When Does Verification Pay Off? A Closer Look at LLMs as Solution Verifiers

Jack Lu*, Ryan Teehan*, Jinran Jin, and Mengye Ren
Agentic Learning AI Lab, New York University
{yl11330, rst306, jj3007, mengye}@nyu.edu
https://agenticlearning.ai/llm-verification
November 20, 2025

Abstract

Large language models (LLMs) can act as both problem solvers and solution verifiers, where the latter select high-quality answers from a pool of solver-generated candidates. This raises the question of under what conditions verification pays off in solver–verifier systems. Prior work has conducted only limited studies of the factors influencing verification performance, focusing primarily on self-verification and examining neither the relationship between solver and verifier model families nor the effects of reasoning post-training. To rectify this, we present a systematic study across 37 models spanning multiple families, sizes, and base vs. post-trained variants, evaluated on 9 benchmarks covering logical reasoning, structured puzzles, symbolic computation, mathematics, commonsense, factual recall, and domain knowledge. In order to support our analysis, we introduce and empirically validate *verifier gain*, a metric that predicts the performance improvements from *test-time verifier-based rejection sampling*. Our experiments find that 1) verification across model families is more effective than either self-verification or verification within the same family, and more generally that the benefits of verification decrease as the solver and verifier become more similar, 2) reasoning post-training weakens self-improvement abilities but strengthens cross-family improvement, and 3) some tasks are inherently more amenable to improvement through verification, particularly mathematical and logical tasks.

1 Introduction

Problem-solving with LLMs has progressed beyond querying a standalone model for a solution to a system where generated solutions are verified by other models. Test-time solution verification is integral to a variety of concrete approaches, including simple strategies, such as filtering multiple generated candidates (Zhao et al., 2025), as well as more complex strategies, such as iterative refinement (Madaan et al., 2023). With test-time solution verification, LLMs can solve more complex problems than when used alone (Cobbe et al., 2021; Lightman et al., 2024). This is particularly true in real deployment settings, when models encounter new, verifiable reasoning questions which they must solve without access to ground-truth answers.

Despite the ubiquity of solution verification, studies of solver–verifier interactions have remained limited in scope. Prior work has largely examined how a single model verifies its own solutions (self-verification) and

* Equal contribution.

improves its own solutions (self-improvement) (Song et al., 2025). However, self-verification is not guaranteed to be effective. Models may be biased toward their own reasoning patterns, and their training may reinforce these tendencies. Additionally, this focus on self-verification offers little insight into verification performance when the solver and verifier differ. We therefore broaden our analysis to additionally include both intra-family and cross-family verification and ask the following central question:

> *When does verification actually pay off, and how do factors such as model family, model size, reasoning post-training, solver–verifier similarity, and task type influence its effectiveness?*

To accomplish this, we evaluate verifiers across a diverse suite of tasks, including synthetic tasks used to test precise logical reasoning or symbolic computation (3SAT, Sudoku, and Matrix Multiplication), mathematical reasoning tasks (AIME (Mathematical Association of America, 2025), GSM8K (Cobbe et al., 2021)), commonsense and factual reasoning (CSQA (Talmor et al., 2019), GPQA (Rein et al., 2024)), and broad domain knowledge (MMLU in STEM and in social sciences (Hendrycks et al., 2021)) using 37 models from 7 model families. The tasks we chose provide ground-truth labels needed to rigorously measure verifier quality, while also capturing a broad range of skills required in real deployment settings. Using open-source model families that have base and post-trained pairs, size variants, reproducible inference pipelines, and explicit reasoning traces, we can study verification systematically.

We find that self-verification does not always "pay off." Models often favor solutions resembling their own reasoning (Sections 5.2 and 5.3) and reasoning post-training can sharpen this bias (Section 5.4), harming performance during self-verification or intra-family verification. Yet, this same bias makes stronger models more effective as cross-family verifiers, where the solver's distribution differs from their own. Additionally, we find that some tasks inherently benefit less from verification than others (Section 5.5). Therefore, we present the following contributions, which offer actionable and empirically supported guidance for how to use verifiers effectively.

- **New Metric: Verifier Gain.** Verifier accuracy alone provides an incomplete picture of verifier usefulness at test time. To address this, we derive *verifier gain*, a metric that simulates the improvement obtained from a verifier during test-time rejection sampling. We empirically study rejection sampling with verifiers and show that this theoretical formulation closely reflects empirical performance trends.
- **Self-Improvement, Intra-Family Improvement, and Cross-Family Improvement.** We find that cross-family verification is often more beneficial than intra-family verification or self-verification, comparing particularly favorably to the latter. Looking deeper, we show that the verifier gain decreases as the solution distributions of solver and verifier become more similar. Furthermore, our results suggest that as models become stronger, whether through increased scale, post-training, or simply higher solver accuracy, they become less effective as self-verifiers and more effective as cross-family verifiers.
- **Dataset Verifiability.** We study whether tasks that are easy to solve are also easy to verify and whether some tasks are inherently more verifiable than others. We find that verification accuracy generally correlates with solver accuracy, though self-verification yields little verifier gain across all tasks. We also observe that a clear subset of tasks involving mathematical or logical reasoning consistently produces higher verifier gains.

To the best of our knowledge, our work is the first systematic study of solver–verifier interactions across self-, intra-family, and cross-family regimes, spanning both base and reasoning post-trained models. Additionally, we introduce and validate verifier gain as a lightweight predictor of rejection-sampling improvements, examine the effects of reasoning post-training on verification, and connect observed differences in verification performance to output-distribution similarity between solver and verifier.

