Calibrated Reasoning: An Explanatory Verifier for Dynamic and Efficient Problem-Solving

Anisha Garg
anisha.garg@cerebras.net

Engin Tekin
engin.tekin@cerebras.net

Yash More
yash.more@cerebras.net

David Bick
david.bick@cerebras.net

Nishit Neema
nishit.neema@cerebras.net

Ganesh Venkatesh
ganesh.venkatesh@cerebras.net

APPLIED AI RESEARCH, CEREBRAS

Abstract

Advanced test-time computing strategies are essential for scaling reasoning models, but their effectiveness is capped by the models' poor self-evaluation. We propose a pairwise Explanatory Verifier, trained via reinforcement learning (GRPO), that produces calibrated confidence scores and associated natural language reasoning for generated solutions. Our verifier improves the accuracy and efficiency of test-time strategies like best-of-n and self-reflection. Crucially, it excels at identifying challenging failure modes, such as when both candidate solutions are identically incorrect, succeeding where standard methods like majority voting fail.

1 Introduction

Advanced test-time strategies like multi-path exploration are key to solving increasingly complex problems [1–3]. However, their effectiveness is capped by a core limitation in reasoning models. These models struggle with reliable self-evaluation [4] and are often biased towards a narrow set of approaches (Figure 4). This critical failure to discern correctness creates a bottleneck, preventing dynamic exploration of alternatives and hindering progress on scaling AI systems to address challenging tasks.

To overcome this bottleneck, we introduce an Explanatory Verifier trained via Reinforcement Learning [5] to provide both a calibrated judgment and a natural language rationale. Rather than assessing solutions in isolation, our verifier takes motivation from prior work [6] to perform a more efficient relational analysis on pairs of reasoning trajectories to identify subtle errors and judge their correctness. This justified comparative judgment framework is designed to directly enhance common test-time reasoning strategies like best-of-n sampling [7] and self-reflection [8].

To our knowledge, this is the first work to systematically train and analyze a pairwise, explanatory verifier to scale a complex reasoning system. Downstream evaluations on challenging benchmarks demonstrate that the verifier significantly improves accuracy in best-of-N sampling and self-reflection settings, often while using fewer computational resources. The foundation of this success is the verifier's robust calibration; our analysis reveals that it can reliably detect incorrect answers even in ambiguous scenarios where both candidates are wrong or identically flawed. These are precisely the cases where voting strategies fail. This reliability enables a more dynamic inference paradigm, where computationally expensive exploration is reserved only for problems where verifier confidence is low. Overall, our approach enables efficient scaling of reasoning models, providing a blueprint for dynamic systems that can tackle increasingly complex tasks with proportional resource allocation.

Accepted to Workshop on Efficient Reasoning@ 39th Conference on Neural Information Processing Systems (NeurIPS 2025)

2 Explanatory Verifier Training

Problem Formulation We frame training the explanatory verifier as a reinforcement learning problem. The goal is to learn a policy π that, given a problem instance consisting of a question Q and two candidate responses (R_A, R_B) [9], generates a completion containing reasoning within <think> tags and ratings $V = (v_A, v_B) \in [0, 10]$ in final answer. Ground-truth labels $y = (y_A, y_B) \in 0, 1$ indicate correctness, but the continuous output scale allows the model to express uncertainty in its judgments. Completion quality is measured by a reward function, $R(c, x)$, which is defined as the binary cross-entropy [7] between the normalized ratings and labels (see Section 2.2). The training objective, optimized using Group Relative Policy Optimization (GRPO) [5], is to find the optimal policy π^* that maximizes the expected reward over the data distribution $\mathcal{D}$:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{x\sim\mathcal{D}, c\sim\pi(\cdot|p(x))}[R(c, x)]$$

2.1 Training Dataset

The foundation of our verifier is a curated dataset derived from Numina Math [10], CodeForces, and LeetCode [11, 12]. For each problem, we generated multiple solution attempts with Qwen3-8B [13] to get correct and incorrect reasoning paths for training.

The base datasets contain many corner cases that are challenging for automated verification approaches. We implemented a rigorous curation process, detailed in Appendix B, so that our data contains only problems where automated verification is reliable to provide high-quality signal. We removed any problems that: had open-ended responses such as proof-based questions; contained multiple sub-questions and answers; were ambiguous or under-specified as determined by a strong LLM-as-a-judge; or whose final answer evaluated to anything other than a single numeric expression.

