The Art of Scaling Reinforcement Learning Compute for LLMs

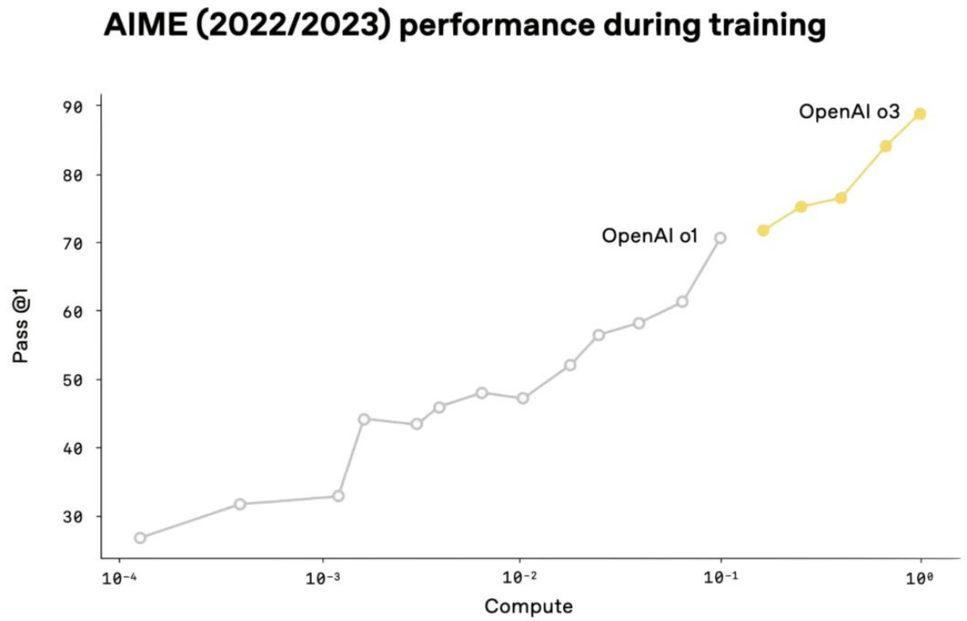
Devvrit Khatri*, **Lovish Madaan***, Rishabh Tiwari, Rachit Bansal, Sai Surya Duvvuri, Manzil Zaheer, Inderjit Dhillon, David Brandfonbrener, Rishabh Agarwal

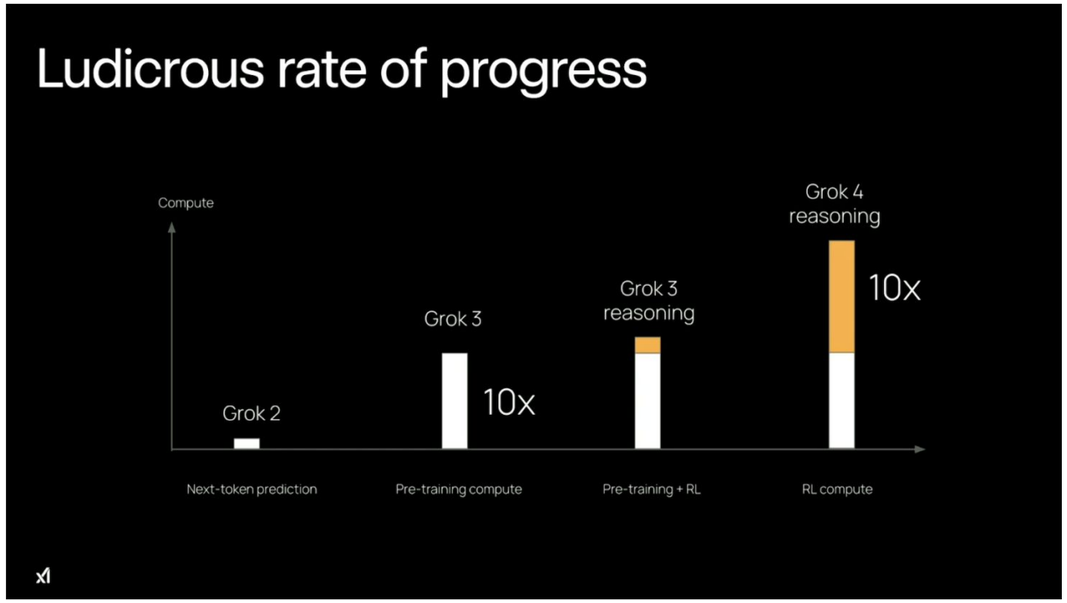
* - equal contribution

Meta

Motivation

- Reinforcement Learning (RL) is becoming the new scaling frontier for large language models (LLMs)