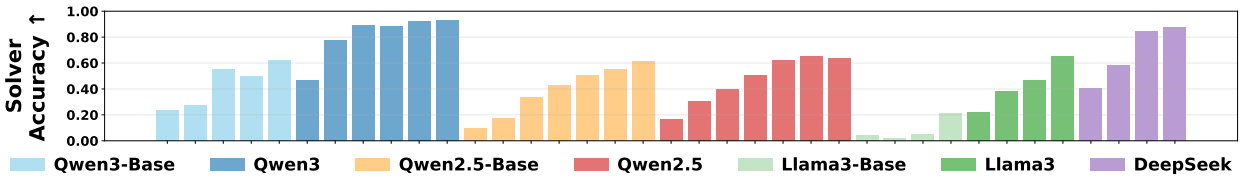


Figure 1: *Average solver accuracy of each model over all datasets. Base model families are suffixed by* ***-Base****. Models within each family are ordered in increasing size. We show information for each evaluated model in Table 1.*

2 Related Work

Verifiers. Weng et al. (2023), Wu et al. (2024), and Jiang et al. (2024) develop outcome-level self-verification methods by predicting parts of the question conditioned on the solution. To reduce hallucinations, Dhuliawala et al. (2024) have language models fact-check their own generations by generating fact-check questions. Researchers have also trained general-purpose outcome-level verifiers (Hosseini et al., 2024; Zhang et al., 2025; Cobbe et al., 2021) and value models (Yu et al., 2024), either independently or jointly with the solver (Shen et al., 2021; Sareen et al., 2025). Unlike these trained outcome reward models, our work studies off-the-shelf LLMs prompted to verify solutions, which requires no task-specific training data or finetuning. Song et al. (2025) investigate the performance improvement caused by using an outcome-level verifier (the GV-Gap), and how this improvement changes as the solver or verifier increases in capacity. However, they primarily focus on cases where the solver and verifier are the same model. Additionally, they only study base models and do not consider post-trained models in their analysis. We finally note concurrent work by Zhou et al. (2025), which also studies the factors that influence test-time verification. In contrast to our work, they focus on the effects of problem difficulty and generator capability, and do not investigate the effects of model family or reasoning post-training. Finally, verifiers also have their limitations, potentially producing false positives (Stroebl et al., 2024), eliminating valid reasoning paths (Yu et al., 2025), and failing to select the right solution (Brown et al., 2024).

Scaling test-time compute. A simple method for scaling test-time compute involves sampling several candidates and selecting one *post hoc*, for example via Best-of-N. This can take the form of sample-and-rank approaches (Nichols et al., 2020), majority vote (Wang et al., 2023), model-based aggregation (Chen et al., 2024), or sampling then filtering (Weng et al., 2023). Recent work has focused on studying scaling test-time compute with verifiers. For example, Zhao et al. (2025) study random sampling with self-verification, Chen et al. (2025) study combining parallel sampling with self-correction, and Singhi et al. (2025) compare Self-Consistency (Wang et al., 2023) to scaling with a generative verifier. Snell et al. (2025) investigate compute-optimal approaches to test-time scaling with process-level verification.

Self-improvement. Researchers have also studied LLM self-improvement and self-evaluation, with some voicing skepticism (Huang et al., 2024b; Kamoi et al., 2024; Olausson et al., 2023; Panickssery et al., 2024; Stechly et al., 2025). On the other hand, recent work has provided a theoretical framework for self-improvement via distribution sharpening, along with empirical support (Huang et al., 2024a). Zhang et al. (2024) look specifically at self-improvement for small models, arguing that they need to be paired with a stronger verifier. Some practical methods for self-improvement use natural language feedback (Madaan et al., 2023; Shinn et al., 2023; Kim et al., 2023) or train models for self-correction explicitly (Welleck et al., 2023). Other methods use tools (Gou et al., 2024), particularly code interpreters (Zhou et al., 2024), to iteratively improve solutions.

3 Preliminaries

In this section, we establish the framework used throughout this work. We define datasets, solvers, and verifiers, introduce the metrics used to evaluate solver and verifier behaviors, and specify the verification settings in our empirical analysis.

3.1 Datasets, Solvers, and Verifiers

Let $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}^\star$ be a dataset of pairs $(x, \mathcal{Y}_x)$, where $x \in \mathcal{X}$ is a problem and $\mathcal{Y}_x \subseteq \mathcal{Y}$ is a non-empty set of correct solutions. A solver $S : \mathcal{X} \to \mathcal{Y}$ is an LLM that produces a solution y for a given problem x, and a verifier $V : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ is an LLM that evaluates a problem–solution pair and returns a binary judgment. Following Song et al. (2025), who find chain-of-thought (CoT) verification more stable than multiple-choice formats, we instruct both solvers and verifiers to generate CoT reasoning before producing their final answers and judgments. We define the correctness indicator as $c(x, y) = \mathbb{1}\{y \in \mathcal{Y}_x\}$.

3.2 Evaluation Metrics

We define the accuracy of a solver S on a dataset $\mathcal{D}$ as the expected correctness of its outputs over all problems in the dataset: $\mathbb{E}_{(x,\mathcal{Y}_x)\sim\mathcal{D},\, y\sim S(x)}\big[\, c(x, y) \,\big]$. Verifier performance has several dimensions. We report the verifier accuracy, F1-Score, and precision (whose definitions are included in Appendix A for reference) for our verification settings. In addition, we compute the true positive rate (TPR), false positive rate (FPR), and false negative rate (FNR):

$$\begin{aligned}
\mathrm{TPR}(S, V; \mathcal{D}) &= \mathbb{E}[\, V(x, y) \mid y \in \mathcal{Y}_x \,] \\
\mathrm{FPR}(S, V; \mathcal{D}) &= \mathbb{E}[\, V(x, y) \mid y \notin \mathcal{Y}_x \,] \\
\mathrm{FNR}(S, V; \mathcal{D}) &= \mathbb{E}[\, 1 - V(x, y) \mid y \in \mathcal{Y}_x \,] = 1 - \mathrm{TPR}(S, V; \mathcal{D}).
\end{aligned}$$

Our primary goal is to evaluate whether using a verifier V can improve a solver S at test time via rejection sampling, where solver outputs are repeatedly sampled until the verifier accepts one. Assuming the solver has a non-zero probability of sampling a correct solution, in the limit of infinite resampling, the expected correctness of the accepted solution converges to the verifier's precision, or the proportion of accepted solutions that are actually correct. To quantify the improvement from combining a solver with a verifier, we define **verifier gain**:

$$\mathrm{Gain}(S, V; \mathcal{D}) = \mathrm{Precision}(S, V; \mathcal{D}) - \mathrm{SolverAcc}(S; \mathcal{D}). \tag{1}$$

This is a simple, yet useful, metric because it quantifies the improvement in correctness induced by filtering with the verifier. TPR/FPR describe the verifier in isolation; solver accuracy describes the generator in isolation, but verifier gain measures the benefit of the combined system. Since this is an asymptotic metric, it serves as a bound on the improvement attainable by verifier-based rejection sampling. Throughout this work, we use verifier gain to compare differences in verification behavior.

