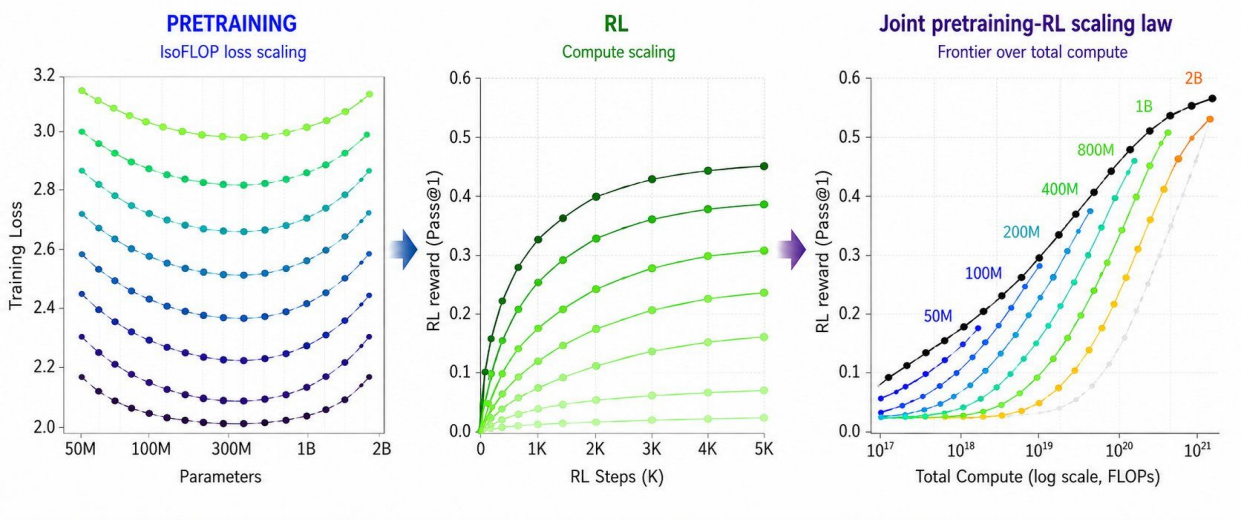
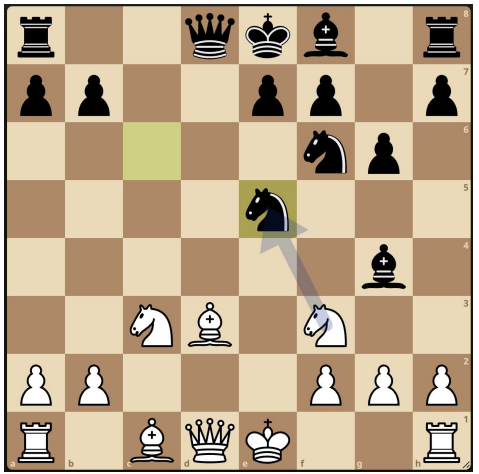
Understanding Reasoning from Pretraining to Post-Training

NeurIPS 2026







Jingyan Shen



Ang Li



Salman Rahman



Yifan Sun



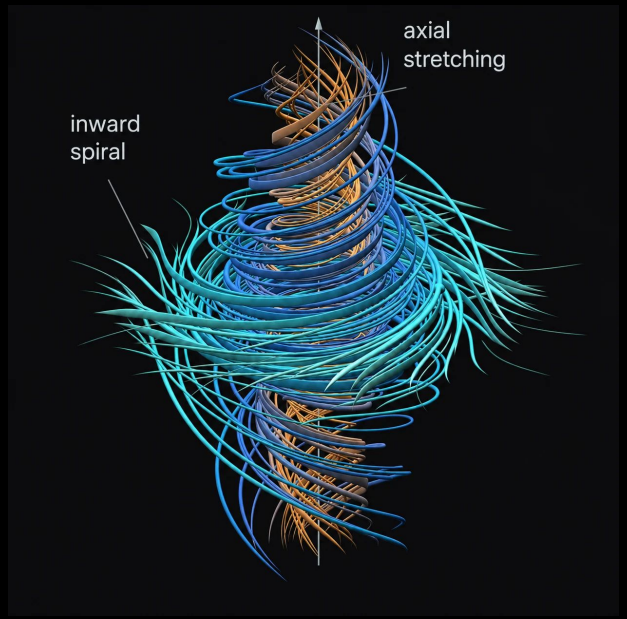
Micah Goldblum

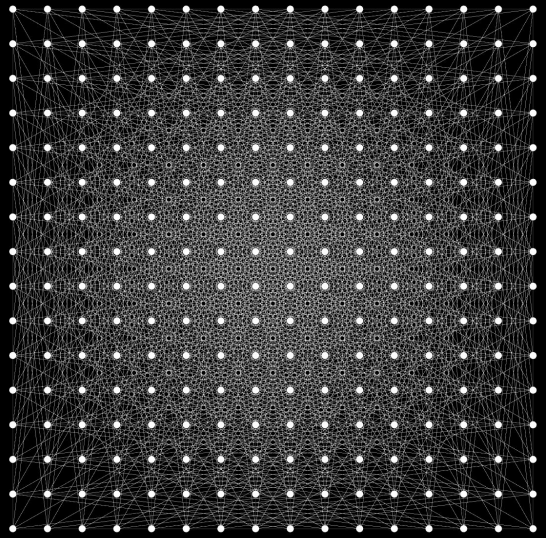


Matus Telgarsky

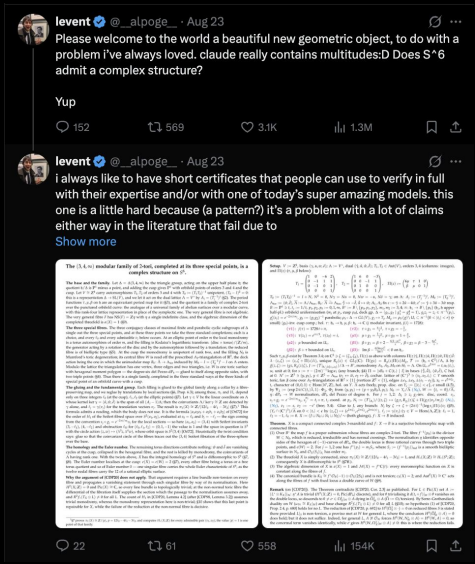


Pavel Izmailov





Previously known construction of many unit distances from a rescaled square grid.



levent @__alpoge__ · Aug 23

Please welcome to the world a beautiful new geometric object, to do with a problem i've always loved. claude really contains multitudes:D Does S^6 admit a complex structure?

Yup

152 569 3.1K 1.3M

levent @__alpoge__ · Aug 23

i always like to have short certificates that people can use to verify in full with their expertise and/or with one of today's super amazing models. this one is a little hard because (a pattern?) it's a problem with a lot of claims either way in the literature that fail due to

Show more

22 61 558 154K

levent @__alpoge__ · Jul 19

hello there the jacobian conjecture is false thanx to my close friend akhil for asking about it and my other close friend fable for working during the world cup final

((1+xy)^3 z + y^2 (1+xy) (4+3xy), y + 3 x (1+xy)^2 z + 3 x y^2 (4+3xy), 2 x - 3 x^2 y - x^3 z): \C^3\to \C^3, has jacobian determinant -2, and sends (0, 0, -1/4), (1, -3/2, 13/2), and (-1, 3/2, 13/2) to (-1/4, 0, 0)

1.7K 8.5K 43K 40M

LLM Training Pipeline



LLM Training Pipeline

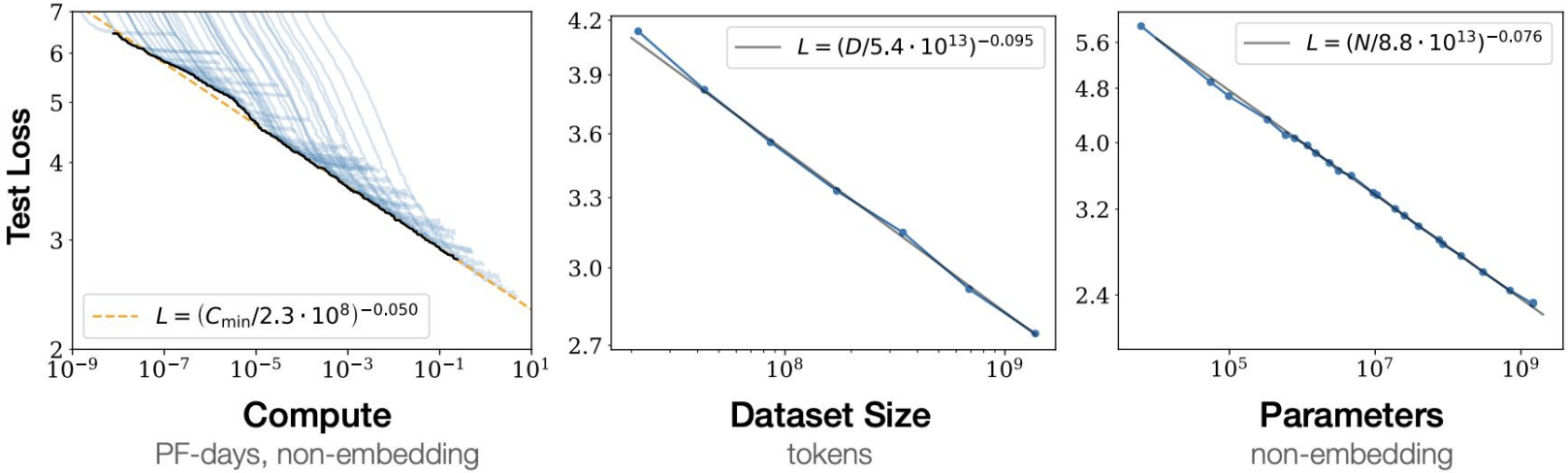


Building a strong prior for RL

Teaching it the RL format, tool use, …

Learning to achieve goals we care about

Scaling Laws



More PT compute ⇒ better models

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$$

Allows you to allocate compute optimally between model size and data size

Reinforcement Learning: AlphaZero



Superhuman go / chess play from zero priors.

Reinforcement Learning: AlphaZero



Superhuman go /
chess play from zero
priors.

How to allocate compute?
Building Prior ↔ Downstream training
Pretraining ↔ RL

What is RL actually doing?

RL Sharpens PT Policy

pass@k is flat for large k
arxiv.org/abs/2504.13837

Composes & Discovers

f(x) & g(x) → f(g(x))
arxiv.org/abs/2509.25123

Both / Dynamic Interplay

Both in some combination
arxiv.org/abs/2512.07783

Intuitively, the allocation question is also related to whether RL can breakthrough beyond the PT priors.

Many open questions about RL

- How does RL scale with compute?
- How does RL interact with Pretraining?
- How do we allocate compute?
- What does RL actually do?

Many open questions about RL

- How does RL scale with compute?
- How does RL interact with Pretraining?
- How do we allocate compute?
- What does RL actually do?

Hard to study on real LLM pretraining!

