Extended Abstract

Motivation Modern reinforcement learning pipelines for language models rely heavily on exact, rule-based verifiers to provide reward signals. However, in most real-world deployment domains, ground-truth algorithmic verifiers do not exist. This project investigates scaling test-time compute through a Generative Verifier (GV) on the Countdown mathematical reasoning task using a computationally constrained 0.5B parameter regime. We evaluate whether a Qwen 2.5 0.5B model can bootstrap its own reasoning capabilities without an oracle, hypothesizing that the "verification gap" will widen at higher values of N due to the model's limited semantic capacity.

Method Our architecture isolates verification mechanics within the 0.5B regime through three phases. First, a baseline policy generator is established using Supervised Fine-Tuning (SFT), Implicit Preference Optimization (IPO), and RLOO policy-gradient optimization. Second, reward modeling is framed as a next-token prediction task. We curate a dataset of 80,000 reasoning trajectories generated by the RLOO policy and fine-tune a fresh 0.5B model for binary classification, predicting exclusively a `[CORRECT]` or `[INCORRECT]` verdict token. Finally, during Best-of-N evaluation ($N \in \{1, 4, 8, 16, 32\}$), the GV scores candidates by extracting the log-probability of the `[CORRECT]` token directly from the final sequence position's logits.

Implementation All models were trained using Modal serverless infrastructure (NVIDIA H100 GPUs) with custom PyTorch training loops. Data generation and inference were accelerated using vLLM. The generated training dataset consisted of 17,271 correct trajectories (21.6%) and 62,729 incorrect trajectories (78.4%). The GV was trained using a learning rate of 5e-6, a batch size of 16, and 2 gradient accumulation steps. The verification inference pipeline utilized pure log-likelihood extraction from the final attention state to avoid the computational overhead of autoregressive sampling.

Results Contrary to inference-time scaling laws observed in frontier models, the 0.5B Generative Verifier failed to improve generation quality as compute scaled. The model established a 34.0% zero-shot baseline ($N = 1$). At $N = 32$, performance regressed to 32.0%, yielding a negative verification gap of -0.02. Qualitative analysis revealed three primary failure modes: arithmetic hallucination (accepting invalid math disguised in confident formatting), constraint forgetting (ignoring rules about which numbers must be used), and length bias (systematically assigning higher probabilities to more verbose traces).

Discussion The results highlight the fundamental capacity constraints of semantic verification. While generative verification provides a functional alternative to regression-based reward models, the 0.5B model lacks the computational depth to simultaneously track arithmetic state and enforce strict generation constraints. At higher values of N, the Best-of-N pipeline effectively acts as an adversarial attack against the verifier, actively surfacing "adversarial false positives" that trick the verifier's heuristics.

Conclusion By framing verification as next-token prediction, we demonstrate the mechanics of inference-time compute scaling. We conclude that sub-1B models are currently insufficient to act as robust judges for multi-step arithmetic. Future work should investigate Process-Supervised Reward Modeling (PRM) to penalize logic errors step-by-step, explore parameter capacity thresholds (e.g., 1.5B or 3B architectures), and analyze dynamic temperature scaling to protect weaker verifiers from highly hallucinated traces.

Scaling Test-Time Compute via Generative Verification in Constrained Parameter Regimes

Mohammad Rehan Ghori
Department of Computer Science
Stanford University - SCPD
rghori7@gmail.com

Abstract

Modern reinforcement learning pipelines for language models rely heavily on exact, rule-based verifiers to provide reward signals. However, in most real-world deployment domains, ground-truth algorithmic verifiers do not exist. To bridge this gap, this project investigates scaling test-time compute through a Generative Verifier (GV) on the Countdown mathematical reasoning task using a computationally constrained 0.5B parameter regime. By framing reward modeling as a next-token prediction task, we fine-tune a Qwen 2.5 0.5B model to judge reasoning trajectories generated by a peer RLOO policy. The GV was trained on a highly imbalanced dataset of 80,000 trajectories and evaluated in a Best-of-N inference setting. Results indicate that the 0.5B verifier lacks the semantic capacity to scale compute effectively, plateauing at a 34.0% Pass@1 rate and exhibiting a negative verification gap (-0.02) at $N = 32$ due to susceptibility to constraint-based reward hacking.

1 Introduction

Scaling compute during the inference phase has emerged as a powerful mechanism for unlocking reasoning capabilities in large language models. However, techniques like Best-of-N sampling require a highly reliable scoring mechanism to rank candidate solutions. While closed environments allow for rule-based oracles, real-world applications require models to act as their own judges.

Small, lightweight models frequently struggle in this capacity due to poor calibration and high susceptibility to reward hacking. This report evaluates the precise limits of self-verification within a constrained sub-1B parameter regime. Specifically, we investigate whether a Qwen 2.5 0.5B model can bootstrap its own reasoning capabilities on the Countdown arithmetic task without an oracle at deployment. We hypothesize that while the Generative Verifier will yield performance gains at low compute budgets, the "verification gap" will widen at higher values of N as the generator discovers out-of-distribution mathematical flaws that the judge lacks the capacity to penalize.

