CS 957, System-2 AI
Test-Time Scaling

Mahdieh Soleymani

April 2025

Sharif University of Technology

Recap

- How to improve solving reasoning problems by test-time compute
  - Prompting
  - Decoding

- Scaling Methods
  - Sequential revision
  - Self-Consistency
  - BoN
  - Beam Search

Approches for deciding on choices and paths

- Prompting a general LLM to get feedback
  - LLMs cannot feedback or self-correct reasoning effectively
- LLM’s distribution: majority voting and confidence
- Training a model-based verifier

Verifier or Reward Model (RM)

- Training reward models to discriminate desirable and undesirable outputs.
  - RM as a LM, trained to predict a binary label as either a ‘correct’ or ‘incorrect’ token

- Conditioned on the problem and a candidate solution, the verifier outputs the probability that the solution is correct

$$RM: \mathcal{P} \times \mathcal{S} \rightarrow [0,1]$$

- The reward model can then be used in a reinforcement learning pipeline or to perform search via rejection sampling

H. Lightman et al., Let’s Verify Step by Step, 2023

Aggregation: Best-of-N weighted

- combine the benefits of verifiers and majority voting
- extend the idea of verifier and weigh each sample by its score from external verifiers

$$a^*_{BoN\_weighted} = \underset{a}{\operatorname{argmax}} \sum_{i=1}^{N} \mathbb{I}(a_i = i) RM(p, S_i)$$

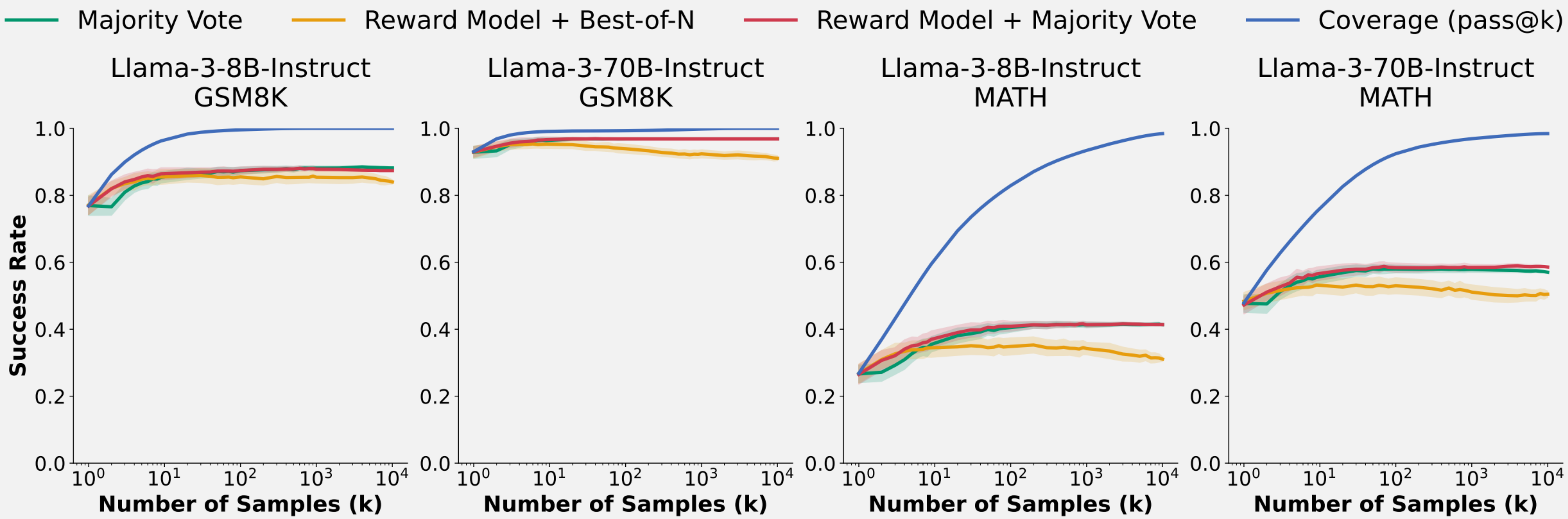
- $RM(p, S_i)$ is the score of the i-th solution assigned by ORM or PRM for problem p

Verification

- Verifying the correctness and consistency of LLM during the test-time scaling is also crucial
    - **Final Selection**: directly selects the output sample among various ones
    - **During Search**: serves as the criteria in the search process
    - **Termination**: determines when to stop, under the Sequential Scaling paradigm
    - **Aggregation**: what sample to aggregate and how to aggregate them

Modifying the sampler

- **Optimizing the verifier**
- Modifying the proposal distribution
  - By conditioning
  - By fine-tuning



B. Brown et al., Large Language Monkeys: Scaling Inference Compute with Repeated Sampling, 2024

Training Verifiers to Solve Math Word Problems

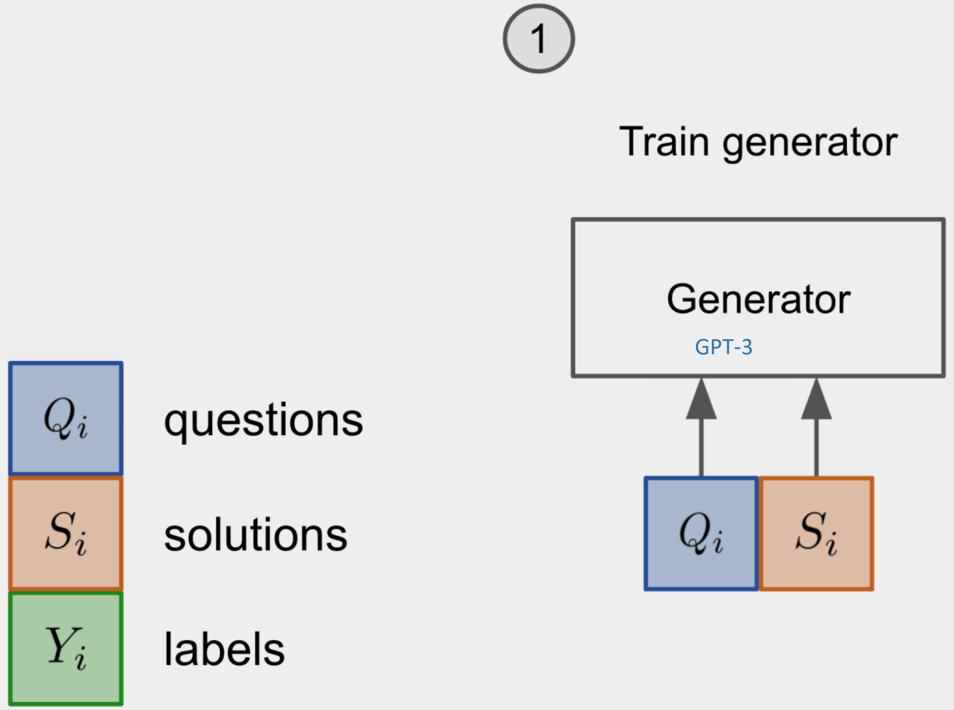
Karl Cobbe* **Vineet Kosaraju*** **Mohammad Bavarian** **Mark Chen**