3.3 Verification Settings

We group models into families (e.g., `Llama3`, `Qwen2.5`), where each family contains related models of varying sizes. Because base and post-trained models often exhibit substantially different behaviors, we treat them as distinct families. For example, the base model `meta-llama/Llama-3.1-70B` belongs to the `Llama3-Base` family and the post-trained model `meta-llama/Llama-3.1-8B-Instruct` belongs to the `Llama3` family. We study three different verification settings, defined by the solver-verifier relationship:

- **Self-Verification.** The solver and verifier are the same model, so the model verifies its own solutions. For example, when a 70B `Llama3` model is used as both the solver and the verifier, the verification metric (e.g., accuracy, FPR) is computed on this single pairing.
- **Intra-Family Verification.** The verifier evaluates solutions produced by other models within the same family. For example, a 70B `Llama3` verifier may evaluate outputs from 8B or 13B `Llama3` solvers. The reported metric is averaged over all such within-family solvers, excluding the self-verification case.
- **Cross-Family Verification.** The verifier evaluates solutions from models of different families. For example, a base `Llama3` verifier may evaluate outputs from `Qwen3` or from a post-trained `Llama3`. The reported metric is averaged over all such cross-family solvers.

During our evaluation, we partition our computed verifier metrics according to the verification setting and average within each partition.

4 Experimental Setup

Models. We evaluate the solver and verifier abilities of 21 post-trained models from the `Llama3` (Grattafiori et al., 2024), `Qwen2.5` (Qwen et al., 2024), `Qwen3` (Yang et al., 2025), and `DeepSeek-R1` (DeepSeek-AI et al., 2025) families. For our study of reasoning post-training effects in Section 5.4, we additionally evaluate 12 base models from the `Qwen2.5-Base` and `Qwen3-Base` families. We specifically choose off-the-shelf, general-purpose models so that each model can serve as both a solver and a verifier, allowing controlled comparison across these roles. Model sizes range from 0.5B to 72B parameters. The full model list, with sizes, families, and HuggingFace identifiers, is provided in Appendix B. The legend of Figure 1 displays the seven model families and the color scheme assigned to each.

Datasets. We compile a broad suite of real-world and synthetic tasks spanning diverse domains, including tasks requiring mathematical reasoning (GSM8K, AIME), commonsense knowledge (CSQA), and domain-specific factual knowledge of varying breadth (MMLU (STEM), MMLU (Social Sciences), and GPQA). We also construct synthetic tasks to assess logical reasoning (3SAT), structured puzzle solving (Sudoku), and symbolic computation (Matrix Multiplication). Further details with synthetic task examples are provided in Appendix C.

Evaluation. Datasets like Matrix Multiplication and the natural-language benchmarks contain a single ground-truth answer per problem, so we extract boxed solver outputs and evaluate via exact matching. In contrast, datasets like Sudoku and 3SAT may admit multiple valid solutions, so we evaluate solver outputs according to each task's rules. To evaluate verifiers, we give each model the problem and solver answer to generate CoT reasoning before producing a boxed "correct" or "incorrect" as the final judgment.

Implementation. For both solvers and verifiers, we generate with temperature 0.7, top-p 0.9, and a maximum output length of 8192 tokens. We discard outputs that do not contain a boxed answer. All inference experiments are run using vLLM on H200 GPUs. Prompts and additional details on output filtering are provided in Appendix D.

Upon publication, we plan to open-source all experiment and data generation code.

5 Results

5.1 Does Verifier Gain Predict Improvements from Resampling?

We first empirically validate our verifier gain metric, which estimates the expected improvement in a solver's accuracy when using a verifier for rejection sampling. To assess how well this metric predicts real performance, we conduct rejection sampling experiments across all 12×12 solver–verifier pairs from a 12-model subset of our post-trained models, consisting of the three smallest models from each of the four post-training families. For each problem, the solver generates solutions until the verifier labels one as correct, up to ten attempts. If no such solution is found, we retain the final attempt.

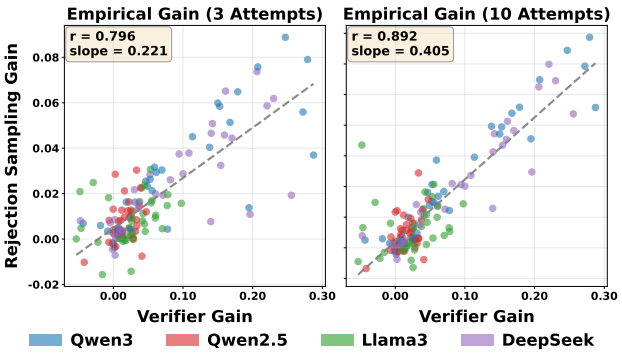


Figure 2: *Verifier gain (Equation 1) predicts empirical rejection sampling gain. Each point is one solver–verifier pair averaged across datasets, colored by verifier family.*

Figure 2 plots the empirical gain against the theoretical verifier gain for each pair, averaged across datasets. The Pearson correlation (r) strengthens considerably from 3 to 10 attempts, consistent with verifier gain being an asymptotic metric that better predicts performance as the resampling attempts grow.

> **Takeaway:** Verifier gain reliably predicts rejection sampling gains and serves as a powerful comparative metric for evaluating solver–verifier pairs. Crucially, it can be estimated from a single verification round without costly rejection sampling experiments.

5.2 Do Better Solvers Make Better Verifiers?

We next study the relationship between solver and verifier performance, and how to best measure the benefits of verification.

Solver performance. We benchmark all 37 models on each of our 9 datasets, averaging performance across tasks (Figure 1) and reporting task-level results in Appendix E. Overall, solver accuracy increases with model capacity. Models in the `Qwen3` and `DeepSeek` families perform particularly well, whereas `Llama3-Base` performs poorly due to base models being unfamiliar with the question–answering instruction format. Within each family, we observe clear performance scaling for `Qwen2.5-Base` and `DeepSeek`, with the remaining families showing similar upward trends.