Finally, our pairwise input format imposes a substantial constraint on context length. We remove the content within <think> tags, and further filter out samples that exceed 6,144 input tokens. In the end, we have 3,634 unique input tuples, (Q, R_A, R_B), spanning 628 distinct questions, Q, for math dataset. We hold out 294 tuples for validation set.

2.2 Reward Shaping

We train the verifier to produce ratings on a continuous scale from 0 to 10, where 0 indicates high confidence that the response is incorrect and 10 indicates high confidence that it is correct. A key challenge in our training was incentivizing the verifier to use the full [0, 10] rating scale despite a binary ground-truth signal ($y \in 0, 1$). Mean squared error often pushes predictions to the extremes, so we designed a reward based on a variant of binary cross-entropy to encourage calibrated outputs. First, we normalize the model's raw rating for a given response, $v \in [0, 10]$, to a probability-like value $p = v/10$ and then clamp it to $[0.1, 0.9]$ to ensure training stability and prevent the logarithm from producing excessively large or infinite values: $\hat{p} = 0.1 + 0.8 \cdot p = 0.1 + 0.08 \cdot v$. This transformed value $\hat{p}$ is then used to calculate a reward based on binary cross-entropy. For a completion c with responses A and B, reward is calculated as the sum of the rewards for each individual judgment:

$$R_{\text{shaped}}(c, x) = \underbrace{[y_A \log(\hat{p}_A) + (1 - y_A) \log(1 - \hat{p}_A)]}_{\text{Reward for response A}} + \underbrace{[y_B \log(\hat{p}_B) + (1 - y_B) \log(1 - \hat{p}_B)]}_{\text{Reward for response B}}$$

In this formulation, while clamping ensures numerical stability, the logarithmic formulation penalizes confident errors sharply and allows nuanced predictions without excessive penalty.

2.3 Implementation Details

We use GRPO [5] to train QWEN3-8B, following the VeRL implementation[1], in a multi-stage manner on math and coding datasets [14]. The learning rate was 1×10^{-6} with a 10-step linear warmup, KL coefficient 0.001, and rollouts with temperature and top_p set to 1.0. Training is strictly on-policy, with one gradient step per rollout group. In **Stage 1**, we train on a math-only dataset with a maximum sequence length (MSL) of 8,192, generating 16 rollouts per prompt with a global batch size of 256. We use dynamic filtering to discard rollouts with no variance. Stage 1 proceeds for 240 steps. In

[1] https://github.com/volcengine/verl

Stage 2, training is extended with a curated coding dataset combined with a subset of math. We observe truncated responses, entropy collapse, and verifier ratings concentrated around the midpoint (score ≈ 5), suggesting the model was struggling with problem difficulty within the given token budget. To mitigate these issues, MSL is increased to 16,384, global batch size to 512, an entropy coefficient of 0.001 is introduced, and high-difficulty problems (difficulty > 0.8) are filtered using pass@k. Stage 2 continues for 120 steps.

3 Evaluating the Explanatory Verifier: Ability to discern, Downstream Performance, and Emergent Capabilities

We comprehensively evaluate our Explanatory Verifier on (i) its judgment accuracy, (ii) its enhancement of downstream reasoning tasks, and (iii) its emergent generative skills.

3.1 Benchmarking the Verifier's Ability to Discern

This section quantifies the judgment accuracy of Explanatory Verifier, benchmarking it against the self-evaluation capability of the baseline reasoning model. The results, summarized in Figure 1, demonstrate a significant improvement in the model's discerning abilities throughout its training.

Improvements in Discernment Figure 1 (Left) shows that, on a held-out validation set, the model progressively improves at evaluating correctness across all judgment scenarios, including identifying the better response and recognizing when both are incorrect. Initially (step 0), performance is highly skewed towards giving a high score to everything, but training develops reliable judgment in all settings. Of particular interest is the model's ability to identify when both inputs are incorrect.

Verifier Response Calibration Figure 1 (Right) shows that the verifier's ratings become significantly more calibrated and consistent, beyond accuracy. This trend towards higher precision and lower variance holds true across problems of varying difficulty levels (grouped by pass@k bins).

