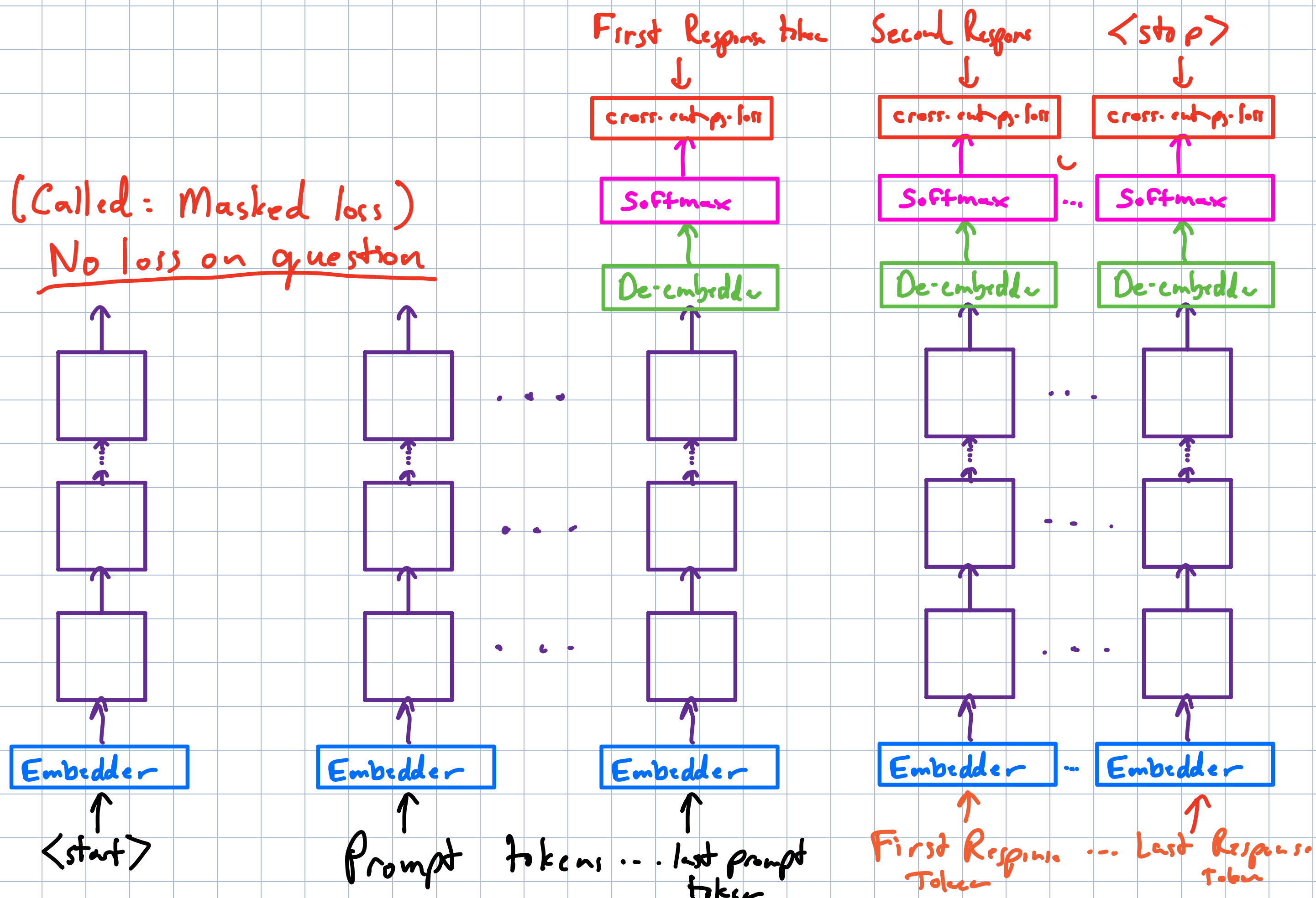
Today: Post-training and test-time compute

(We'll return to generative models after Post-training interlude)

Announce: Fill out survey
Extra Credit for everyone (3%) if 75% of class does the survey
Due: Fri of Thanksgiving week.
This Friday!!!
Finish your peer review Today!!!

Post-training including RLVR.
Recall basic SFT for instruction following.



For LLMs, one key advance was "Chain-of-thought." So it is useful to have examples that show their work while solving problems.

But how can we make a model better at solving problems?