Correlating verifier and solver performance. After establishing solver accuracy, we analyze whether a model's solver performance correlates with its performance as a verifier (Figure 3). For each of our 21 post-trained models and each dataset, we evaluate verification on the same set of solver models to obtain verifier accuracy, F1-Score, precision, FPR, FNR, and gain for every solver–verifier pair. For each verifier, we then partition the results

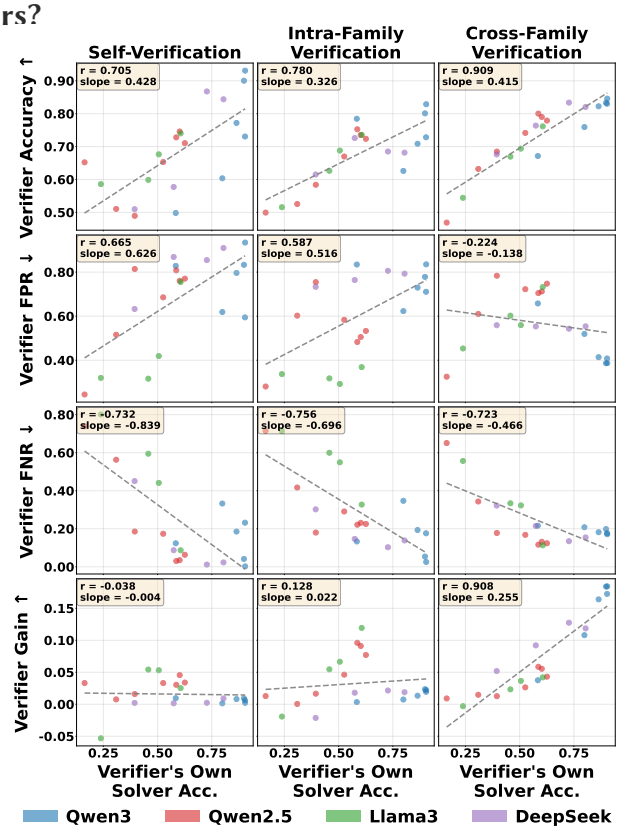


Figure 3: *Correlation between each verifier's metrics (rows) and its own solver accuracy for all 21 post-trained models, averaged over all datasets. Each metric is computed over our three verification settings (columns).*

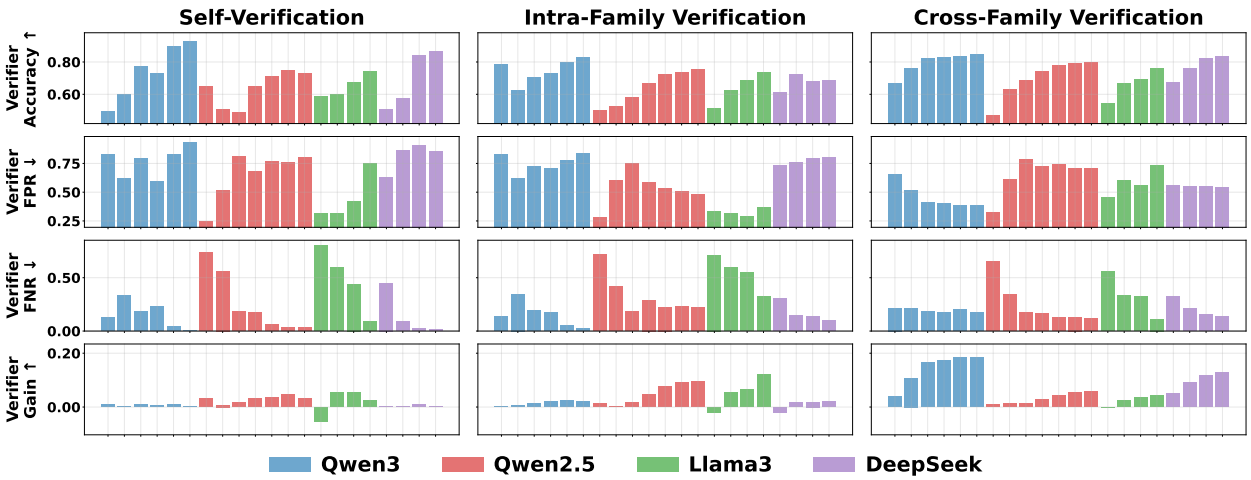


Figure 4: *We show each verifier's metric (rows) against model size for all 21 post-trained models, averaged over all datasets. Models are separated by family and ordered by increasing size. Each metric is computed over our three verification settings (columns).*

by our three verification settings and average within each setting over solvers and datasets.

While verifier accuracy tends to improve with the verifier's own solver accuracy, the relationship becomes more nuanced when examining other metrics. The FPR increases during self-verification and intra-family verification but decreases slightly during cross-family verification, indicating that when a strong solver is used as a verifier, it is more likely to falsely label a solution as correct if it was generated by itself or another model in its family. We provide additional visualizations for F1-Score and precision in Appendix F.

To better interpret the trends suggested by the accuracy and FPR results, we examine verifier gain in the final row. This visualization offers a clearer view of verification quality: self-verification yields the smallest gains, and more accurate solvers do not exhibit greater self-improvement. Gains increase slightly in intra-family verification, while cross-family verification provides the greatest potential benefits.

Examining verifier performance at different model families and sizes. In Figure 4, we repeat the experiments from Figure 3 but plot each verifier metric as a function of model size within each model family. Verification accuracy and FNR consistently improve as models become larger, whereas FPR behaves more inconsistently, often increasing with model size (e.g., intra-family verification for `DeepSeek`).

For state-of-the-art reasoning post-trained models such as `Qwen3` and `DeepSeek`, we find that verifier gains are largest in the cross-family setting, smaller in the intra-family setting, and minimal during self-verification. At first glance, this appears to contradict Song et al. (2025), who report that self-verification GV-Gaps increase with more pre-training FLOPs. However, their analysis focuses on older model families, and Figure 3 likewise shows larger verifier gains for older models such as `Qwen2.5` and `Llama3`.

We hypothesize that stronger reasoning post-trained models like `DeepSeek` and `Qwen3` show negligible gains in self-verification and limited gains in intra-family verification because they may already engage in *spontaneous* self-verification when used as solvers, reducing the benefit of an additional *forced* verification round. To test this, we measure the spontaneous self-verification rate of all 21 post-trained models across all 9 datasets by scanning solver outputs for self-verification keywords (e.g., "let me check," "wait," "reconsider," "that's wrong"; full list in Appendix G). Indeed, `Qwen3` and `DeepSeek` self-verify in 96% and 73% of outputs, respectively, while `Llama3` and `Qwen2.5` self-verify in only 1–2%. This aligns with our hypothesis: models that already spontaneously self-verify during solving leave little room for an additional verification pass

to add value. More broadly, it suggests the relevant question is whether reasoning post-training introduces spontaneous self-verification.