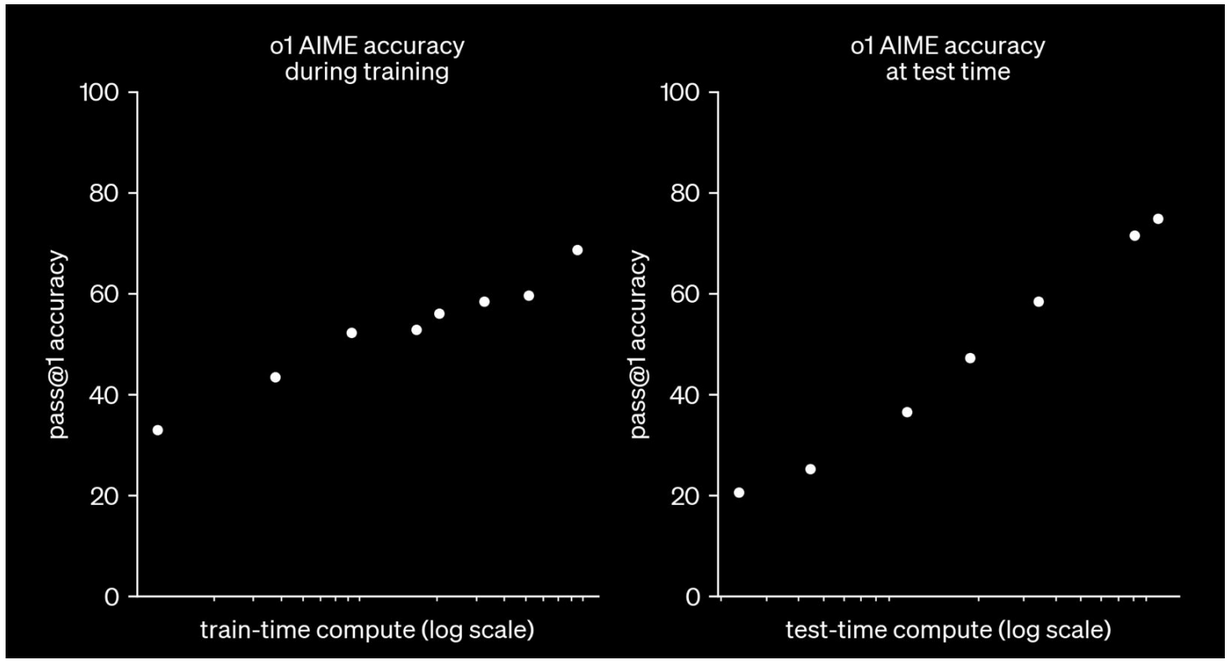


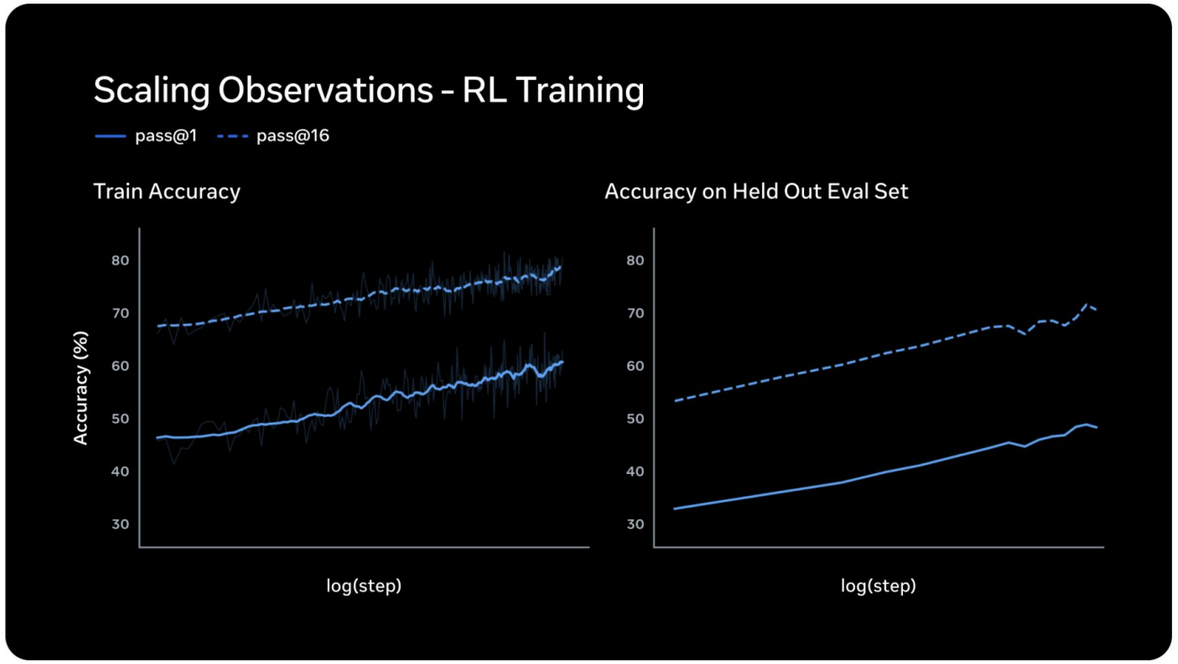


Sources: https://epoch.ai/gradient-updates/how-far-can-reasoning-models-scale, https://x.ai/news/grok-4

Motivation

- RL unlocks new capabilities in a base model like test-time thinking and agentic capabilities





Sources: https://openai.com/index/learning-to-reason-with-llms/, https://ai.meta.com/blog/introducing-muse-spark-msl/

Motivation

- Many design choices while building the RL infra stack
  - Off-policyness, loss, clipping, numerics, adaptive prompt filtering, generation length, batch size, etc.