Heewoo Jun **Łukasz Kaiser** **Matthias Plappert** **Jerry Tworek**

Jacob Hilton **Reiichiro Nakano** **Christopher Hesse** **John Schulman**

OpenAI

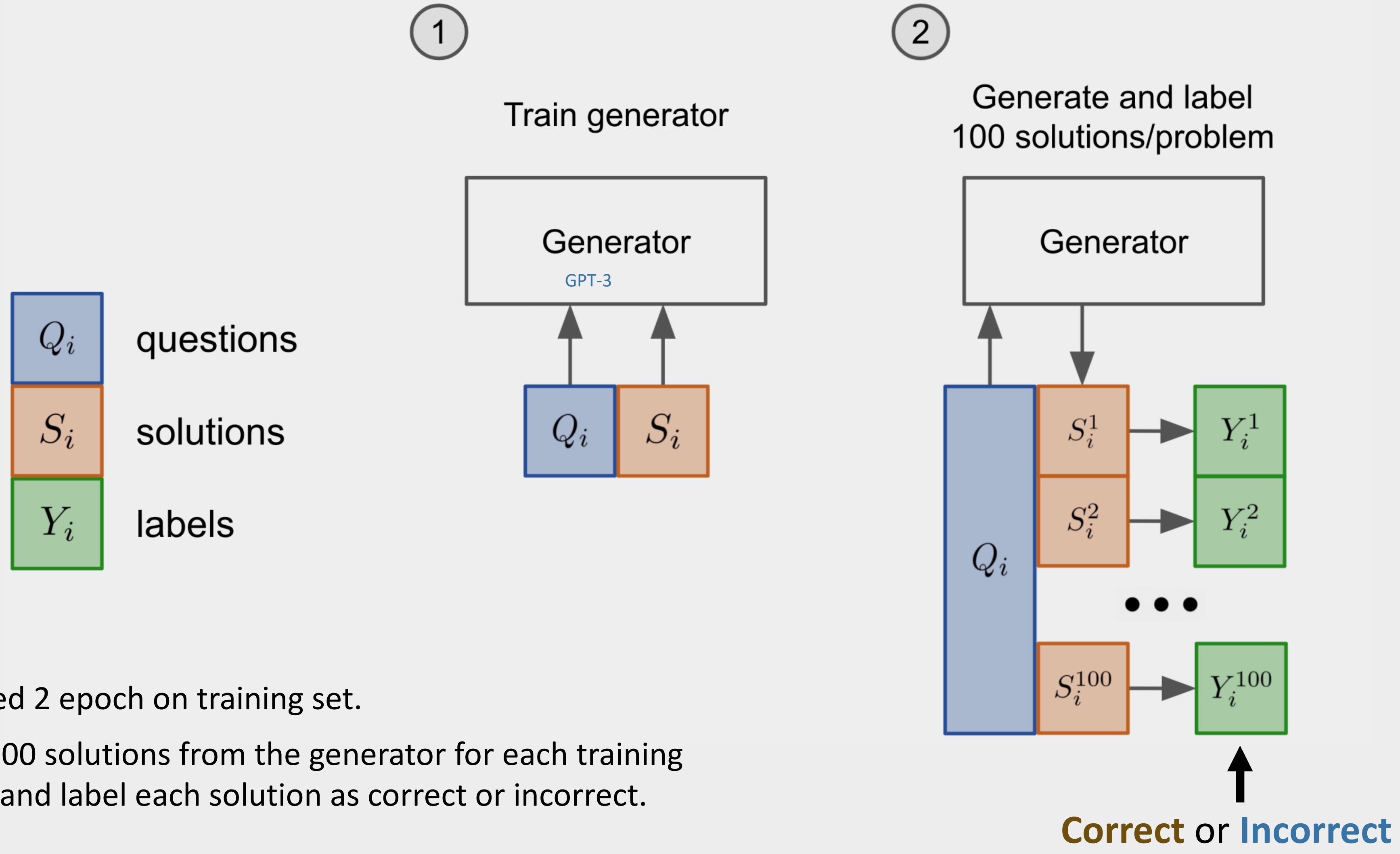
Fine-tuning + Verification



1. Fine-tuned 2 epoch on training set.

Cobbe et al., Training verifiers to solve math word problems, 2021

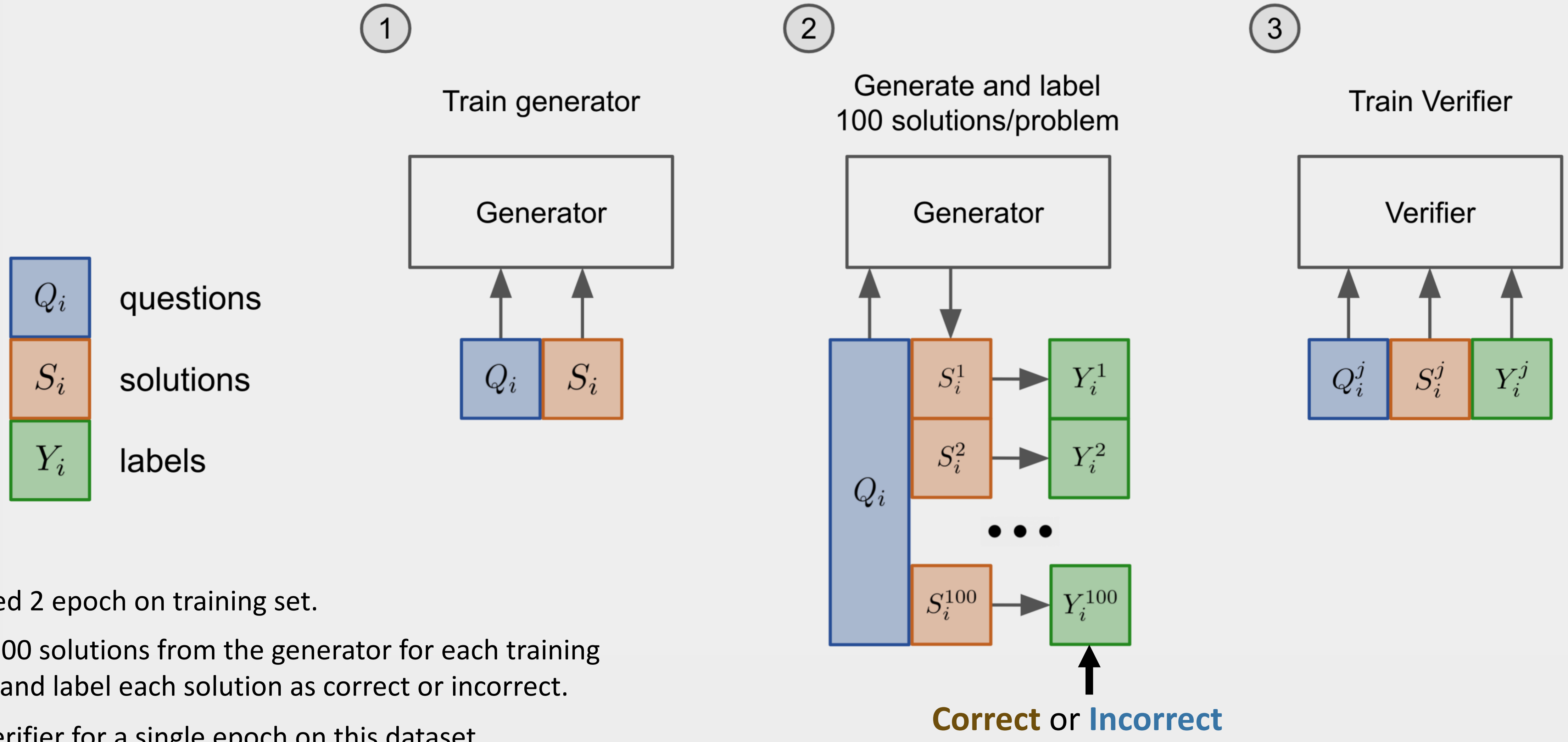
Fine-tuning + Verification



1. Fine-tuned 2 epoch on training set.
2. Sample 100 solutions from the generator for each training problem and label each solution as correct or incorrect.

Cobbe et al., Training verifiers to solve math word problems, 2021

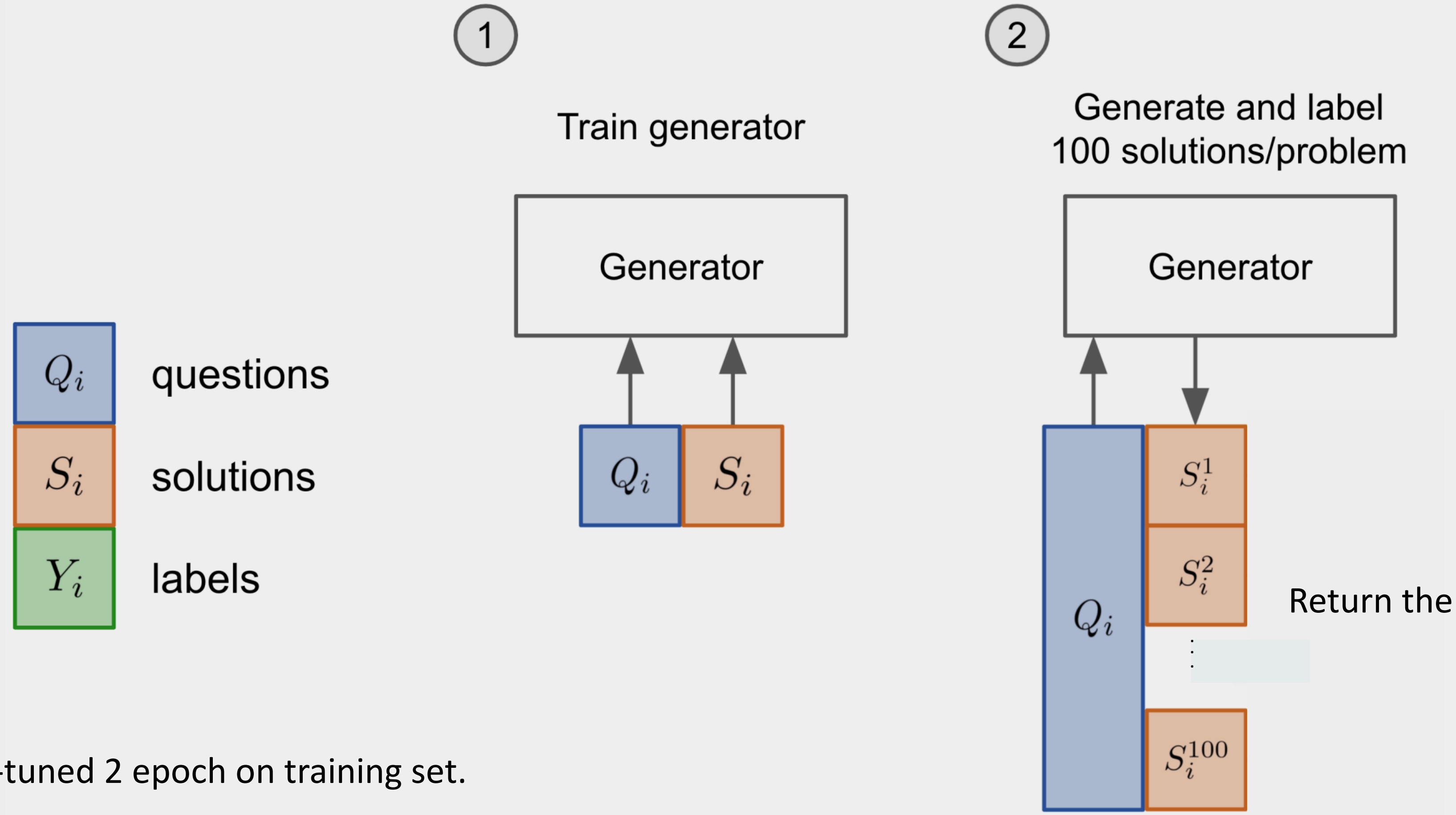
Fine-tuning + Verification



1. Fine-tuned 2 epoch on training set.
2. Sample 100 solutions from the generator for each training problem and label each solution as correct or incorrect.
3. Train a verifier for a single epoch on this dataset.

Cobbe et al., Training verifiers to solve math word problems, 2021

Fine-tuning + Verification



Return the one with the highest verifier score

1. Fine-tuned 2 epoch on training set.

Cobbe et al., Training verifiers to solve math word problems, 2021

Verifier or Reward Model (RM): Training

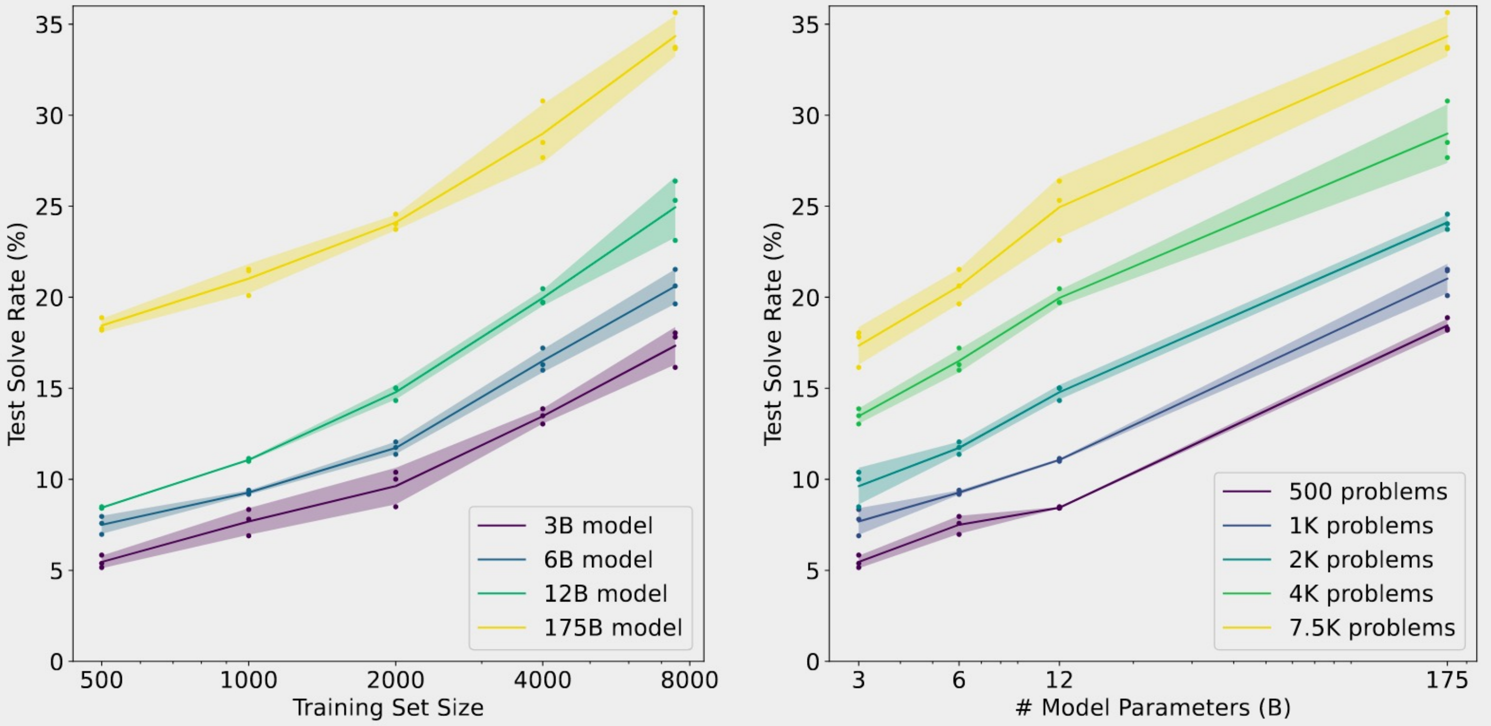
- Steps of training:
  1. **Finetune a model** (the “generator”) for 2 epochs on the training set.
  2. **Sample** 100 completions from the generator for each training problem and **label** each solution as correct or incorrect.
  3. **Train a verifier** for a single epoch on this dataset.

- In addition to predicting solution correctness, the same language modeling objective as the generator is also considered

Cobbe et al., Training verifiers to solve math word problems, 2021

Finetuned Model

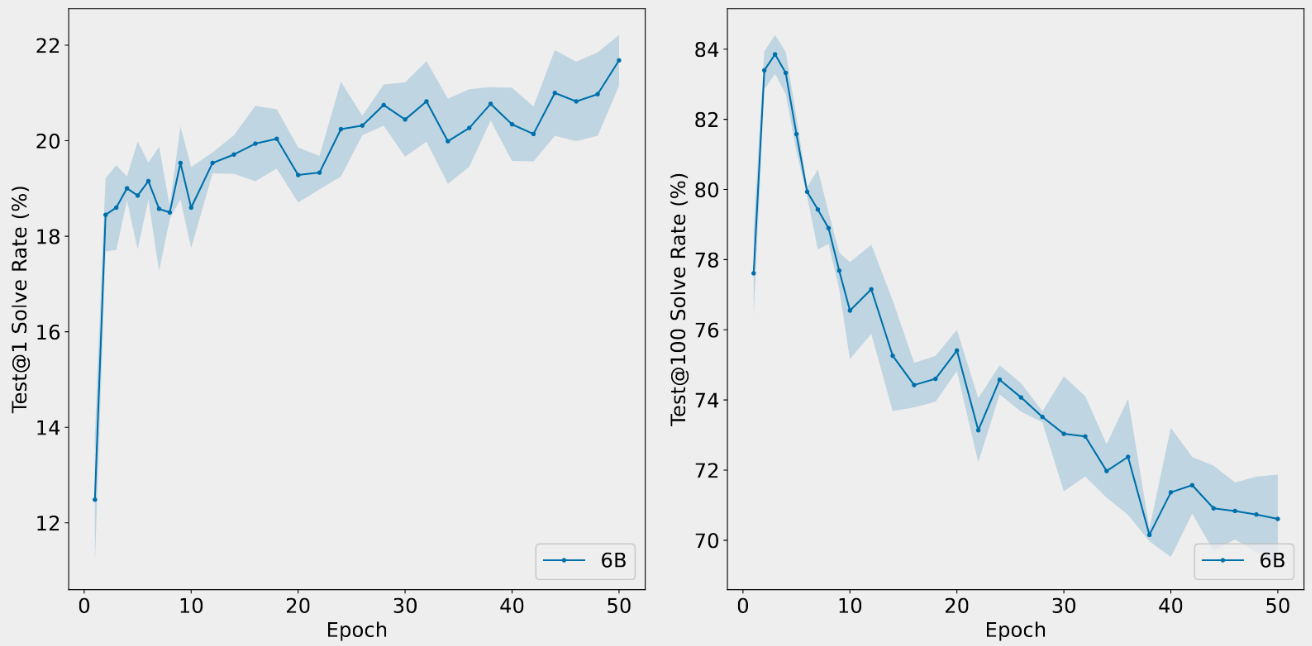
- Updating model parameters to minimize the cross-entropy loss over all training tokens



