Verification Methodologies for Reliable Mathematical Reasoning

A Comparative Study of Outcome, Process, and Generative Verification

Siddharth Magesh (smagesh@usc.edu)

Problem Setting, Motivation, and Research Contributions

Generative language models can solve mathematical problems that require multiple reasoning steps to reach the correct answer. However, a single incorrect intermediate step can alter the model's reasoning trajectory and lead to an incorrect solution. To reduce this problem, verification methods generate N candidate solutions for a given problem and use a verifier to select the best candidate according to a scoring strategy, typically the highest verification score. Cobbe et al. [1] introduce GSM8K and demonstrate the effectiveness of **outcome supervised verification**. Lightman et al. [2] use **steplevel process supervision** to examine whether feedback on individual reasoning steps improves solution selection on harder mathematical reasoning tasks. Zhang et al. [3] reformulate verification as **text generation**, using chain-of-thought reasoning to explain and evaluate candidate solutions while also sampling alternative verification rationales. Together, these works explore different verification strategies, although they do not share a single common benchmark that allows a direct quantitative comparison across all three methods.

Verification Mechanisms: Approaches and Trade-offs

All three methods rely on the same underlying **Best-of-N** selection mechanism. Given a problem Q, the generator samples a set of candidate solutions and the verifier assigns each candidate a score $R(Q, S_i)$ representing it's correctness. The selected solution is $\mathbf{S}^* = \arg\max_{\mathbf{S_i} \in \{\mathbf{S_1}, \mathbf{S_2}, \ldots, \mathbf{S_N}\}} \mathbf{R}(\mathbf{Q}, \mathbf{S_i})$. Increasing N does expands the search space but it does not guarantee finding optimal solution. If the generator fails to produce any correct solution, the verifier has no correct candidate to select.

Outcome Supervision. Cobbe et al. [1] fine-tune GPT-3 generators for **2 epochs**. For each training problem Q_i, the generator samples $N = 100$ candidate solutions at temperature $\tau = 0.7$ and each candidate is labeled by checking its final answer $y \in \{0, 1\}$. The verifier is initialized from the corresponding fine-tuned generator and trained for **1 epoch**.

$$\text{Loss} = \underbrace{-\sum_{t=1}^{T} \log P_\theta(s_t \mid Q, s_{<t})}_{\mathcal{L}_{\text{LM}}} + \underbrace{\frac{1}{T} \sum_{t=1}^{T} \left(V_\theta(Q, s_{\leq t}) - y\right)^2}_{\mathcal{L}_{\text{ver}}}$$

Training uses an equal mixture of original GSM8K solutions for $\mathcal{L}_{LM}$ and generated outcome labeled candidates for $\mathcal{L}_{ver}$. Question-token losses are masked, and the verifier weights θ are updated while the generator remains fixed. Since every token receives the same final outcome label y, this is **not step-level supervision**.

Process Supervision. Lightman et al. [2] train a Process Reward Model (PRM) on **PRM800K**, which contains approximately **800,000 human step-level labels**. The large-scale PRM is initialized from a GPT-4-based model with additional MathMix training. For each reasoning step s_t, the supervision label is $z_t \in \{+, 0, -\}$.

$$\theta^* = \arg\min_\theta \left[-\sum_{i=1}^{M} \sum_{t=1}^{T_i'} \log P_\theta(z_{i,t} \mid x_i, s_{i,\leq t}) \right], \qquad R(x, S) = \prod_{t=1}^{m} \left[P_\theta(+ \mid x, s_{\leq t}) + P_\theta(0 \mid x, s_{\leq t}) \right].$$

Here, M is the number of training solutions and T_i' is the number of labeled steps in solution i. The PRM receives local supervision on where reasoning fails, which improves credit assignment but requires expensive human labeling. In phase 2, annotation stops at the first incorrect step. To reduce labeling cost, active learning prioritizes **convincing wrong solutions**.

Generative Verification. GenRM trains a language model to predict `Yes/No` and scores a candidate using $P_\theta(\texttt{Yes} \mid x, S)$. GenRM-CoT first generates a verification rationale v_{CoT} before producing the final judgment. The same parameters θ are trained jointly for verification and correct-solution generation, allowing the two tasks to share representations.

$$\mathcal{L}_{\text{GenRM}} = \mathcal{L}_{\text{SFT}}(\theta, \mathcal{D}_{\text{verify}}) + \lambda \mathcal{L}_{\text{SFT}}(\theta, \mathcal{D}_{\text{correct}}), \qquad r_{\text{MajV@K}}(x, S) = \frac{1}{K} \sum_{i=1}^{K} P_\theta \left(\texttt{Yes} \mid x, S, I_{\text{CoT}}, v_{\text{CoT}}^{(i)}, I \right).$$

Here, λ weights solution-generation training, while I_{CoT} and I denote the verification-reasoning and Yes/No prompts. Both objectives use next-token cross-entropy. For GSM8K, **Gemini 1.0 Pro** generates candidate solutions and reference-guided synthetic verification rationales, while **Gemma models** are fine-tuned as verifiers. The reference solution and is not provided during verifier fine-tuning or inference. By default, $K = 32$ verification rationales are sampled and their `Yes` probabilities are averaged rather than using hard binary voting.