- Question: What design choices actually work best at scale? And can you achieve smooth scaling like pre-training?

Sources: Rishabh Agarwal, The Art of Scaling RL for LLMs

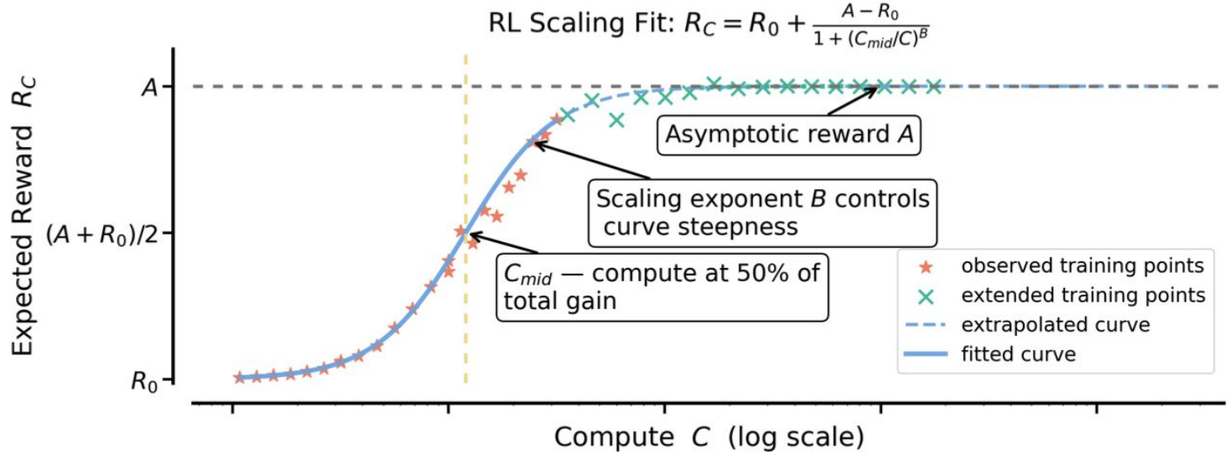
Predictable Scaling in RL

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

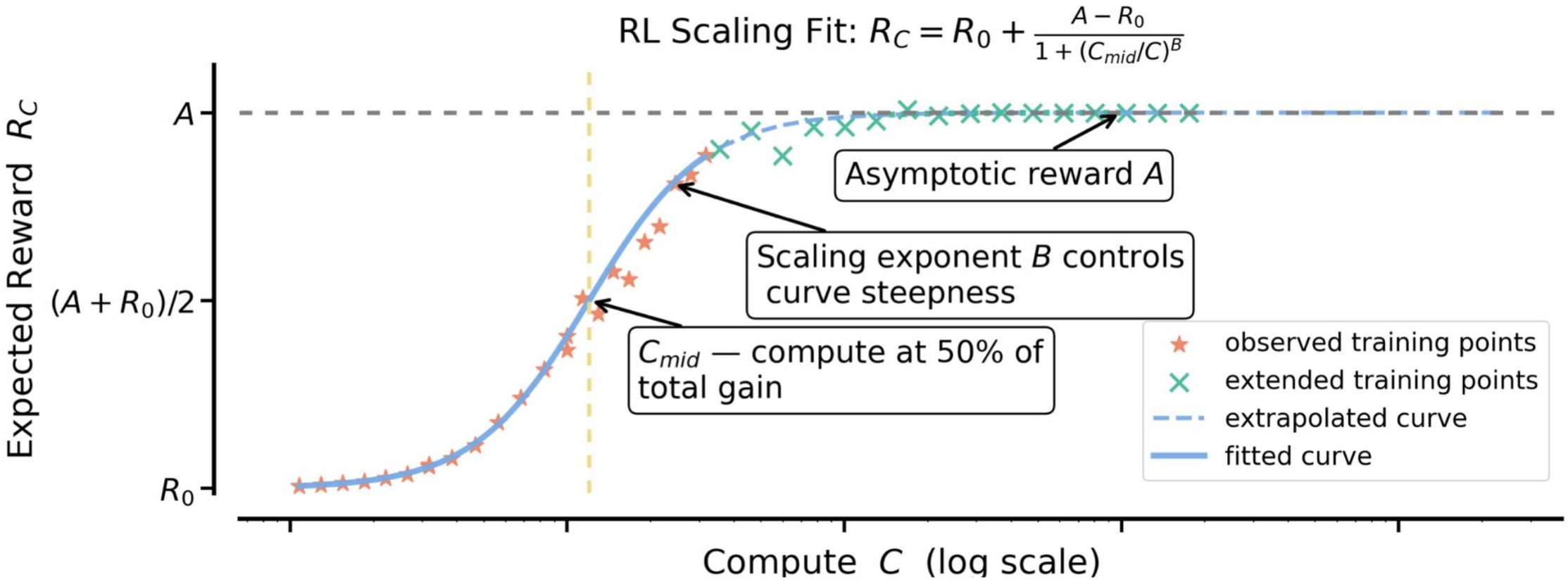
(fixed model and training data)