Cobbe et al., Training verifiers to solve math word problems, 2021

Finetuned Model

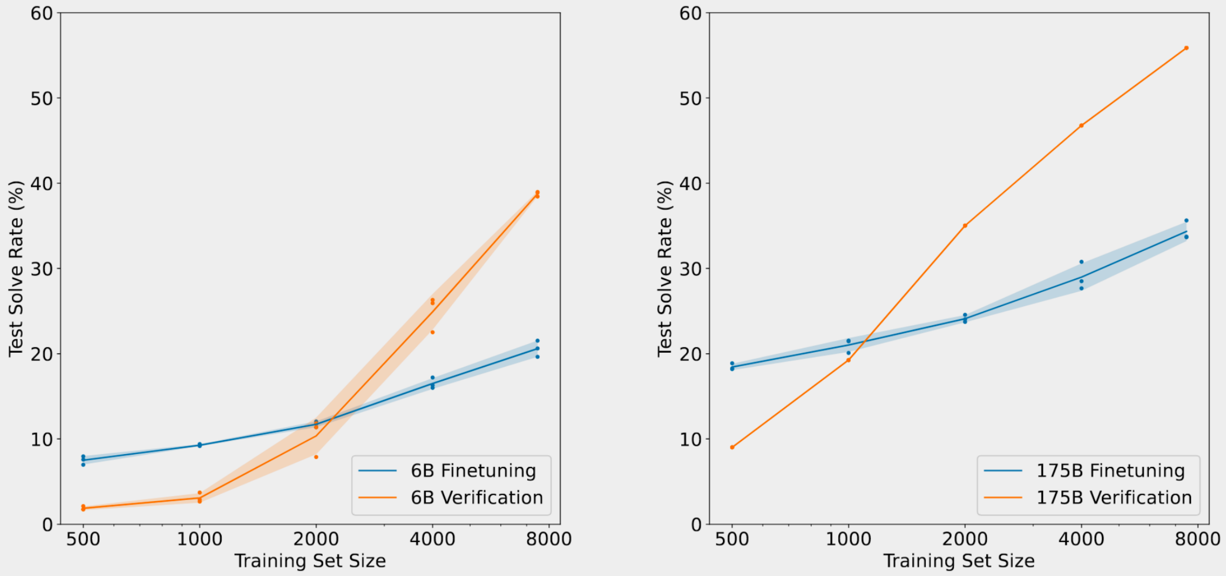
- Test solve rate
  - percentage of problems solved correctly at least once when allowing the model to make N separate guesses



Cobbe et al., Training verifiers to solve math word problems, 2021

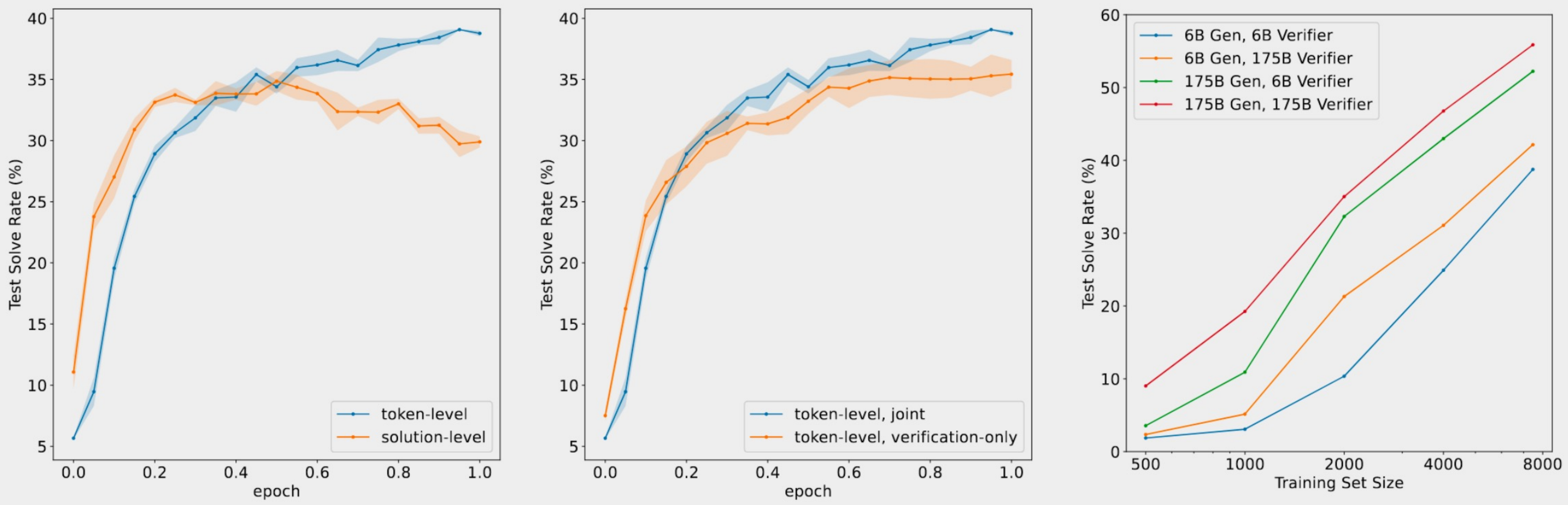
Fine-tuning + Verification

- Verification: Best-of-100
- Finetuning: autoregressively sampling a single greedy solution

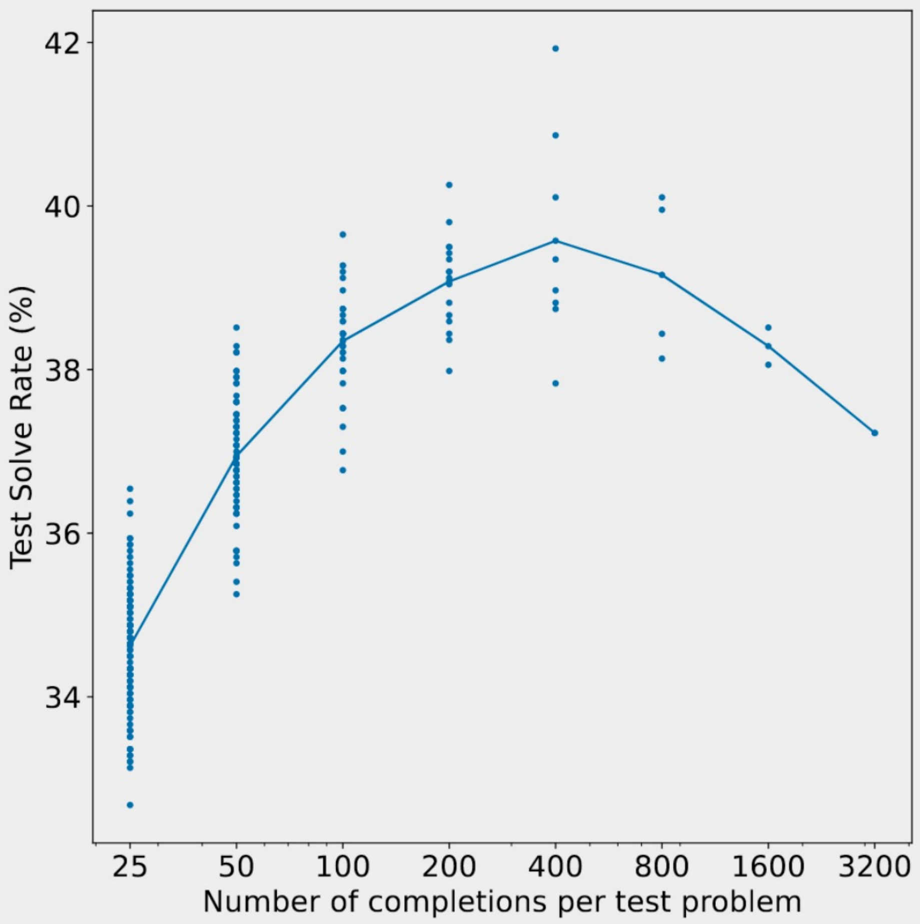


Cobbe et al., Training verifiers to solve math word problems, 2021

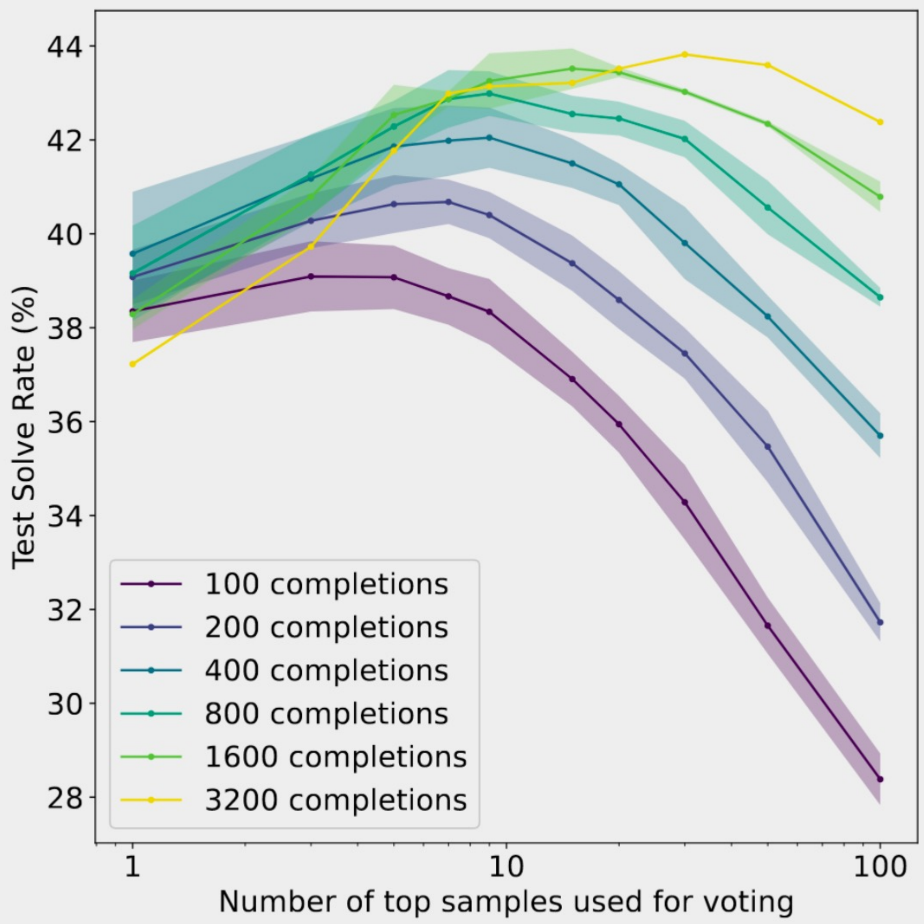
Fine-tuning + Verification: Ablations



Cobbe et al., Training verifiers to solve math word problems, 2021



(a) 6B verification test performance when given varying numbers of completions per problem to rank.



(b) 6B verification test performance when varying the number of top ranked samples allowed to vote on the answer.

Cobbe et al., Training verifiers to solve math word problems, 2021

ORM Shortcomings

- Correct final answer while incorrect reasoning traces
  - problematic in many real-world domains such as education
  - Within the domain of logical reasoning, models trained with outcome supervision regularly use incorrect reasoning to reach the correct final answer

- However, for correct reasoning steps it is necessary to use PRMs

H. Lightman et al., Let’s Verify Step by Step, 2023

J. Uesato et al., Solving math word problems with process and outcome-based feedback, 2022

Verifier or Reward Model (RM)

- **Outcome Reward Model (ORM)**: for each step indicates whether the resulting final answer of that full sample matched the reference final answer

- **Process Reward Model (PRM):** the binary label after each step indicates whether the steps so far are correct.
  - provides more precise feedback
  - human annotations for the correctness of intermediate steps

H. Lightman et al., Let's Verify Step by Step, 2023
J. Uesato et al., Solving math word problems with process and outcome-based feedback, 2022

PRMs for TTS

- For search-based methods, PRMs guide the selection at each response step

- For parallel sampling-based methods, PRMs evaluate the responses after generation.