- Cost
- Attribution: hard to reason about what's in PT data vs learned in RL
- Evaluation: how to understand how CoT evolves during RL

(a) LLM-style Pre-to-Post Training Pipeline on Chess Data

Pretraining: Human Chess Game Data

Player 1 Move 1: P e2e4 Player 2 Move 1: P e7e5





<P> <e2> <e4> <P> <e7> <e5>

Training data: move sequence of the game

<P>	<e2>	<e4>	<P>	<e7>	<e5>
<N>	<g1>	<f3>	<N>	<b8>	<c6>
<B>	<f1>	<b5>	<P>	<a7>	<a6>

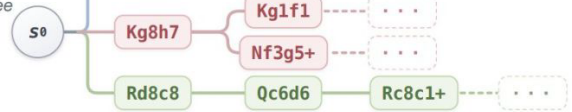
. . .

SFT: Synthetic Reasoning Data



root state s_0

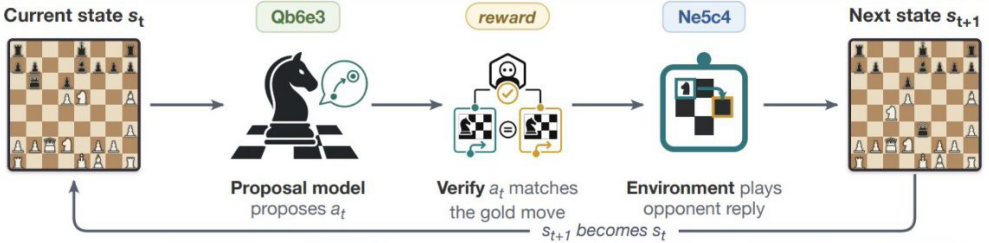
sample N trajectories →  *proposal model* → *merge into search tree*



→ *serialize root-to-leaf paths into a token sequence*

<T> <R> <d8> <d2> <Q> <c6> <e8> <+> . . . <SEP> <K> <g8> <h7> . . . </T>

RL: Verifiable Puzzle Environment



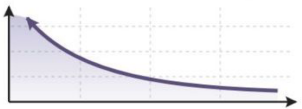
Rollout 1 Rh6h3 ✓ → Pg7g6 ✗ Reward = 0

Rollout 2 Rh6h3 ✓ → Pe3e2 ✓ → Rh8h3 ✓ Reward = 1

(b) Scaling Analysis: Pretraining-RL Scaling Law

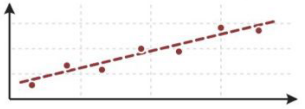
$$R(C_{\mathrm{RL}}, N, T) = f(L_{\mathrm{pt}}) + g(N,T) \cdot \log_{10}\left(\frac{C_{\mathrm{RL}}}{C_{\mathrm{ref}}}\right)$$

Mean Reward at C_{ref} $f(L_{pt})$



Pretrain Loss L_{pt}

Slope of Mean Reward $g(N, T)$

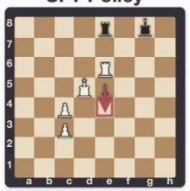


Pretrain Tokens($\log_{10} T$)

(c) Mechanism Analysis: Is RL Just Sharpening?

Ground truth amplification on easy puzzles

SFT Policy



★ e5 → e4
g8 → f7
e8 × e6

RL Training

RL Policy



★ e5 → e4
g8 → f7
e8 × e6

Tail discovery on hard puzzles

SFT Policy



d6 × e5
b6 → b4
★ b6 → e3

RL Training

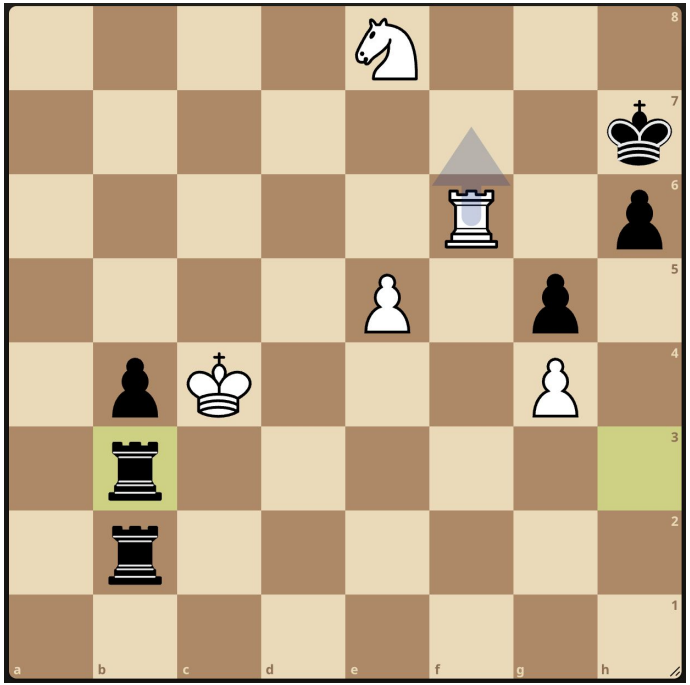
RL Policy



★ b6 → e3
b6 → b4
d6 × e5

Chess as a testbed

- Complex but compact task
- Lots of human games for pretraining
- Lots of downstream tasks (puzzles)
- Exact verification, we have ground truth
- Interpretable actions



Chess as a testbed for reasoning

LLM Training Pipeline

STAGE 1
Pre-training

STAGE 2
SFT

STAGE 3
RL

Human games

54B tokens of Lichess Blitz + Rapid (2022). Autoregressive next-token prediction over move sequences.

Synthetic reasoning traces

Sample K possible continuations from the current position, organize as CoT

Verifiable Puzzles

GRPO with a binary outcome reward.

Everything happens in moves, no natural language.

Representation

A move is four tokens.

N	g1	f3	—
piece	source	dest	flag

81 tokens in the vocabulary

A game is both players' moves in sequence.

```
Pe2e4  Pe7e5  Ng1f3  Nb8c6  Bf1b5  Pa7a6  ...
```

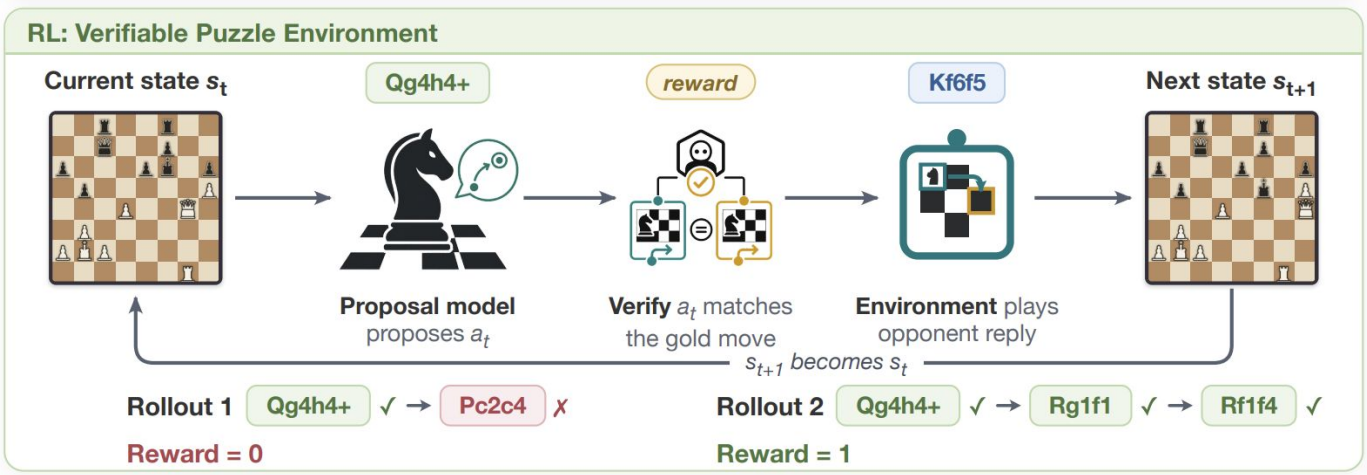


Puzzles as the RL task

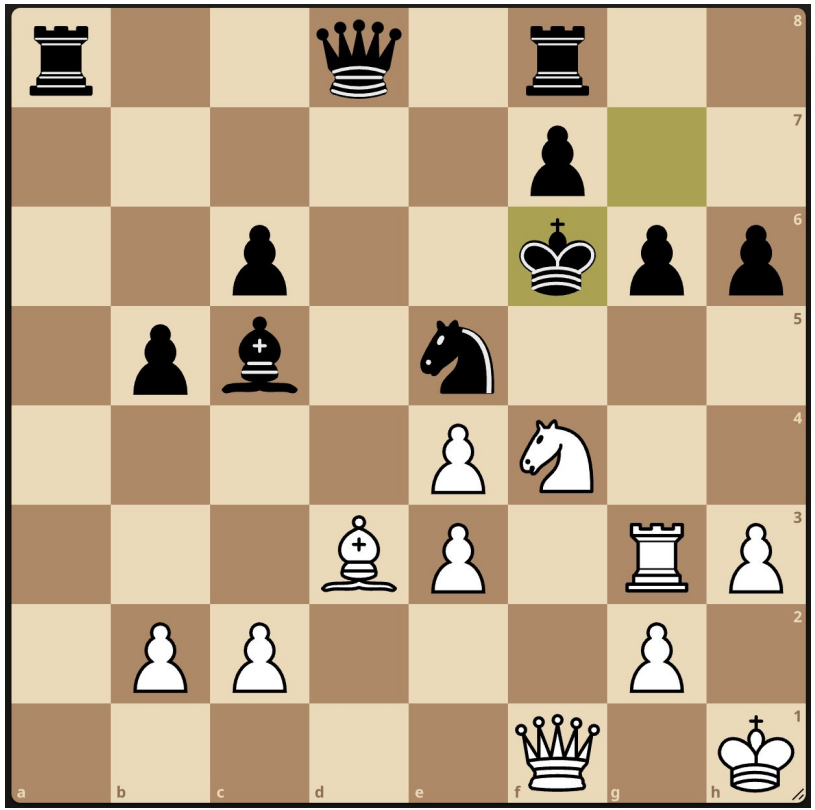
- The model sees the board, picks a move
- Wrong move ends the episode
- Right move: the opponent replies
- Repeat until the line is done

