Budget-aware Test-time Scaling via Discriminative Verification

Kyle Montgomery[1*], **Sijun Tan**[2*], **Yuqi Chen**[1], **Siyuan Zhuang**[2], **Tianjun Zhang**[2], **Raluca Ada Popa**[2], **Chenguang Wang**[1†]
[1]UC Santa Cruz, [2]UC Berkeley
{kylemontgomery, chenguangwang}@ucsc.edu, sijuntan@berkeley.edu

Abstract

Test-time scaling is a powerful strategy for boosting the performance of large language models on complex reasoning tasks. While state-of-the-art approaches often employ generative verifiers to select the best solution from a pool of candidates, this method incurs prohibitive computational costs, limiting its practicality. In this work, we shift the focus to a more budget-aware paradigm: discriminative verification. We conduct a thorough empirical analysis and demonstrate that while discriminative verifiers may underperform in isolation, combining them with self-consistency in a hybrid approach creates a powerful and efficient test-time scaling mechanism. Notably, under a fixed compute budget, this hybrid approach surpasses state-of-the-art generative verification by a significant margin: achieving up to 15.3% higher accuracy on AIME2025. Our findings establish that for practical, real-world applications, budget-aware scaling with discriminative verifiers is not only a "free" upgrade over self-consistency, but also a more effective and efficient alternative to costly generative techniques. Code is available at https://github.com/wang-research-lab/verification.

1 Introduction

Since the release of OpenAI's o1 OpenAI (2024), there has been substantial progress in enhancing the reasoning capabilities of large language models (LLMs) by scaling test-time compute (Snell et al., 2024). Test-time scaling aims to improve model performance by allocating additional computational resources during inference. A canonical example is self-consistency (SC) (Wang et al., 2023b), which involves sampling multiple completions and selecting the final answer via a majority vote.

Alternatively, one can enhance answer selection by employing a generative "verifier" model to score each solution. Generative verifiers are themselves sophisticated LLMs that produce a detailed chain-of-thought (CoT) rationale, critically evaluating a candidate solution before rendering a final verdict (Zhang et al., 2024; Mahan et al., 2024). The approach is intuitively appealing and opens up a new axis for scaling: if one verification pass is good, multiple passes should be even better (Shi & Jin, 2025; Zhao et al., 2025).

While generative verifiers generally offer strong performance, it comes at a staggering computational cost. Indeed, Singhi et al. (2025) demonstrates that generative verifiers underperform SC under low inference budgets and require up to 8× more compute just to match SC, and deliver marginal gains (3.8%) even when granted 128× the compute budget. There are two reasons for this. First, performance is bottlenecked by the quality of the candidate solutions. If all candidates are incorrect, not even an oracle verifier can recover the correct answer. Second, the SC baseline is strong, nearing

[*]Equal contribution.
[†]Corresponding author.

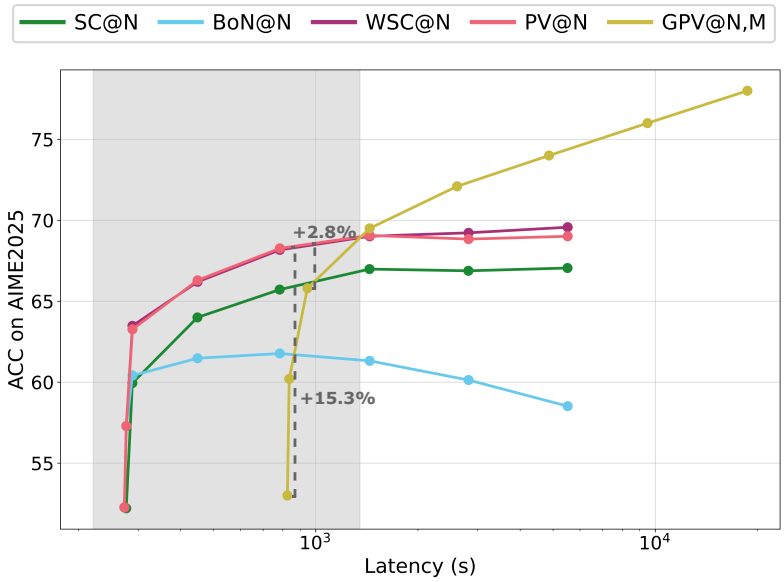


Figure 1: Hybrid discriminative verification techniques (e.g., weighted self-consistency (WSC) (Welleck et al., 2024) and pessimistic verification (PV) (Shi & Jin, 2025)) outperform generative pessimistic verification (GPV) under equalized compute budgets of less than 22.5 minutes (shaded region). For example, at latency budgets of 13.8 minutes and 15.7 minutes, hybrid discriminative verification can outperform generative verification by 15.3% and 2.8%, respectively. N is doubled at each point along the x-axis. For GPV, each solution is verified twice ($M = 2$).

pass@N on many tasks. To surpass SC, a verifier must both (1) agree with the majority when it is correct, and (2) successfully identify the correct minority solution when the majority is wrong. As a result, allocating additional compute to generating candidate solutions typically yields better returns than spending it on verification.

Given these limitations, it is preferable to minimize the cost of verification under constrained budgets. In this regard, discriminative verifiers are promising due to their computational efficiency. Unlike generative verifiers, which require both prefilling and sequential decoding stages, discriminative verifiers only perform a single forward pass, avoiding the decoding bottleneck. However, despite their speed advantage, discriminative verifiers exhibit limited capabilities on complex reasoning tasks (Tan et al., 2025b), often underperforming SC as the pool of candidate solutions grows, which has limited their practical use.

In this work, we show that hybrid approaches combining discriminative verification with self-consistency can offer the best trade-off between effectiveness and efficiency under practical compute budgets. For instance, under inference budgets of 13.8 minutes and 15.7 minutes, hybrid discriminative verification methods (Welleck et al., 2024; Shi & Jin, 2025) outperform state-of-the-art generative verification by 15.3% and 2.8%, respectively. Moreover, although discriminative verifiers underperform SC in isolation, we show that by leveraging these hybrid methods, the resulting test-time scaling pipeline can obtain consistent improvements over SC on AIME2025 by up to 5.1%, while having only 2% compute overhead. These results highlight hybrid discriminative verification as a practical and scalable alternative, delivering strong accuracy gains with negligible overhead and outperforming more expensive generative approaches under realistic budget constraints.

