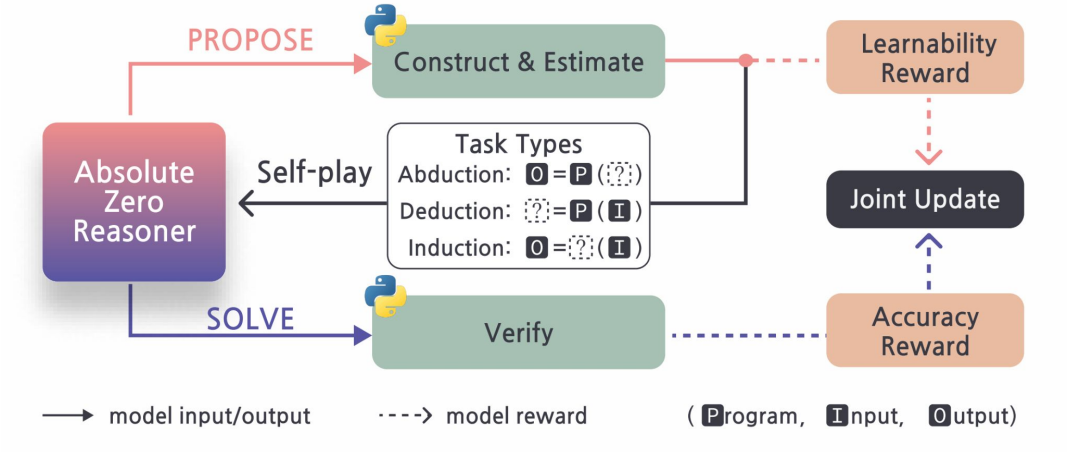
$$\mathcal{J}(\theta) := \max_{\theta} \; \mathbb{E}_{z \sim p(z)} \left[\mathbb{E}_{(x,y^\star) \sim f_e(\cdot|\tau), \tau \sim \pi_\theta^{\text{propose}}(\cdot|z)} \left[\lambda r_e^{\text{propose}}(\tau, \pi_\theta) + \mathbb{E}_{y \sim \pi_\theta^{\text{solve}}(\cdot|x)} \left[r_e^{\text{solve}}(y, y^\star) \right] \right] \right]$$

1. $\pi_\theta^{\text{propose}}$ proposes task τ
2. f transforms/validates it to a valid reasoning task $(x, y^\star)$
   a. $\pi_\theta^{\text{propose}}$ receives learnability reward $r_e^{\text{propose}}(\tau, \pi_\theta)$
   b. Measures expected improvement in π_θ after training on τ
3. Standard RL step:
   a. Solver produces answer $y \sim \pi_\theta^{\text{solve}}(\cdot \mid x)$
   b. $\pi_\theta^{\text{solver}}$ receives solution reward $r_e^{\text{solve}}(y, y^\star)$ using environment *e* as verifier
   c. λ balances tradeoff between task exploration vs. solution learning

Georgia Tech

Absolute Zero Reasoner (AZR)

- Concrete implementation of the Absolute Zero paradigm
  - Single language model used as proposer and solver
  - Generates its own reasoning tasks during training
  - Uses executable environment as verifier
  - model proposes task → solves task → receives reward → updates policy



Georgia Tech

AZR Reward Design

- **Proposer** reward based on solver success rate
  - Uses Monte-Carlo estimate of solve accuracy
  - Encourages tasks with intermediate difficulty
- **Solver** reward from execution verification
  - Binary reward based on correctness
- Composite reward structure
  - Integrates both proposer and solver reward
  - Format aware penalty

$$r_{\text{propose}} = \begin{cases} 0, & \text{if } \bar{r}_{\text{solve}} = 0 \\ 1 - \bar{r}_{\text{solve}}, & \text{otherwise.} \end{cases}$$

$$r_{\text{solve}} = \mathbb{I}_{(y=y^\star)}$$

$$R(y_\pi) = \begin{cases} r_{\text{role}} & \text{correctly formatted, role} \in \{\text{propose,solve}\} \\ -0.5 & \text{response is wrong but well-formatted,} \\ -1 & \text{answer has formatting errors,} \end{cases}$$

Georgia Tech

AZR Modes of Reasoning

- **Tasks defined as program triplet (p, i, o)**
  - p = program, i = input, o = output
  - Environment executes p(i) → o
- **Deduction**
  - proposer generates (p, i),
  - solver predicts o, and correctness is checked by executing p(i)
- **Abduction**
  - proposer generates (p, o)
  - solver predicts i, valid if executing p(î) reproduces o
- **Induction**
  - proposer generates {(i, o)} examples from program p
  - solver predicts p, verified on unseen inputs

program space $\mathscr{P}$, input space $\mathscr{I}$ and output space $\mathscr{O}$