Training ORM and PRM

- Train the ORM using samples from the policy for that approach
  - taking K = 96 samples with temperature 1.

- For the PRM
  - 3 samples per problem from the SFT policy is annotated
    - restricting to problems where the SFT majority prediction was incorrect
  - Offline human-generated reasoning traces from the GSM8K are also included
  - Due to the small size of our annotated dataset, the PRM is initialized by ORM

J. Uesato et al., Solving math word problems with process and outcome-based feedback, 2022

Comparison of ORM and PRM

- This study found that ORM and PRM led to similar final performance in the domain of grade school math.

J. Uesato et al., Solving math word problems with process and outcome-based feedback, 2022

Let's Verify Step by Step

Hunter Lightman* **Vineet Kosaraju*** **Yura Burda*** **Harri Edwards**

Bowen Baker **Teddy Lee** **Jan Leike** **John Schulman** **Ilya Sutskever**

Karl Cobbe*

OpenAI

PRM800K Dataset

- **presents human data-labelers with step-by-step solutions** to MATH problems sampled by the large-scale generator.

- contains 800K step-level labels across 75K solutions to 12K problems.
    - include also data from 4.5K MATH test

- assign each step in the solution a label of positive, negative, or neutral.

H. Lightman et al., Let’s Verify Step by Step, 2023

PRM800K Dataset and Training

- **Strategically select** which solutions to show data-labelers:
  - wrong-answer: solutions that reach an incorrect final answer.
  - convincing wrong-answer solutions: solutions that are wrong but rated highly by the current best PRM

- **Iteratively re-train PRM**:
  - At each iteration, generate N solutions per problem
  - surface only the top K most convincing wrong-answer solutions to data-labelers.

- chooses to supervise only up to the first incorrect step to provide similar supervision

H. Lightman et al., Let's Verify Step by Step, 2023

Training Set of PRM vs. ORM

- training the large-scale PRM using the step-level labels in **PRM800K**.
- Training the ORM solely on PRM800K solutions would be problematic
  - since active learning strategy has heavily biased the dataset towards wrong answer solutions.
- To prepare the large-scale ORM baseline, the verifier is trained on 100 uniform samples per problem from the generator.

H. Lightman et al., Let’s Verify Step by Step, 2023

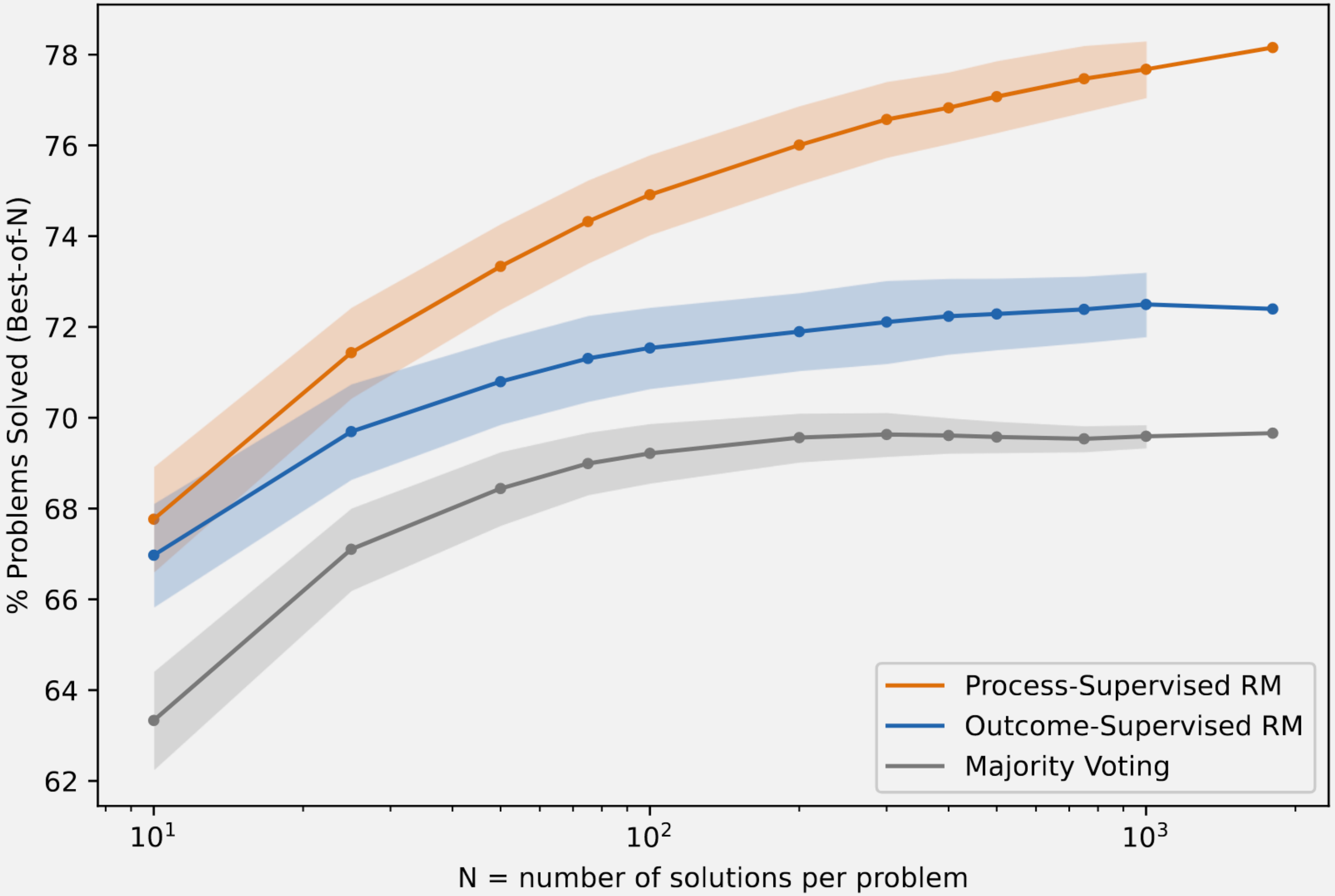
Preparation to compare ORM and PRM

- Stepwise aggregation: PRM score as the product of the correctness probabilities for each step is computed

- Inter-answer aggregation: RM-weighted voting (Uesato et al., 2022) but this did not noticeably improve performance

H. Lightman et al., Let’s Verify Step by Step, 2023

	ORM	PRM	Majority Voting
% Solved (Best-of-1860)	72.4	**78.2**	69.6



H. Lightman et al., Let's Verify Step by Step, 2023

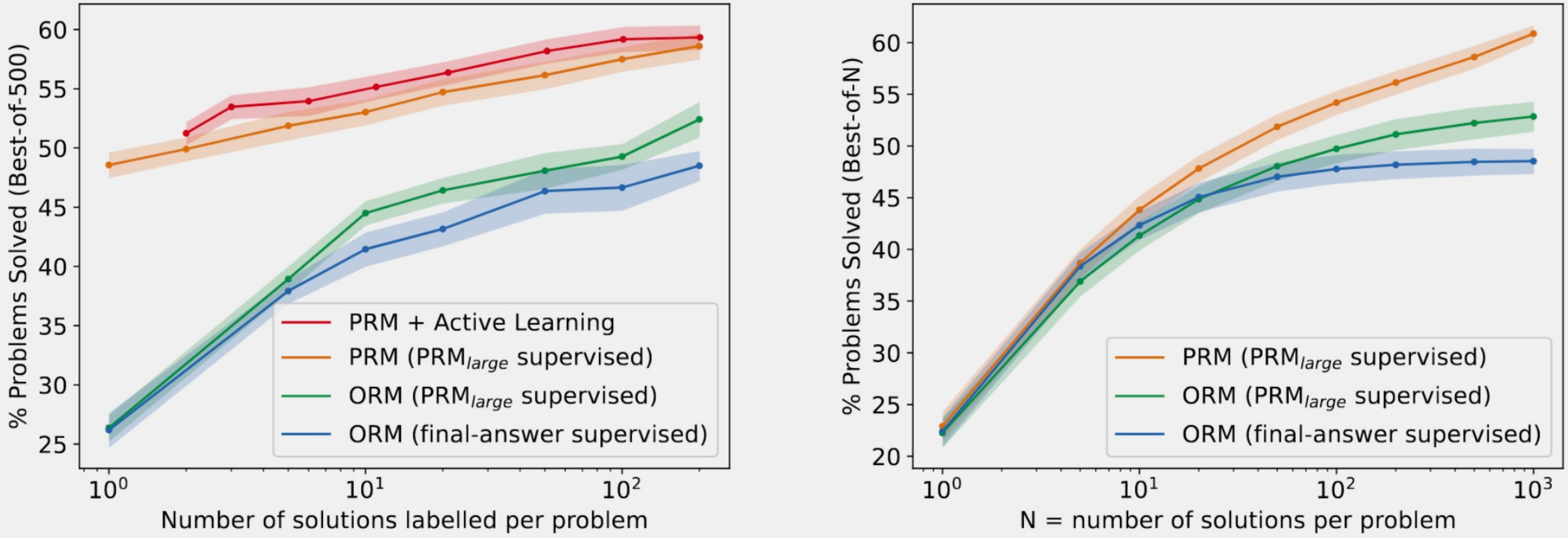
There is an incomplete picture

- PRM outperforms the ORM at large-scale
- To better compare PRM and ORM, confounding factors must be isolated.
  - the training sets for the ORM and the PRM are not directly comparable
  - the final-answer grading will provide positive labels to spurious solutions

Synthetic Supervision

- We first sample between 1 and 200 solutions per problem from a small-scale generator.

- For each dataset, we provide three forms of supervision:
  - process supervision from PRMlarge
  - outcome supervision from PRMlarge
  - outcome supervision from final-answer checking.



(a) Four series of reward models trained using different data collection strategies, compared across training sets of varying sizes.

(b) Three reward models trained on 200 samples/problem using different forms of supervision, compared across many test-time compute budgets.

H. Lightman et al., Let's Verify Step by Step, 2023

Out-of-Distribution Generalization

- 224 STEM questions, pulled from AP Physics, AP Calculus, AP Chemistry, AMC10, and AMC12 exams released after the pre-training

	ORM	PRM	Majority Vote	# Problems
AP Calculus	68.9%	**86.7%**	80.0%	45
AP Chemistry	68.9%	**80.0%**	71.7%	60
AP Physics	77.8%	**86.7%**	82.2%	45
AMC10/12	49.1%	**53.2%**	32.8%	84
Aggregate	63.8%	**72.9%**	61.3%	234

H. Lightman et al., Let's Verify Step by Step, 2023

Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations

Peiyi Wang[1] **Lei Li**[3] **Zhihong Shao**[4] **Runxin Xu**[2] **Damai Dai**[1] **Yifei Li**[5]
Deli Chen[2] **Yu Wu**[2] **Zhifang Sui**[1]

[1]State Key Laboratory of Multimedia Information Processing, School of Computer Science, Peking University.
[2]DeepSeek-AI [3]The University of Hong Kong
[4]Tsinghua University [5]The Ohio State University
{wangpeiyi9979, nlp.lilei}@gmail.com
li.14042@osu.edu szf@pku.edu.cn