2 Related Work

This work intersects with online policy gradients, reward modeling, and inference-time scaling. Standard online RL methods like REINFORCE Leave-One-Out (RLOO) successfully reduce variance in policy gradients but traditionally map to single-generation decoding. Recent literature demonstrates that increasing compute during inference, such as parallel sampling, can rival parameter scaling during pre-training.

To solve the ranking problem without an oracle, we draw on generative verification, which casts reward modeling as a standard language modeling objective (next-token prediction) rather than scalar

regression. Our work extends this literature by strictly isolating the verification mechanics within the sub-1B parameter regime on a multi-step arithmetic dataset.

3 Method

The architecture is executed in three primary phases:

- **Baseline Alignment:** The Qwen 2.5 0.5B Base model is first warm-started using Supervised Fine-Tuning (SFT). We subsequently apply Implicit Preference Optimization (IPO) and an online RLOO policy-gradient optimization using a rule-based verifier to establish high-performing generator checkpoints.
- **Generative Verifier Training:** We curate a binary classification dataset by sampling trajectories per prompt from the Countdown training data using the RLOO policy via vLLM. This yielded 80,000 distinct reasoning trajectories. A fresh Qwen 0.5B model is then fine-tuned with a masked cross-entropy loss applied exclusively to the final verdict token (`[CORRECT]` or `[INCORRECT]`).
- **Test-Time Inference:** During evaluation, the oracle is disabled. The RLOO policy generates $N \in \{1, 4, 8, 16, 32\}$ candidate solutions. The GV scores each candidate by extracting the log-probability of the `[CORRECT]` token directly from the logits at the final sequence position. The trajectory with the highest likelihood is selected as the final answer.

4 Experimental Setup

All experiments utilize the `Qwen/Qwen2.5-0.5B` architecture and the Countdown arithmetic reasoning dataset (`asingh15/countdown_tasks_3to4`).

Data generation and inference are accelerated using vLLM on NVIDIA H100 GPUs. Training generation utilized a temperature of 1.0. The generated training dataset consisted of 17,271 correct trajectories (21.6%) and 62,729 incorrect trajectories (78.4%), providing a robust distribution of negative examples. The GV was trained using a learning rate of 5e-6, a batch size of 16, and 2 gradient accumulation steps.

Final Best-of-N generation utilizes a temperature of 0.6, top_k of 20, and top_p of 0.95. Performance is measured via the Pass@1 metric, evaluating strict algorithmic correctness and proper number utilization.

5 Results

5.1 Quantitative Evaluation

Contrary to inference-time scaling laws observed in larger frontier models, the 0.5B Generative Verifier failed to improve generation quality as compute scaled. The model's baseline zero-shot performance ($N = 1$) established a 34.0% Pass@1 rate. As N increased to 32, the performance fluctuated and ultimately regressed to 32.0%, yielding a negative verification gap of -0.02. This indicates that as the candidate pool grows, the likelihood of the generator producing an adversarial "false positive" that successfully tricks the verifier outweighs the probability of finding a genuinely correct mathematical trace.

Table 1: Pass@N Scaling Performance on Countdown Test Set

Method	$N = 1$	$N = 4$	$N = 8$	$N = 16$	$N = 32$
RLOO + Generative Verifier	0.3400	0.3400	0.2800	0.3400	0.3200

5.2 Qualitative Analysis

An analysis of the test-time trajectories reveals that the Generative Verifier frequently falls victim to three primary failure modes: arithmetic hallucination, constraint forgetting, and length bias.

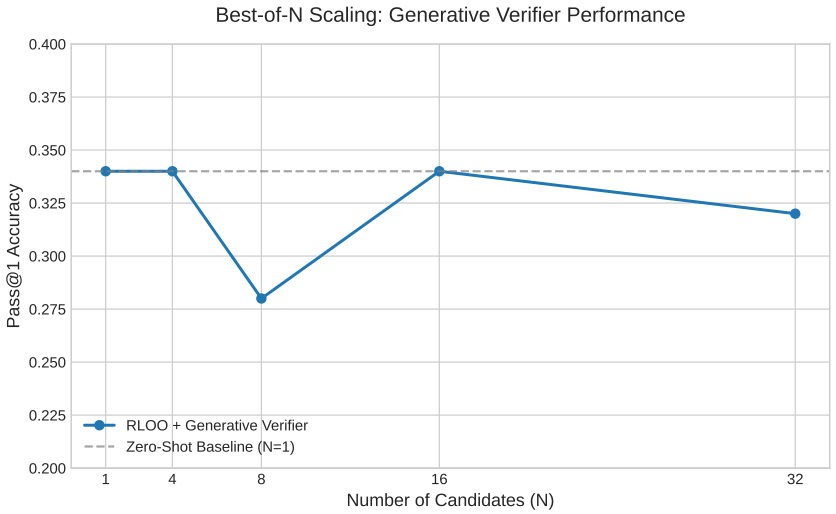

Figure 1: Best-of-N scaling performance of the 0.5B Generative Verifier. The model fails to reliably exceed the zero-shot baseline at higher compute budgets, exhibiting a negative verification gap at $N = 32$.