Reward

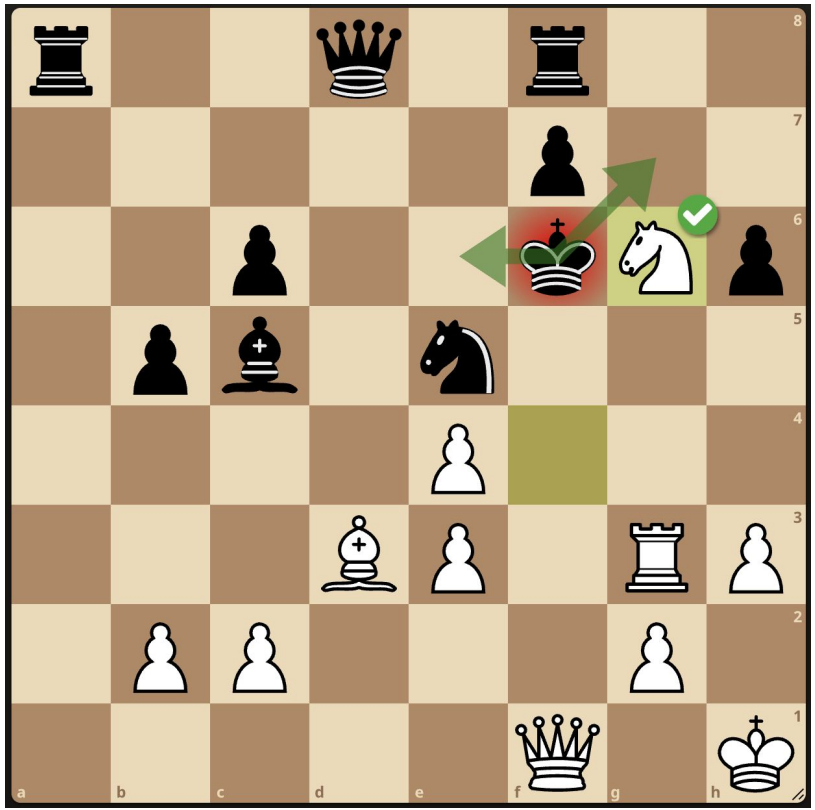
1 if every move matches, else 0



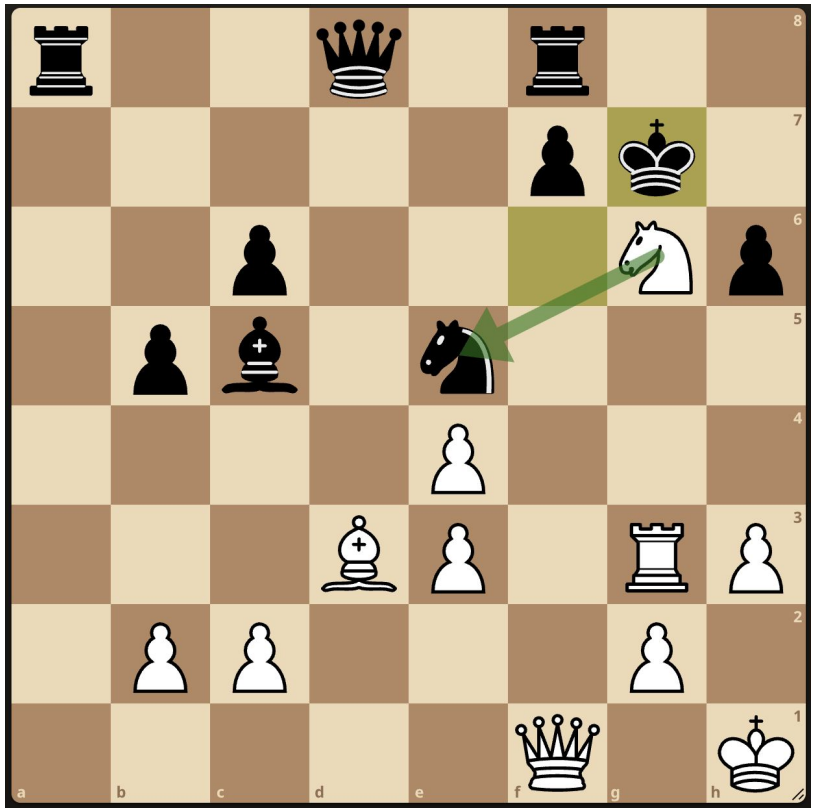
Puzzle example



Puzzle example



Puzzle example

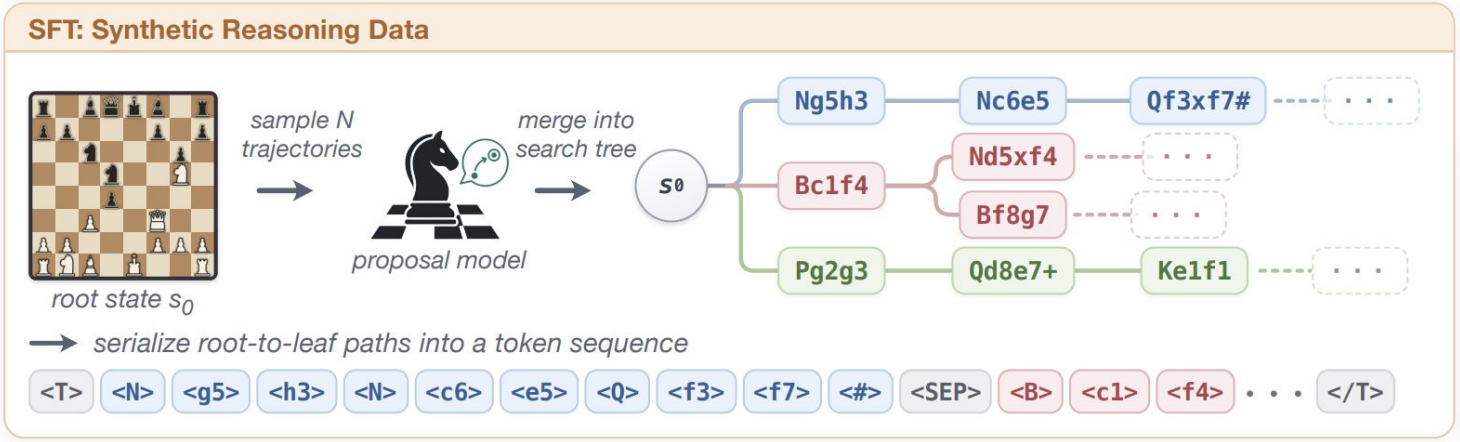


CoT reasoning traces

1. Sample continuations from current position
2. Structure as a tree of positions
3. Serialize into a token sequence
4. Commit to a move

Reason in context

The model writes its own trace, then commits to a move



What a trace looks like

Input	Pe2e4 Pc7c5 Ng1f3 Nb8c6 Pd2d4 Pc5xd4 Bf1c4 Pg7g6 Nf3g5 Pe7e6 Pc2c3 Pd7d5 Pe4xd5 Pe6xd5 Qd1f3 Ng8f6 Bc4xd5 Nf6xd5 <T>
Reasoning	Ng5h3 Bf8g7 <sep> Ng5h3 Nc6e5 Qf3xf7# Ne5xf7 <sep> Ng5h3 Bc8g4 Qf3xf7# <sep> Ng5h3 Qd8d6 Bc1f4 Qd6e7+ <sep> Ng5h3 <sep> Ng5xf7 Nd5e7 Bc1f4 Rh8g8 Nf7xd8 Ne7f5 Qf3xc6+ <sep> Bc1f4 Nd5xf4 Qf3xf4 Bc8f5 Qf4e5+ Bf8e7 <sep> Bc1f4 Bf8g7 Ng5xf7 Rh8f8 <sep> Bc1f4 Bf8e7 Bf4e5 <sep> Qf3xf7# <sep> O-O Nd5f6 Nc6e5 Qf7xd5 Ne5c6 <sep> Qf3xf7# <sep> Pg2g3 Qd8f6 Qf3xf6 Rh8g8 <sep> Pg2g3 Qd8e7+ Ke1f1 Bc8e6 Ng5xe6 Qe7xe6 <sep> Qf3xf7# </T>
Answer	Qf3xf7#

Setup

54B
pretraining tokens

10
model sizes, 5M to 1B

156K
training puzzles

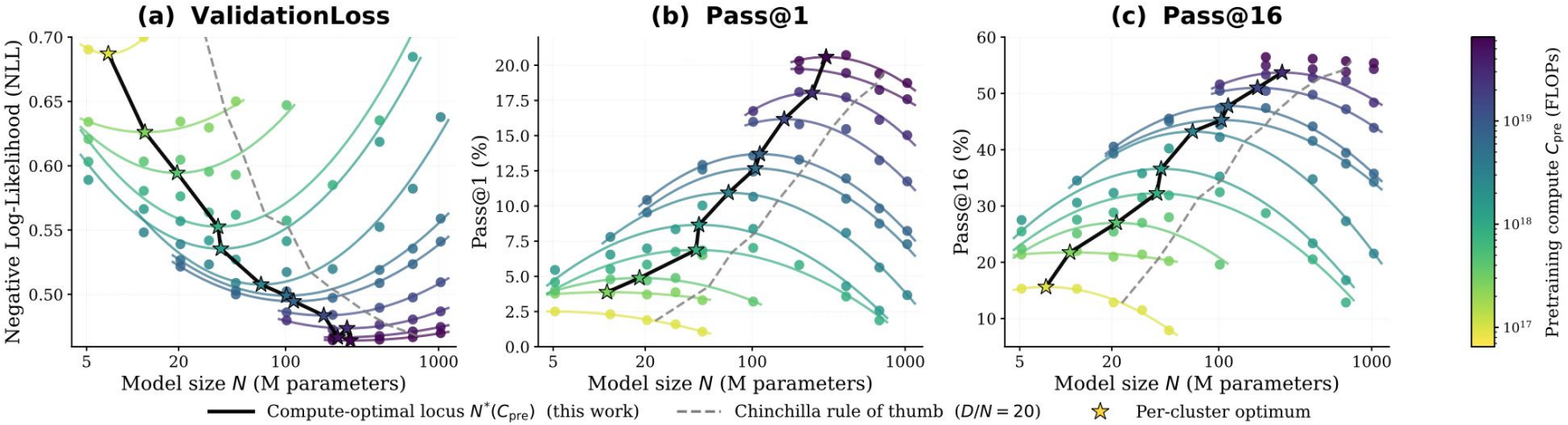
1,480
benchmark puzzles

- 36 pretraining and RL combinations, at four model sizes
- Five puzzle difficulty bins, B1 easiest to B5 hardest
- Training and test sets are disjoint at the board-position level
- Qwen3 architecture, GRPO