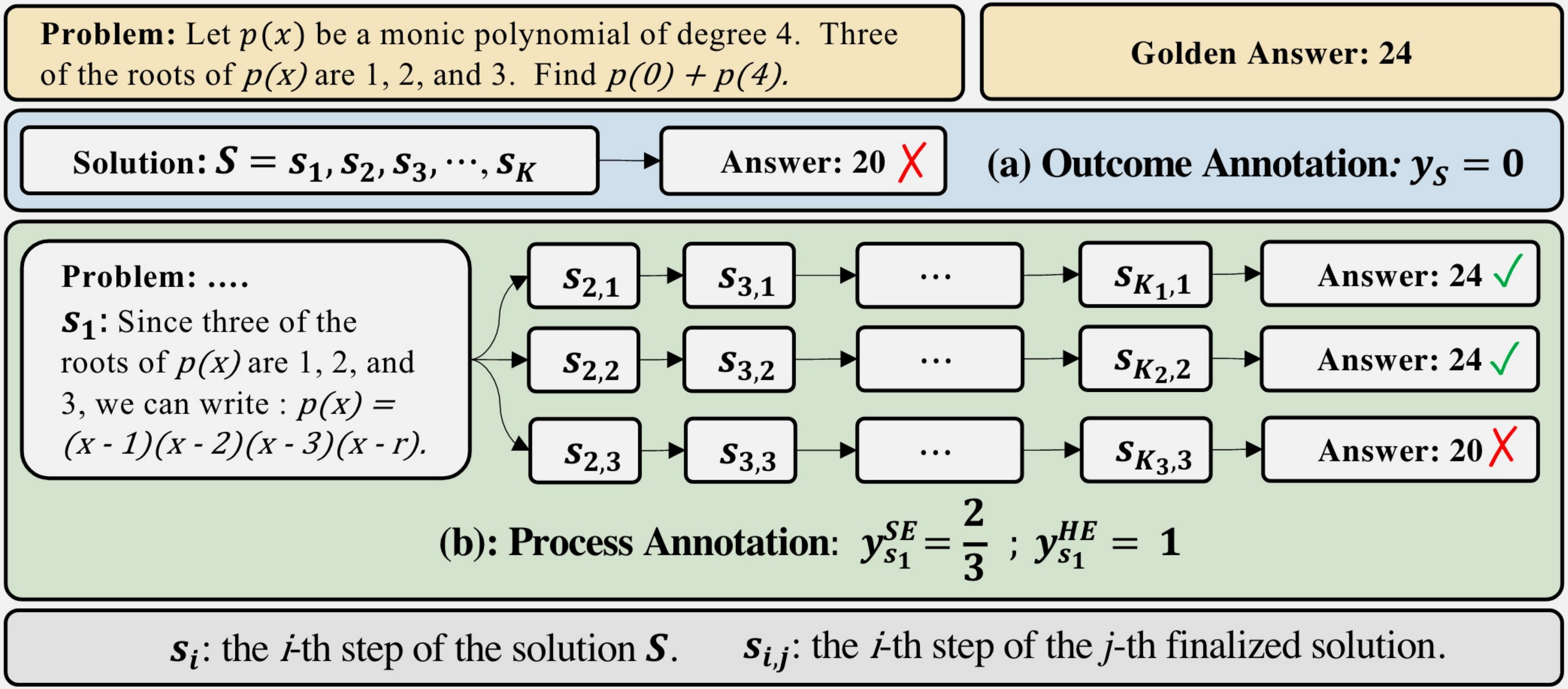
Math-Sheferd

- (Uesato et al., 2022) and (Lightman et al., 2023) utilize human annotators to provide process supervision annotations

- Human feedback is inherently costly and limited in scope

- proposes an automatic process annotation framework
  - Consequently, it creates a more larger dataset

P. Wang et al., Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations, 2024



P. Wang et al., Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations, 2024

Process supervision definition

- A step that has the potential to deduce a well-founded result can be considered a good reasoning step.
- This definition also introduces some degree of noise (like ORM)

Completion

- Completer: finalize N subsequent reasoning processes from this step $\left\{\left(s_{i+1,j}, \cdots, s_{K_j,j}, a_j\right)\right\}_{j=1}^{N}$
  - a_j: the decoded answer
  - K_j: total number of steps for the j-th finalized solution

- Estimate the potential of this step based on the correctness of all decoded answers $A = \left\{a_j\right\}_{j=1}^{N}$.

P. Wang et al., Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations, 2024

Estimation

- Estimate the quality y_{s_i} for the step s_i,

- Hard estimation (HE) supposes that a reasoning step is good as long as it can reach the correct answer a^*:

$$y_{s_i}^{HE} = \begin{cases} 1 & \exists a_j \in A, a_j = a^* \\ 0 & \text{otherwise} \end{cases}$$

- Soft estimation (SE) assumes the quality of a step as the frequency with which it reaches the correct answer

$$y_{s_i}^{SE} = \frac{\sum_{j=1}^{N} \mathbb{I}(a_j = a^*)}{N}$$

P. Wang et al., Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations, 2024

Dataset & Training

- The generator (and completer) is trained for 3 epochs on MetaMATH

- To construct the training dataset of ORM and PRM
  - 7B and 13B models are trained for a single epoch on GSM8K and MATH training sets.
  - 15 solutions per problem are sampled from each model for the training set.
  - LLemma-7B is used as the completer with the decoded number N=8.
  - 170k solutions for GSM8K and 270k solutions for MATH are obtained.

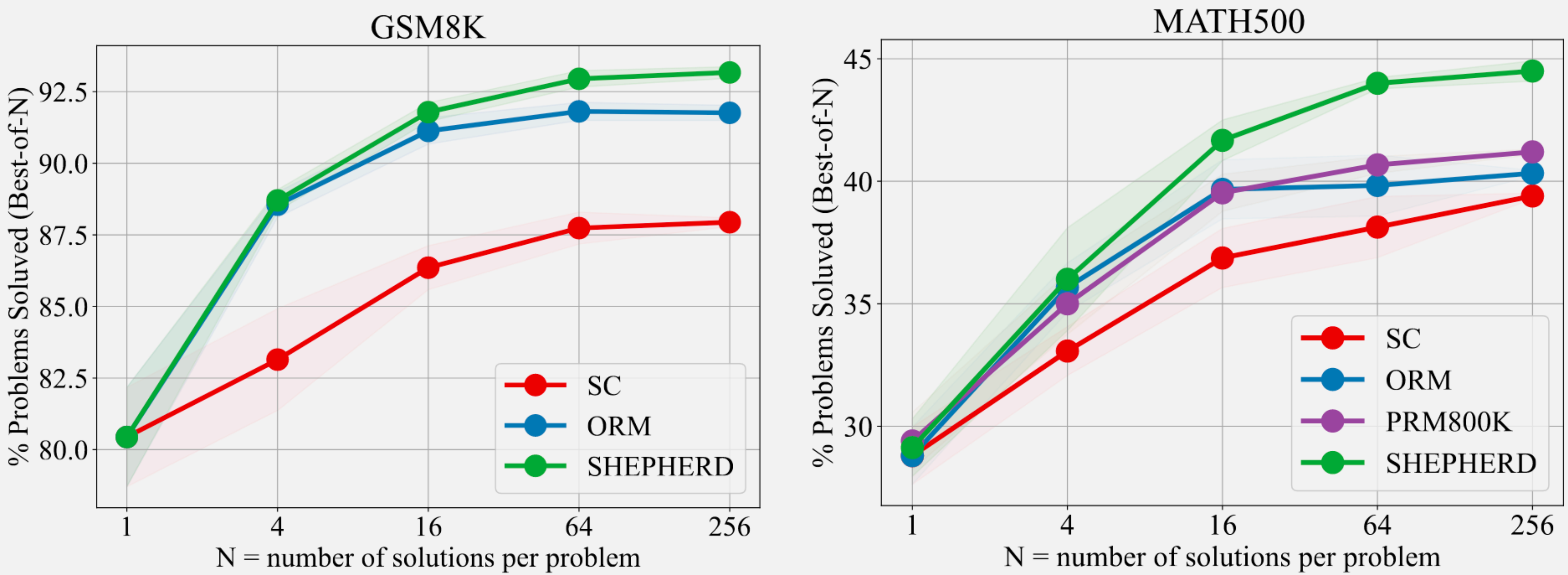
- LLaMA2-70B and LLemma34B are chosen as the base models to train verifiers for GSM8K and MATH, respectively

P. Wang et al., Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations, 2024

Verifier

- PRMs is trained by
  - estimates of per-step correctness are obtained from running Monte Carlo rollouts from each step in the solution.
  - PRM's per-step predictions therefore correspond to value estimates of reward-to-go for the base model's sampling policy, similar to recent work
- The minimum score across all steps is adopted to represent the final score of a solution.

P. Wang et al., Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations, 2024



P. Wang et al., Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations, 2024

Models	Verifiers	GSM8K	MATH500
LLaMA2-70B: MetaMATH	Self-Consistency	88.0	39.4
	ORM	91.8	40.4
	Self-Consistency + ORM	92.0	42.0
	MATH-SHEPHERD (Ours)	93.2	44.5
	Self-Consistency + MATH-SHEPHERD (Ours)	92.4	45.2
LLemma-34B: MetaMATH	Self-Consistency	82.6	44.2
	ORM	90.0	43.7
	Self-Consistency + ORM	89.6	45.4
	MATH-SHEPHERD (Ours)	90.9	46.0
	Self-Consistency + MATH-SHEPHERD (Ours)	89.7	47.3
DeepSeek-67B: MetaMATH	Self-Consistency	88.2	45.4
	ORM	92.6	45.3
	Self-Consistency + ORM	92.4	47.0
	MATH-SHEPHERD (Ours)	**93.3**	47.0
	Self-Consistency + MATH-SHEPHERD (Ours)	92.5	**48.1**

P. Wang et al., Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations, 2024

Results of PRM vs. ORM

- PRM and ORM yield similar results on GSM8K, whereas PRM significantly outperforms ORM on the MATH dataset.
  - GSM8K dataset necessitates fewer steps for problem-solving.

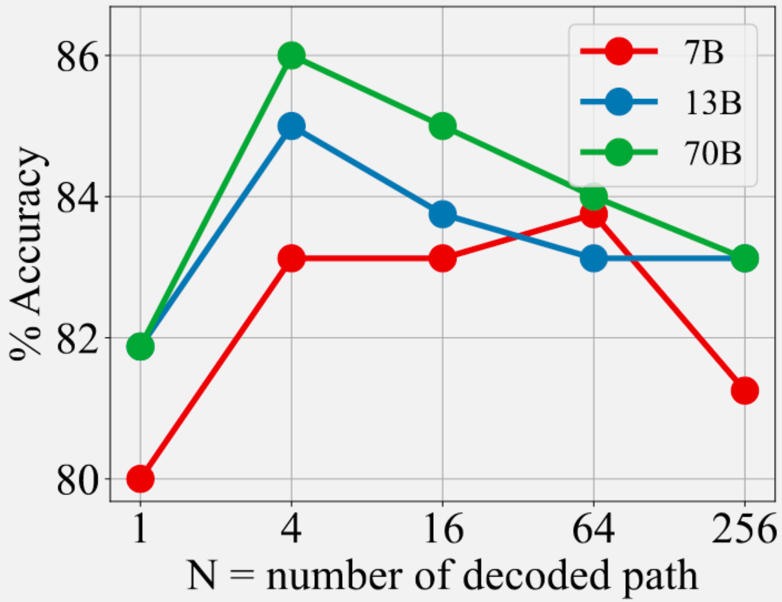
- In GSM8K, when combined with self-consistency, there's a drop in performance, whereas in MATH, performance improves.