Takeaways:

1. Verifier models are biased toward accepting incorrect solutions when performing self-verification or intra-family verification.
2. Verification accuracy alone is not a reliable predictor of how much a verifier can improve a solver at test time. Instead, computing verifier gain using solver accuracy and verifier precision provides a more reliable metric.
3. While model families like `Llama3` and `Qwen2.5` show some ability to self-improve, stronger model families like `DeepSeek` and `Qwen3` do not, which we find is linked to the latter already spontaneously self-verifying during solving (73–96% vs. 1–2%).

5.3 Are Verifiers Biased Toward Solutions That Resemble Their Own?

Humans tend to judge solutions that resemble their own reasoning as more likely correct. This mirrors self-enhancement bias (Krueger, 1998), where individuals evaluate themselves more favorably than evidence suggests. Our results in Figures 3 and 4 show that strong reasoning models benefit the least from self-verification and the most from cross-family verification, suggesting an analogous effect in solver–verifier interactions.

To directly investigate this, we study the relationship between verifier performance metrics and the **solver–verifier similarity score**, defined as the average cosine similarity between two models' solution embeddings across all dataset problems. We conduct cross-verification experiments using 12 post-trained models (the three smallest from each of four families) and compute all verifier metrics for each pair. For intra-family verification, each solver has 2 same-family verifiers (excluding itself), yielding $12 \times 2 = 24$ pairs. For cross-family verification, each solver has 9 verifiers from other families, giving $12 \times 9 = 108$ pairs. Solutions are embedded using `sentence-transformers /all-mpnet-base-v2`.

Figure 5 shows that, for both intra-family and cross-family settings, the more similar the solver and verifier distributions, the more likely the verifier is to accept incorrect answers. While intra-family verifier gains are too small to yield a strong correlation, cross-family gains decrease significantly as similarity increases, indicating that choosing a dissimilar verifier leads to more reliable verification. We repli-

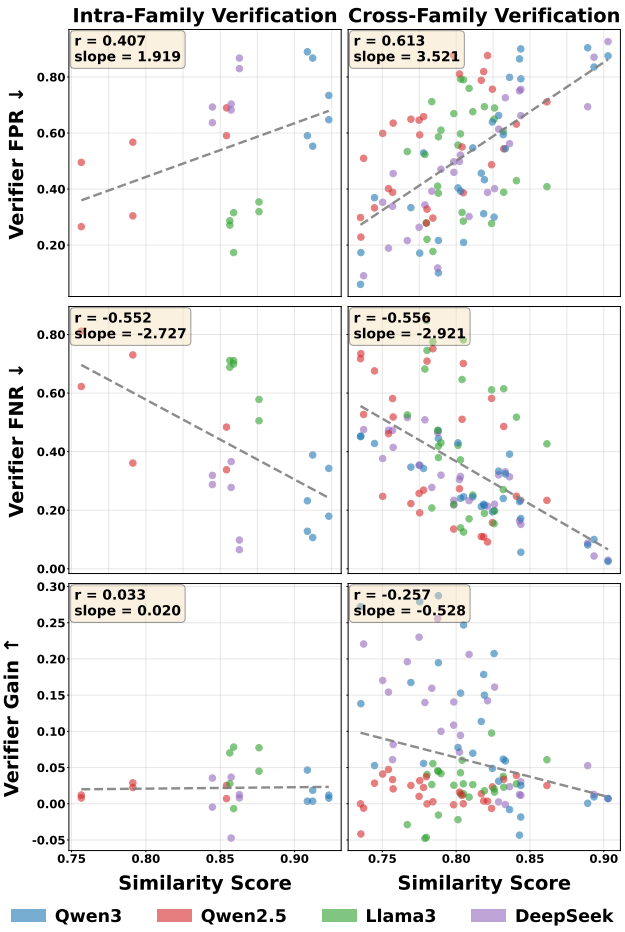


Figure 5: *Correlation between verifier metrics and similarity scores between solver-verifier pairs. Each marker is colored based on the verifier model family. Self-verification is omitted as it yields only one data point per model, too few for reliable correlation analysis.*

cate this analysis using log-likelihood as an alternative similarity metric with the same directional findings, but we prefer cosine similarity as the primary metric because log-likelihood conflates distributional similarity with intrinsic predictability of the solver's text (see Appendix H for details).

> **Takeaway:** Higher similarity between solver and verifier solution distributions increases the verifier's tendency to accept incorrect solver outputs, reducing verifier gain. Using a verifier with a meaningfully different solution distribution mitigates this bias.

5.4 How Does Reasoning Post-Training Affect Solver and Verifier Performance?

We examine how post-training influences verifier behavior, focusing on the `Qwen2.5-Base`/`Qwen2.5` and `Qwen3-Base`/`Qwen3` pairs, since other families are either too weak (`Llama3-Base`) or lack base models (`DeepSeek`). As both `Qwen2.5` and `Qwen3` use GRPO for reasoning post-training, our analysis primarily concerns its effects. Verification metrics are computed across all 37 base and post-trained models.

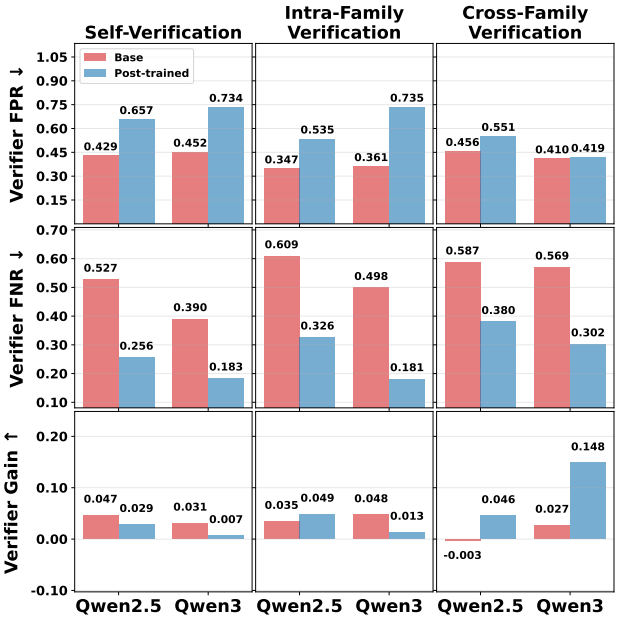


Figure 6: *Changes in verifier metrics of the* `Qwen2.5-Base` *and* `Qwen3-Base` *models from reasoning post-training.*

Reasoning post-training's effect on solver performance. We evaluate how solver accuracy changes after reasoning post-training, computing accuracy averaged across model families and datasets. As expected, post-training yields substantial improvements: `Qwen2.5` solvers improve by 8.2% on average, while `Qwen3` solvers show a striking 35.4% gain.