- Function of compute
- Bounded (because of accuracy metric)

Predictable Scaling



Predictable Scaling in RL

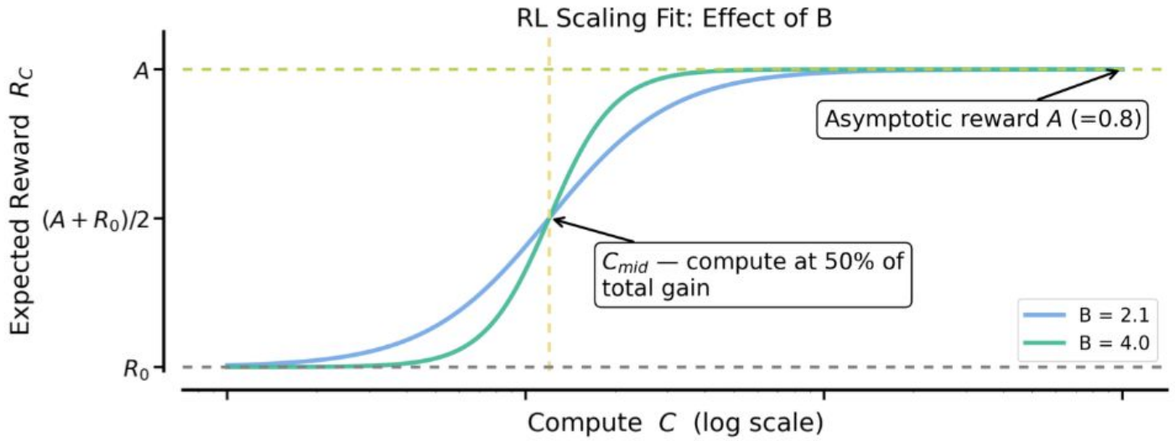


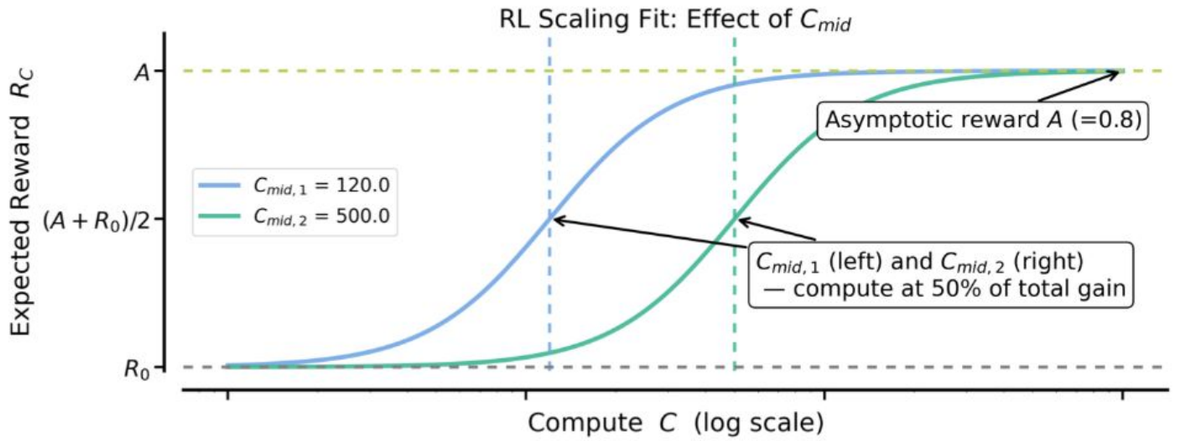
A **Asymptotic Ceiling**
How high?

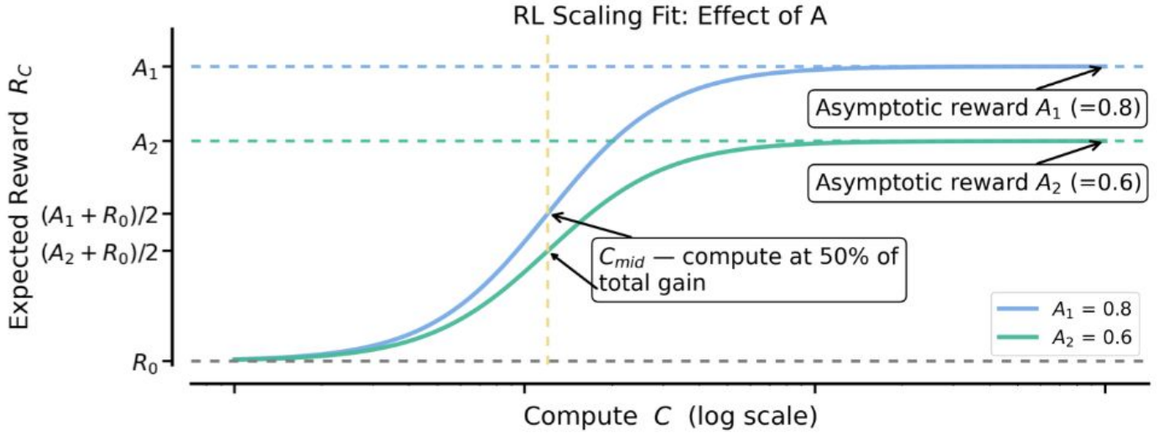
B **Compute Efficiency**
How fast?