Our contributions are as follows:

- We conduct a thorough empirical analysis of discriminative verification techniques, exploring how different selection strategies perform across scaling regimes. To our knowledge, this is the first study to systematically examine the test-time scaling properties of discriminative verification.

- Building on this analysis, we present a compute-centric comparison of discriminative and generative verification, showing that discriminative methods offer a more practical and efficient alternative under realistic inference budgets.

2 Effective Discriminative Verification

Hybrid Discriminative Verification Discriminative verification often underperforms SC when the pool of candidate solutions is large. Hybrid discriminative verification methods address this by combining the consensus signal from SC with the verifier's signal. We study two hybrid approaches:

- *Weighted self-consistency* (WSC) (Welleck et al., 2024) groups solutions by their final answers and selects the answer with the largest total verifier score, i.e., $a^* = \arg\max_a \sum_{i:a_i=a} r(s_i)$. The approach prioritizes answers that are not only common but also favored by the verifier. Pseudocode for this method is provided in Algorithm 3.
- *Pessimistic verification* (PV) (Shi & Jin, 2025) groups solutions by their final answer and penalizes small answer clusters to reduce the chance of selecting low-support answers. Formally, $a^* = \arg\max_a \left(\frac{1}{n_a} \sum_{i:a_i=a} r(s_i) \; - \; \alpha \frac{\ln N}{n_a+1} \right)$, where α controls the strength of the penalty. When $\alpha = 0$, selection is based exclusively on the mean verifier score. As $\alpha \to \infty$, the penalty dominates and the selection collapses to SC. Empirically, we find that $\alpha = 0.5$ provides a good tradeoff (see Appendix G.1). Pseudocode for this method is provided in Algorithm 4.

Dataset curation. We sample 32k math problems from NuminaMath (LI et al., 2024) and generate responses to each from ten LLMs: DeepSeek-R1 and its six distilled variants (DeepSeek-AI et al., 2025), DeepScaleR-1.5B-Preview (Luo et al., 2025b), and both the preview and production releases of QWQ-32B (Team, 2024, 2025). We grade each response against its reference solution, and throw out problems for which all ten solutions are either correct or incorrect, leaving just 11,420 response groups for training.

Verifier training. Following prior work (Qwen et al., 2025; Yang et al., 2024), we replace the language modeling head of the LLM (specifically DeepSeek-R1-Distill-Qwen-1.5B) with a two-layer scaler value head. We train our verifier using a Bradley-Terry ranking loss combined with an L_2 regularization term (Ouyang et al., 2022; Kirchner et al., 2024). Concretely, our loss is

$$\mathcal{L} = -\frac{1}{|P|\,|N|} \sum_{i \in P} \sum_{j \in N} \log \sigma\big(r_i - r_j\big) \; + \; \frac{\lambda}{2} \mathbb{E}\left(r^2\right),$$

where $r = (r_1, \ldots, r_m)$ are the logits assigned by the verifier to a batch of m responses, $\sigma(x)$ is the logistic function, and P and N are the sets of correct and incorrect responses, respectively. The first term maximizes the probability $\sigma(r_i - r_j)$ that every correct response $i \in P$ outranks every incorrect response $j \in N$ (Bradley & Terry, 1952), and the second term keeps score head well-behaved and centered around zero. Additional training details, including hyperparameters, are provided in Appendix F.

3 Results

We analyze the performance of our trained discriminative verifier under various discriminative verification techniques on several challenging benchmarks: AIME2024, AIME2025, LiveBench Math (White et al., 2025), and GPQA (Rein et al., 2023). For each AIME problem, we sample 128 candidate responses no longer than 16k tokens from DeepSeek-R1-Distill-Qwen-32B. On LiveBench Math and GPQA, we sample only 64 candidate responses. Similar to the construction of our training dataset, we exclude the reasoning content (i.e., the tokens between the <think> and </think> tags) during inference (see Appendix G.2). To ensure our metric estimates (e.g., Pass@N or PV@N) are precise, we report the mean over 1000 resampled draws of size N per problem and report 95% confidence intervals. Our results are provided in Table 1.

Across the board in Table 1, hybrid verification methods like WSC and PV consistently outperform competing selection methods. For example, on AIME2025, PV@32 improves over Pass@1 by 17.2%,

Method	AIME2024	AIME2025	LiveBench Math	GPQA
Pass@1	67.0 ± 0.5	51.9 ± 0.6	62.1 ± 0.2	56.9 ± 0.2
SC@32	83.4 ± 0.4	66.6 ± 0.5	67.0 ± 0.2	63.5 ± 0.2
BoN@32	79.1 ± 0.5	60.8 ± 0.6	64.1 ± 0.2	63.9 ± 0.2
WSC@32	$\mathbf{85.6 \pm 0.4}$	68.8 ± 0.5	67.5 ± 0.2	65.0 ± 0.2
PV@32	85.5 ± 0.4	$\mathbf{69.1 \pm 0.5}$	$\mathbf{67.8 \pm 0.2}$	$\mathbf{65.6 \pm 0.2}$

Table 1: Accuracy rates of DeepSeek-R1-Distill-Qwen-32B ($N = 32$) with various discriminative verification techniques (highlighted in yellow). Pass@1 and SC@32 are included for comparison.

and beats SC@32 and BoN@32 by 2.5% and 8.3%, respectively. Amazingly, even on an out-of-distribution task like GPQA, which includes questions on biology, physics, and chemistry, PV@32 can outperform SC@32 by 2.1%. Appendix E provides additional scaling analysis of discriminative verification techniques.

Comparison of discriminative and generative verification. We compare discriminative and generative verification under equalized compute budgets. Following prior work (Singhi et al., 2025), we measure the total inference compute, i.e., the compute required to *generate and verify* candidate solutions. Specifically, we focus on latency, which we measure on a single H100 GPU using vLLM (Kwon et al., 2023) and its many optimizations to reflect real-world usage. We compare against Heimdall (Shi & Jin, 2025), a state-of-the-art generative verifier trained from DeepSeek-R1-Distill-Qwen-32B, which leverages pessimistic verification to incorporate the consensus signal from SC. We refer to this approach as GPV (see Algorithm 5). Appendix D conducts a similar analysis with FLOPs.

	$N = 1$	$N = 2$	$N = 4$	$N = 8$	$N = 16$	$N = 32$	$N = 64$	$N = 128$
Repeated Sampling	273.1	276.6	288.4	448.4	782.9	1434.0	2815.5	5514.1
Discriminative	0.05	0.10	0.21	0.42	0.83	1.66	3.32	6.65
Generative ($M = 2$)	552.0	558.8	656.6	992.8	1825.7	3423.7	6668.8	13160.7

Table 2: The average wall-clock time (s) for repeatedly sampling N candidate solutions, as well as the average time to verify each candidate solution using discriminative and generative verification.