Reasoning post-training's effect on verifier performance. We next analyze how reasoning post-training affects verifier behavior (Figure 6). For each model, we compute verifier metrics against all solvers and datasets, partition by verification setting, and average within families. For both Qwen families, post-training increases FPR and reduces verifier gain in self-verification, despite improvements in FNR. Although `Qwen3` benefits more in solver accuracy (Figure 12), its post-trained verifiers show higher FPRs and lower gains in both self- and intra-family verification. In contrast, both families, especially `Qwen3`, show substantial improvements in cross-family verification. Overall, reasoning post-training exacerbates trends identified previously, increasing false positives and limiting self-verification benefits.

> **Takeaway:** Reasoning post-training significantly improves problem-solving but can reduce self- and intra-family verification gains, while boosting cross-family verification performance.

5.5 How Does Task Type Affect Verifiability?

Thus far, we have examined model-related factors. We now shift to a task-level perspective and ask: *are tasks that are easy to solve also easy to verify?* In Figure 7, we recompute the verifier metrics from Section 5.2, average across all verifier models, and plot them against solver accuracies. Solver accuracy and verifier accuracy correlate strongly in all settings, but the picture is more mixed for verifier gains. During self-verification, we find essentially no correlation between verifier gain and solver accuracy, but a clear

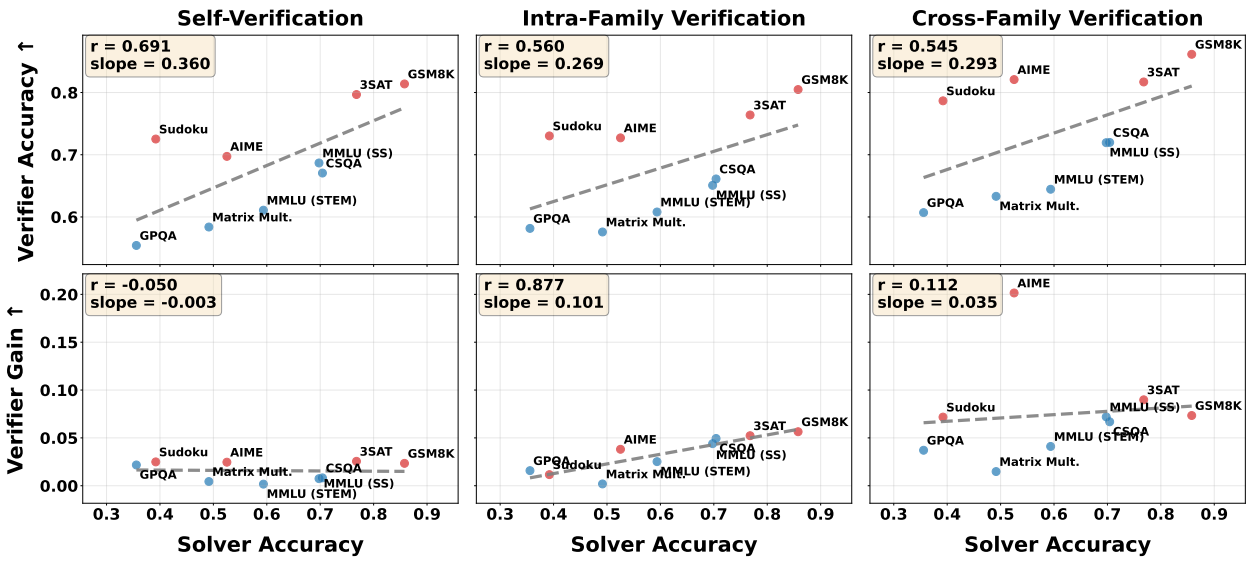


Figure 7: *Correlation of verifier metrics (rows) with solver accuracies, averaged over solver-verifier pairs that belong to each verification setting (columns).*

positive relationship emerges during intra-family and cross-family verification. Notably, AIME appears as an outlier, potentially because some models encountered similar problems during post-training.

The best-fit lines for verifier accuracy reveal two distinct task clusters (colored red and blue), leading us to ask: *are some tasks inherently easier to verify than others?* In Figure 7, AIME, GSM8K, 3SAT, and Sudoku exhibit a higher ratio of verifier accuracy to solver accuracy and deliver higher gains across all settings. Sudoku and 3SAT require exponential solving time but allow polynomial-time verification, whereas verifying a matrix product offers no such shortcut. Among real-world datasets, GSM8K and AIME involve high-school-level mathematics, whereas MMLU (Social Sciences) requires domain-specific knowledge, CSQA relies on world knowledge, and GPQA and MMLU (STEM) draw on specialized natural science knowledge. For these latter tasks, verifying an answer requires essentially the same knowledge as solving it, confirming and extending Song et al. (2025)'s finding that models cannot self-improve on factual recall tasks but can on Sudoku.

Takeaways:

1. Tasks that are easy to solve tend to be easy to verify.
2. Tasks that are easier to solve tend to be more improvable through intra-family and cross-family verification, but not necessarily through self-verification.
3. Synthetic problems with logical or structured reasoning, as well as real-world tasks which rely on mathematical reasoning, are inherently easier to verify, and yield larger verifier gains, than those which require factual recall.

6 Conclusion

This work presents a comprehensive study of LLM-based verification for problem solving. We show that verification accuracy alone provides an incomplete picture, motivating *verifier gain*, which measures the expected improvement from using a verifier for rejection sampling. We find lower verifier gains from self- and intra-family verification compared to cross-family, trends exacerbated by reasoning post-training

and increasing model size. Further analysis reveals that these decreases correlate with greater similarity between solver and verifier solution distributions, indicating that verifiers are biased to accept solutions resembling their own. We also find that tasks involving logical or mathematical reasoning are inherently easier to verify than knowledge-recall tasks.

Our results yield an actionable checklist for designing effective solver-verifier systems:

- Use verifier gain, not accuracy, to evaluate a solver-verifier pair. Verification accuracy can be misleading (Section 5.2), while verifier gain strongly predicts actual rejection sampling gains (Section 5.1).
- Check whether the task is easier to verify than to solve. Logical and mathematical reasoning tasks yield higher verifier gains than knowledge-recall tasks (Section 5.5).
- Prefer verifiers that "think differently" from the solver. Solution-distribution similarity increases false positives and reduces gains (Section 5.3).
- Avoid using strong reasoning models as their own verifiers due to their minimal self-verification gain from reasoning post-training (Sections 5.2, 5.4).