Two parts to the answer:

A) Be willing to spend more compute while answering.
↖ test-time compute

B) Train it to be better.

__Test-time compute:__ 0) Oldest Approach: pure prompting.

"Think step by step. Be careful."

1) Repeated generation. Instead of one try, generate N responses.

At Inference/Test time: A) Take Majority/Plurality Vote
B) Take highest prob. — variant
C) Train a reward model to grade.

2) Sample better...

Recent Paper: Karan & Du, "Reasoning with Sampling: Your base model is smarter than you think" Oct 16, 2025.

We want a good answer. That includes thinking. If we sample from original model, what can go wrong? Moving forward one token at a time, we can make a mistake... and then flounder.

Older Approach: Beam-Search with some k.

To do what? Sample from higher-likelihood __sequences.__

Alternative: Sample not from $p(\overset{\text{sequence.}}{\vec{x}})$
but from normalized $(p(\vec{x}))^{\alpha}$ when $\alpha > 1$

Is this just low temperature sampling?

No. Consider position t.

$$P_{desired}(x_t \mid x_0, \ldots, x_{t-1}) = \frac{\sum_{x_{>t}} \left(p(x_0, \ldots, x_{t-1}, x_t, \overbrace{x_{t+1}, \ldots, x_T}^{\text{"integrate out"}})\right)^{\alpha}}{\sum_{\tilde{x}_t} \left[\sum_{x_{>t}} \left(p(x_0, \ldots, x_{t-1}, \tilde{x}_t, x_{t+1}, \ldots, x_T)\right)^{\alpha} \right]}$$

The corresponding low-temp sampling doesn't raise the future completions to the desired power and then sum them.

Instead

$$P_{\text{LowTem}}(x_t \mid x_0, \ldots, x_{t-1}) = \frac{\left(p(x_t \mid x_0, \ldots, x_{t-1})\right)^{\alpha}}{\sum_{\tilde{x}_t} \left(p(\tilde{x}_t \mid x_0, \ldots, x_{t-1})\right)^{\alpha}}$$

$$= \frac{\left(\sum_{x_{>t}} p(x_0, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots, x_T)\right)^{\alpha}}{\sum_{\tilde{x}_t} \left[\left(\sum_{x_{>t}} p(x_0, \ldots, x_{t-1}, \tilde{x}_t, x_{t+1}, \ldots, x_T)\right)^{\alpha}\right]}$$

Raising a sum to a power is not the same as the sum of powers

Key Idea in the paper: Use MCMC Techniques to sample

Algorithm 1: Power Sampling for Autoregressive Models

Input : base p; proposal p_{prop}; power α; length T

Hyperparams: block size B; MCMC steps N_{MCMC}

Output : $(x_0, \ldots, x_T) \sim p^{\alpha}$

Notation: Define the unnormalized intermediate target

$$\pi_k(x_{0:kB}) \propto \boxed{p(x_{0:kB})^{\alpha}}.$$

Goal

for $k \leftarrow 0$ **to** $\lceil \frac{T}{B} \rceil - 1$ **do**

Given prefix $x_{0:kB}$, we wish to sample from π_{k+1}. Construct initialization $\mathbf{x}^0$ by extending autoregressively with p_{prop}:

$$x_t^{(0)} \sim p_{\text{prop}}(x_t \mid x_{<t}), \qquad \text{for } kB + 1 \leq t \leq (k+1)B.$$

Set the current state $\mathbf{x} \leftarrow \mathbf{x}^0$.

for $n \leftarrow 1$ **to** N_{MCMC} **do**

Sample an index $m \in \{1, \ldots, (k+1)B\}$ uniformly.

Construct proposal sequence $\mathbf{x}'$ with prefix $x_{0:m-1}$ and resampled completion:

$$x_t' \sim p_{\text{prop}}(x_t \mid x_{<t}), \qquad \text{for } m \leq t \leq (k+1)B.$$

← Take random steps towards more likely generations

Compute acceptance ratio (9)

$$A(\mathbf{x}', \mathbf{x}) \leftarrow \min\left\{1, \frac{\pi_k(\mathbf{x}')}{\pi_k(\mathbf{x})} \cdot \frac{p_{\text{prop}}(\mathbf{x} \mid \mathbf{x}')}{p_{\text{prop}}(\mathbf{x}' \mid \mathbf{x})}\right\}.$$

→ Traditional way to write in MCMC. In this case, reduces to $\frac{P_{\text{prop}}(x)}{P_{\text{prop}}(x')}$ Because of AR Generation in P_{prop}.