Table 2 shows the average time to sample and verify N candidate solutions using discriminative and generative verification methods. For instance, verifying 32 solutions sampled from DeepSeek-R1-Distill-Qwen-32B with our 1.5B discriminative verifier takes only 1.66 seconds, just 0.1% of the generation time. On the other hand, verifying 32 candidate solutions with Heimdall at $M = 2$ takes 3423.7 seconds, over twice the time needed for solution generation, and more than 2000$\times$ the cost of discriminative verification. Indeed, as shown in Figure 1, hybrid discriminative verification methods dominate generative verification for all inference budgets shorter than 22.5 minutes (1350s) on AIME2025 with $M = 2$. This threshold is dependent on a range of factors, including the number of verifications per solution (M), the specific solver, the size of the verifier, and the dataset, but it highlights a broader trend: under realistic latency constraints, discriminative verification almost always gives better performance than generative verification.

4 Conclusion

We studied hybrid discriminative verification as a practical alternative to costly generative approaches. Discriminative methods achieve comparable or superior accuracy in practical compute regimes, where the high cost of CoT generation limits generative approaches. Our results highlight hybrid discriminative verification as the more efficient choice for realistic test-time scaling.

References

R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3–4):324–345, December 1952.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In Kevin Knight, Hwee Tou Ng, and Kemal Oflazer (eds.), *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 173–180, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219862. URL https://aclanthology.org/P05-1022/.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Nicholas Crispino, Kyle Montgomery, Fankun Zeng, Dawn Song, and Chenguang Wang. Agent instructs large language models to be general zero-shot reasoners. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 9458–9549, 2024.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy S Liang, and Tatsunori B Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback. *Advances in Neural Information Processing Systems*, 36, 2024.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*, 2025.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms, 2024. URL https://arxiv.org/abs/2406.18495.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*, 2024.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, et al. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*, 2023.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models. *arXiv preprint arXiv:2405.01535*, 2024.

Jan Hendrik Kirchner, Yining Chen, Harri Edwards, Jan Leike, Nat McAleese, and Yuri Burda. Prover-verifier games improve legibility of llm outputs, 2024. URL https://arxiv.org/abs/2407.13692.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath. [https://huggingface.co/AI-MO/NuminaMath-CoT](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024.

Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, Hai Zhao, and Pengfei Liu. Generative judge for evaluating alignment. *arXiv preprint arXiv:2310.05470*, 2023.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

Chris Yuhao Liu and Liang Zeng. Skywork reward model series. https://huggingface.co/Skywork, September 2024. URL https://huggingface.co/Skywork.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.

Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepcoder: A fully open-source 14b coder at o3-mini level. https://pretty-radio-b75.notion.site/DeepCoder-A-Fully-Open-Source-14B-Coder-at-O3-mini-Level-1cf81902c14680b3bee5eb349a512a51, 2025a. Notion Blog.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2, 2025b. Notion Blog.

Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. Generative reward models. *arXiv preprint arXiv:2410.12832*, 2024.

Justus Mattern, Sami Jaghouar, Manveer Basra, Jannik Straube, Matthew Di Ferrante, Felix Gabriel, Jack Min Ong, Vincent Weisser, and Johannes Hagemann. Synthetic-1: Two million collaboratively generated reasoning traces from deepseek-r1, 2025. URL https://www.primeintellect.ai/blog/synthetic-1-release.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

OpenAI. Learning to reason with language models. https://openai.com/index/learning-to-reason-with-llms/, 2024. Accessed: 2025-04-25.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Richard Yuanzhe Pang, Weizhe Yuan, He He, Kyunghyun Cho, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization. *Advances in Neural Information Processing Systems*, 37:116617–116637, 2024.

Junsoo Park, Seungyeon Jwa, Meiying Ren, Daeyoung Kim, and Sanghyuk Choi. Offsetbias: Leveraging debiased data for tuning evaluators, 2024.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A Graduate-Level Google-Proof Q&A Benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Swarnadeep Saha, Xian Li, Marjan Ghazvininejad, Jason Weston, and Tianlu Wang. Learning to plan & reason for evaluation with thinking-llm-as-a-judge. *arXiv preprint arXiv:2501.18099*, 2025.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*, 2022.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

Wenlei Shi and Xing Jin. Heimdall: test-time scaling on the generative verification, 2025. URL https://arxiv.org/abs/2504.10337.

Nishad Singhi, Hritik Bansal, Arian Hosseini, Aditya Grover, Kai-Wei Chang, Marcus Rohrbach, and Anna Rohrbach. When to solve, when to verify: Compute-optimal problem solving and generative verification for llm reasoning, 2025. URL https://arxiv.org/abs/2504.01005.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Sijun Tan, Michael Luo, Colin Cai, Tarun Venkat, Kyle Montgomery, Aaron Hao, Tianhao Wu, Arnav Balyan, Manan Roongta, Chenguang Wang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. rllm: A framework for post-training language agents. `https://pretty-radio-b75.notion.site/rLLM-A-Framework-for-Post-Training-Language-Agents-21b81902c146819db63cd98a54ba5f31`, 2025a. Notion Blog.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. Judgebench: A benchmark for evaluating llm-based judges, 2025b. URL `https://arxiv.org/abs/2410.12784`.

Qwen Team. Qwq: Reflect deeply on the boundaries of the unknown, November 2024. URL `https://qwenlm.github.io/blog/qwq-32b-preview/`.

Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL `https://qwenlm.github.io/blog/qwq-32b/`.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Y Wu, and Zhifang Sui. Math-shepherd: A label-free step-by-step verifier for llms in mathematical reasoning. *arXiv preprint arXiv:2312.08935*, 2023a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023b. URL `https://arxiv.org/abs/2203.11171`.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, et al. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. *arXiv preprint arXiv:2306.05087*, 2023c.

Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J. Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. Helpsteer2: Open-source dataset for training top-performing reward models, 2024. URL `https://arxiv.org/abs/2406.08673`.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. From decoding to meta-generation: Inference-time algorithms for large language models. *arXiv preprint arXiv:2406.16838*, 2024.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, et al. Livebench: A challenging, contamination-free llm benchmark. *arXiv preprint arXiv:2406.19314*, 2024.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. Livebench: A challenging, contamination-limited llm benchmark, 2025. URL `https://arxiv.org/abs/2406.19314`.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024. URL `https://arxiv.org/abs/2409.12122`.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.