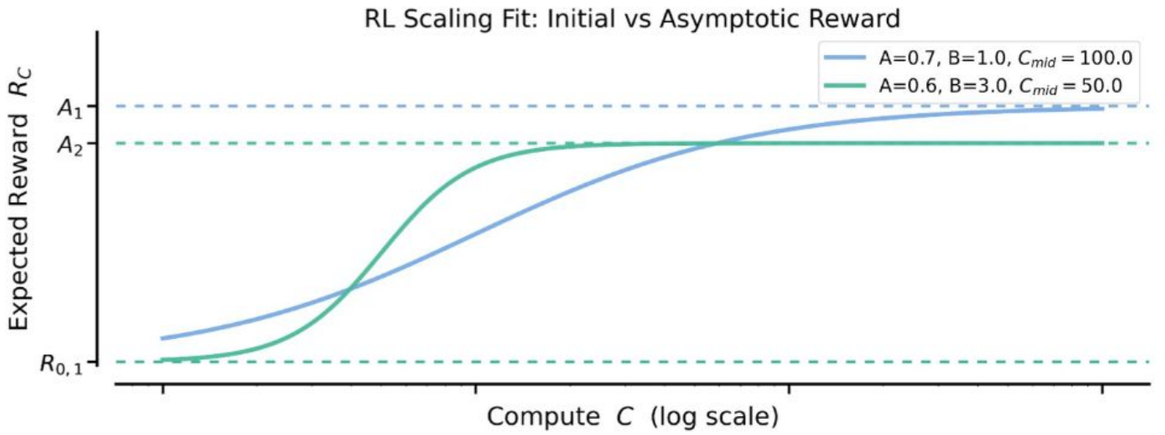
C_mid **Inflection**
When saturate?

Predictable Scaling in RL

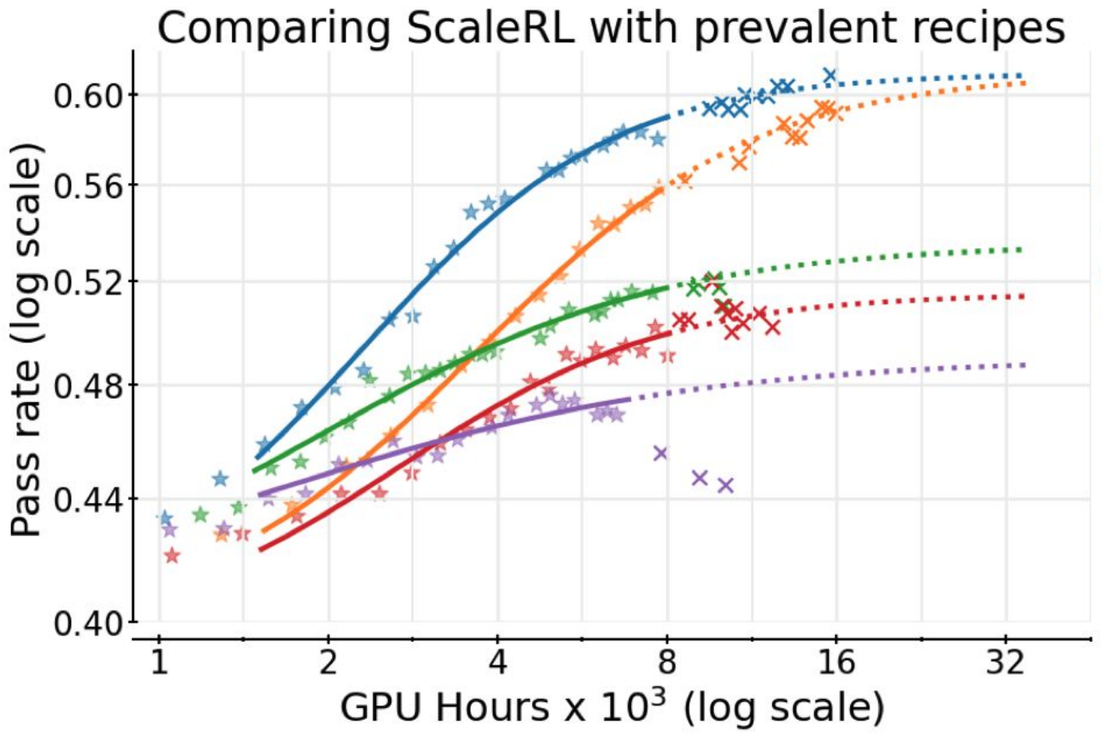








Different recipes have different ceilings



- ScaleRL B: 1.97, A: 0.610
- MiniMax B: 1.77, A: 0.610
- Magistral B: 1.44, A: 0.535
- Qwen2.5 (DAPO) B: 1.85, A: 0.515
- DeepSeek (GRPO) B: 1.17, A: 0.490