$p \in \mathscr{P} \quad i \in \mathscr{I} \quad o \in \mathscr{O}$

Georgia Tech

AZR Learning Algorithm

Algorithm 1 Self-Play Training of Absolute Zero Reasoner (AZR)

Require: Pretrained base LLM π_θ; batch size B; #references K; iterations T

$\mathcal{D}_{\text{ded}}, \mathcal{D}_{\text{abd}}, \mathcal{D}_{\text{ind}} \leftarrow \textsc{InitSeeding}(\pi_\theta)$ ▷ see §3.3.1

for $t \leftarrow 1$ to T **do**

  for $b \leftarrow 1$ to B **do** ▷ **PROPOSE PHASE**

    $p \sim \mathcal{D}_{\text{abd}} \cup \mathcal{D}_{\text{ded}}$ ▷ sample a program for induction task proposal

    $\left(\{i_\pi^n\}_{n=1}^N,\ m_\pi\right) \leftarrow \pi_\theta^{\text{propose}}(\text{ind}, p)$ ▷ generate N inputs and a description

    if $\{(i_\pi^n, o_\pi^n)\}_{n=1}^N \leftarrow \textsc{ValidateAndConstruct}(p, \{i_\pi^n\}, \textsc{syntax})$ **then** ▷ validate I/Os, see §3.3.3

      $\mathcal{D}_{\text{ind}} \leftarrow \mathcal{D}_{\text{ind}} \cup \{(p, \{(i_\pi^n, o_\pi^n)\}, m_\pi)\}$ ▷ update *induction* buffer

    for $\alpha \in \{\text{ded}, \text{abd}\}$ **do**

      $(p_k, i_k, o_k)_{k=1}^K \sim \mathcal{D}_\alpha$ ▷ sample K reference examples

      $(p_\pi, i_\pi) \leftarrow \pi_\theta^{\text{propose}}(\alpha, \{(p_k, i_k, o_k)\})$ ▷ propose new task

      if $o_\pi \leftarrow \textsc{ValidateAndConstruct}(p_\pi, i_\pi, \textsc{syntax,safety,determinism})$ **then** ▷ see §3.3.3

        $\mathcal{D}_\alpha \leftarrow \mathcal{D}_\alpha \cup \{(p_\pi, i_\pi, o_\pi)\}$ ▷ update *deduction* or *abduction* buffers

  for all $\alpha \in \{\text{ded}, \text{abd}, \text{ind}\}$ **do** ▷ **SOLVE PHASE**

    $(x, y^\star) \leftarrow \textsc{SamplePrepareTasks}(\mathcal{D}_\alpha, B, t)$ ▷ $x, y^\star$ prepared based on α and t, see §3.3.3

    $y_\pi \sim \pi_\theta^{\text{solve}}(x)$

  Reward: Use proposed task triplets and solved answers to get $r_{propose}$ & r_{solve} ▷ see §3.1

  RL update: use Task Relative REINFORCE++ to update π_θ ▷ see §3.3.5

Georgia Tech

AZR Learning Algorithm - Task Buffers

- Maintain task buffers for each reasoning mode
  - store valid task triplets (p, i, o)
  - initialized using pretrained model
  - used as references for future proposals
- Each iteration samples from buffers $(p_k, i_k, o_k) \sim D_\alpha$
  - $\alpha \in \{ded, abd, ind\}$
  - K reference examples used for prompting
  - enables self-generated curriculum
- Composite reward structure
  - Integrates both proposer and solver reward
  - Format aware penalty

Program Triplet

Input: `"Hello World"`

```
1 def f(x):
2     return x
```

Output: `"Hello World"`

Figure 5. **The Seed AZR Zero Triplet.**

Georgia Tech

AZR Learning Algorithm - Propose Tasks

- Proposer generates new tasks using policy
  - $\tau \sim \pi_\theta^{propose}$
  - conditioned on task type α
  - conditioned on reference examples
- Environment validates proposed task
  - o = p(i)
  - check syntax, safety, **determinism** $\forall p \in \mathcal{P}_{\text{deterministic}},\ \forall i \in \mathcal{I},\ \left(\lim_{j \to \infty} p(i)^{(1)} = p(i)^{(2)} = \cdots = p(i)^{(j)} \right)$
  - discard invalid programs
- Valid tasks added to buffer
  - $D_\alpha \leftarrow D_\alpha \cup (p,i,o)$
  - grows dataset during training
  - creates automatic curriculum

Georgia Tech

AZR Learning Algorithm - Solve & Update

- Sample tasks from buffers
  - $(x, y^*) \leftarrow \textit{SamplePrepareTasks}(D_\alpha)$
  - convert triplet → (prompt, expected output)
  - used for solver training
- Solver predicts answer
  - $y \sim \pi_\theta^{solve}(x)$
  - Reward: $r_{solve} = \mathbf{1}(y = y^*)$
  - $r_{propose}$ depends on solve success rate
- Update shared policy with RL
  - $J(\theta) = E[\lambda r_{propose} + r_{solve}]$
  - REINFORCE++ update
  - Same policy for propose + solve