PT ↔ RL Scaling Laws

Chess pretraining

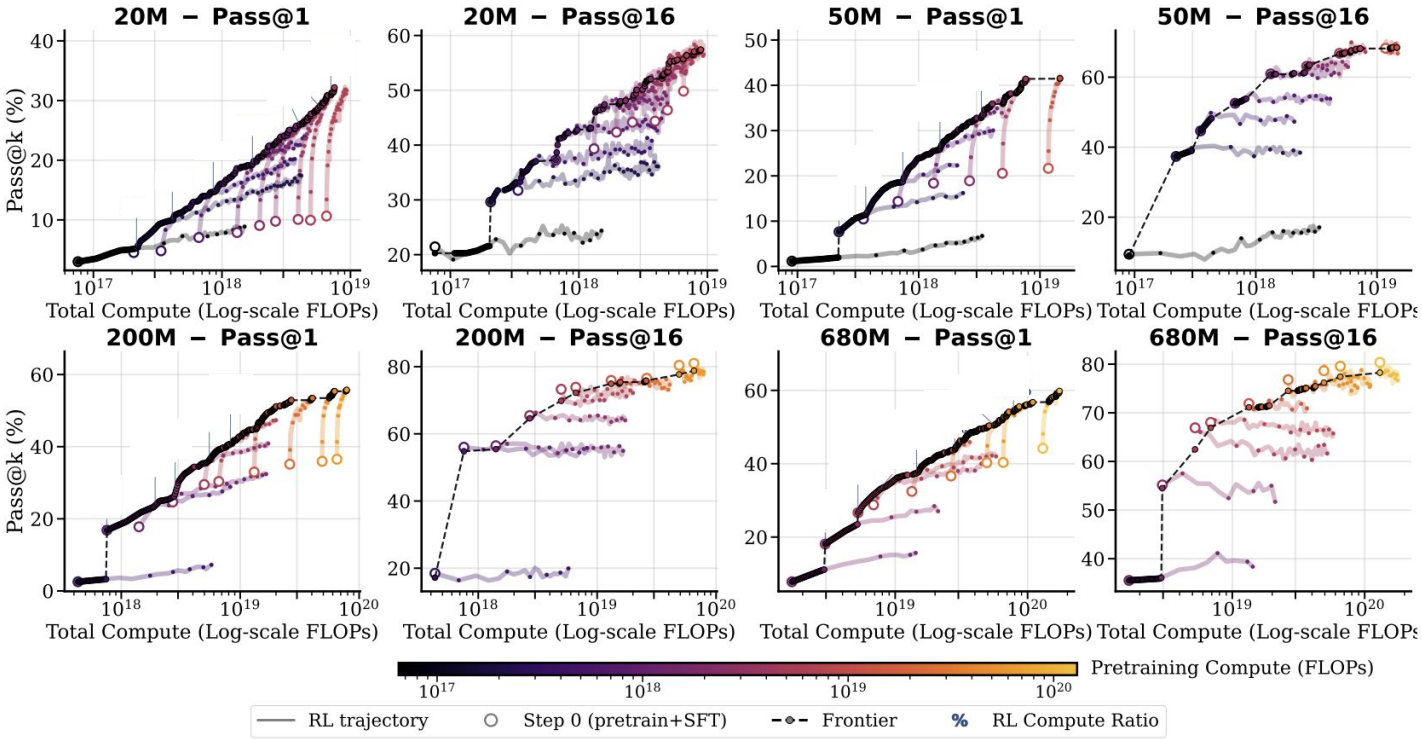
$$L(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$$



We are able to fit Chinchilla-style scaling laws.

The exponents are a bit different from natural language.

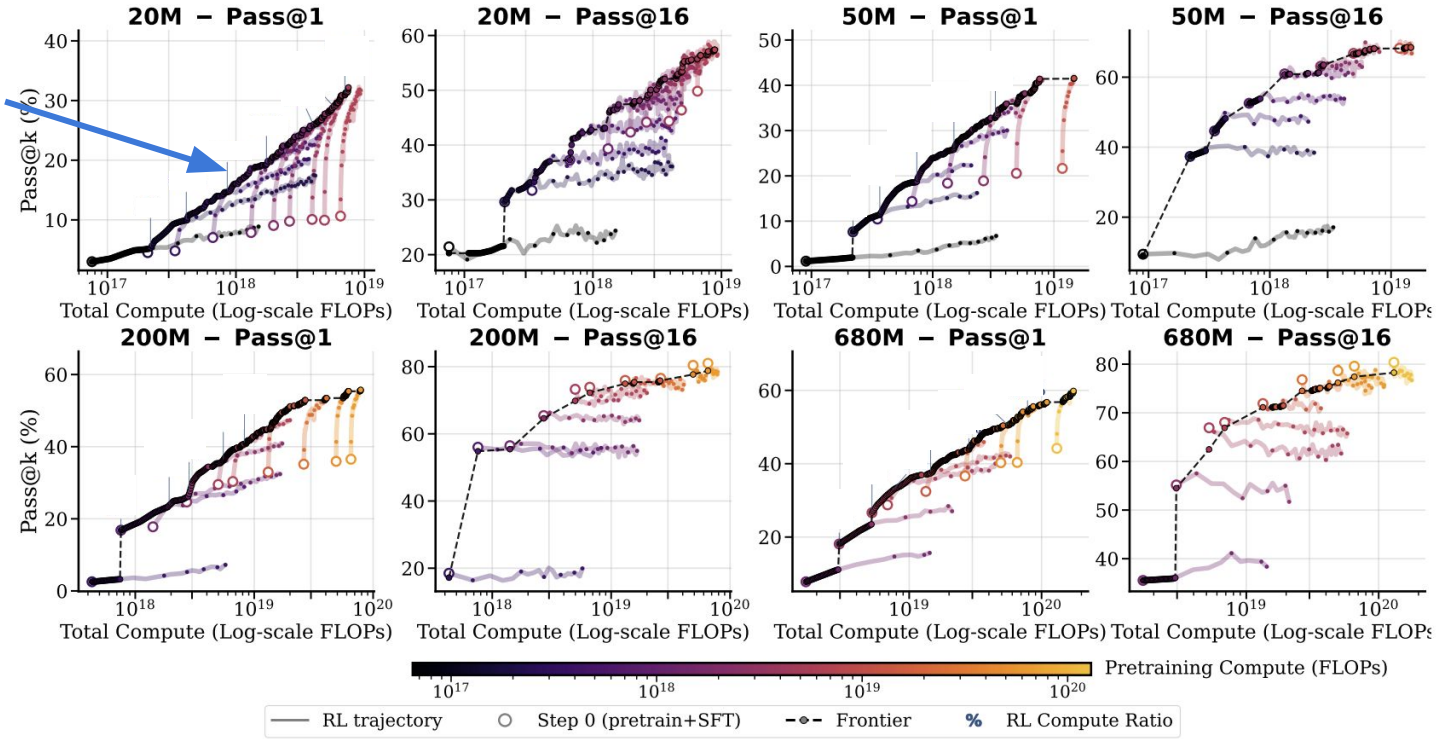
Reinforcement Learning



Next, we run RL initializing from different model sizes and pretraining token counts.

Reinforcement Learning

pass@1 improves with more RL

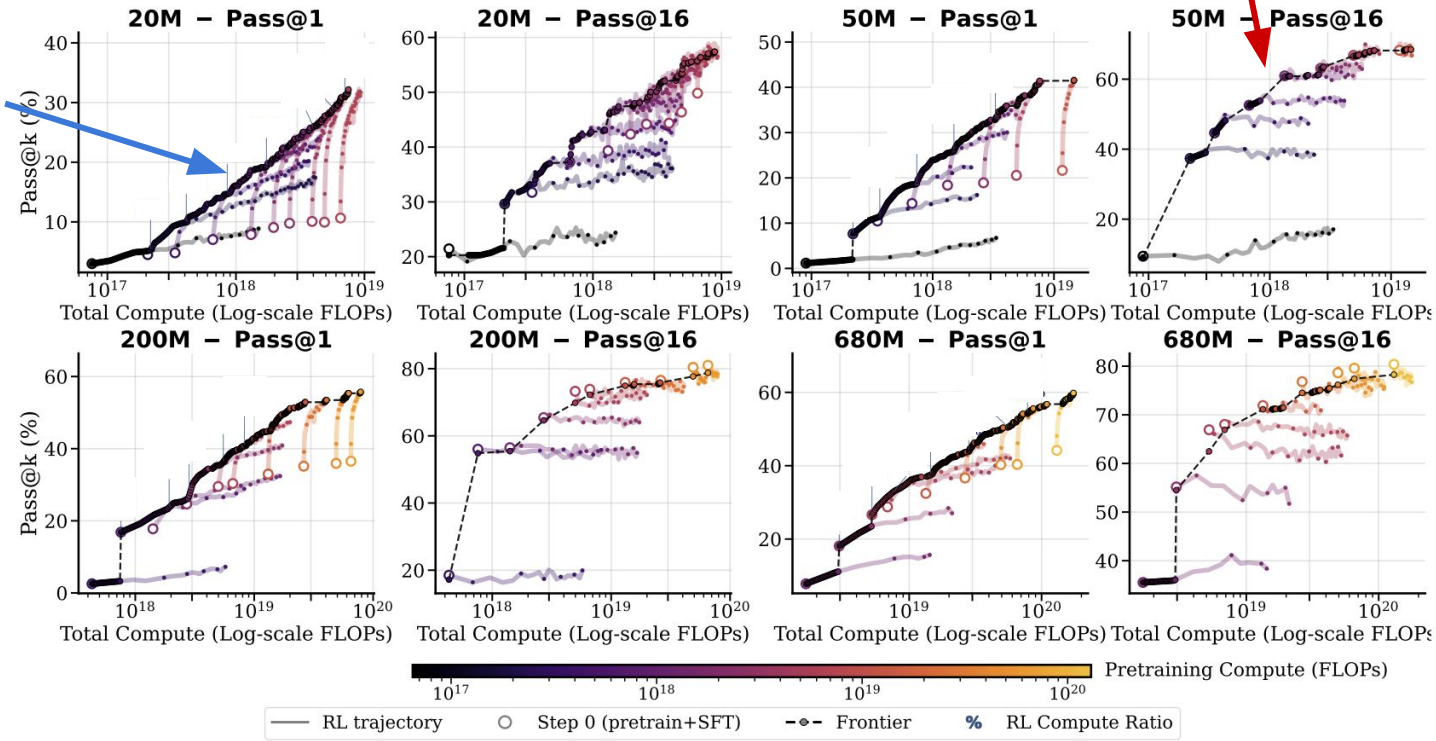


Next, we run RL initializing from different model sizes and pretraining token counts.

Reinforcement Learning

pass@16 looks flat

pass@1 improves with more RL

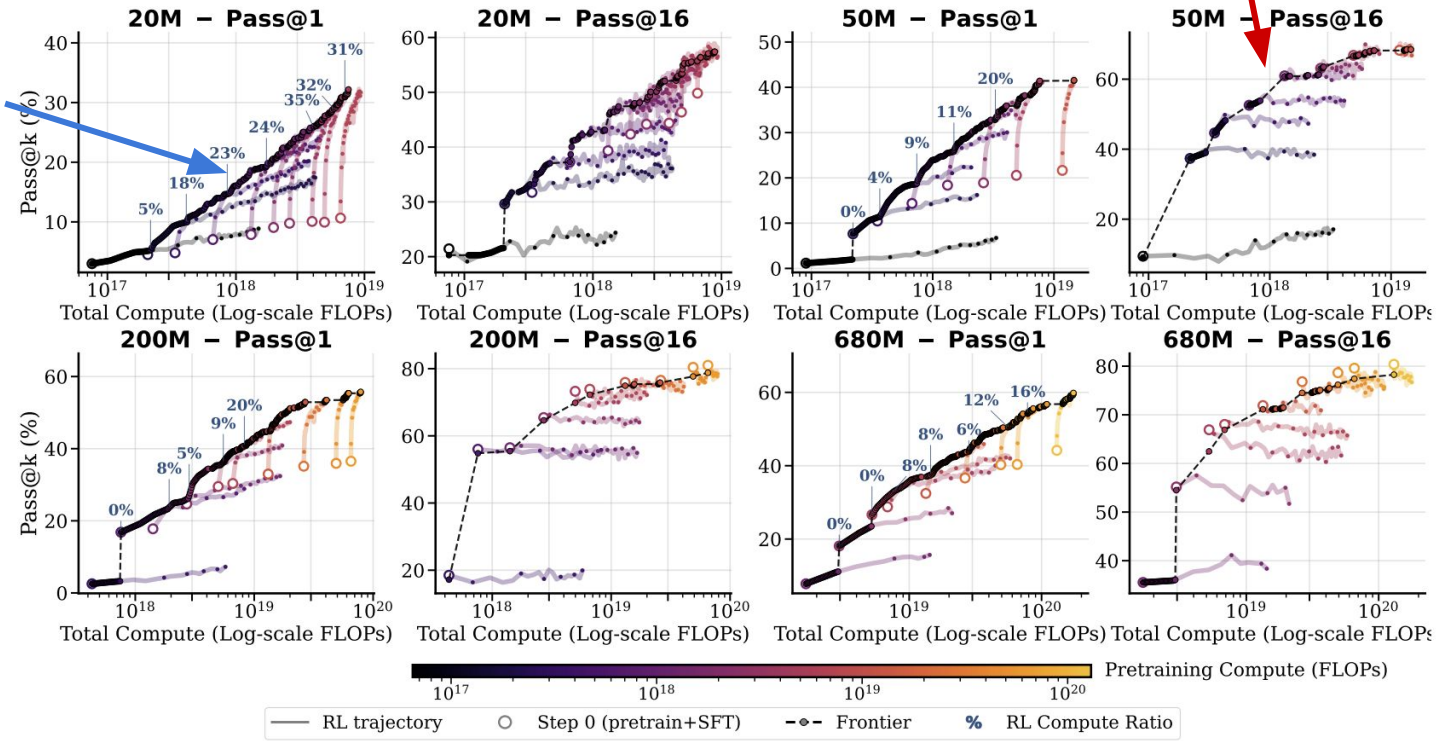


Next, we run RL initializing from different model sizes and pretraining token counts.