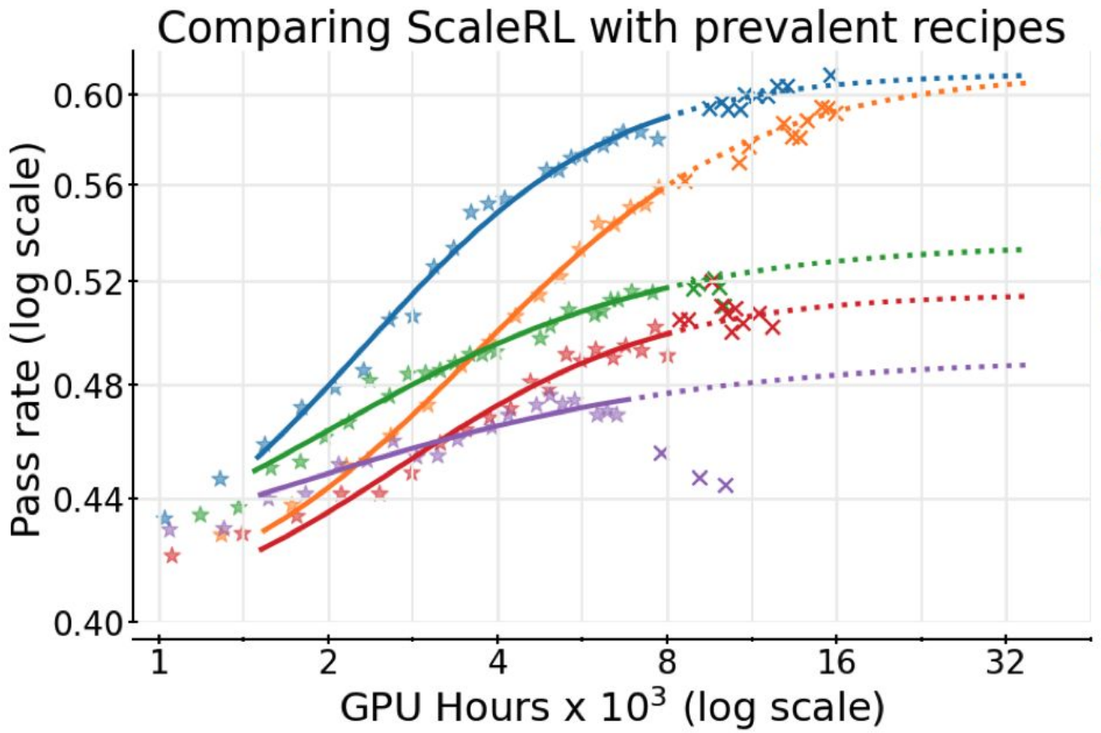
★ training points
× extended training points
— fitted curve
-- extrapolated curve

ScaleRL recipe

$$\mathcal{J}_{\textbf{ScaleRL}}(\theta) = \mathop{\mathbb{E}}_{\substack{x\sim D,\\ \{y_i\}_{i=1}^{G}\sim\pi_{gen}^{\theta_{old}}(\cdot|x)}}\left[\frac{1}{\sum_{g=1}^{G}|y_g|}\sum_{i=1}^{G}\sum_{t=1}^{|y_i|}\textbf{sg}(\min(\rho_{i,t},\epsilon))\hat{A}_i^{\text{norm}}\,\log\pi_{train}^{\theta}(y_{i,t})\right],$$

$$\rho_{i,t}=\frac{\pi_{train}^{\theta}(y_{i,t})}{\pi_{gen}^{\theta_{old}}(y_{i,t})},\ \hat{A}_i^{\text{norm}}=\hat{A}_i/\hat{A}_{\text{std}},\ 0<\text{mean}(\{r_j\}_{j=1}^{G})<1,\ \text{pass\_rate}(x)<0.9,$$

Different recipes have different ceilings



ScaleRL	B: 1.97, A: 0.610
MiniMax	B: 1.77, A: 0.610
Magistral	B: 1.44, A: 0.535
Qwen2.5 (DAPO)	B: 1.85, A: 0.515
DeepSeek (GRPO)	B: 1.17, A: 0.490

★ training points
× extended training points
— fitted curve
-- extrapolated curve