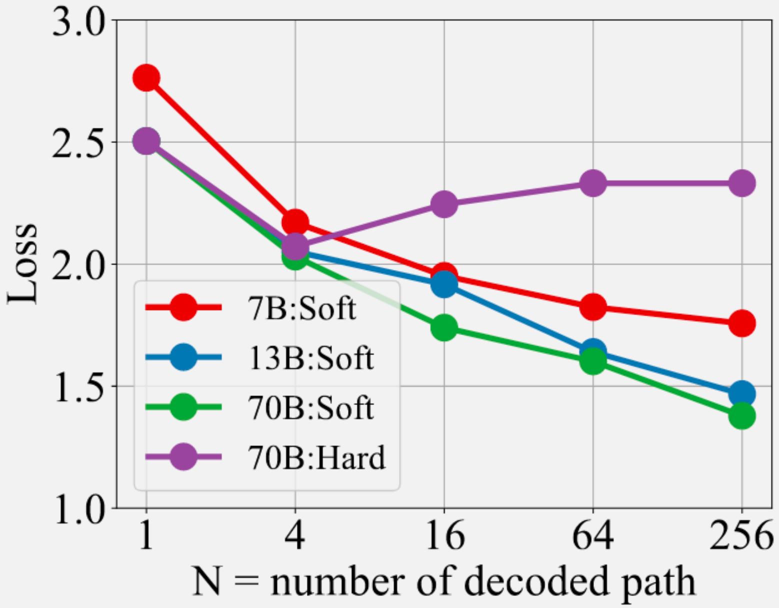
P. Wang et al., Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations, 2024

Quality of Process Annotation on GSM8K

- Quality assessment of different completors
- manually annotates 160 steps sampled from the training set of GSM8K
- further increases in N may lead to false positives

Accuracy of HE

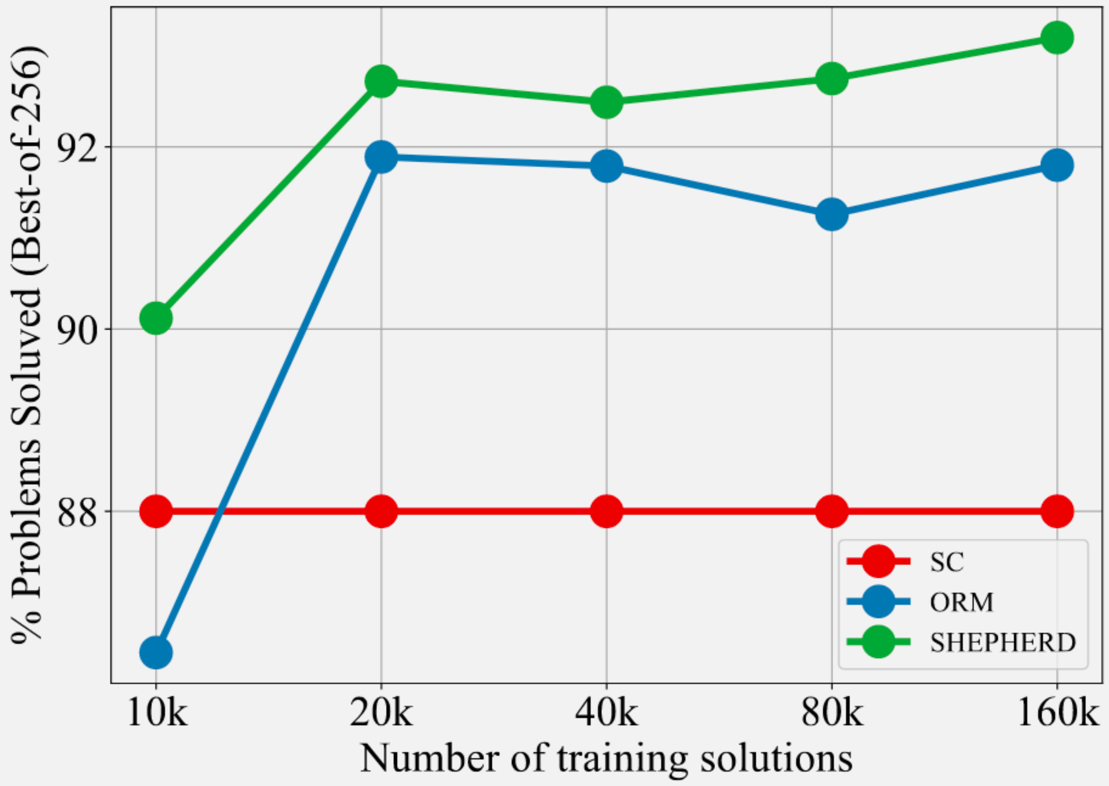




cross-entropy loss of SE/HE compared to the human-annotation

P. Wang et al., Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations, 2024

Quantities of Training Data



P. Wang et al., Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations, 2024

Google DeepMind

2024-8-7

Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters

Charlie Snell[♦, 1], **Jaehoon Lee**[2], **Kelvin Xu**[♣, 2] **and Aviral Kumar**[♣, 2]

[♣]Equal advising, [1]UC Berkeley, [2]Google DeepMind, [♦]Work done during an internship at Google DeepMind

Test-Time Compute-Optimal Scaling

- Allocate computational resources efficiently during inference
  - dynamically allocates computation based on an estimate of the question's complexity
  - Strategically choosing the test-time scaling approach

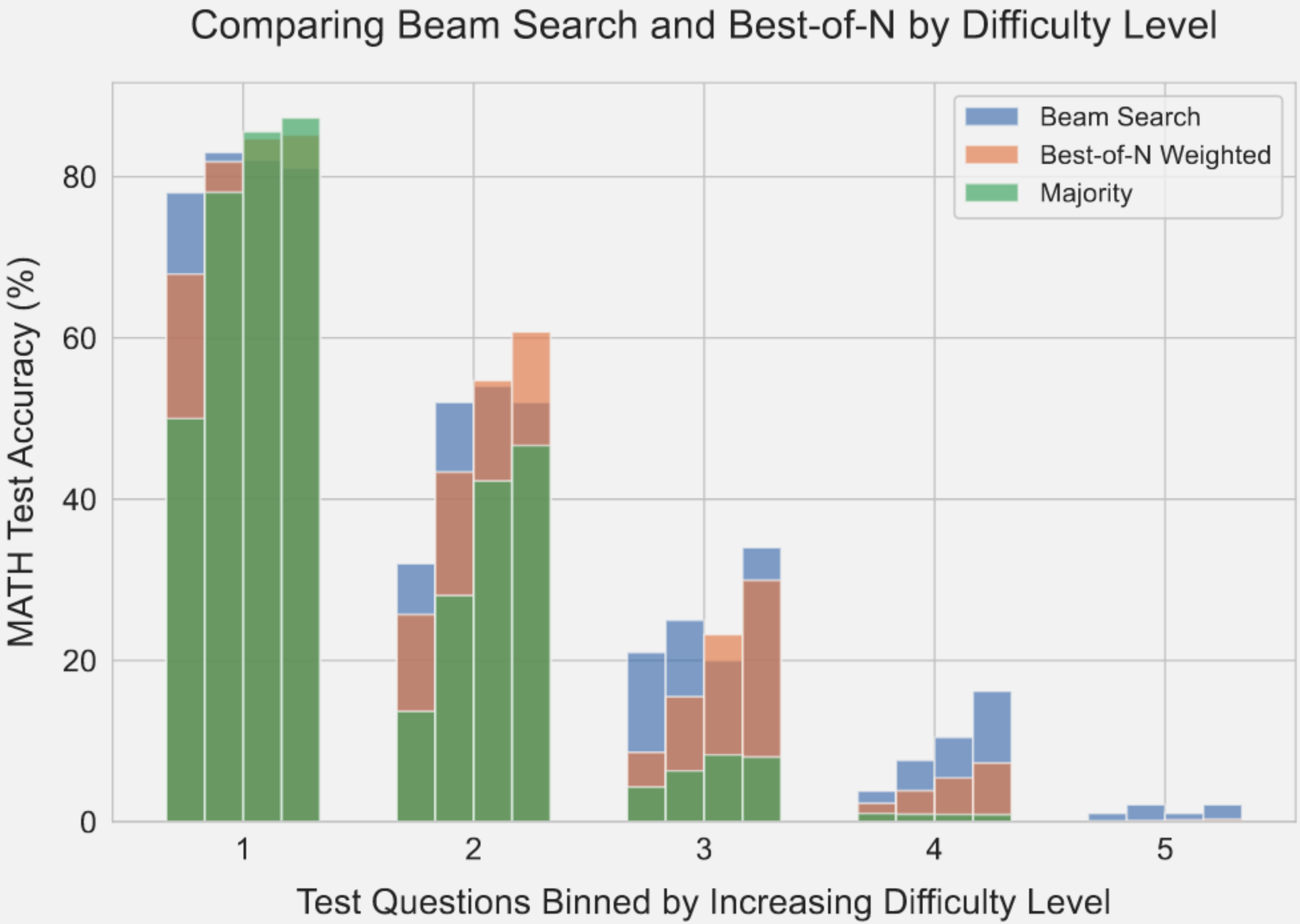
C. Snell et al., "Scaling llm test-time compute optimally can be more effective than scaling model parameters", 2024.

Categorizing Problems

- Prompts are categorized into five difficulty levels

- Difficulty of a problem: is defined as (a function of a given base LLM) according to 2048 samples
  - Oracle difficulty: pass@1 rates are binned into five quantiles (needs ground truth)
  - Model-predicted difficulty: the same binning over the the averaged final answer score from a learned verifier

C. Snell et al., “Scaling llm test-time compute optimally can be more effective than scaling model parameters”, 2024.



C. Snell et al., “Scaling llm test-time compute optimally can be more effective than scaling model parameters”, 2024.

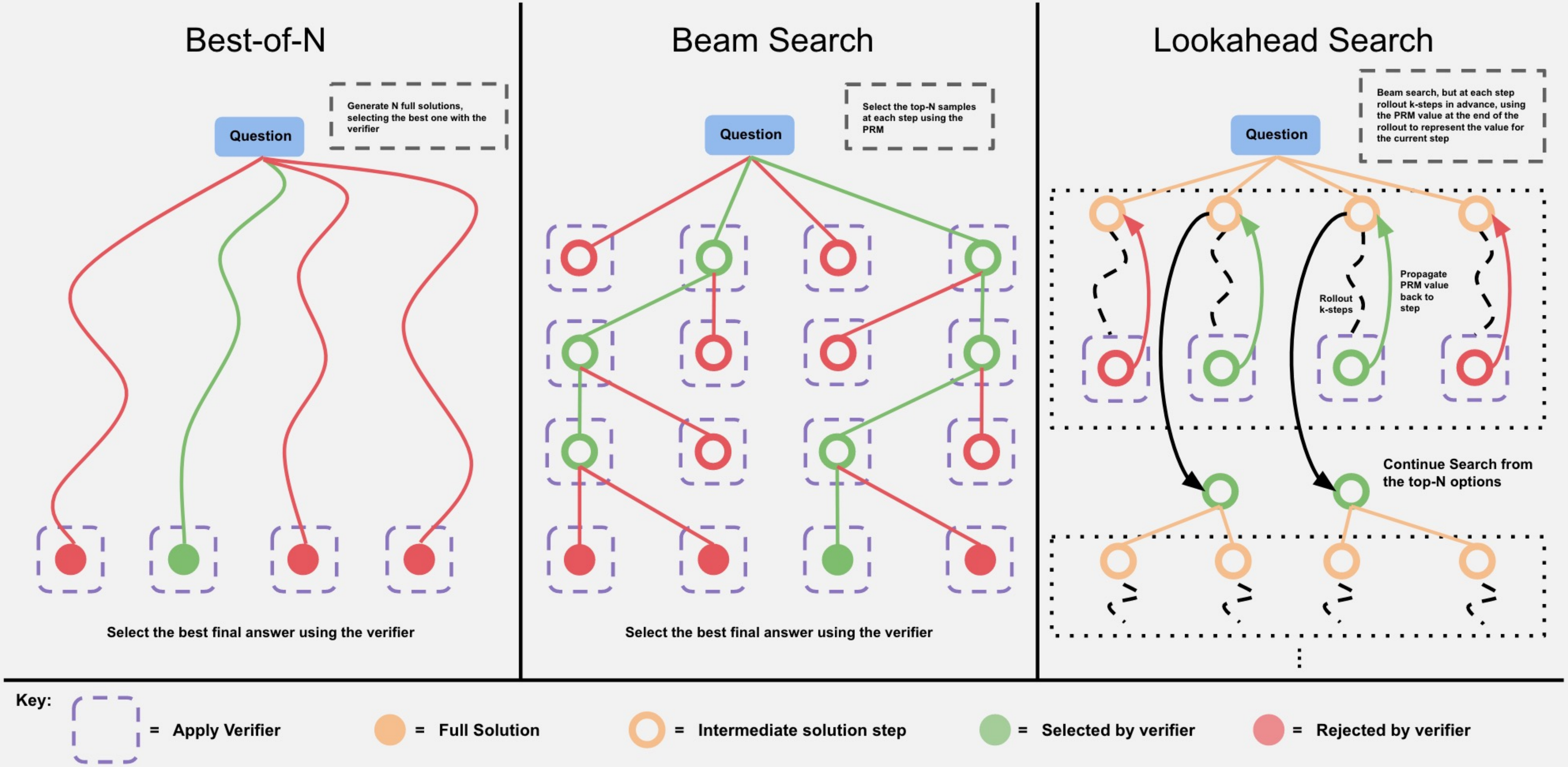
Beam Search vs. BoN

- Beam search (effective for harder questions) outperforms best-of-N sampling at low compute budgets

- Best-of-N scales better for easier tasks with high compute budget

- efficacy of any given verifier search method depends critically on both the **compute budget** and the **question at hand**