Limitations and future work. Section 5.3 shows that LLMs are biased toward accepting incorrect solutions that resemble their own reasoning, indicating that it will be worthwhile to examine the origins of this bias in pre-training and/or post-training. Additionally, while we note that our solver-verifier setting corresponds to a number of real deployment settings, multi-turn problem solving is an increasingly popular way to solve difficult problems with LLM agents. A promising avenue for future research is to study verification in the multi-turn setting, where future interactions incorporate verifier feedback and also depend on previous solver outputs in the conversation.

Acknowledgement

We thank members of the NYU Agentic Learning AI Lab for their helpful discussions. JL is supported by the NSERC PGS-D Scholarship. The work is supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) under grant RS-2024-00469482, funded by the Ministry of Science and ICT (MSIT) of the Republic of Korea in connection with the Global AI Frontier Lab International Collaborative Research. The compute is supported by the NYU High Performance Computing resources, services, and staff expertise. We also thank Modal for providing additional compute resources.

References

Brown, B., Juravsky, J., Ehrlich, R., Clark, R., Le, Q. V., Ré, C., and Mirhoseini, A. (2024). Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Chen, J., Ren, J., Chen, X., Yang, C., Sun, R., Yoon, J., and Arık, S. Ö. (2025). Sets: Leveraging self-verification and self-correction for improved test-time scaling. *arXiv preprint arXiv:2501.19306*.

Chen, X., Aksitov, R., Alon, U., Ren, J., Xiao, K., Yin, P., Prakash, S., Sutton, C., Wang, X., and Zhou, D. (2024). Universal self-consistency for large language models. In *ICML 2024 Workshop on In-Context Learning*.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. (2021). Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B.,

Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. (2025). Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dhuliawala, S., Komeili, M., Xu, J., Raileanu, R., Li, X., Celikyilmaz, A., and Weston, J. (2024). Chain-of-verification reduces hallucination in large language models. In *ACL Findings*.

Gou, Z., Shao, Z., Gong, Y., yelong shen, Yang, Y., Duan, N., and Chen, W. (2024). CRITIC: Large language models can self-correct with tool-interactive critiquing. In *ICLR*.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Wyatt, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Guzmán, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Thattai, G., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Zhang, J., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Prasad, K., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Lakhotia, K., Rantala-Yeary, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Tsimpoukelli, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Zhang, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Maheswari, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Albiero, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Wang, X., Tan, X. E., Xia, X.,

Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Srivastava, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Teo, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Dong, A., Franco, A., Goyal, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Liu, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Gao, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Le, E.-T., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Kokkinos, F., Ozgenel, F., Caggioni, F., Kanayet, F., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Inan, H., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Zhan, H., Damlaj, I., Molybog, I., Tufanov, I., Leontiadis, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Lam, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K. H., Saxena, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Jagadeesh, K., Huang, K., Chawla, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Liu, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Mehta, N., Laptev, N. P., Dong, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Parthasarathy, R., Li, R., Hogan, R., Battey, R., Wang, R., Howes, R., Rinott, R., Mehta, S., Siby, S., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Mahajan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay, S., Feng, S., Lin, S., Zha, S. C., Patil, S., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Deng, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Koehler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Ionescu, V., Poenaru, V., Mihailescu, V. T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wu, X., Wang, X., Wu, X., Gao, X., Kleinman, Y., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Zhao, Y., Hao, Y., Qian, Y., Li, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., Zhao, Z., and Ma, Z. (2024). The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. (2021). Measuring massive multitask language understanding. In *ICLR*.

Hosseini, A., Yuan, X., Malkin, N., Courville, A., Sordoni, A., and Agarwal, R. (2024). V-STar: Training verifiers for self-taught reasoners. In *COLM*.

Huang, A., Block, A., Foster, D. J., Rohatgi, D., Zhang, C., Simchowitz, M., Ash, J. T., and Krishnamurthy, A. (2024a). Self-improvement in language models: The sharpening mechanism. *arXiv preprint arXiv:2412.01951*.

Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X., and Zhou, D. (2024b). Large language models cannot self-correct reasoning yet. In *ICLR*.

Jiang, W., Shi, H., Yu, L., Liu, Z., Zhang, Y., Li, Z., and Kwok, J. (2024). Forward-backward reasoning in large language models for mathematical verification. In *ACL Findings*.

Kamoi, R., Zhang, Y., Zhang, N., Han, J., and Zhang, R. (2024). When can LLMs actually correct their own mistakes? a critical survey of self-correction of LLMs.

Kim, G., Baldi, P., and McAleer, S. (2023). Language models can solve computer tasks. In *NeurIPS*.

Krueger, J. (1998). Enhancement bias in descriptions of self and others. *Personality and Social Psychology Bulletin*.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. (2024). Let's verify step by step. In *ICLR*.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. (2023). Self-refine: Iterative refinement with self-feedback.

Mathematical Association of America (2025). 1983-2025 American Invitational Mathematics Examination (AIME) I: Problems and Solutions. Art of Problem Solving Wiki entry.

Nichols, E., Gao, L., and Gomez, R. (2020). Collaborative storytelling with large-scale neural language models. In *ACM SIGGRAPH Conference on Motion, Interaction and Games*.

Olausson, T. X., Inala, J. P., Wang, C., Gao, J., and Solar-Lezama, A. (2023). Is self-repair a silver bullet for code generation? *arXiv preprint arXiv:2306.09896*.

Panickssery, A., Bowman, S. R., and Feng, S. (2024). Llm evaluators recognize and favor their own generations.

Qwen, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. (2024). Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. (2024). GPQA: A graduate-level google-proof q&a benchmark. In *COLM*.

Sareen, K., Moss, M. M., Sordoni, A., Agarwal, R., and Hosseini, A. (2025). Putting the value back in rl: Better test-time scaling by unifying llm reasoners with verifiers. *arXiv preprint arXiv:2505.04842*.

Shen, J., Yin, Y., Li, L., Shang, L., Jiang, X., Zhang, M., and Liu, Q. (2021). Generate & rank: A multi-task framework for math word problems. In *EMNLP*.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K. R., and Yao, S. (2023). Reflexion: language agents with verbal reinforcement learning. In *NeurIPS*.