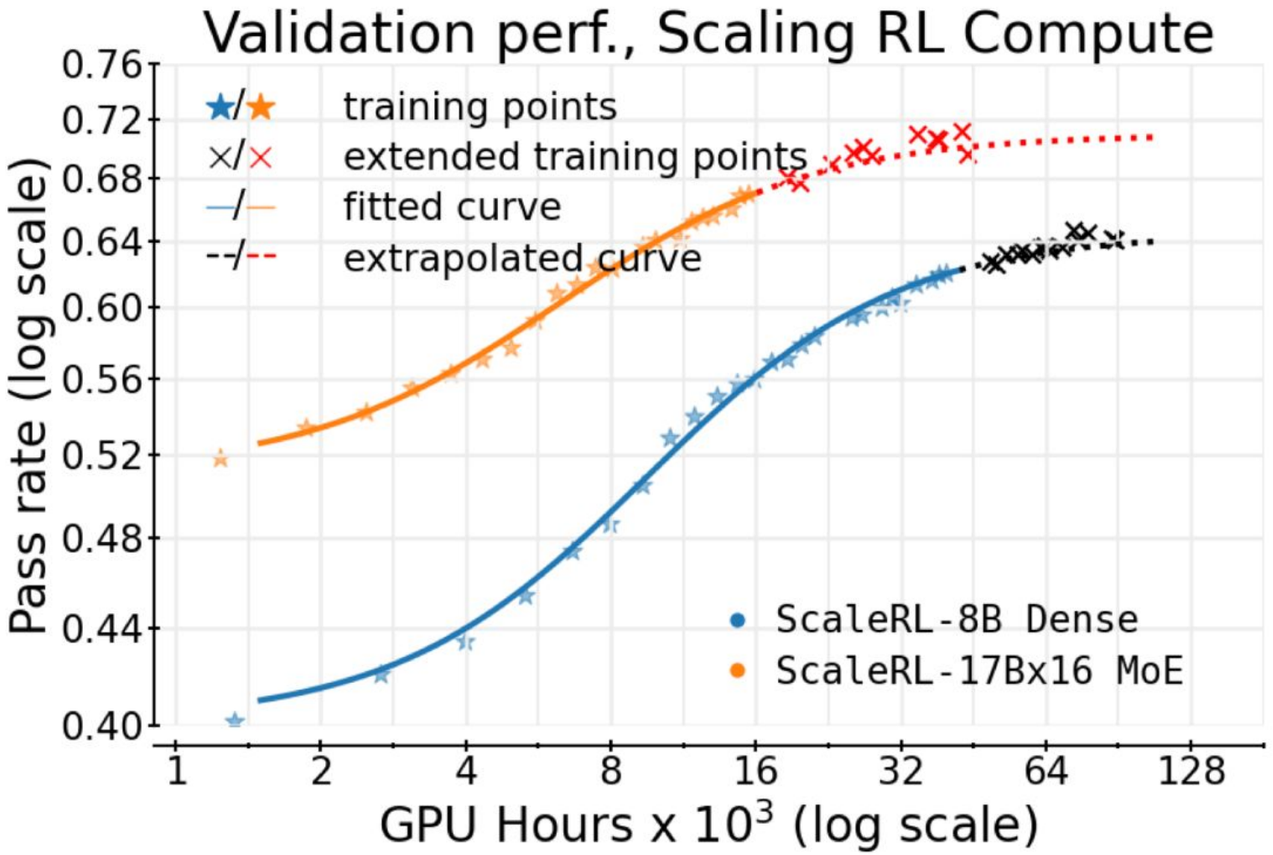
ScaleRL recipe

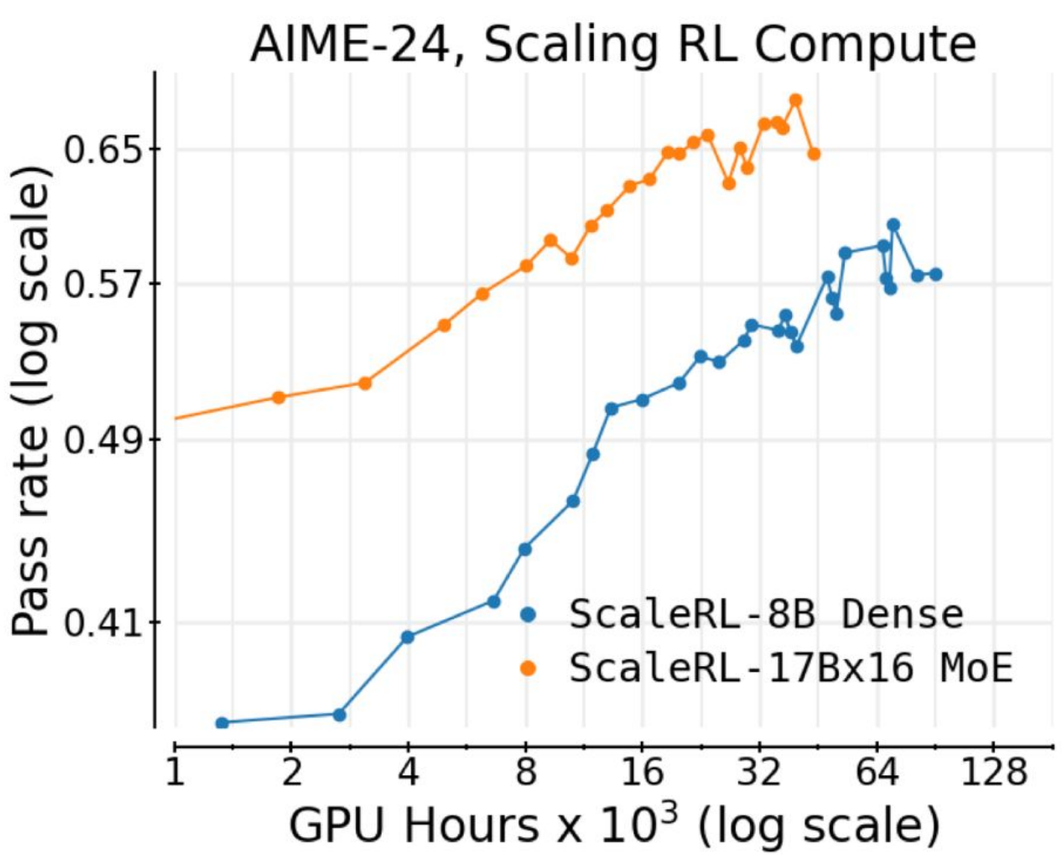
$$\mathcal{J}_{\textbf{ScaleRL}}(\theta) = \mathop{\mathbb{E}}_{\substack{x\sim D,\\ \{y_i\}_{i=1}^{G}\sim\pi_{gen}^{\theta_{old}}(\cdot|x)}}\left[\frac{1}{\sum_{g=1}^{G}|y_g|}\sum_{i=1}^{G}\sum_{t=1}^{|y_i|}\textbf{sg}(\min(\rho_{i,t},\epsilon))\hat{A}_i^{\text{norm}}\log\pi_{train}^{\theta}(y_{i,t})\right],$$