Reinforcement Learning

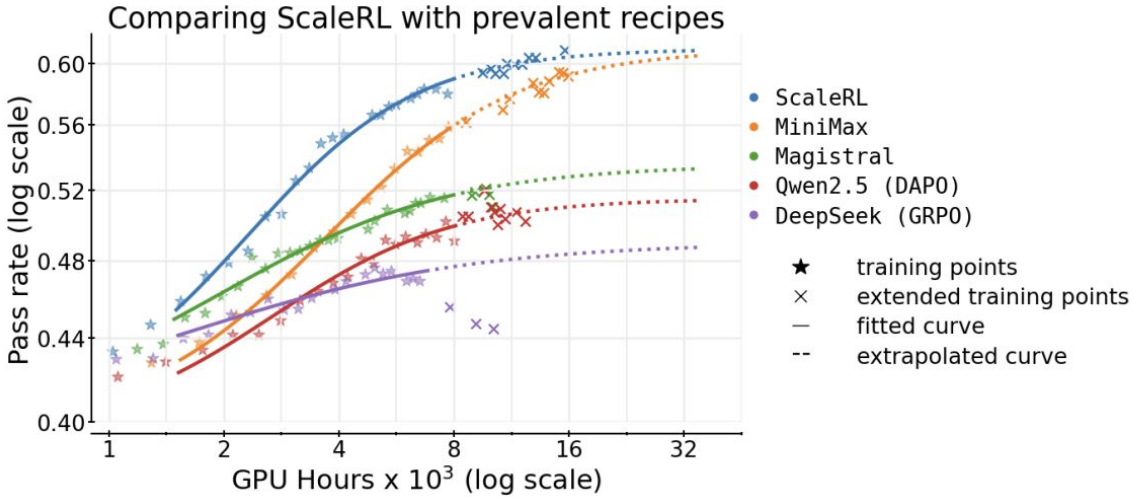
pass@16 looks flat

pass@1 improves with more RL



We can also track what fraction compute goes to RL on the pareto frontier

Scaling Law for RL

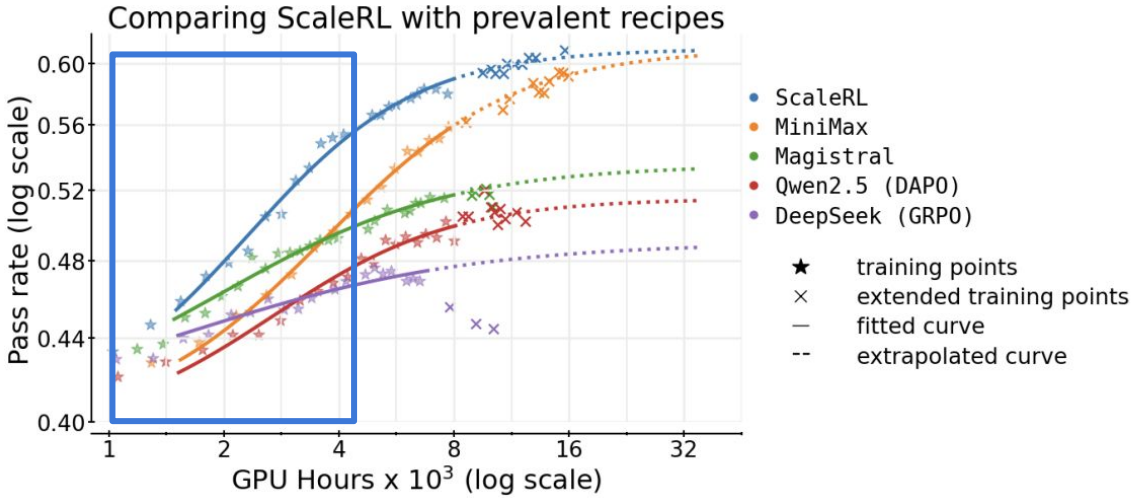


$$\overbrace{R_C - R_0}^{\text{Reward Gain}} = \overbrace{(A - R_0)}^{\text{Asymptotic Reward Gain}} \times \underbrace{\frac{1}{1 + (C_{\text{mid}}/C)^B}}_{\text{Compute Efficiency}}$$

In very high compute regime, RL pass rate looks like a sigmoid…

The Art of Scaling Reinforcement Learning Compute for LLMs; Khatri et al., 2026

Scaling Law for RL

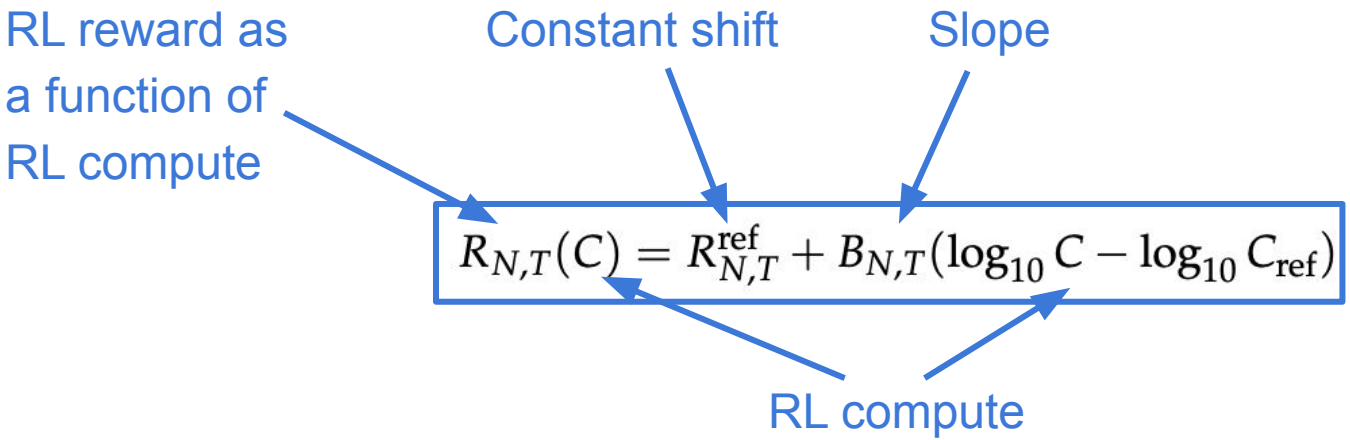


In very high compute regime, RL pass rate looks like a sigmoid…

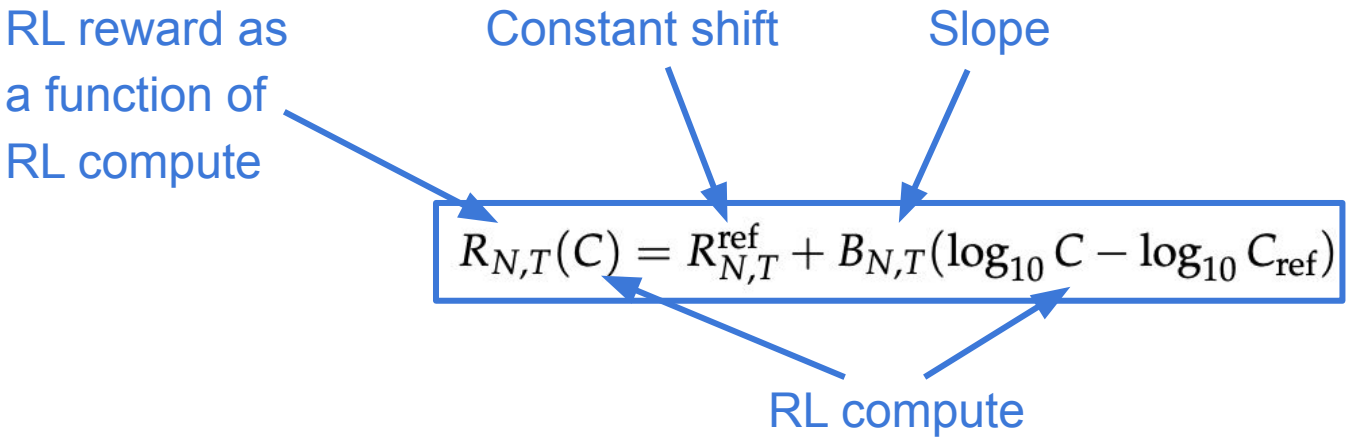
but early in training it looks log-linear!

$$R_{N,T}(C) = R^{\text{ref}}_{N,T} + B_{N,T}(\log_{10} C - \log_{10} C_{\text{ref}})$$

Scaling Law for RL

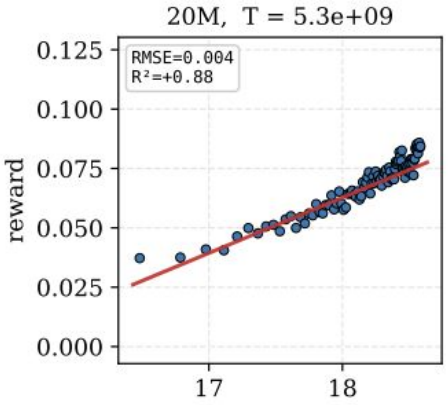


Scaling Law for RL



reward = level + slope × (decades of RL compute)