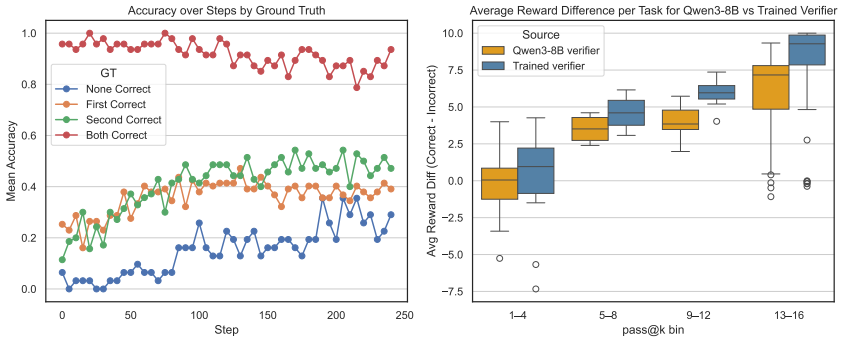

Figure 1: **(Left)** Training progression of the verifier's accuracy, broken down by the ground-truth (GT) configuration of the input pair. **(Right)** A comparison of the trained verifier (RL) against the baseline on generations from AIME 2024. The RL-trained verifier exhibits a much larger separation in ratings between correct and incorrect responses and shows lower variance in its predictions.

Refer to Appendix C for more analysis on the underlying behavior of the verifier. Owing to these properties, we observe improvements when leveraging the verifier in test-time scaling, as discussed next. All results are reported using greedy decoding.

3.2 Improving Token Efficiency in Best-of-N Sampling

We evaluate the verifier as a retry mechanism during inference-time scaling. As shown in Figure 2, the verifier achieves higher accuracy at lower values of k compared to self-consistency, and comparable accuracy at higher k, while requiring 1–3$\times$ fewer tokens. Here, k is the maximum number of candidate answers per task. In self-consistency, the final answer is chosen by majority vote over k generations. In verifier-based approach, two candidates are scored first; additional generations are produced only if both are deemed incorrect, continuing until a correct answer is found or k is reached.

Although trained on QWEN3-8B outputs, the 8B verifier effectively evaluates larger models. For example, on AIME 2025, incorporating the verifier with Qwen3-32B generations achieves 0.77 accuracy while using only 75% of the tokens compared to self-consistency.

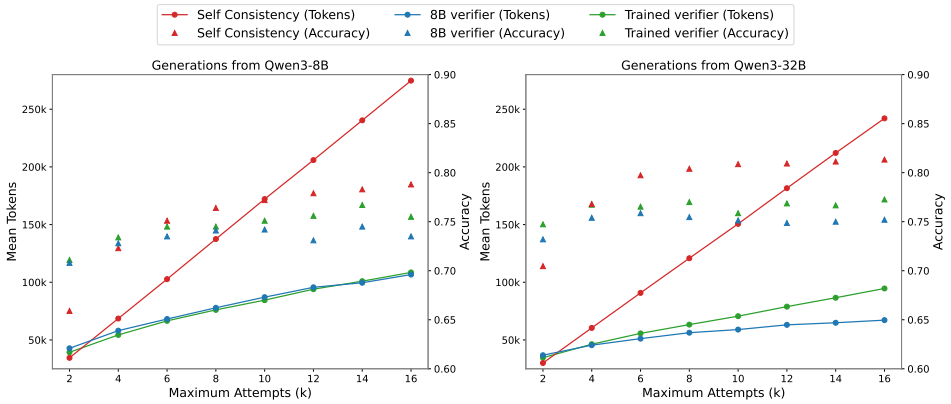

Figure 2: Across different values of k, the trained verifier consistently achieves higher accuracy than Qwen3-8B and provides a significant boost to the performance over self-consistency at lower k. Importantly, it requires substantially fewer tokens than self-consistency demonstrating both efficiency and stable performance across scales.