Fei Yu, Anningzhe Gao, and Benyou Wang. Ovm, outcome-supervised value models for planning in mathematical reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 858–875, 2024.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024.

Eric Zhao, Pranjal Awasthi, and Sreenivas Gollapudi. Sample, scrutinize and scale: Effective inference-time search by scaling verification, 2025. URL https://arxiv.org/abs/2502.01839.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2024.

Banghua Zhu, Evan Frick, Tianhao Wu, Hanlin Zhu, and Jiantao Jiao. Starling-7b: Improving llm helpfulness & harmlessness with rlaif, November 2023a.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. Judgelm: Fine-tuned large language models are scalable judges. *arXiv preprint arXiv:2310.17631*, 2023b.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Preliminaries

Repeated sampling is a test-time scaling technique that involves generating a batch of N independent candidate solutions $\{s_i\}_{i=1}^{N}$ for a given problem Q. Each solution s_i is a chain of reasoning that terminates in a final answer $a_i = \text{Ans}(s_i)$. As N increases, the probability that at least one answer is correct also rises (i.e., Pass@N improves; see Figure 1) (Cobbe et al., 2021). However, this leaves open the central challenge of selecting a single answer a^* from among the candidates in the absence of ground truth.

Self-consistency. A common approach for this selection problem is self-consistency (SC) (Wang et al., 2023b). Since correct answers tend to reoccur across independent solutions, SC groups responses by their final answer and selects the most frequent one. Formally, each distinct answer a has support size $n_a = |\{i : a_i = a\}|$, and SC chooses $a^* = \arg\max_a n_a$. While this approach is robust when the correct answer is common, it can fail when the majority converges on an incorrect answer. Pseudocode for this method is provided in Algorithm 1.

Best-of-N. Another strategy is best-of-N (BoN) selection (Charniak & Johnson, 2005; Cobbe et al., 2021), which uses a *discriminative verifier* to assign each solution a scalar score (e.g., in $[0, 1]$), and selects the final answer from the highest-scoring solution. Formally, each solution s_i receives a scalar score $r(s_i)$, then BoN chooses $a^* = \text{Ans}(s^*)$ where $s^* = \arg\max_{s_i} r(s_i)$. A strong verifier can identify correct but rare responses that SC might miss. However, as N increases, it can also be misled by confident yet incorrect responses, highlighting a long-tail vulnerability (see Figure 1). Pseudocode for this method is provided in Algorithm 2.

B Related Work

LLM-based verifiers can be broadly categorized into generative and discriminative approaches. Generative verifiers use large language models as judges that assess the correctness or quality of outputs by generating natural language rationales. A growing body of work explores this direction, employing LLMs as judges for modeling human preferences (Dubois et al., 2024; Zheng et al., 2024; Li et al., 2024; Wang et al., 2023c; Kim et al., 2023, 2024; Li et al., 2023; Zhu et al., 2023b; Mahan et al., 2024), or as verifiers for evaluating solution correctness in reasoning tasks (Zhang et al., 2024; Singhi et al., 2025; Shi & Jin, 2025; Saha et al., 2025).

In contrast, discriminative verifiers, such as reward models, assign scalar scores to candidate responses based on human preference data (Christiano et al., 2017; Ziegler et al., 2019; Zhu et al., 2023a; Liu & Zeng, 2024; Wang et al., 2024; Park et al., 2024; Han et al., 2024). These models are central to reinforcement learning from human feedback and are also used to rank or select responses in BoN inference settings (Lightman et al., 2023; Wang et al., 2023a; Luo et al., 2024; Saunders et al., 2022; Uesato et al., 2022; Yu et al., 2024). Together, generative and discriminative verifiers provide complementary paradigms for evaluating, selecting, and aligning LLM outputs at inference time.

A substantial body of work has investigated improving the mathematical reasoning capabilities of LLMs through prompting (Wei et al., 2022; Kojima et al., 2022; Crispino et al., 2024), training (Cobbe et al., 2021; Guan et al., 2025; Hosseini et al., 2024; Lightman et al., 2023; Pang et al., 2024; Ye et al., 2025; Luo et al., 2025a,b; Tan et al., 2025a), and test-time scaling (Snell et al., 2024; Brown et al., 2024; Setlur et al., 2024). Following the release of o1 (OpenAI, 2024), there has been a surge of interest in test-time scaling methods for LLM reasoning (Snell et al., 2024; Brown et al., 2024; Singhi et al., 2025; Zhao et al., 2025), which improve performance by sampling multiple solutions and aggregating them via majority voting or LLM-based verification. Our work builds on this line of research, demonstrating that discriminative LLM verifiers can serve as an effective and efficient verification approach for test-time scaling in complex math reasoning tasks.

C Algorithms

Algorithm 1 Self-Consistency (SC@N)

Require: problem Q, solver LM, slate size N

Candidates $\leftarrow \{s_i\}_{i=1}^{N} \sim \mathrm{LM}(Q)$ ▷ **Stage 1: Generate Candidates**

Extract final answers $\{a_i\}_{i=1}^{N}$ and partition into clusters $\{\mathcal{C}_a\}$ by a **Stage 2: Group Answers**

for each cluster $\mathcal{C}_a$ **do**

  $n_a \leftarrow |\mathcal{C}_a|$

$a^* \leftarrow \arg\max_a n_a$ ▷ **Stage 3: Plurality Vote**

return a^*

Algorithm 2 Best-of-N (BoN@N)

Require: problem Q, solver LM, slate size N, verifier V

Candidates $\leftarrow \{s_i\}_{i=1}^{N} \sim \mathrm{LM}(Q)$ ▷ **Stage 1: Generate Candidates**

Verifications $\leftarrow \{\, r_i = V(s_i) \,\}_{i=1}^{N}$ ▷ **Stage 2: Verify Candidates**

$i^* \leftarrow \arg\max_{i \in \{1,\ldots,N\}} r_i$ ▷ **Stage 3: Select Highest-Scoring Solution**

$a^* \leftarrow \mathrm{Ans}(s_{i^*})$ ▷ **Stage 4: Extract Final Answer**

return a^*

Algorithm 3 Weighted Self-Consistency (WSC@N)

Require: problem Q, solver LM, slate size N, verifier V

Candidates $\leftarrow \{s_i\}_{i=1}^{N} \sim \mathrm{LM}(Q)$ **Stage 1: Generate Candidates**