Experimental Evidence: Benchmark Results and Interpretation

The common evaluation metric across all three papers is the **solve rate**, or Best-of-N accuracy after verification. This differs from pass@N: it is not enough for a correct solution to exist among the generated candidates, since the verifier must also successfully rank and select that candidate.

Cobbe et al. On GSM8K, a **6B verification model selecting from 100 samples** slightly outperforms the **175B fine-tuned baseline**, which uses a single greedy solution after 20 fine-tuning epochs. This supports allocating inference compute to candidate generation and verification rather than relying only on additional fine-tuning, although it does not establish superiority under equal compute. The paper also shows that **token-level verification** improves selection performance, while retaining the auxiliary language modeling loss further strengthens the verifier.

Lightman et al. On the **500 held-out MATH problems**, Best-of-1860 achieves **78.2% with the PRM**, compared with **72.4% for the ORM** and **69.6% for majority voting**. The authors use 4,500 problems from the original MATH test split for training and reserve the remaining 500 for evaluation. Because the large PRM and ORM use different training data. To create common baseline, they compares smaller models trained on identical solutions with different supervision. Step-level labels outperform both PRM-generated outcome labels and final-answer labels, supporting the benefit of process supervision.

Zhang et al. With **Gemini 1.0 Pro** generating candidates and a **Gemma 2 9B** verifier, GSM8K performance increases from the generator's **73.0%** baseline to **93.4% under Best-of-16**. On MMLU college mathematics, a matched Best-of-32 comparison gives **56.1% for GenRM-CoT** versus **53.0% for discriminative verification**. Direct GenRM is only comparable to, or slightly better than, the discriminative baseline, while larger gains come from verification rationales and repeated sampling. This suggests that the main improvements are not explained simply by replacing a scalar score with a `Yes/No` token.

Comparative Analysis and Critical Assessment

Together, these papers improve verification through complementary directions: Cobbe et al. study scaling and search, Lightman et al. compare supervision signals, and Zhang et al. compare verifiers on fixed candidate pools. GSM8K and PRM800K also provide reusable resources for studying reasoning failures.

Their limitations also constrain the conclusions that can be drawn. In Cobbe et al., final-answer labels can mark a solution as correct even when its intermediate reasoning is invalid, and searching over larger candidate pools can expose verifier errors rather than guarantee better selection. In Lightman et al., the controlled comparison relies on a large PRM as the teacher, so it does not measure the same effect using independent human step labels. Its product-based scoring also slightly penalizes longer solutions.

GenRM compares verifiers on the same candidate solutions, so performance differences can be attributed more directly to the verification method. However, GenRM-CoT uses additional inference compute to generate verification rationales, so its accuracy gains do not establish better performance under equal computational cost, which the paper does not directly compare. The generated rationales are also not guaranteed to be fully correct: the paper shows a case where the verifier reaches the correct final judgment while incorrectly criticizing a valid intermediate reasoning step.

My assessment is that the evidence supports improved answer selection within the tested mathematical settings. Better credit assignment and explicit verification reasoning are plausible explanations for the gains, but neither guarantees valid reasoning. Different generators, supervision sources, and sampling budgets prevent direct cross-paper accuracy comparisons. Process labels and generative rationales should therefore be viewed as complementary verification choices rather than directly comparable alternatives.

Follow-up Research: A Focused Question and Proposed Experiment

Under a fixed inference-compute budget, does GenRM-CoT achieve higher solution-selection accuracy than discriminative verification or direct GenRM? To test this, use the same benchmark, fixed generator, and equally sized verifiers, while measuring total candidate-generation and verification cost in GPU-seconds on identical hardware. Vary the number of candidate solutions and, for GenRM-CoT, the number of verification rationales within each compute budget, then compare selected-answer accuracy across budgets. This would show whether evaluating more candidates with cheaper verifiers can match GenRM-CoT, and when its additional reasoning cost provides a worthwhile accuracy gain.

References.

[1] K. Cobbe et al., *Training Verifiers to Solve Math Word Problems*, arXiv:2110.14168, 2021.

[2] H. Lightman et al., *Let's Verify Step by Step*, arXiv:2305.20050, 2023.

[3] L. Zhang et al., *Generative Verifiers: Reward Modeling as Next-Token Prediction*, ICLR, 2025.