3.3 Improving Self-Reflection with Verifier-Guided Feedback

Beyond judging correctness, the verifier's natural language reasoning provides valuable feedback for iterative self-reflection (Table 1). We generate two candidate answers with QWEN3-8B, score them with the verifier, and produce a final answer guided by its reasoning. This approach improves accuracy by up to 6 percentage points on AIME 2024 and 2025 compared to generating an answer using simple consistency check. Notably, gains are larger when using GPT-OSS-20B to generate final answer, likely due to its stronger instruction-following and reasoning. Beyond math, verifier feedback also boosts coding performance, demonstrating early applicability to this growing domain.

Table 1: Accuracy of models with and without feedback from a verifier.

Model	**AIME 2024**		**AIME 2025**		**LCB v6** (01/01 - 08/01)	
	W/o Verifier	W/ Verifier	W/o Verifier	W/ Verifier	W/o Verifier	W/ Verifier
Qwen3-8B	0.77	0.77	0.67	0.68	0.39	0.44
GPT-20B	0.77	0.80	0.65	0.71	0.49	0.49
Qwen3-32B	0.77	0.76	0.67	0.69	–	–

3.4 Emergent Generative Capabilities of the Verifier

A particularly promising finding is the emergent generative capability of the Explanatory Verifier. In a single-shot generation comparison against the baseline model, the verifier achieves statistically similar pass@1 accuracy (Table 2). This indicates that the intensive training for critical evaluation does not degrade, and possibly even enhances, the model's core reasoning abilities. This result paves the way for a new training paradigm that co-optimizes reasoning models for generating diverse solutions while accurately verifying their correctness, making them better suited for test-time scaling.

Table 2: Performance on AIME 2024 and 2025 at $n\_repeat = 15$

Benchmark	**Qwen3-8B**	**Verifier**
AIME 2024	0.77	0.78
AIME 2025	0.66	0.68

4 Conclusion And Future Work

In this work, we demonstrated that test-time strategies can be significantly enhanced by using an Explanatory Verifier, trained with reinforcement learning, to overcome the core self-evaluation bottleneck of reasoning models. This approach opens promising avenues for future research, from training verifiers for test-time strategies using natural language feedback to the holistic co-design of integrated generator-verifier models. Ultimately, our work is a foundational step towards the next generation of AI systems: efficient, agentic systems where multi-faceted models can autonomously tackle problems of increasing complexity.

References

[1] OpenAI. Openai o3 model documentation, 2025. URL https://platform.openai.com/docs/models/o3. Accessed: 2025-09-08.

[2] OpenAI. Introducing deep research, 2025. URL https://openai.com/index/introducing-deep-research/. Accessed: 2025-09-08.

[3] Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence, 2025. URL https://arxiv.org/abs/2508.15260.

[4] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet, 2024. URL https://arxiv.org/abs/2310.01798.

[5] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

[6] Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. Inference-time scaling for generalist reward modeling, 2025. URL https://arxiv.org/abs/2504.02495.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN 0262035618.

[8] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

[9] Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. Inference-time scaling for generalist reward modeling. *arXiv preprint arXiv:2504.02495*, 2025.

[10] 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-1.5](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024.

[11] Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. Taco: Topics in algorithmic code generation dataset. *arXiv preprint arXiv:2312.14852*, 2023.

[12] Yunhui Xia, Wei Shen, Yan Wang, Jason Klein Liu, Huifeng Sun, Siyue Wu, Jian Hu, and Xiaolong Xu. Leetcodedataset: A temporal dataset for robust evaluation and efficient training of code llms, 2025. URL https://arxiv.org/abs/2504.14655.

[13] Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

[14] Yang Chen, Zhuolin Yang, Zihan Liu, Chankyu Lee, Peng Xu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acereason-nemotron: Advancing math and code reasoning through reinforcement learning. *arXiv preprint arXiv:2505.16400*, 2025.

[15] Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. Reinforcement learning for reasoning in large language models with one training example, 2025. URL https://arxiv.org/abs/2504.20571.

[16] Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D. Goodman. Quiet-star: Language models can teach themselves to think before speaking, 2024. URL https://arxiv.org/abs/2403.09629.

[17] Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Quentin Carbonneaux, Taco Cohen, and Gabriel Synnaeve. Rlef: Grounding code llms in execution feedback with reinforcement learning. *arXiv preprint arXiv:2410.02089*, 2024. URL https://arxiv.org/abs/2410.02089.