C. Snell et al., “Scaling llm test-time compute optimally can be more effective than scaling model parameters”, 2024.

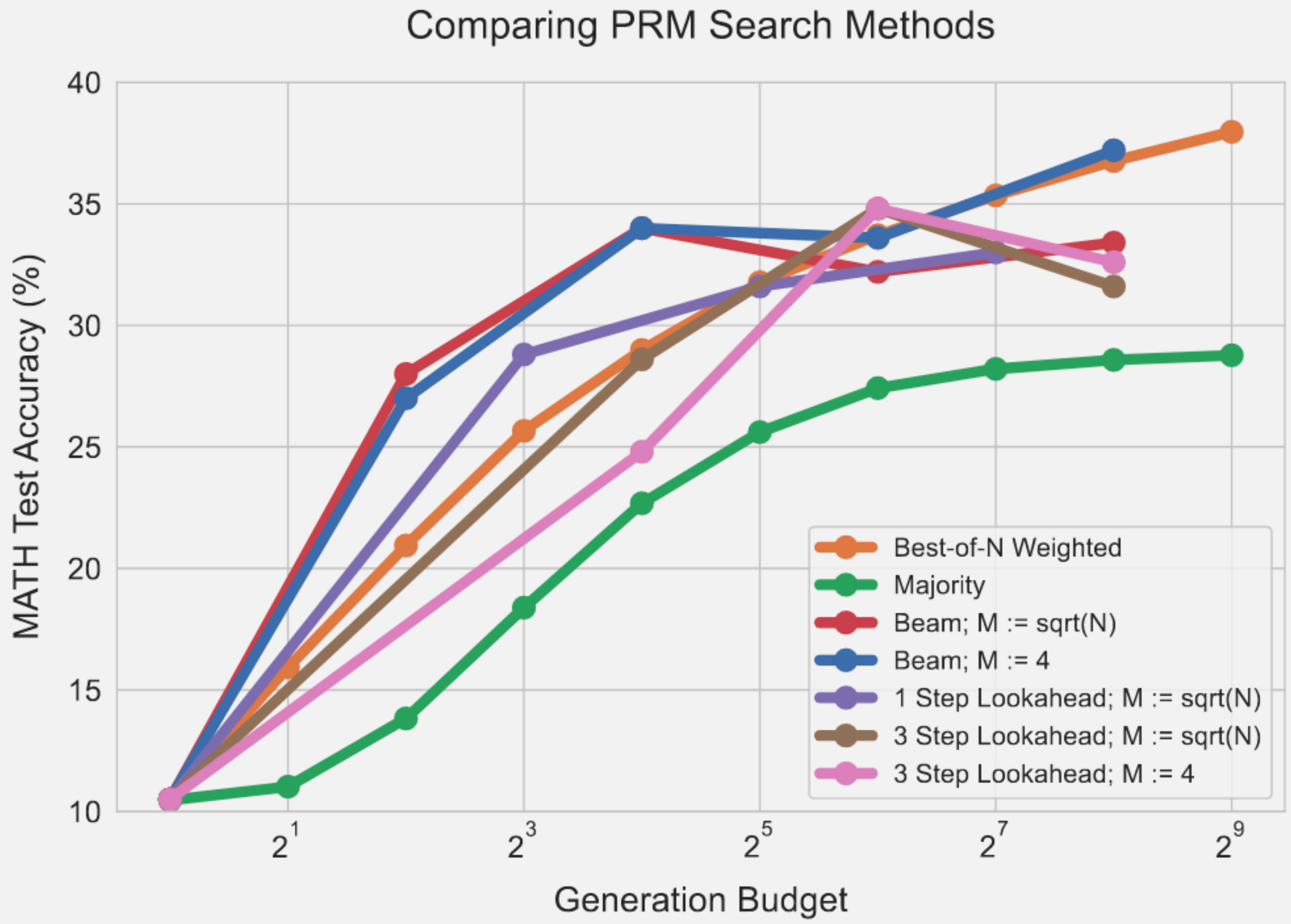


C. Snell et al., "Scaling llm test-time compute optimally can be more effective than scaling model parameters", 2024.

Look-ahead Search

- It rollouts to improve the accuracy of the PRM's value estimation in each step
  - rather than using the PRM score at the current step to select the top candidates, lookahead search performs a simulation
  - at each step of the search, we sample k additional steps ahead.
    - Therefore, we define the cost of lookahead-search to be $N \times (k + 1)$ samples.

C. Snell et al., "Scaling llm test-time compute optimally can be more effective than scaling model parameters", 2024.



PRM: Math-Sheperd

C. Snell et al., “Scaling llm test-time compute optimally can be more effective than scaling model parameters”, 2024.

The most effective way to utilize TTC

- chooses a test-time strategy θ of maximal benefits on a given prompt q

$$\theta^*_{q,a^*(q)}(N) = \operatorname{argmax}_\theta \left(\mathbb{E}_{y\sim\mathrm{Target}(\theta,N,q)}\left[\mathbb{1}_{y=y^*(q)}\right]\right),$$

  - Target(θ, N, q): distribution over output tokens for a given prompt q, using strategy θ, and a compute budget of N
  - $y^*(q)$ denotes the ground-truth correct response for q
  - $\theta^*_{q,a^*(q)}(N)$: test-time compute-optimal strategy for the problem q and compute budget N

C. Snell et al., "Scaling llm test-time compute optimally can be more effective than scaling model parameters", 2024.

Question Difficulty for Compute-Optimal Scaling

- $\theta^*_{q,a^*(q)}(N)$ is defined as a function of the difficulty of this prompt.
  - $\theta^*_{q,a^*(q)}(N)$ is estimated on a validation set for a given test-time compute budget.
    - the best performing test-time compute strategy for each difficulty bin is selected independently.
  - Question difficulty acts as a sufficient statistic of a question when designing the compute-optimal strategy.

- We then apply these compute-optimal strategies on the test-set.

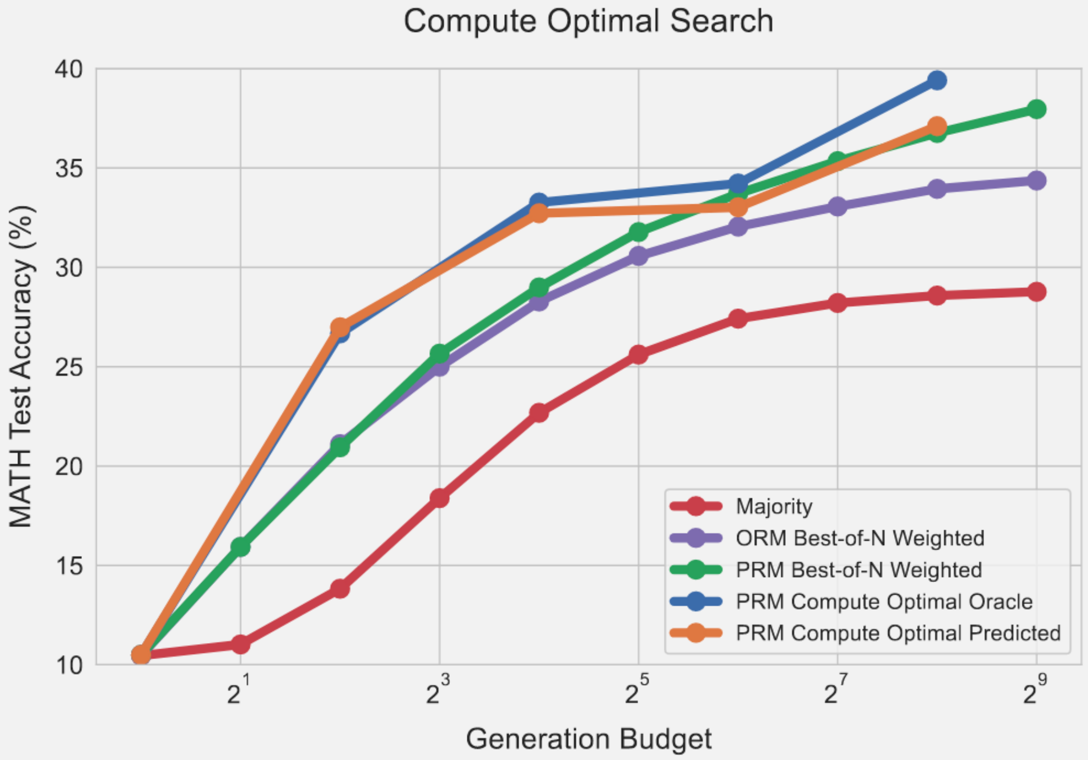
C. Snell et al., "Scaling llm test-time compute optimally can be more effective than scaling model parameters", 2024.

Compute-optimal Scaling Strategy

- It first assesses the problem difficulty
  - via a model-predicted notion of difficulty
- then utilizes the right scaling strategy to solve this problem

Compute-optimal scaling trend

- The best performing search strategy at each difficulty level



compute-optimal scaling can nearly outperform best-of-N using up to 4x less test-time compute (e.g. 16 verses 64 generations)

in the higher budget regime, some of these benefits diminish with the use of predicted difficulty

C. Snell et al., "Scaling llm test-time compute optimally can be more effective than scaling model parameters", 2024.

Training and Using Revision Models

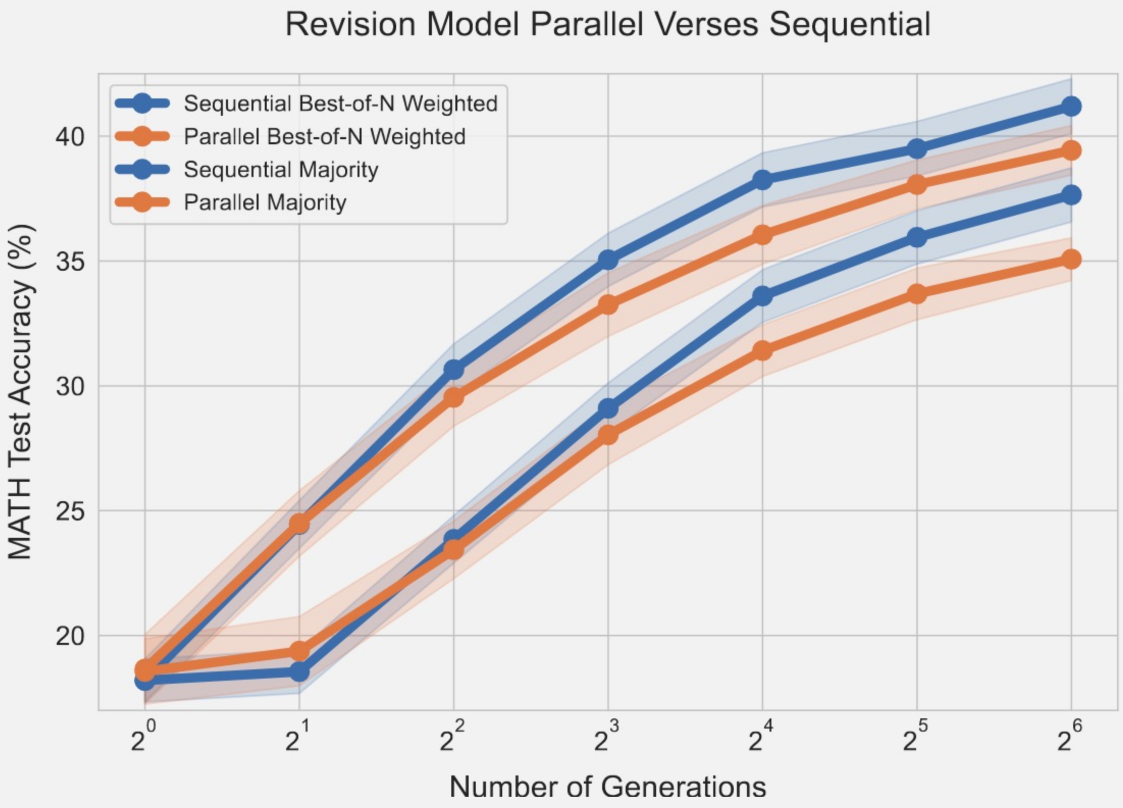
- Train and use models that refine their own proposal distribution

- For finetuning, we need trajectories consisting of a sequence of incorrect answers followed by a correct answer, that we can then run SFT on.
  - 64 responses in parallel at a higher temperature are sampled
  - multi-turn rollouts are constructed from these independent samples
    - Up to four incorrect answers in context (a uniform distribution over categories 0 to 4)
    - edit distance metric to prioritize selecting incorrect answers which are correlated with the final correct answer