Arithmetic Hallucination & Constraint Forgetting: At higher generation budgets, the generator often produces traces that mimic the syntactic structure of a correct solution while failing the semantic logic. For example, at $N = 4$ (Target: 99, Numbers: [93, 4, 51, 61]), the policy proposed the equation $(61 - 51) + 4 + 93$. The trace hallucinated the final arithmetic step, stating that the result equals 99, when the actual sum is 107. The verifier was successfully tricked by the confident, step-by-step formatting of the reasoning trace. Similarly, at $N = 16$ (Target: 82), the model correctly calculated $83 - 1 = 82$ but entirely forgot the constraint to use the remaining numbers provided in the prompt.

Length (Verbosity) Bias: Furthermore, an empirical review of the adversarial candidates selected at $N = 32$ suggests the 0.5B verifier is highly susceptible to verbosity bias. When confronted with a large pool of candidate solutions, the verifier systematically assigned higher log-probabilities to longer, more convoluted reasoning chains. The model appears to conflate token length and formatting complexity with reasoning quality, selecting verbose but mathematically flawed traces over succinct, correct solutions.

6 Discussion

The results highlight the fundamental capacity constraints of semantic verification. While generative verification provides a functional alternative to regression-based reward models, parsing multi-step arithmetic traces requires substantial computational depth.

The 0.5B model possesses enough capacity to learn the formatting, syntax, and superficial heuristics of a correct answer (such as preferring longer reasoning traces). However, it is too small to simultaneously perform independent mathematical verification, track arithmetic state, and enforce rigid game constraints. At higher values of N, the Best-of-N pipeline essentially acts as an adversarial attack against the verifier. By generating 32 diverse candidates, the policy inevitably produces "adversarial false positives"—traces that perfectly trigger the verifier's heuristics (length, confident formatting) while hiding critical arithmetic hallucinations.

7 Conclusion and Future Work

This study maps the compute-optimal frontier of self-verification within the 0.5B parameter regime. By framing verification as a next-token prediction task, we demonstrate the mechanics of inference-time compute scaling without environment oracles. However, we conclude that sub-1B models are

currently insufficient to act as robust judges for arithmetic reasoning, as they succumb rapidly to reward hacking, constraint forgetting, and length bias at larger sampling budgets.

To resolve these failure modes, future work should proceed along three distinct proposed steps:

1. **Process-Supervised Reward Modeling (PRM):** The current architecture utilizes an Outcome Reward Model (ORM) that evaluates only the final `[CORRECT]` token. Future pipelines should train a PRM to output intermediate verification tokens after every reasoning step. This step-by-step supervision would forcefully mitigate the arithmetic hallucinations observed in this study by penalizing logic errors exactly where they occur.
2. **Investigating Parameter Capacity Thresholds:** We hypothesize the existence of a parameter scale threshold for effective self-verification. Future experiments should replicate this pipeline across a model family (e.g., 1.5B, 3B, and 7B architectures) to identify the precise parameter count at which a model gains sufficient semantic capacity to hold both arithmetic logic and constraint logic in its attention span simultaneously, thereby overcoming verbosity bias.
3. **Temperature Scaling during Verification:** To protect weaker verifiers from adversarial traces, future Best-of-N evaluations should investigate dynamic temperature scaling. Generating candidates at a lower temperature (e.g., $\tau = 0.2$) may reduce the variance of the candidate pool, limiting the creation of the highly hallucinated, complex traces that successfully tricked the 0.5B model at $N = 32$.

8 Conclusion

This study maps the compute-optimal frontier of self-verification within the 0.5B parameter regime. By framing verification as a next-token prediction task, we demonstrate the mechanics of inference-time compute scaling. However, we conclude that sub-1B models are insufficient to act as robust judges for arithmetic reasoning, as they succumb rapidly to reward hacking and constraint forgetting at larger sampling budgets. Future directions involve exploring step-by-step Process-Supervised Reward Modeling (PRM) to localize verification errors, or scaling the verifier parameter count independently of the generator policy.

9 Team Contributions

- **Rehan Ghori:** Sole contributor. Responsible for the SFT, IPO, and RLOO baseline implementations, data curation, Generative Verifier custom training loop, vLLM inference optimization, and comprehensive Best-of-N evaluation pipeline.

Changes from Proposal To optimize compute efficiency, the Generative Verifier dataset was constrained to a high-quality subset of 80,000 trajectories, and training was capped at 1 epoch, which achieved 97.5% training accuracy for binary verdict classification.

References

[1] Ahmadian, A., et al. (2024). Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms.

[2] Snell, C., et al. (2024). Scaling llm test-time compute optimally can be more effective than scaling model parameters.

[3] Zhang, L., et al. (2024). Generative verifiers: Reward modeling as next-token prediction.

A Implementation Details

All models were trained using Modal serverless infrastructure (H100 GPUs) with custom PyTorch training loops. The verification inference pipeline utilized pure log-likelihood extraction from the final attention state to avoid the computational overhead of autoregressive sampling during the Best-of-N ranking phase.