[18] OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen,

Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

[19] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

[20] Kusha Sareen, Morgane M Moss, Alessandro Sordoni, Rishabh Agarwal, and Arian Hosseini. Putting the value back in rl: Better test-time scaling by unifying llm reasoners with verifier. *arXiv preprint arXiv:2505.04842*, 2025. URL https://arxiv.org/abs/2505.04842.

[21] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

[22] Xinyan Guan, Yanjiang Liu, Xinyu Lu, Boxi Cao, Ben He, Xianpei Han, Le Sun, Jie Lou, Bowen Yu, Yaojie Lu, and Hongyu Lin. Search, verify and feedback: Towards next generation post-training paradigm of foundation models via verifier engineering, 2024. URL https://arxiv.org/abs/2411.11504.

[23] Fei Yu, Yingru Li, and Benyou Wang. Scaling flaws of verifier-guided search in mathematical reasoning, 2025. URL https://arxiv.org/abs/2502.00271.

[24] Xuzhao Li, Xuchen Li, Shiyu Hu, Yongzhen Guo, and Wentao Zhang. Verifybench: A systematic benchmark for evaluating reasoning verifiers across domains, 2025. URL https://arxiv.org/abs/2507.09884.

[25] Ding Chen, Qingchen Yu, Pengyuan Wang, Wentao Zhang, Bo Tang, Feiyu Xiong, Xinchi Li, Minchuan Yang, and Zhiyu Li. xverify: Efficient answer verifier for reasoning model evaluations, 2025. URL https://arxiv.org/abs/2504.10481.

[26] Zhangchen Xu, Yuetai Li, Fengqing Jiang, Bhaskar Ramasubramanian, Luyao Niu, Bill Yuchen Lin, and Radha Poovendran. Tinyv: Reducing false negatives in verification improves rl for llm reasoning, 2025. URL https://arxiv.org/abs/2505.14625.

[27] Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards, 2025. URL https://arxiv.org/abs/2502.01456.

[28] Ryan Marten, Trung Vu, Charlie Cheng-Jie Ji, Kartik Sharma, Shreyas Pimpalgaonkar, Alex Dimakis, and Maheswaran Sathiamoorthy. Curator: A tool for synthetic data creation. https://github.com/bespokelabsai/curator, January 2025.

[29] 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, 2023. URL https://arxiv.org/abs/2309.06180.

[30] Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. Skywork open reasoner 1 technical report. *arXiv preprint arXiv:2505.22312*, 2025.

[31] Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Yang Liu, and Yahui Zhou. Skywork open reasoner series. https://capricious-hydrogen-41c.notion.site/Skywork-Open-Reaonser-Series-1d0bc9ae823a80459b46c149e4f51680, 2025. Notion Blog.

[32] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, Amit Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Fransesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrmam, and Anthony Scopatz. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3, January 2017. doi: 10.7717/peerj-cs.103. URL `https://doi.org/10.7717/peerj-cs.103`.

[33] Hynek Kydlíček. Math-Verify: Math Verification Library. URL `https://github.com/huggingface/math-verify`.

A Related Work

A.1 RLVR

Reinforcement learning with verifiable feedback(RLVR) has emerged as a commonly used paradigm for aligning LLMs with task-specific goals such as problem-solving or code-generation. It [5, 15] can be seen as a simplified form of bootstrapping LM reasoning [16] or a simpler form of RL with execution feedback [17], in which one uses answer matching or constraint verification as a binary signal to train large language models. RLVR allows an LLM to learn to reason better, especially in verifiable domains such as mathematics and programming.

A.2 Scaling Test-Time Compute

Test-time compute scaling improves the performance of language models by allocating additional inference budget combining strategies such as search and voting. These approaches increase the diversity of candidate answers and, when combined with majority voting or verifier models, allow systems to reason for longer and arrive at more accurate solutions. This paradigm enables smaller open models to match or even surpass larger ones, as demonstrated in OpenAI's o1 [18]. Test-time scaling also integrates naturally with RLVR, since stronger verifiers directly improve inference-time selection [19–21].

Verifiers [22] play a central role in these scaling-test time strategies, guiding inference and post-training optimization with scalable, generalizable supervision signals. However, as sample sizes grow, the effectiveness of verifier-guided approaches can decline [23] due to misranking and pruning errors, particularly on challenging or out-of-distribution tasks, highlighting the need for improved calibration and hybrid candidate selection for robust scaling [23, 24].