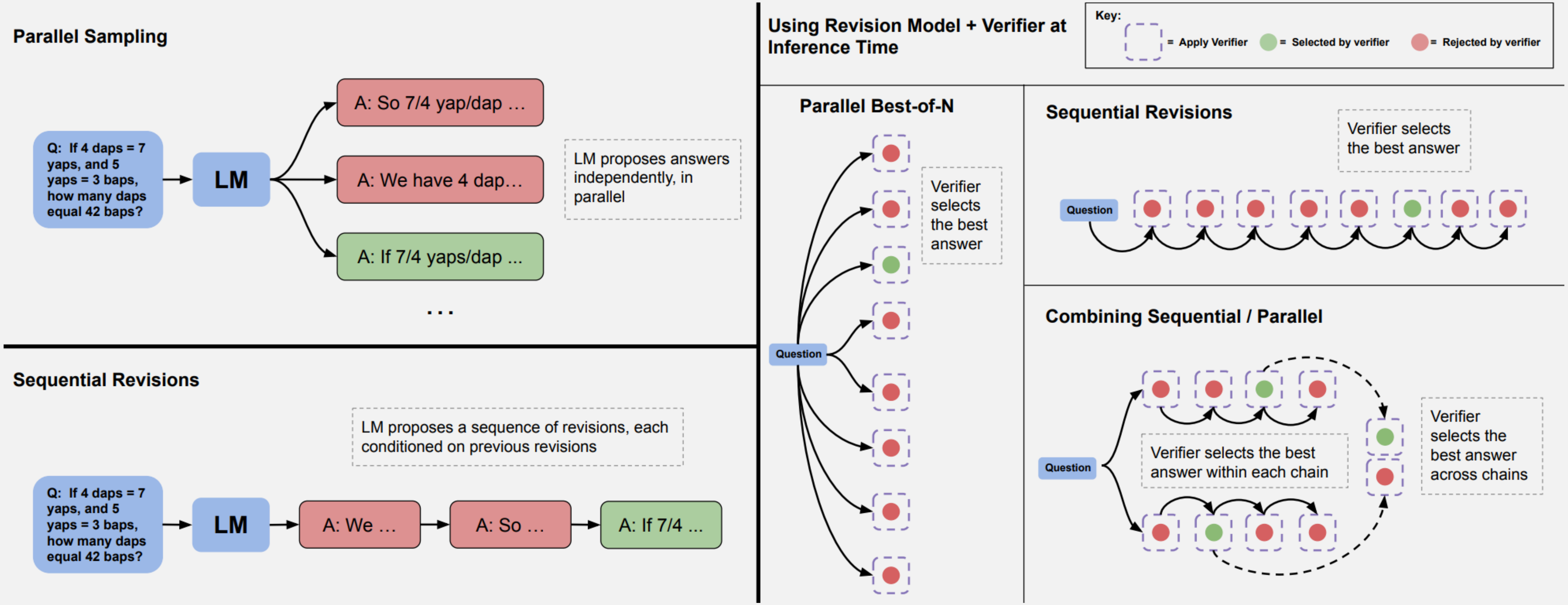
Using revisions at inference-time

- Given a finetuned revision model, sample a sequence of revisions from the model at test-time.
- Sample longer chains by truncating the context to the most recent four revised responses.
- It may incidentally turn the correct answer into an incorrect answer in the next revision step.
  - A verifier to select the most correct answer from the sequence of revisions is required

Sequential vs. Parallel



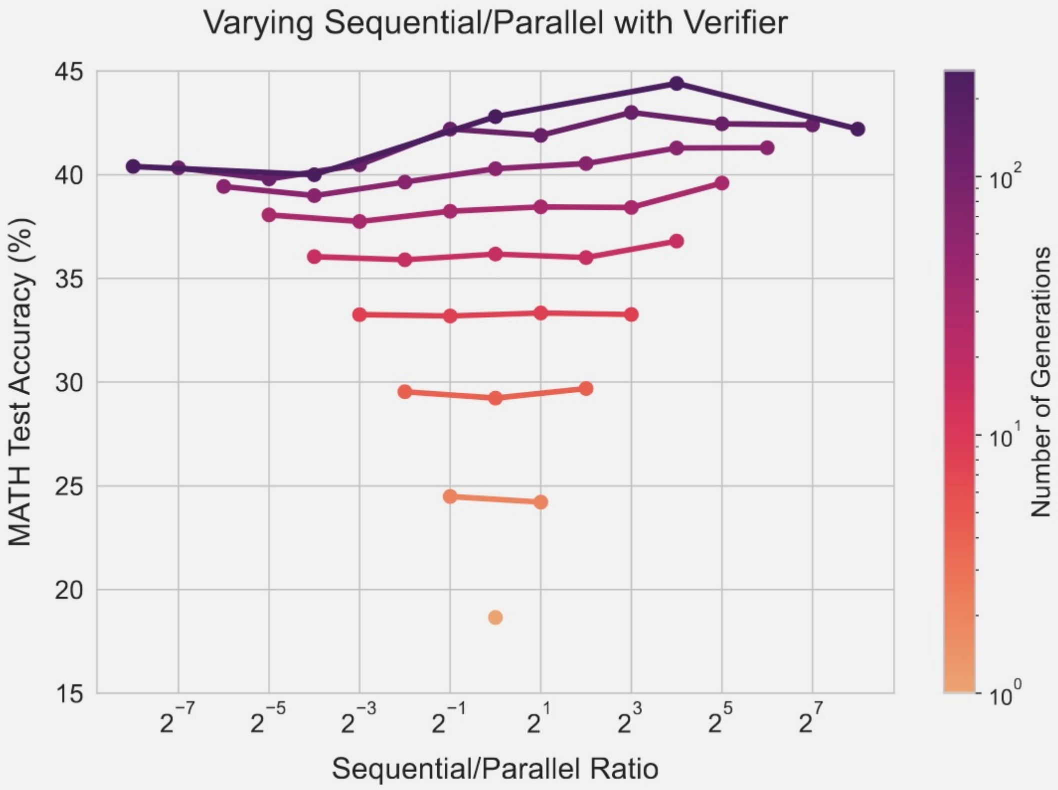
C. Snell et al., Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters, 2024

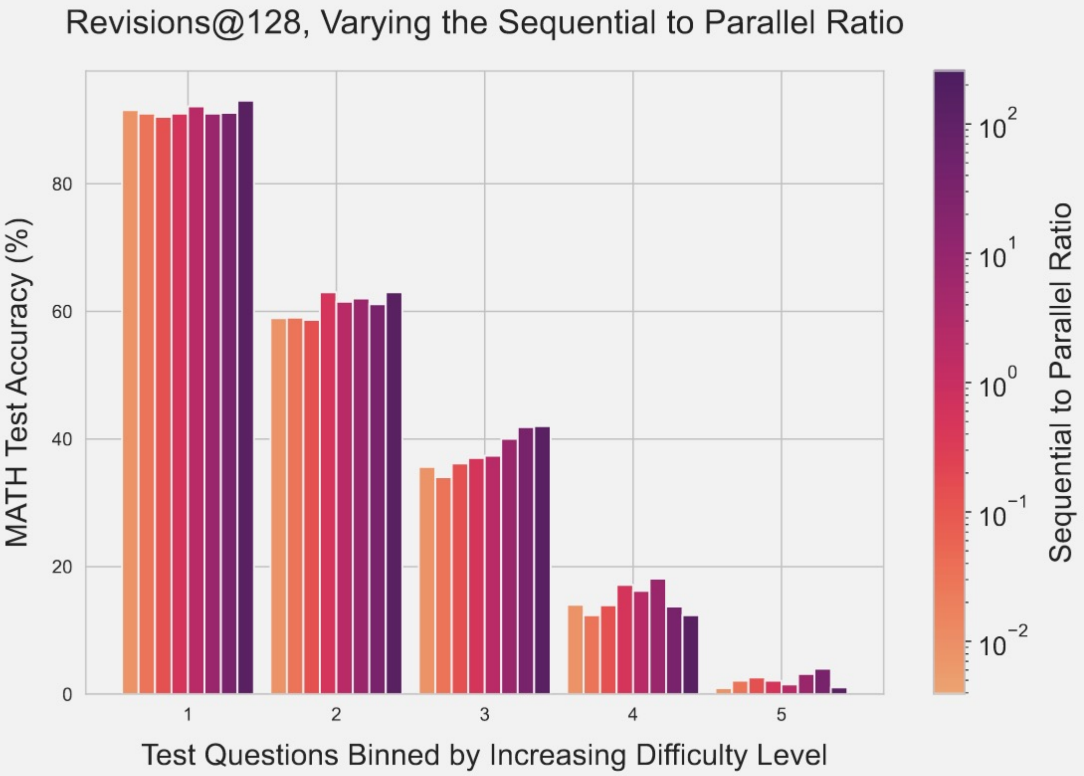


C. Snell et al., Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters, 2024

Varying Sequential/Parallel Ratio

- Easy questions benefit more from sequential revisions and difficult questions benefit from a balance between sequential and parallel computation

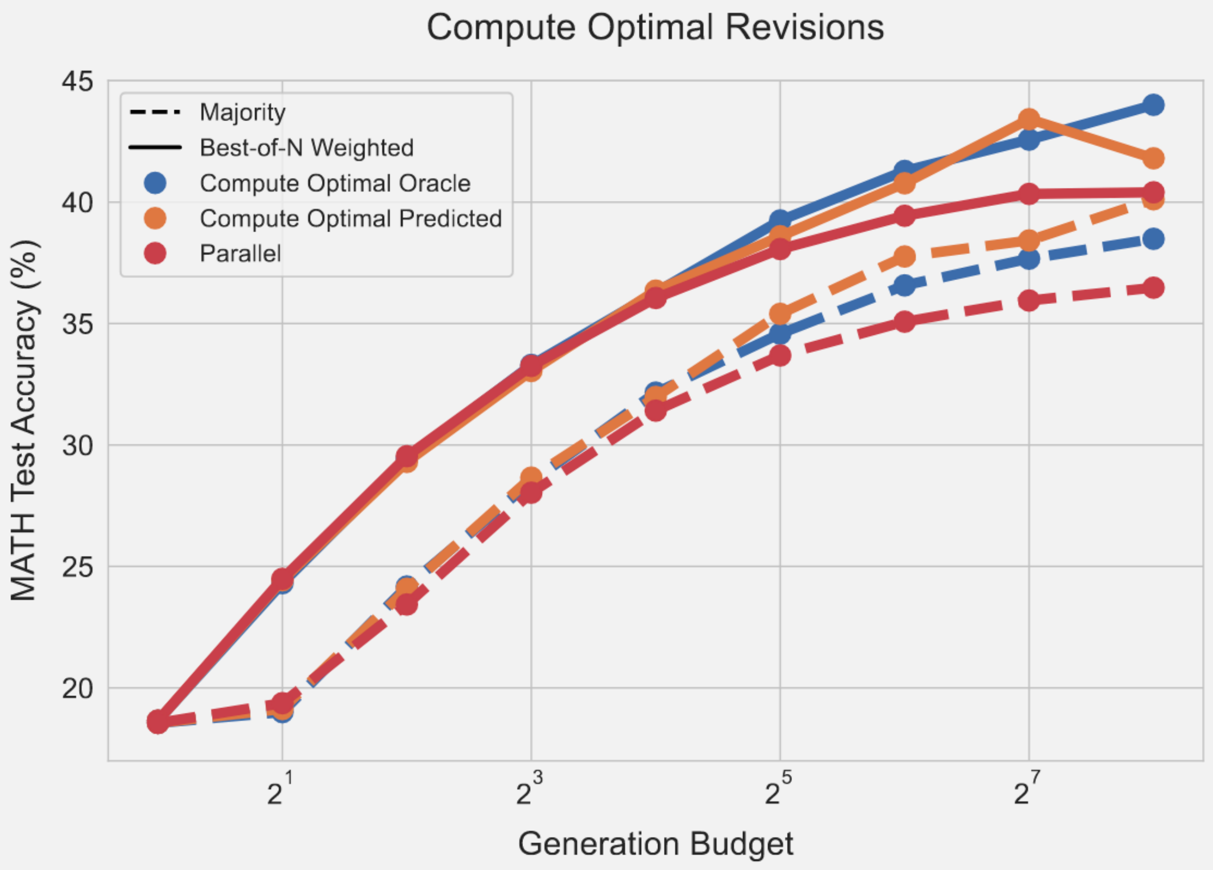




C. Snell et al., "Scaling llm test-time compute optimally can be more effective than scaling model parameters", 2024.

Compute-Optimal Revision

- selects the ideal ratio of sequential to parallel compute per difficulty bin.
- compute-optimal scaling can outperform best-of-N using up to 4x less test-time compute (e.g. 64 samples verses 256)



C. Snell et al., “Scaling llm test-time compute optimally can be more effective than scaling model parameters”, 2024.

Compute-Optimal Revision

- Strategy adapts compute allocation:
  - easier prompts undergo sequential refinement
  - harder prompts trigger parallel sampling or beam search, which explores multiple response variations to increase the likelihood of finding a correct solution.
- This dual approach balances exploration (for challenging inputs) and refinement (for near correct responses)