Verifications $\leftarrow \{\, r_i = V(s_i) \,\}_{i=1}^{N}$ ▷ **Stage 2: Verify Candidates**

Extract final answers $\{a_i\}_{i=1}^{N}$ and partition into clusters $\{\mathcal{C}_a\}$ by a **Stage 3: Group Answers**

for each cluster $\mathcal{C}_a$ **do**

  $W_a \leftarrow \sum_{i \in \mathcal{C}_a} r_i$

$a^* \leftarrow \arg\max_a W_a$ **Stage 4: Select Highest-Weight Answer**

return a^*

Algorithm 4 Pessimistic Verification (PV@N)

Require: problem Q, solver LM, slate size N, verifier V, penalty weight α

Candidates $\leftarrow \{s_i\}_{i=1}^{N} \sim \mathrm{LM}(Q)$ ▷ **Stage 1: Generate Candidates**

Verifications $\leftarrow \{\, r_i = V(s_i) \}_{i=1}^{N}$ ▷ **Stage 2: Verify Candidates**

Extract final answers $\{a_i\}_{i=1}^{N}$ and partition into clusters $\{\mathcal{C}_a\}$ by a **Stage 3: Group Answers**

for each cluster $\mathcal{C}_a$ **do**

  $n_a \leftarrow |\mathcal{C}_a|$

  $\bar{r}(a) \leftarrow \frac{1}{n_a} \sum_{i \in \mathcal{C}_a} r_i$

  $\psi_a \leftarrow \frac{\ln N}{n_a + 1}$

$a^* \leftarrow \arg\max_a \, [\, \bar{r}(a) - \alpha \, \psi_a \,]$ ▷ **Stage 4: Select Best Answer**

return a^*

Algorithm 5 Generative Pessimistic Verification (GPV@N, M)

Require: problem Q, solver LM, slate size N, generative verifier V, # of verifications M, penalty weight α

Candidates $\leftarrow \{s_i\}_{i=1}^{N} \sim \mathrm{LM}(Q)$ ▷ **Stage 1: Generate Candidates**
for $i = 1$ to N **do** ▷ **Stage 2: Generative Verifications (repeat M times)**
  for $m = 1$ to M **do**
    $(\mathrm{CoT}_{i,m},\, r_{i,m}) \leftarrow V(s_i)$
  $\tilde{r}_i \leftarrow \frac{1}{M} \sum_{m=1}^{M} r_{i,m}$
Extract final answers $\{a_i\}_{i=1}^{N}$ and partition into clusters $\{\mathcal{C}_a\}$ by a **Stage 3: Group Answers**
for each cluster $\mathcal{C}_a$ **do**
  $n_a \leftarrow |\mathcal{C}_a|$
  $\bar{r}(a) \leftarrow \frac{1}{n_a} \sum_{i \in \mathcal{C}_a} \tilde{r}_i$
  $\psi_a \leftarrow \frac{\ln(NM)}{n_a M + 1}$
$a^* \leftarrow \arg\max_a [\, \bar{r}(a) - \alpha\, \psi_a \,]$ ▷ **Stage 4: Select Best Answer**
return a^*

D FLOPs Analysis of Discriminative and Generative Methods

FLOPs provide a theoretical measure of the intrinsic compute required, independent of hardware and other implementation details, allowing us to study how compute requirements scale for discriminative and verification techniques. For a decoder-only transformer model with hidden size d, intermediate size m, L layers, and vocabulary size V, the FLOPs roughly decompose into three components:

1. **Layer projections.** Each token per layer requires $8d^2 + 4dm$ FLOPs for Q, K, V, O projections and the MLP.
2. **Attention.** With KV caching, prefill compute is quadratic in T_{in}: each of the T_{in} tokens attends to all previous tokens, giving $4d \cdot \frac{T_{\text{in}}(T_{\text{in}}+1)}{2}$ FLOPs per layer. During decoding, cached keys/values avoid recomputation, so each of the T_{out} generated tokens only attends to the fixed prefix and prior outputs, costing $4d \cdot (T_{\text{in}} T_{\text{out}} + \frac{T_{\text{out}}(T_{\text{out}}-1)}{2})$ FLOPs per layer.
3. **LM Head.** Finally, output projection adds $2dVT_{\text{out}}$ FLOPs, where V is the vocabulary size. For discriminative verification, we set $V = 1$ and $T_{\text{out}} = 1$, corresponding to a single scalar output.

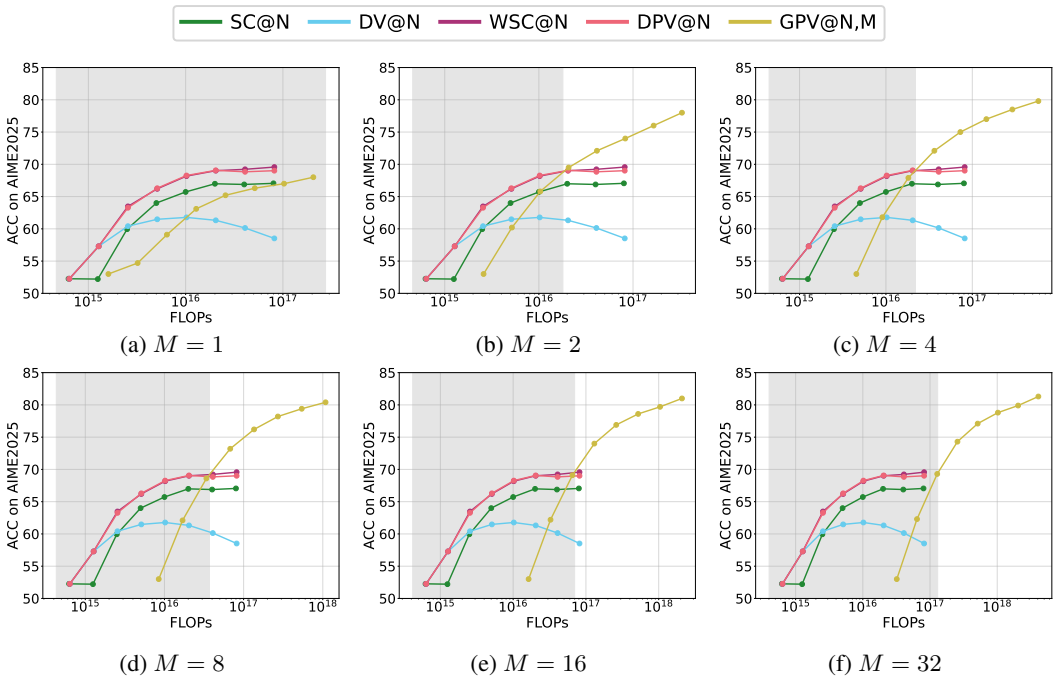
Note that this formulation omits smaller terms such as normalization layers, activation functions, or positional encodings.