Scaling Law for RL



Parameters that depend on PT

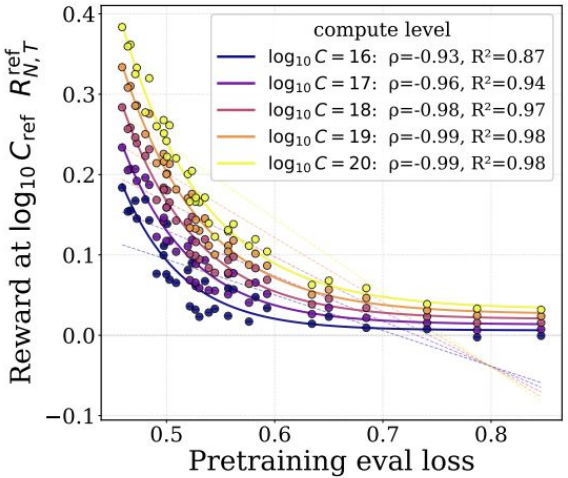
$$R_{N,T}(C) = R^{\text{ref}}_{N,T} + B_{N,T}(\log_{10} C - \log_{10} C_{\text{ref}})$$

We fit these parameters to each individual RL run

$$(R^{\text{ref}}_{N,T}, B_{N,T}) = \arg\min_{a,b} \frac{1}{N_C} \sum_{i=1}^{N_C} \left(R_{N,T}(C_i) - a - b(\log_{10} C_i - \log_{10} C_{\text{ref}})\right)^2$$

Scaling Law for RL

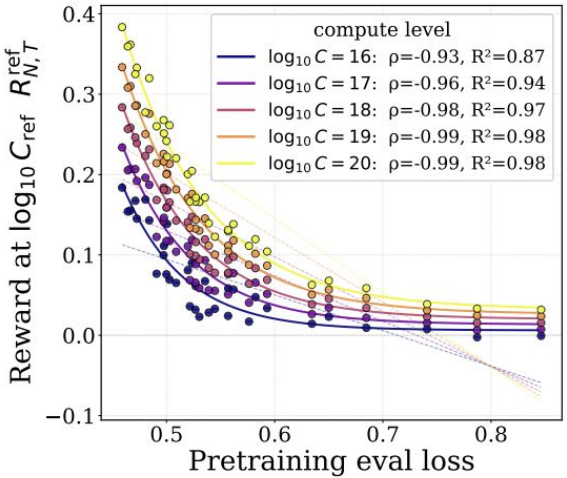
$$R_{N,T}(C) = R^{\text{ref}}_{N,T} + B_{N,T}(\log_{10} C - \log_{10} C_{\text{ref}})$$

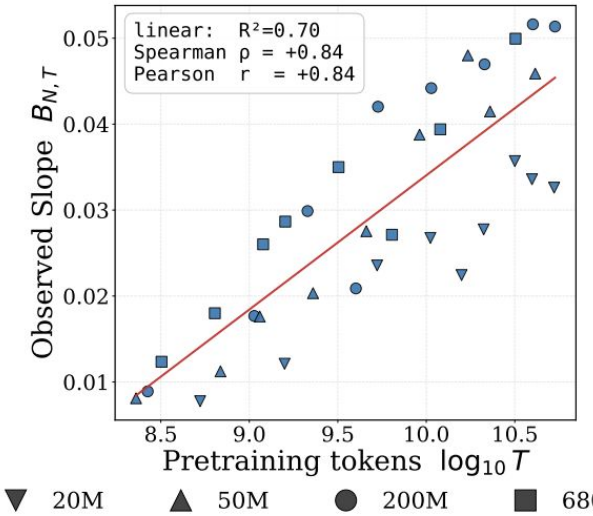


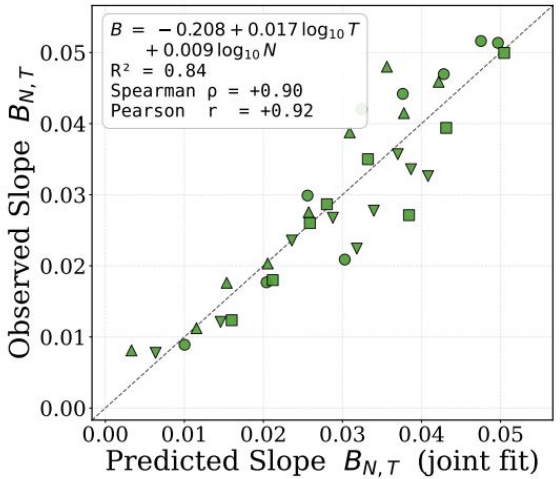
Reward at reference compute is well-predicted by PT loss

Scaling Law for RL

$$R_{N,T}(C) = R^{\text{ref}}_{N,T} + B_{N,T}(\log_{10} C - \log_{10} C_{\text{ref}})$$







Reward at reference compute is well-predicted by PT loss

The slope is well predicted by PT tokens or joint model in (Tokens, Parameters)

Scaling Law for RL

Joint Pretraining–RL Scaling Law

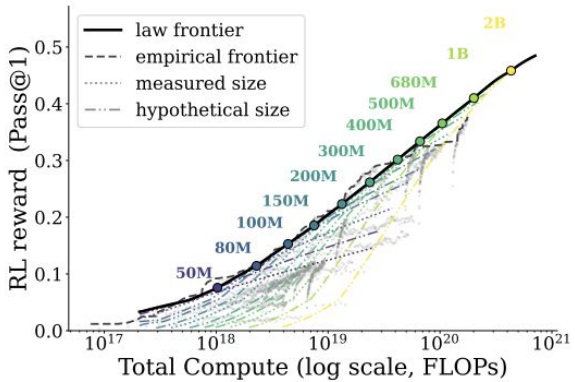
$$R(C_{\mathrm{RL}}, N, T) = f\big(L_{\mathrm{pt}}(N, T)\big) + g(N, T)\,\big(\log_{10} C_{\mathrm{RL}} - \log_{10} C_{\mathrm{ref}}\big)$$

We now have a joint scaling law connecting PT and RL!

Compute Allocation

Joint Pretraining–RL Scaling Law

$$R(C_{\mathrm{RL}}, N, T) = f\big(L_{\mathrm{pt}}(N, T)\big) + g(N, T)\,\big(\log_{10} C_{\mathrm{RL}} - \log_{10} C_{\mathrm{ref}}\big)$$

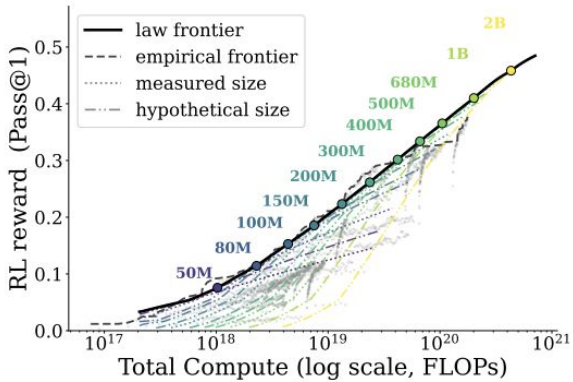


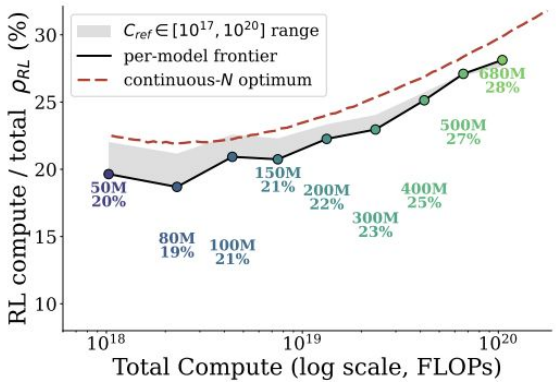
We can now optimize for (C_{RL}, N, T) for a fixed total compute

Compute Allocation

Joint Pretraining–RL Scaling Law

$$R(C_{\mathrm{RL}}, N, T) = f\big(L_{\mathrm{pt}}(N, T)\big) + g(N, T)\,\big(\log_{10} C_{\mathrm{RL}} - \log_{10} C_{\mathrm{ref}}\big)$$



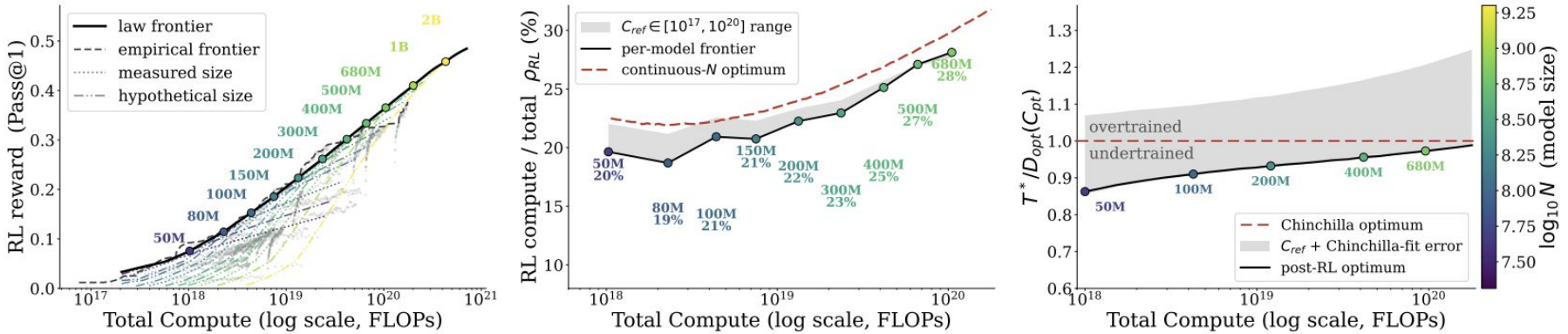


We can now optimize for (C_{RL}, N, T) for a fixed total compute

Compute Allocation

Joint Pretraining–RL Scaling Law

$$R(C_{\mathrm{RL}}, N, T) = f\big(L_{\mathrm{pt}}(N,T)\big) + g(N,T)\,\big(\log_{10} C_{\mathrm{RL}} - \log_{10} C_{\mathrm{ref}}\big)$$



$$R(C_{RL}, N, T) = f(L) + g(N,T)\,(\log_{10} C_{RL} - 20)$$

$$f(L) = 0.0314 + \exp(4.87 - 12.85\,L) \qquad g(N,T) = -0.216 + 0.0172\log_{10} T + 0.0098\log_{10} N \qquad L(N,T) = 0.414 + 95.4\,N^{-0.404} + 1.62\times 10^{5}\,T^{-0.677}$$

We can now optimize for (C_{RL}, N, T) for a fixed total compute

What is RL doing to the policy

What is RL actually doing?