$$\rho_{i,t}=\frac{\pi_{train}^{\theta}(y_{i,t})}{\pi_{gen}^{\theta_{old}}(y_{i,t})},\ \hat{A}_i^{\text{norm}}=\hat{A}_i/\hat{A}_{\text{std}},\ 0<\text{mean}(\{r_j\}_{j=1}^{G})<1,\ \text{pass\_rate}(x)<0.9,$$

Cannot fix a low-ceiling recipe by training for longer!

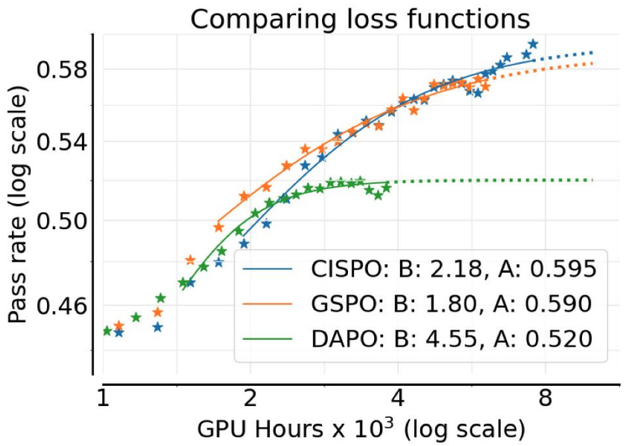
Predictions at O(100k) GPU-hours

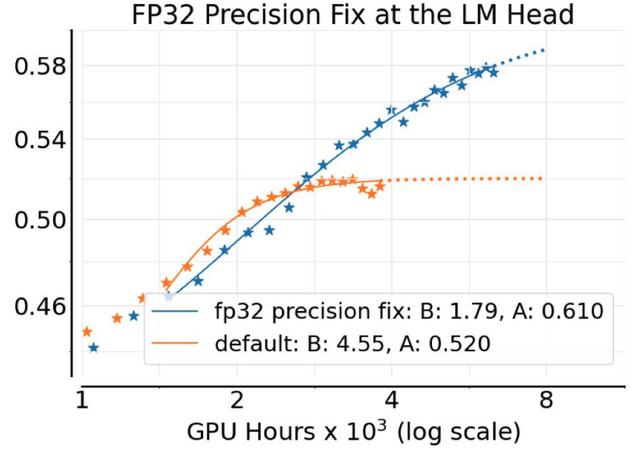


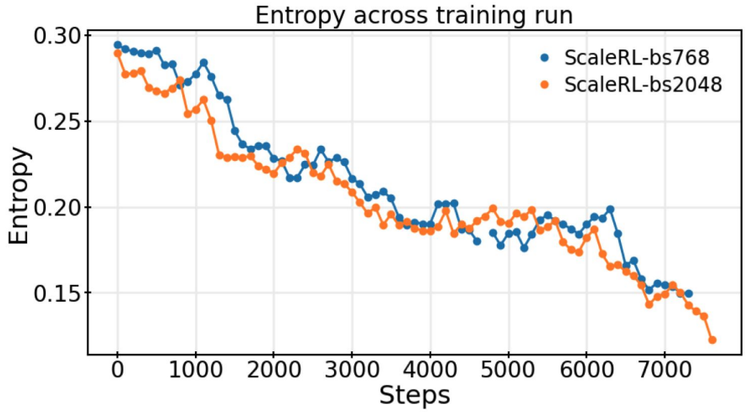


Granular Findings

- Higher asymptote
  - Loss function (CISPO over DAPO/GRPO), numerics (FP32 at the output head)
- Training efficiency
  - Off-policy (PipelineRL), data curriculum
- Scaling obvious axes
  - Generation length, batch size, model size







Thank You!

https://arxiv.org/abs/2510.13786