We compare discriminative and generative verification methods on AIME2025. For each, we vary the number of candidate solutions $N \in 2, 4, 8, 16, 32, 64, 128$ and, for generative verification, the number of verifications per response $M \in 1, 2, 4, 8, 16, 32$. Results are presented in Figure 2.

Repeated sampling provides a natural compute baseline: generating N candidate solutions requires $O(N)$ long CoT traces. For example, generating 32 candidate solutions to a problem from AIME2025 with DeepSeek-R1-Distill-Qwen-32B costs 2.0×10^{16} FLOPs on average. SC selects the most common answer from the candidate solutions and uses no additional compute beyond that of repeated sampling. By contrast, verification-based techniques incur additional compute cost. For example, verifying 32 solutions with our discriminative verifier trained in Section 2 costs just 4.1×10^{14} FLOPs on average, just 2.0% of the compute used for repeated sampling. All discriminative verification techniques (BoN, WSC, PV) use the same amount of verification compute. While BoN tends to underperform SC when N is large, hybrid discriminative verification methods consistently outperform the SC baseline by up to 5.1% for a negligible amount of additional compute.

Conversely, generative verification techniques are significantly less efficient. For example, verifying the same 32 solutions with Heimdall (Shi & Jin, 2025) just once ($M = 1$) requires 3.1×10^{16} FLOPs, over 50% more FLOPs than solution generation and nearly 76× more FLOPs than discriminative verification. While generative verification can be made more effective by scaling the number of verifications per candidate solution (i.e., increasing M), the compute requirements scale linearly.

Critically, under practical FLOP budgets, hybrid discriminative verification techniques outperform generative verification. This is because discriminative methods allocate nearly all of the compute



(a) $M = 1$ (b) $M = 2$ (c) $M = 4$

(d) $M = 8$ (e) $M = 16$ (f) $M = 32$

Figure 2: Accuracy vs. FLOPs on AIME2025 under equalized compute budgets. Each subplot varies the number of verifications per candidate solution (M). Along each curve, successive points correspond to doubling the number of candidate solutions (N). The shaded region highlights the FLOPs budgets where hybrid discriminative verification techniques strictly outperform generative verification under equalized compute budgets.

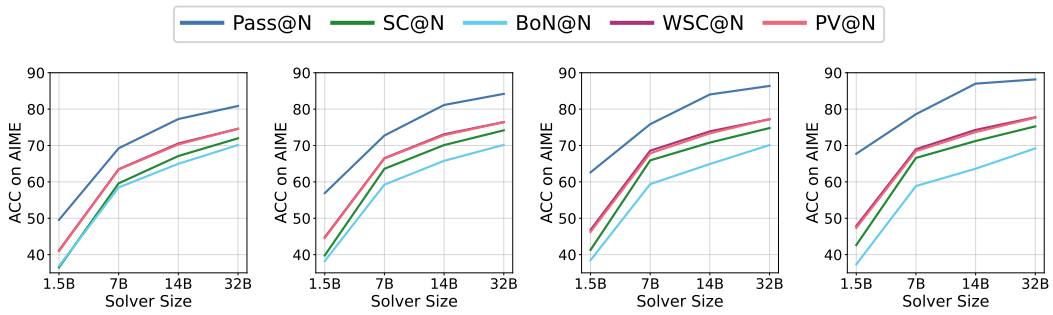
budget towards sampling candidate solutions, while generative verification splits its compute budget between sampling and verifying candidates. Under realistic compute budgets, scaling the number of candidate solutions produces greater returns than scaling verifications; even an oracle-level verifier will fail to produce the correct answer if no correct solutions were sampled. With a large enough budget, however, the gain from sampling additional candidates begins to saturate, and generative verification techniques begin to dominate. The critical threshold at which generative verification becomes superior depends on M (Figure 2). For example, when $M = 1$, hybrid discriminative verification techniques outperform generative verification for any $N \leq 128$. The optimal generative configuration occurs when $M = 2$, but even still, hybrid discriminative verification methods remain optimal for compute budgets less than 2.2×10^{16} FLOPs.

E Scaling Analysis of Discriminative Verification

E.1 Scaling Model Size For Discriminative Verification

Here, we analyze how discriminative verification techniques scale with respect to the size of the solver model, which generates the candidate solutions. To do so, we generate 128 candidate solutions per question in AIME2024 and AIME2025 using DeepSeek-R1-Distill-Qwen models with 1.5B, 7B, 14B, and 32B parameters, and verify each using our trained discriminative verifier. We plot the aggregate results in Figure 3 for several values of N.

We observe that increasing the solver's size produces consistent but diminishing performance increases on AIME. Specifically, hybrid methods like WSC and PV scale similarly to SC as the size of the solver is increased, with WSC and PV maintaining a consistent edge over SC regardless of the solver's size, across various Ns. BoN, on the other hand, exhibits poor scaling behavior: when N is small, BoN only slightly underperforms SC, but when N is large, BoN trails far behind. These results suggest that hybrid approaches can effectively mitigate BoN's long-tail vulnerability.



(a) $N = 8$ (b) $N = 16$ (c) $N = 32$ (d) $N = 64$

Figure 3: Accuracy rates on AIME 2024/2025 for various discriminative verification methods across four solver sizes for several values of N. Pass@N and SC@N are included as baselines.