Georgia Tech

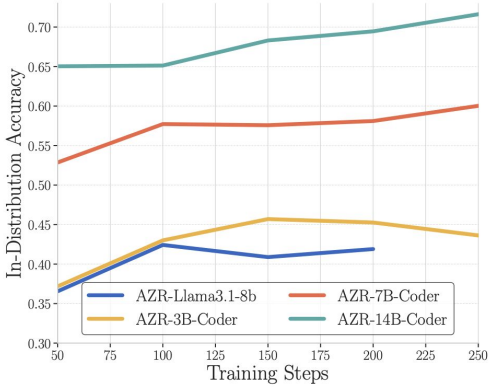
Task-Relative REINFORCE++

$$A^{\text{norm}}_{\text{task,role}} = \frac{r - \mu_{\text{task,role}}}{\sigma_{\text{task,role}}}, \quad \text{task} \in \{\text{ind,ded,abd}\}, \text{role} \in \{\text{propose,solve}\}$$

Georgia Tech

Results

- AZR improves reasoning performance across model sizes
  - Gains in both code and math benchmarks
  - Consistent improvement from 3B → 14B models
- Self-play training achieves strong results without human data
  - No expert-labeled tasks required
  - Performance comparable to RLVR-based methods
- Learnability reward enables stable self-generated curriculum
  - Avoids trivial or unsolvable tasks
  - Leads to steady accuracy improvement over training steps



Model Family	Variant	Code Avg	Math Avg	Total Avg
Llama3.1-8b		28.5	3.4	16.0
Llama3.1-8b	+ SimpleRL[99]	33.7$^{+5.2}$	7.2$^{+3.8}$	20.5$^{+4.5}$
Llama3.1-8b	+ AZR (Ours)	31.6$^{+3.1}$	6.8$^{+3.4}$	19.2$^{+3.2}$
Qwen2.5-3B Coder		51.2	18.8	35.0
Qwen2.5-3B Coder	+ AZR (Ours)	54.9$^{+3.7}$	26.5$^{+7.7}$	40.7$^{+5.7}$
Qwen2.5-7B Coder		56.6	23.9	40.2
Qwen2.5-7B Coder	+ AZR (Ours)	61.6$^{+5.0}$	39.1$^{+15.2}$	50.4$^{+10.2}$
Qwen2.5-14B Coder		60.0	20.2	40.1
Qwen2.5-14B Coder	+ AZR (Ours)	63.6$^{+3.6}$	43.0$^{+22.8}$	53.3$^{+13.2}$

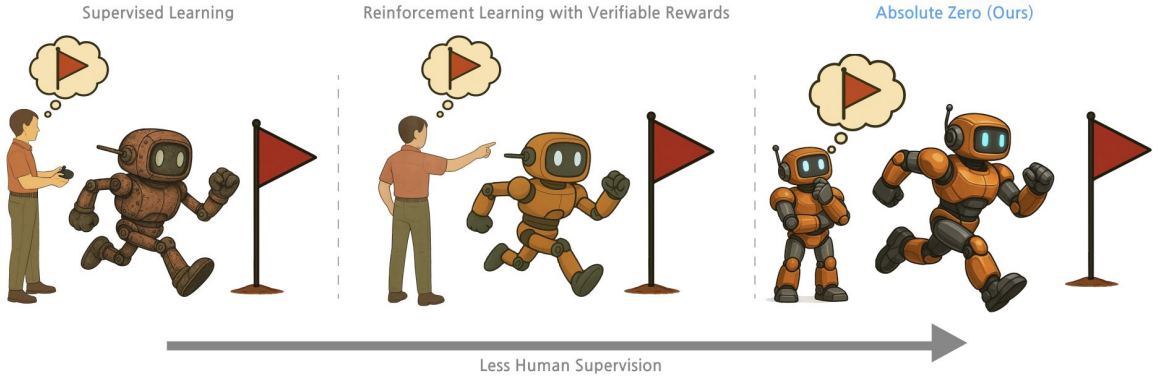
Georgia Tech

Critical Perspective

- Limited to verifiable program-based tasks
  - Requires executable environment for reward
  - Hard to extend to open-ended reasoning
- Curriculum depends on model-generated tasks
  - May limit coverage of reasoning space
  - No guarantee of optimal task distribution
- Expensive training loop
  - Multiple rollouts per task for learnability reward
  - RL + execution + validation increases cost

Main Takeaways

1. Reasoning models can learn from self-generated tasks using verifiable rewards
2. Learnability-based task generation creates an automatic curriculum
3. Shifts training from fixed datasets to experience-driven learning



Georgia Tech