Draw $u \sim \text{Uniform}(0, 1)$;

if $u \leq A(\mathbf{x}', \mathbf{x})$ **then accept** and set $\mathbf{x} \leftarrow \mathbf{x}'$

end

Set $x_{0:(k+1)B} \leftarrow \mathbf{x}$ to fix the new prefix sequence for the next stage.

end

return $x_{0:T}$

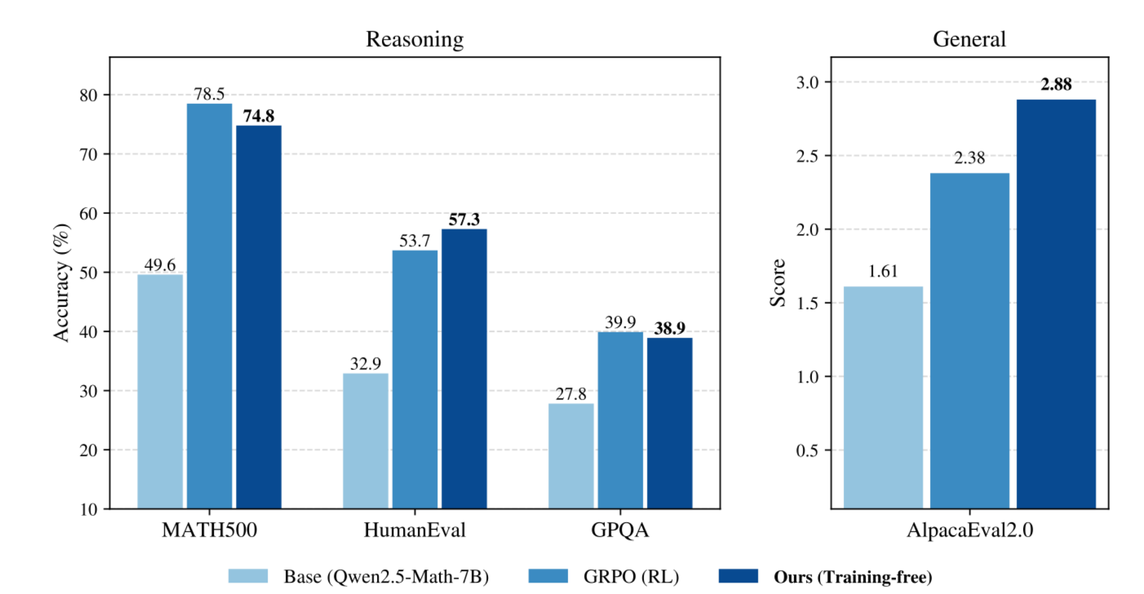


Fig 1 from Paper.

Exercise: What simpler alternative should also have been in this paper?

1) Try "best-of-N" with highest-prob seq chosen

1') Try beam search

2) Pure prompt optimization

As distinct from RLHF: -- with human feedback.

RLVR: Reinforcement Learning with Verifiable Rewards

Key new paper: Khatri, et al "The Art of Scaling Reinforcement Learning Compute for LLMs," Oct 15, 2025.

Treatment here influenced by this, along with DeepSeek paper.

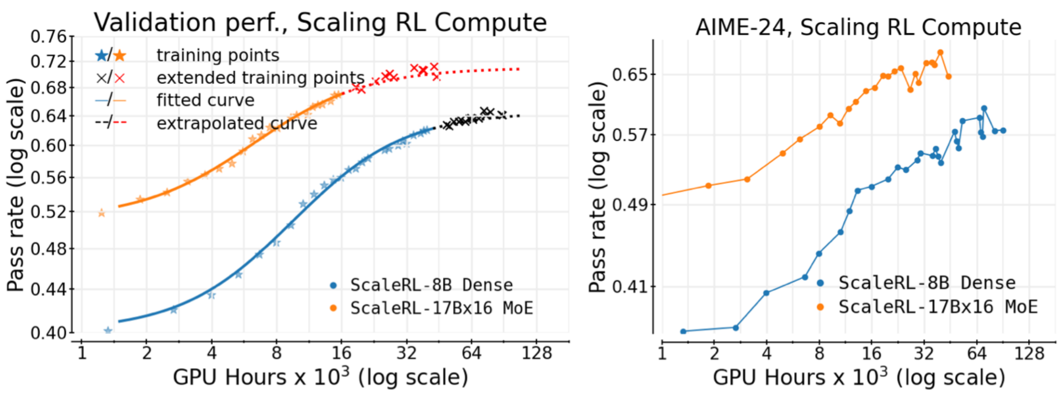


Fig 1 in ScaleRL..