E.2 Inference-time Scaling of Discriminative Verification

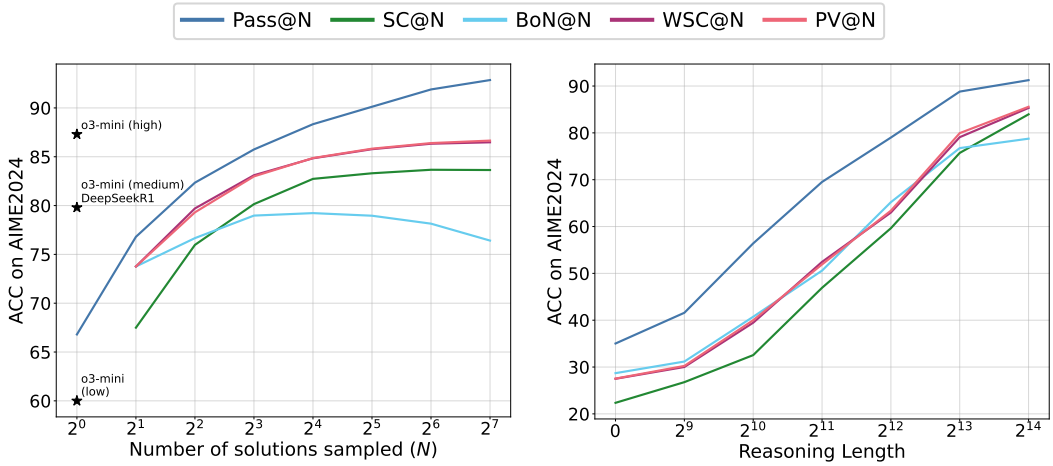


Figure 4: **Left:** Unlike BoN, hybrid techniques show consistent but diminishing improvements on AIME2024 from increasing the number of candidate results N sampled from DeepSeek-R1-Distill-Qwen-32B. **Right:** The performance of DeepSeek-R1-Distill-Qwen-32B on AIME2024 scales logarithmically with the reasoning budget regardless of verification method. Here, $N = 32$.

We study how each discriminative verification method benefits from increased inference-time compute along two axes: the number of candidate solutions sampled from the solver and the reasoning budget allocated to the solver. First, we observe that scaling N produces consistent but diminishing improvements in performance on AIME (i.e., Pass@N increases). BoN struggles to benefit from scaling N, with performance quickly saturating and even falling. On the other hand, hybrid approaches like WSC and PV show consistent improvements as more solutions are sampled, maintaining a 2.2% to 5.6% edge over SC as N is scaled from 2 to 128. On AIME2024, WSC and PV boost the accuracy of DeepSeek-R1-Distill-Qwen-32B from 66.8% to 79.7% with only 4 candidate solutions, matching the performance of o3-mini (medium) or DeepSeek-R1, and outperforming SC by 3.7%.

To control the reasoning budget, we use budget forcing (Muennighoff et al., 2025) and truncate the candidate solutions $T \in \{0, 512, 1024, 2048, 4096, 8192, 16384\}$ tokens after the opening think tag, manually append the closing think tag, then allow the model to continue generating its final answer. In doing so, we collect solutions under constrained reasoning budgets. We observe that even as the reasoning budget is scaled from 0 to 16k tokens, WSC and PV maintain an edge over SC, even while BoN falls off, showcasing the reliability of hybrid verification methods under various constraints.

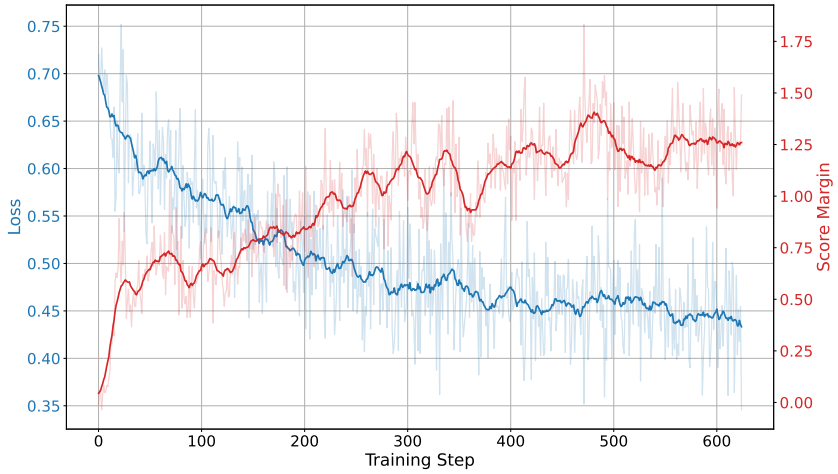


Figure 5: **Blue:** The loss decreases over one epoch of training. **Red:** The score margin—the difference in score assigned to correct solutions and incorrect solutions on average across a global batch—increases during training. Together, these indicate that the discriminative verifier learns to discriminate between correct and incorrect solutions.

F Additional Technical Details

Our training data is based on a subset of Numina-Math (LI et al., 2024), which was released under an Apache license 2.0. DeepSeek-R1 responses were collected from Mattern et al. (2025) (also Apache 2.0). Meanwhile, the majority of the responses from six DeepSeek-R1-Distill models, DeepScaleR-1.5B-Preview, and the two QwQ models were generated on a local cluster of NVIDIA A100 GPUs, with a minority coming from 3rd party API providers.

Our evaluation datasets are AIME2024 (MIT), AIME2025 (MIT), LiveBench-Math (White et al., 2024) (Apache 2.0), and GPQA (Rein et al., 2023) (CC-by-4.0). Combined, they include 596 questions. We decontaminate the training dataset by excluding any problem whose fuzzy-match similarity to an entry in our evaluation sets exceeds 80. For each AIME problem, we sample 128 candidate solutions, while on LiveBench Math and GPQA, we sample only 64 candidate solutions.

When rolling out solutions during training and evaluation, we follow the model's usage recommendations, namely prefilling the opening think token, sampling with a temperature of 0.6 and a top-p value of 0.95, and instructing the model to output its final answer within `\boxed{}`.

Our 1.5B discriminative verifiers was trained on 4xA100s using the hyperparameters listed in Table 3. Figure 5 shows the training dynamics (i.e., loss and score margin) for our discriminitive verifier.

Hyper-parameter	**Value**
Global batch size	32
LR	5×10^{-5}
LR scheduler	Linear with 20 warmup steps
Optimizer (AdamW)	$\beta_1 = 0.9,\ \beta_2 = 0.999$
λ	0.01
Max gradient norm	1.0

Table 3: Hyper-parameters for training discriminative verifiers.

G Additional Ablation Experiments

In addition to our main experiments, we include two further ablations conducted on a held-out validation set. To construct this set, we removed 250 problems from the training dataset and generated

32 responses per problem with 1.5B, 7B, 14B, and 32B variants of deepseek-ai/DeepSeek-R1-Distill-Qwen. We discarded items where all sampled responses were correct or all incorrect, leaving 691 problems for validation. This setup ensures that both correct and incorrect responses are available, making it suitable for evaluating the performance of a verifier.