A.3 Efficient Verifiers

Several recent approaches explicitly target efficiency of the verifier used in RLVR, Chen et al. [25] introduces a small verifier (0.5B–3B params) that rivals closed source models like GPT-4o in answer equivalence detection across 10+ LLMs and datasets. Building on this, Xu et al. [26] reduces false negatives in rule-based verification by training a compact LLM verifier on false negatives and positives, enabling more accurate reward estimates while maintaining computational efficiency when used in tandem with rule-based methods like Cui et al. [27]. Other methods improve data efficiency by training on feedback-rich rationales or by co-training verifiers with generators in reinforcement learning frameworks [20, 27]. Our verifier extends on this work by demonstrating that small yet robust RL verifiers can deliver high calibration and accuracy while reducing computational cost.

B Dataset Preparation

LLM Response Generation We use the Curator package from Bespoke Labs for high-throughput generation of LLM responses to these questions, by sending asynchronous inference requests [28] to an online inference server from vLLM [29]. This combination of tools handles continuous batching and kernel optimizations to maximize GPU utilization.

Initial dataset selection Our dataset is prepared starting with the math and coding subsets from the Skywork-OR1-RL-Data dataset [30, 31], particularly the subsets sourced from NuminaMath [10], TACO [11] and LeetCode [12].

Existing Dataset and Software Issues Though these datasets are meant to be easily verifiable, we still find issues in the data when trying to check correctness. The Math-Verify package by default returns False when there is an internal error in the correctness check, rather than exposing the error. This is misleading, as a False correctness check should indicate that the model had a valid response that was different than a valid ground-truth. Instead we find that some of the errors are because Math-Verify cannot handle certain forms of equations found in the dataset. We adjusted the implementation to account for this and removed examples that created errors.

NuminaMath is also a noisy dataset, as it is created through optical character recognition (OCR) on pdf files of heterogeneous formats at scale. We refer to the Skywork Open Reasoner Technical Report for full details of their initial filtering, but it generally involved heuristics to remove proof-based and other open-ended questions; removed questions with 0% or 100% success-rate from a moderate-sized model; and deduplicated questions based on embedding similarity. They also perform LLM-as-a-judge quality assessment of questions with LLama-3.3-70B-Instruct and Qwen2.5-72B-Instruct, with 16 samples per model, and retain only questions with >9 positive judgments on a binary scale.

Additional Filtering Contributions To further increase the quality of correctness signal provided to our verifier, we perform additional filtering. We eliminate questions that have multiple answers, as the comparison becomes more challenging. We remove questions that have answers in multiple choice format (such as A, B, C or D), as we find that the dataset is inconsistent where the question asks for the multiple choice letter, but contains as ground-truth the *option* text from that choice, and vice-versa. We finally use sympy [32] to parse the ground-truth answer and determine if the answer evaluates to a single number. This checks that there are no free symbols or undefined functions; floats and similar are retained.

The coding datasets are cleaner, and we simply use the Skywork version which checked that all test cases pass in the original solutions, and remove samples with empty, incomplete, or corrupted test cases.

Final Verification Methods For final math correctness checking, we use the Math-Verify package [33], where the ground-truth answer is formatted into a latex environment before parsing, and extract response from within \boxed{} delimiters for comparison (we request the use of \boxed{} in the prompt). For code, we execute the test cases from the dataset on the generated code in a sandbox environment [28]. All test cases must pass for a code sample to be deemed correct.

C Calibrated Ratings

As illustrated in Figure 3, the ratings produced by the trained verifier exhibit strong alignment with calibrated confidence estimates. The left panel reports results from the baseline QWEN3-8B model, where ratings are overwhelmingly concentrated at the maximum value of 10, largely independent of response correctness. The reference black line denotes the expected proportion of correct responses for a perfectly calibrated verifier, whereas the red line reflects empirical accuracy within each rating bucket. The flat red line at approximately 50% demonstrates that the baseline model offers little discriminatory power across rating levels. By contrast, the right panel shows the trained verifier, which distributes ratings more broadly across the scale and, crucially, yields accuracy curves (red line) that track much more closely with the calibration reference (black line). This indicates that training substantially improves the reliability and interpretability of verifier scores as confidence measures.