RL Sharpens PT Policy

pass@k is flat for large k
arxiv.org/abs/2504.13837

Composes & Discovers

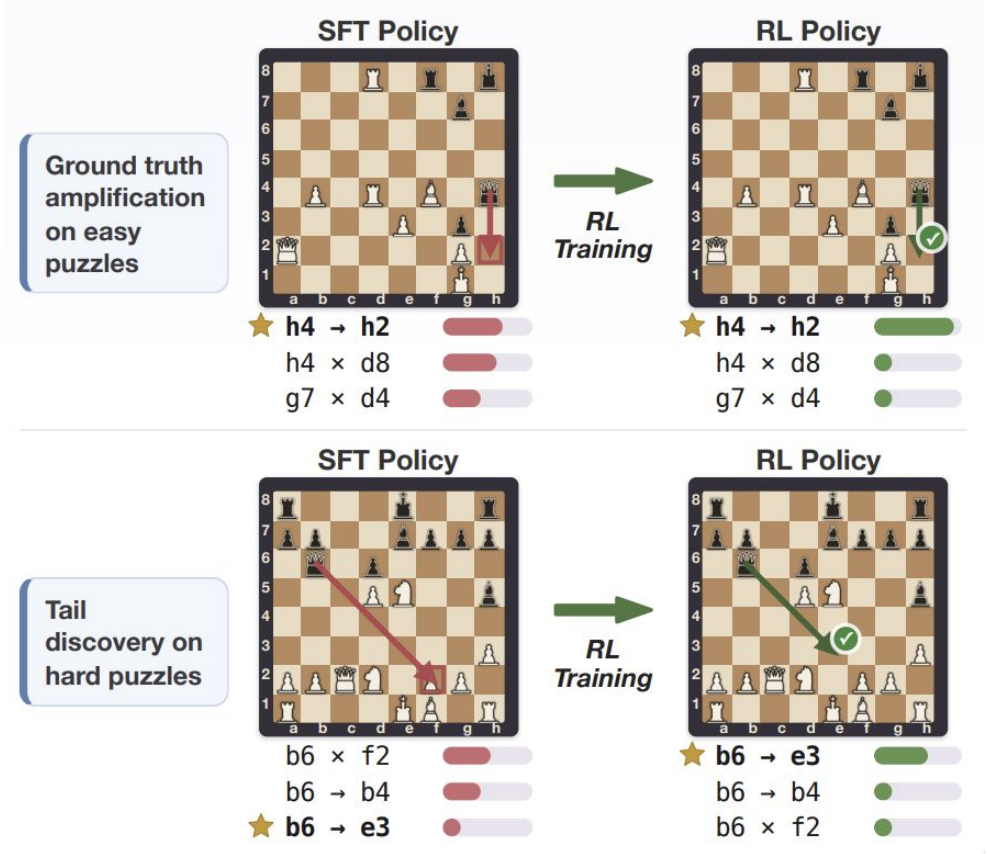
f(x) & g(x) → f(g(x))
arxiv.org/abs/2509.25123

Both / Dynamic Interplay

Both in some combination
arxiv.org/abs/2512.07783

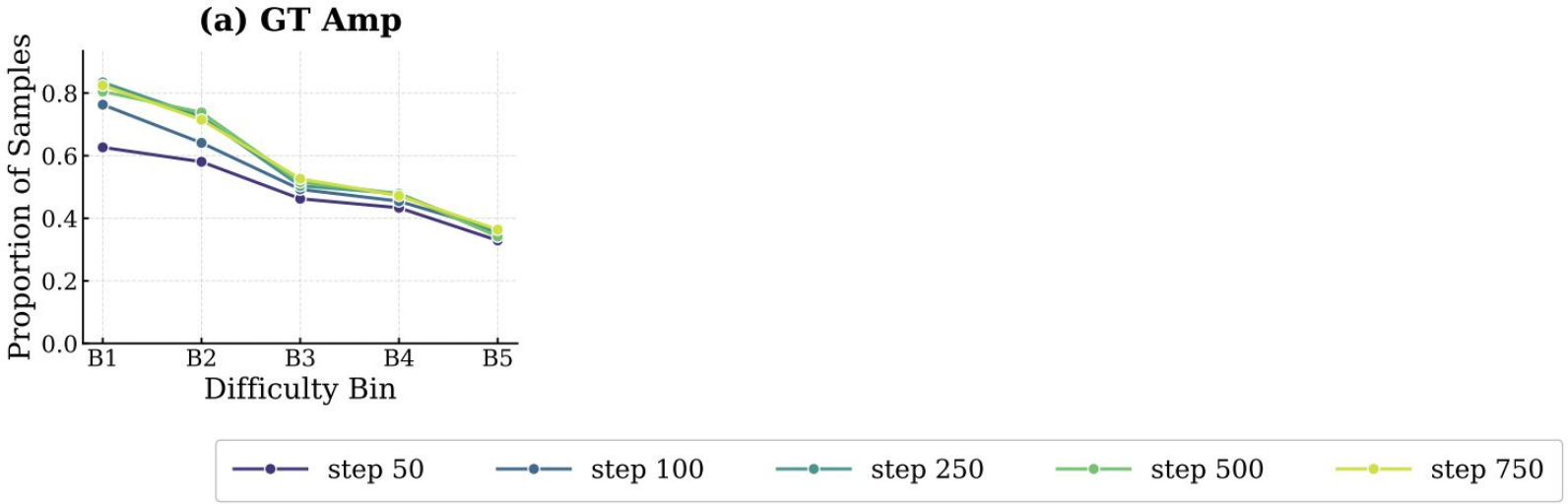
Intuitively, the allocation question is also related to whether RL can breakthrough beyond the PT priors.

Categorization of policy updates



Categorization of policy updates

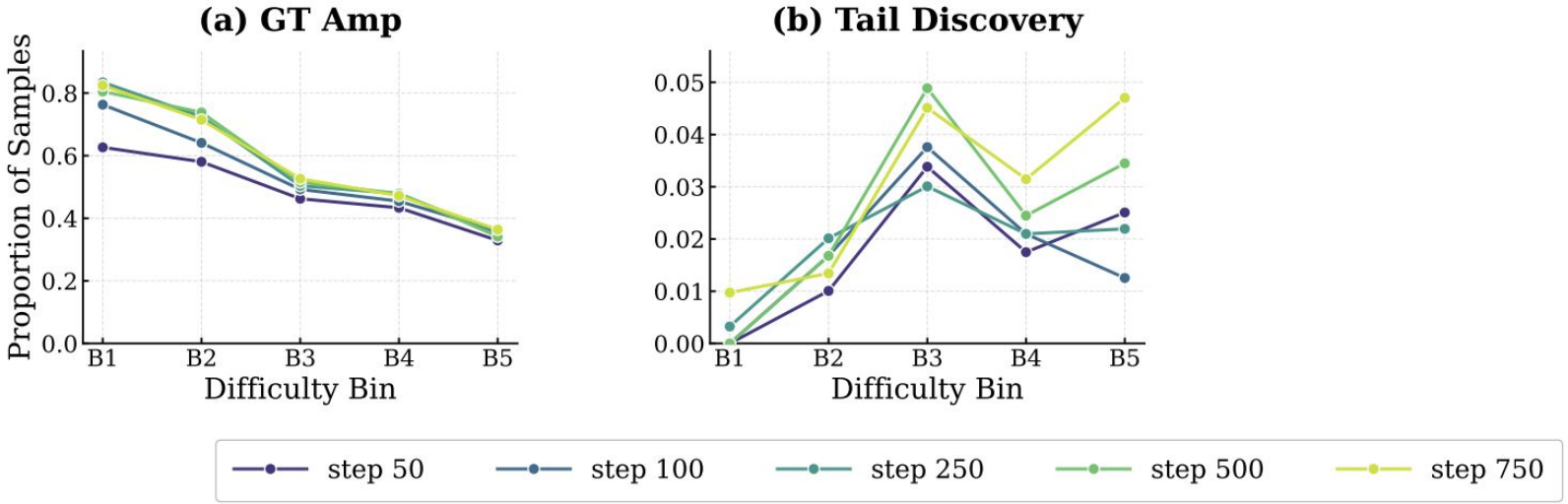
Correct move is already in top-k and RL reinforces it



Categorization of policy updates

Correct move is already in top-k and RL reinforces it

Correct move is in the low-probability tail, but RL discovers it

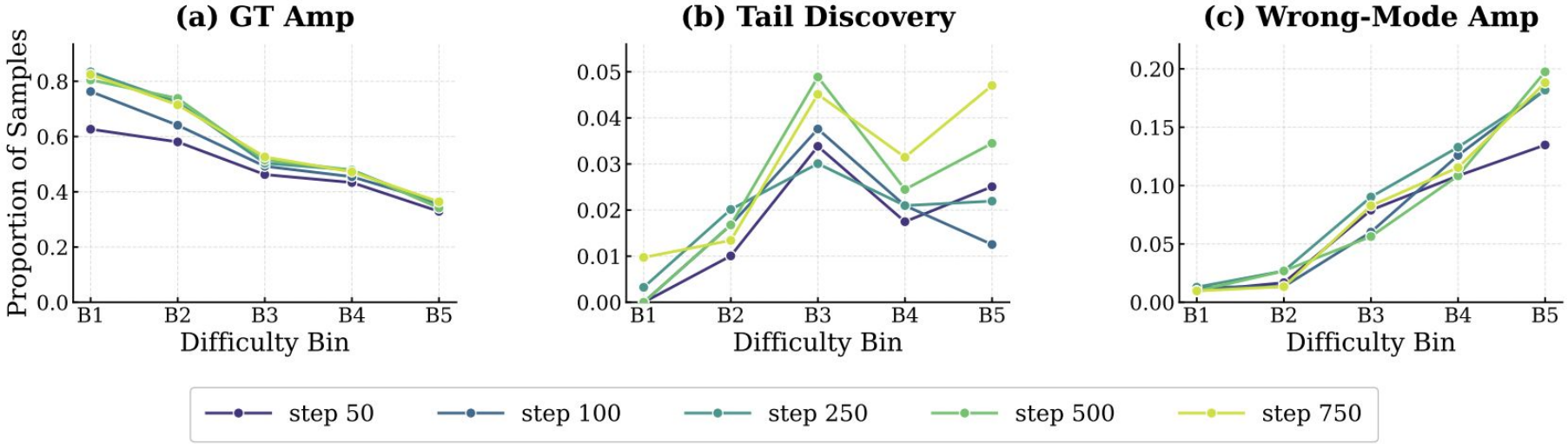


Categorization of policy updates

Correct move is already in top-k and RL reinforces it

Correct move is in the low-probability tail, but RL discovers it

Incorrect high-probability moves are amplified

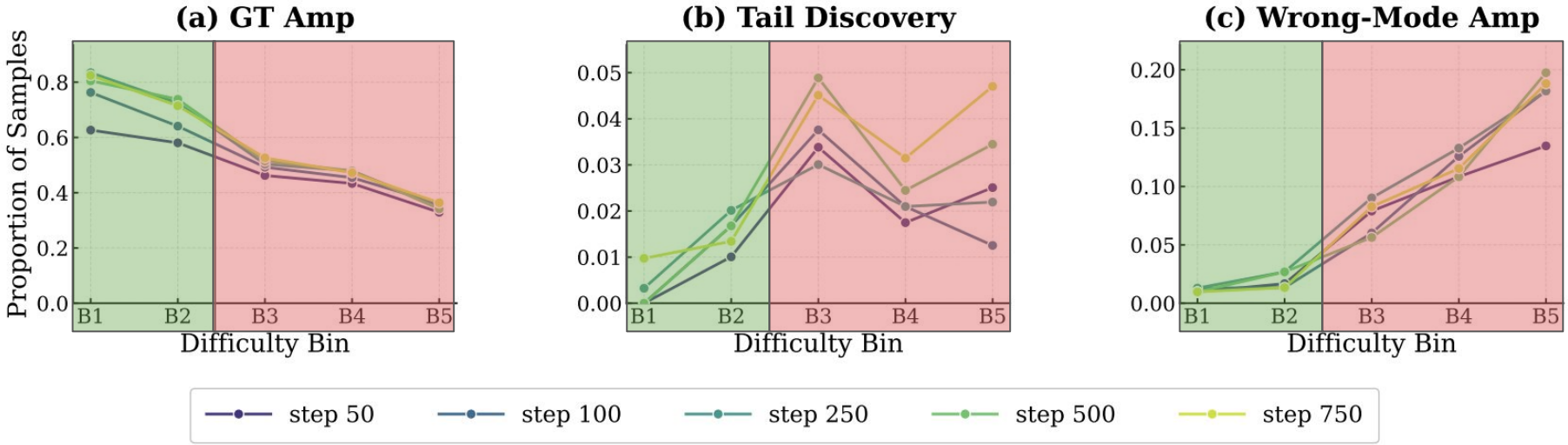


Categorization of policy updates

Correct move is already in top-k and RL reinforces it

Correct move is in the low-probability tail, but RL discovers it

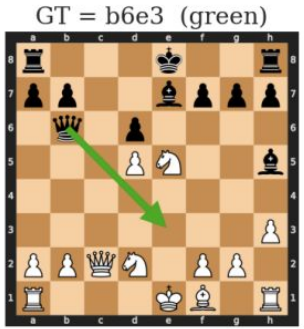
Incorrect high-probability moves are amplified