G.1 Effect of the Pessimism Weight α

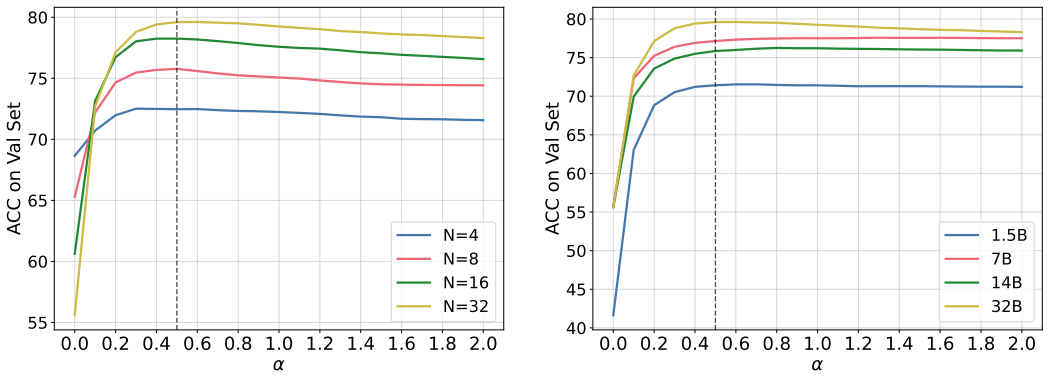


Figure 6: **Left:** Validation accuracy of PV as a function of the pessimism weight α for various numbers of independent candidate solutions (N). **Right:** Validation accuracy of PV as a function of the pessimism weight α for various-sized solver models.

We first ablate the effect of the pessimism weight α in pessimistic verification (PV). As shown in Figure 6 (left), which only includes 147 response groups generated by deepseek-ai/DeepSeek-R1-Distill-Qwen-32B, performance peaks around $\alpha \approx 0.5$ for $N \in 4, 8, 16, 32$ and slowly decays. Figure 6 (right) demonstrates that $\alpha = 0.5$ is a reasonable choice for 4 solver models of various sizes. Based on this result, we set $\alpha = 0.5$ for all main experiments. Notably, in Shi & Jin (2025), the authors use an $\alpha = 0.1$ for experiments with Heimdall. This makes sense: with a stronger verifier and sufficiently large M, you can reduce α and put more weight on the verifier.

G.2 Effect of Reasoning Content on the Verifier

We next ablate whether to pass the reasoning content (the tokens between `<think>` and `</think>`) to the verifier during training and inference. Our main experiments exclude reasoning, i.e., the verifier observes only the final solution string. For comparison, we trained and evaluated a second verifier that retains the reasoning content. As shown in Figure 7, including reasoning consistently degrades performance across all selection methods: BoN, WSC, and PV all achieve lower accuracy when reasoning traces are present. This suggests that the additional reasoning text introduces noise rather than a useful signal, reinforcing our choice to exclude it during both training and evaluation.

H Limitations and Broader Impacts

Limitations Verification techniques can improve answer selection only when at least one correct candidate is present, so its ceiling is still bounded by the solver's Pass@N. Additionally, like SC, hybrid methods assumes that responses can be clustered into equivalence classes and thus would likely not be suitable for domains lacking a reliable mechanism for determining answer equivalence (e.g., open-ended natural-language tasks). Also, under extreme compute budgets, generative verification techniques outperform our hybrid verification techniques. Lastly, our latency analysis between discriminative and generative verification is grounded in current software and hardware; with rapidly advancing progress on both fronts, generative verification is sure to grow more efficient.

Broader Impacts Discriminative verification techniques enable highly efficient yet effective test-time scaling. This may lower the hardware barrier for academic labs or other groups that need strong reasoning but cannot afford massive inference clusters. On the flip side, better low-cost reasoning may

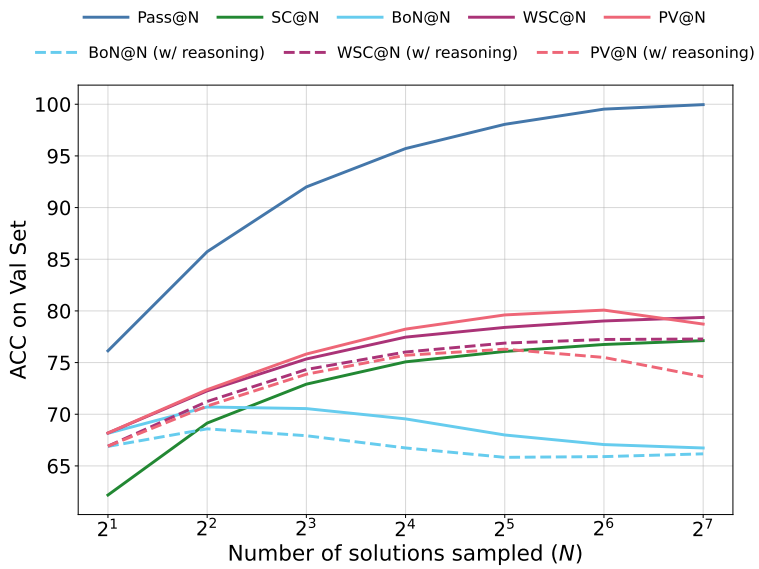


Figure 7: Validation accuracy on the held-out set when including vs. excluding reasoning content in verifier inputs for both training and inference.

accelerate misuse scenarios, which can be mitigated by techniques such as rate-limiting, watermarking, or alignment training.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction and grounded in the experimental results in Figure 1 and Section 3.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed Appendix H.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: This work is empirical.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 2 and Appendix F detail the data curation, training methodology, hyperparameters used. Section 3 specifies the evaluation settings necessary to replicate our work.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code and data is available at `https://github.com/wang-research-lab/verification`.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Section 2 and Appendix F detail the data curation, training methodology, hyperparameters used. Section 3 specifies the evaluation settings necessary to replicate our work. Full details are provided code at `https://github.com/wang-research-lab/verification`.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The main results in Table 1 include 95% confidence intervals, and Section 3 details the methodology used to derive these confidence intervals.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Some analysis of the compute resources used are provided in Section 3, and additionally in Appendix F.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors made every effort to conform with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Broader impacts are discussed in Appendix H.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The work releases only a small discriminative verifier; risk of direct misuse is minimal.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The creators of the assets used in the work are properly cited and their respective licenses were respected and mentioned in Appendix F.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: New assets are released and documented at `https://github.com/wang-research-lab/verification`.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.