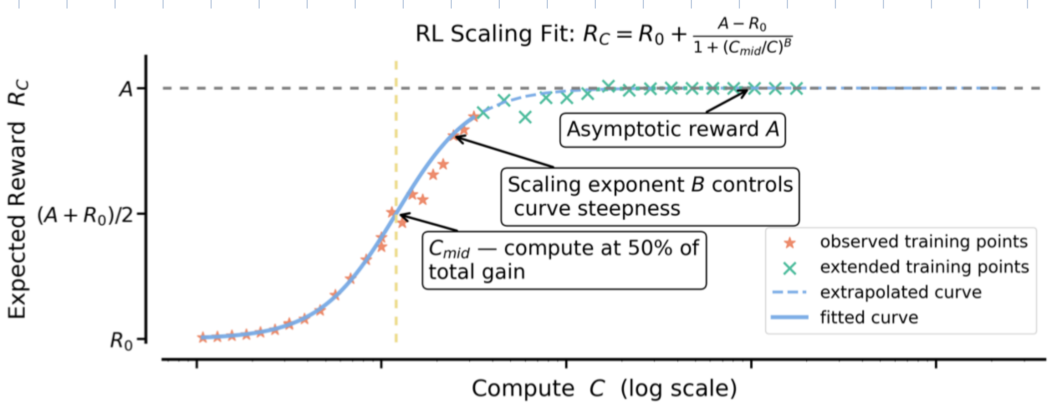


Fig 3 in ScaleRL

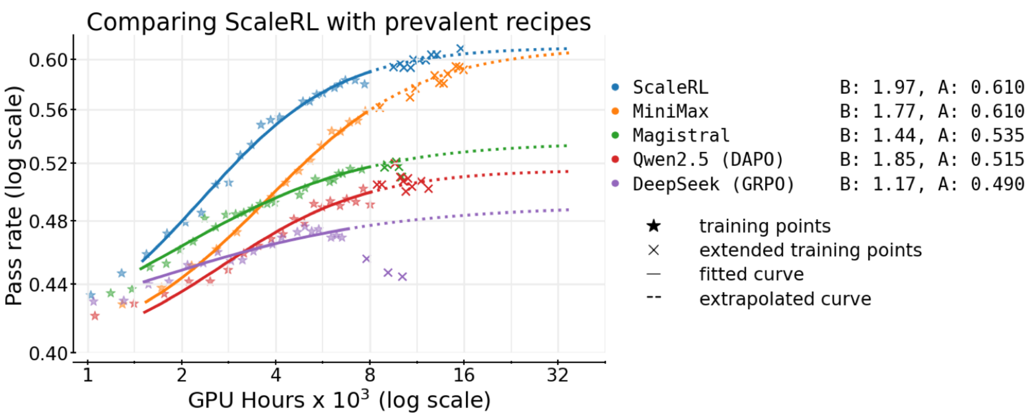


Fig 2 in ScaleRL...

"Bag of tokens perspective"

Unpacking what to do...

Generations   Stop Gradient   HW 3 log P_θ

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

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

"Importance Sampling" Distribution Mismatch.

Demean across G generations reward

Don't Bother if we get 90% right.

VR gives this.

Recall HW3 Trick (Reparameterization Gradient Estimator)

$$\nabla_\theta \mathbb{E}_{y\sim p_\theta}[F(y)] = \mathbb{E}_{y\sim p_\theta}[F(y) \nabla_\theta \log p_\theta]$$

sg$(\cdots)$ ← To stop the gradient from inadvertently going into $\rho_{i,t}$.

$$\min(\rho_{i,t}, \epsilon) = \begin{cases} \epsilon & \text{if } \rho_{i,t} \geq \epsilon \\ \rho_{i,t} & \text{if } \rho_{i,t} < \epsilon \end{cases}$$

→ Caps the overweighting of this sample in case it is strongly preferred by π_{train} as opposed to π_{gen}

In general $\pi_{gen}^{\theta_{old}}$ closely tracks π_{train}^{θ} since it gets updated with a lag.