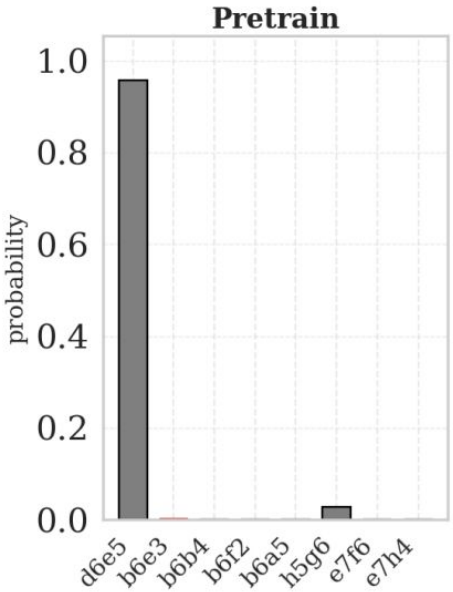


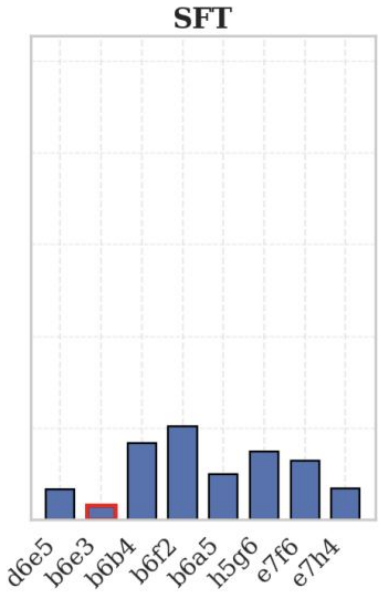
More on easy problems

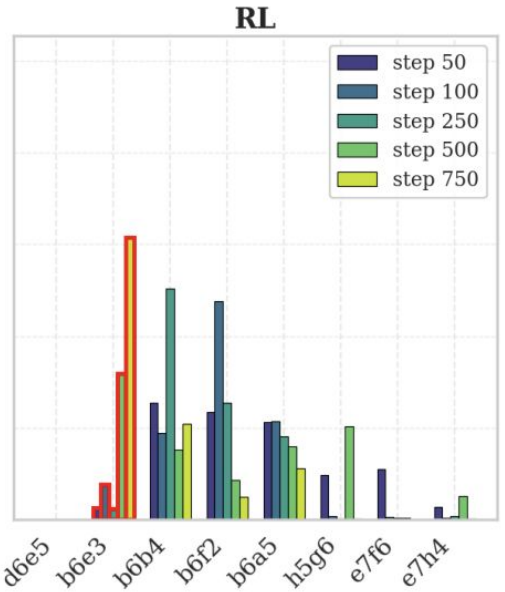
More on hard problems

Tail discovery example



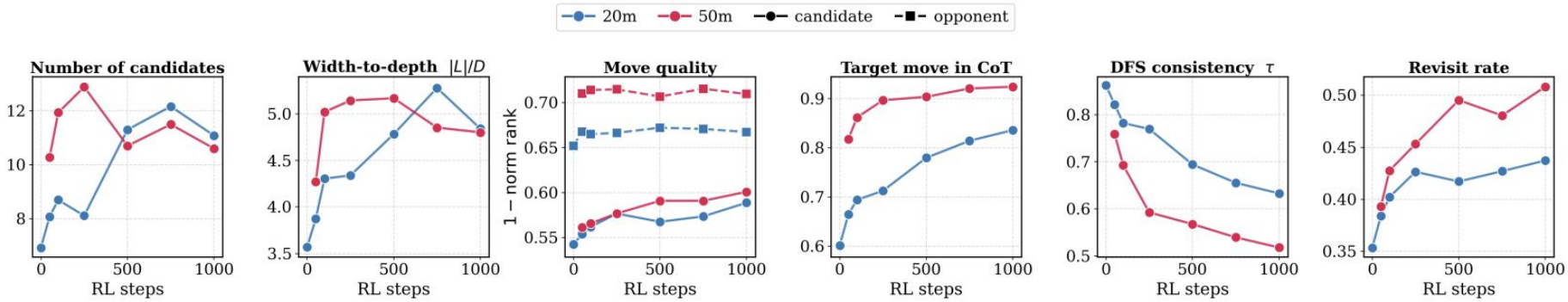






Correct move is in the low-probability tail, but RL discovers it

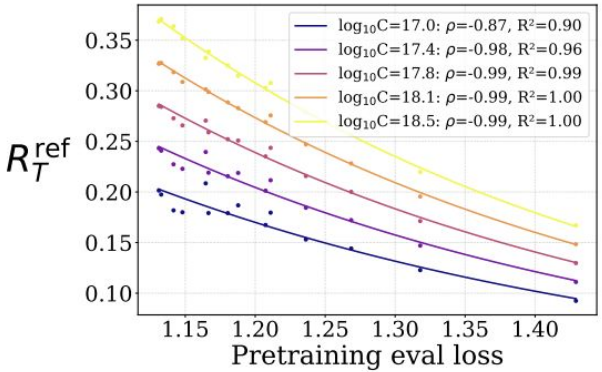
CoT analysis

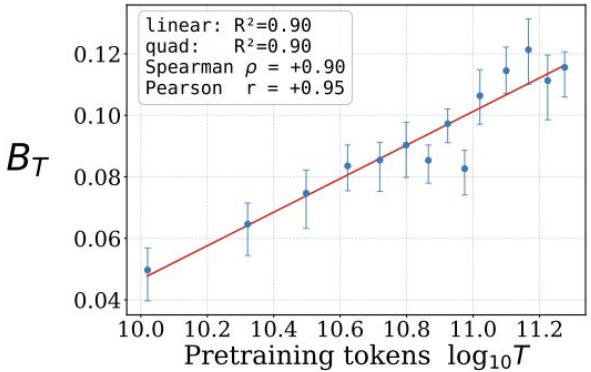


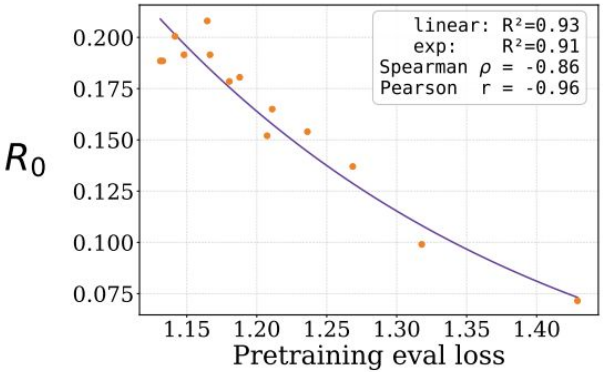
The structured CoT format also allows us to see what happens to CoT during RL.

Generalization beyond chess

Transfer beyond chess



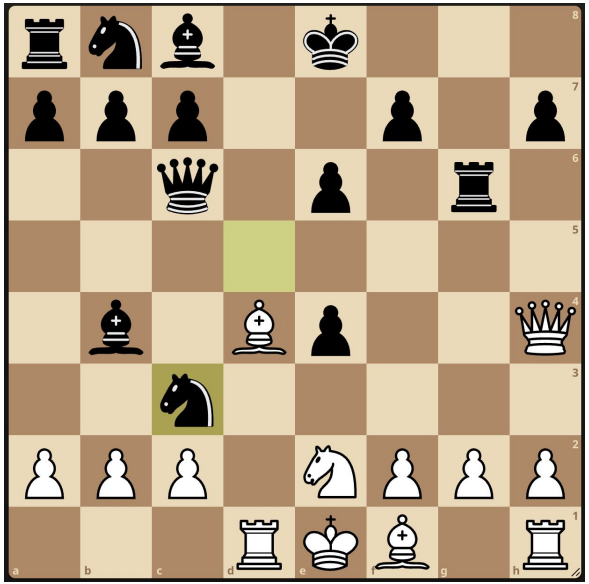




We replicate a part of the scaling analysis with a 1B parameter LLM on math.

Thank you!

- Chess as testbed for reasoning
- Joint scaling law for PT and RL
- Optimal PT ↔ RL compute allocation
- RL largely sharpens but there is also tail discovery

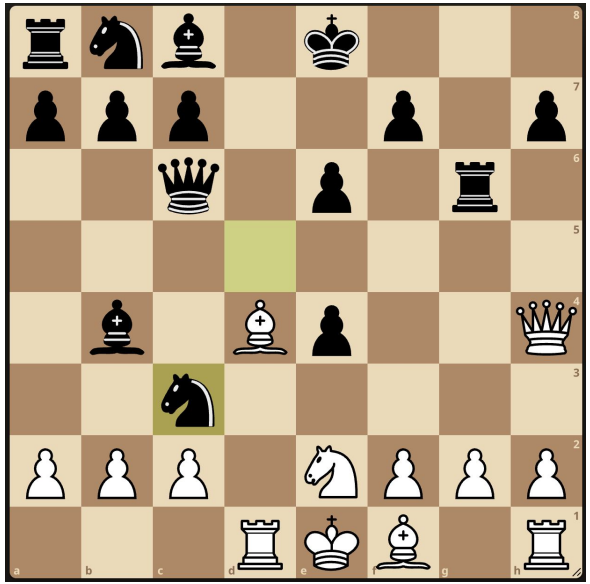


Models & datasets huggingface.co/pavelslab-nyu/pre2post-chess

Code github.com/pavelslab-nyu/pre2post-chess

Next questions

- Should RL and PT interact more?
- How does PT affect RL asymptotic ceiling performance?
- How to reduce wrong mode amplification and increase tail discovery?
- Using the testbed study weak-to-strong generalization, self-play, …
- Generalization to other domains



Models & datasets huggingface.co/pavelslab-nyu/pre2post-chess
Code github.com/pavelslab-nyu/pre2post-chess