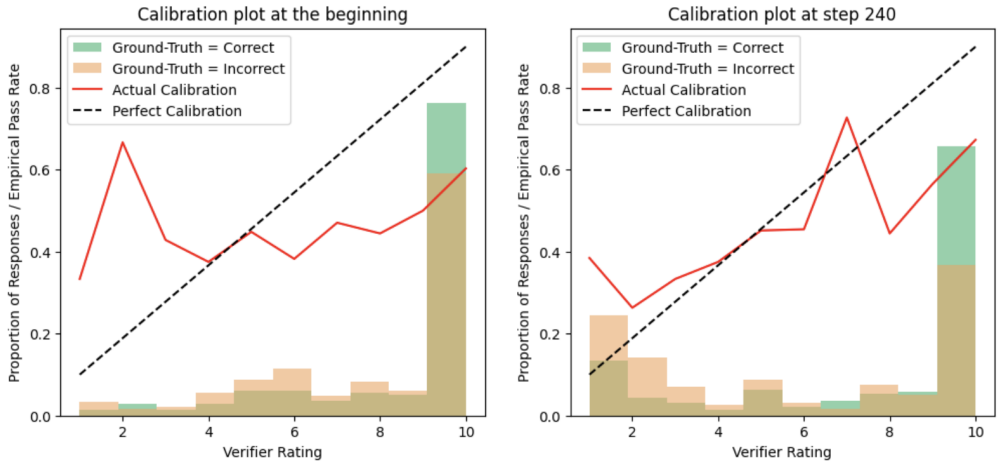

Figure 3: **Left**: baseline QWEN3-8B, where ratings are skewed toward 10 and show little correlation with correctness. **Right**: trained verifier, which produces a broader spread of ratings and improved calibration, with accuracy (red) aligning more closely with the ideal (black).

D Repeated Incorrect answers

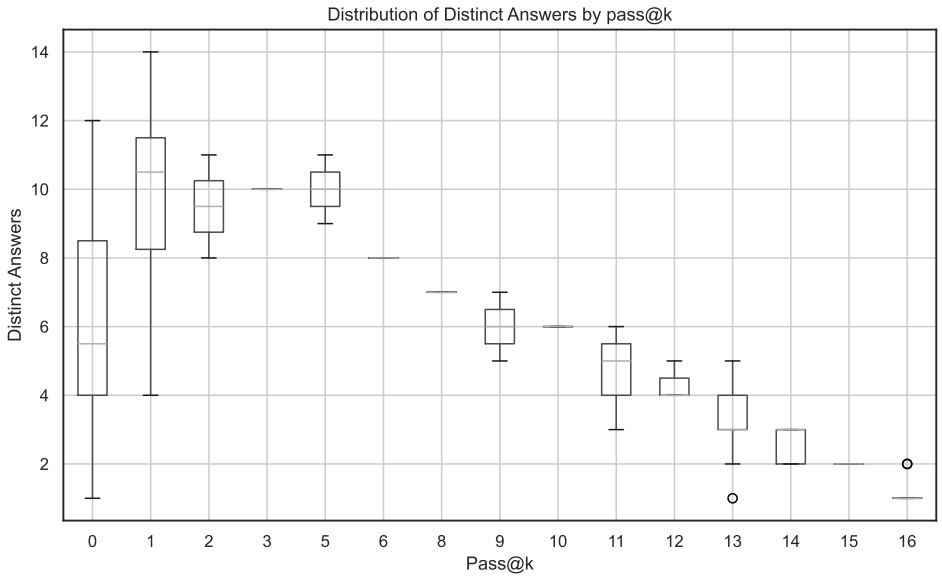

Figure 4: **Model Bias Analysis** This figure illustrates the response diversity of QWEN3-8B over 16 generation attempts on problems of varying difficulty. For moderately complex problems (pass@k > 5), the model produces mostly correct solutions or diverse incorrect solutions (Y-axis count). In stark contrast, when faced with a difficult problem that yields no correct answers (pass@16 == 0), the model repeatedly generates the same incorrect solution, demonstrating a collapse into a narrow, biased failure mode.