Singhi, N., Bansal, H., Hosseini, A., Grover, A., Chang, K.-W., Rohrbach, M., and Rohrbach, A. (2025). When to solve, when to verify: Compute-optimal problem solving and generative verification for llm reasoning. *arXiv preprint arXiv:2504.01005*.

Snell, C. V., Lee, J., Xu, K., and Kumar, A. (2025). Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *ICLR*.

Song, Y., Zhang, H., Eisenach, C., Kakade, S. M., Foster, D., and Ghai, U. (2025). Mind the gap: Examining the self-improvement capabilities of large language models. In *ICLR*.

Stechly, K., Valmeekam, K., and Kambhampati, S. (2025). On the self-verification limitations of large language models on reasoning and planning tasks. In *The Thirteenth International Conference on Learning Representations*.

Stroebl, B., Kapoor, S., and Narayanan, A. (2024). Inference scaling flaws: The limits of llm resampling with imperfect verifiers. *arXiv preprint arXiv:2411.17501*.

Talmor, A., Herzig, J., Lourie, N., and Berant, J. (2019). Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *NAACL 2019*.

Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. (2023). Self-consistency improves chain of thought reasoning in language models. In *ICLR*.

Welleck, S., Lu, X., West, P., Brahman, F., Shen, T., Khashabi, D., and Choi, Y. (2023). Generating sequences by learning to self-correct. In *ICLR*.

Weng, Y., Zhu, M., Xia, F., Li, B., He, S., Liu, S., Sun, B., Liu, K., and Zhao, J. (2023). Large language models are better reasoners with self-verification. In *EMNLP Findings*.

Wu, Z., Zeng, Q., Zhang, Z., Tan, Z., Shen, C., and Jiang, M. (2024). Large language models can self-correct with key condition verification. In *EMNLP*.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. (2025). Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yu, F., Gao, A., and Wang, B. (2024). OVM, outcome-supervised value models for planning in mathematical reasoning. In *NAACL Findings*.

Yu, F., Li, Y., and Wang, B. (2025). Scaling flaws of verifier-guided search in mathematical reasoning. *arXiv preprint arXiv:2502.00271*.

Zhang, L., Hosseini, A., Bansal, H., Kazemi, M., Kumar, A., and Agarwal, R. (2025). Generative verifiers: Reward modeling as next-token prediction. In *ICLR*.

Zhang, Y., Khalifa, M., Logeswaran, L., Kim, J., Lee, M., Lee, H., and Wang, L. (2024). Small language models need strong verifiers to self-correct reasoning. In *ACL*.

Zhao, E., Awasthi, P., and Gollapudi, S. (2025). Sample, scrutinize and scale: Effective inference-time search by scaling verification. In *ICML*.

Zhou, A., Wang, K., Lu, Z., Shi, W., Luo, S., Qin, Z., Lu, S., Jia, A., Song, L., Zhan, M., and Li, H. (2024). Solving challenging math word problems using GPT-4 code interpreter with code-based self-verification. In *ICLR*.

Zhou, Y., Xu, A., Zhou, Y., Singh, J., Gui, J., and Joty, S. (2025). Variation in verification: Understanding verification dynamics in large language models. *arXiv preprint arXiv:2509.17995*.

Appendix

A Additional Details on Verifier Metrics

We show the mathematical definitions of relevant verifier metrics below. For clarity, we include dependencies (e.g., $(S, V; \mathcal{D})$) in the definitions, but sometimes omit them for brevity when the context is clear.

$$\begin{aligned}
\text{VerifierAcc}(S, V; \mathcal{D}) &= \mathbb{E}_{(x,\mathcal{Y}_x)\sim\mathcal{D},\, y\sim S(x)}\left[\, \mathbb{1}\{\, V(x,y) = c(x,y)\,\}\,\right] \\
\text{Precision}(S, V; \mathcal{D}) &= \mathbb{E}[\, c(x,y) \mid V(x,y) = 1\,] = \frac{\text{SolverAcc}\cdot\text{TPR}}{\text{SolverAcc}\cdot\text{TPR} + (1-\text{SolverAcc})\cdot\text{FPR}} \\
\text{Recall}(S, V; \mathcal{D}) &= \text{TPR} \\
\text{F1}(S, V; \mathcal{D}) &= \frac{2\cdot\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}
\end{aligned}$$

B Additional Details on Models

We show the information for each of our 37 evaluated models in Table 1.

C Additional Details on Datasets

C.1 Real-World Datasets

Note that for MMLU (STEM) and MMLU (Social Sciences), we concatenate questions from all subjects that belong to the STEM and Social Sciences supercategories in Hendrycks et al. (2021), respectively.

C.2 Synthetic Datasets

We generate three synthetic datasets, named 3SAT, Matrix Multiplication, and Sudoku, with 1000 samples each. We explain each synthetic dataset's generation parameters below.

Each 3SAT CNF contains uniformly sampled numbers of variables and clauses from 2 to 8 (inclusive). Each Sudoku puzzle is a 9x9 grid with 12 randomly missing cells. Each Matrix Multiplication problem is about multiplying 2 4x4 integer matrices with values uniformly sampled from $[-5, 5]$. All data are generated in a way that ensures the existence of a valid solution. Note that while Matrix Multiplication has a singular correct answer for each problem, Sudoku and 3SAT are allowed multiple correct answers as long as the solver's answer is correct by their rules.

The generation code files for all synthetic datasets are seeded for reproducibility.

An example of a generated 3SAT problem:

```
## Problem Definition
```

```
**SAT (Boolean Satisfiability Problem)** is a fundamental problem in computer science where we need to determine if there exists an assignment of Boolean values (True/False) to variables that makes a given Boolean formula evaluate to True.
**Variables**: In this problem, variables are named as single letters. Each variable can be assigned either True (T) or False (F).
**Literals**: A literal is either a variable (like a) or its negation (like ~a, meaning "not a"). If a is True, then ~a is False, and vice versa.
**Clauses**: A clause is a disjunction (OR operation) of literals. A clause is satisfied (True) if at least one of its literals is True. For example, the clause (a or ~b) is True if either a is True OR b is False (or both).
**CNF (Conjunctive Normal Form)**: The Boolean formula is given in CNF, which is a conjunction (AND operation) of multiple clauses. The entire formula is satisfied only if ALL clauses are satisfied simultaneously.
**3SAT**: This is a special case of SAT where every clause contains exactly 3 literals.

## The Problem

Find a satisfying assignment for the following CNF formula: (~c or ~b or d) and
(d or ~b or ~c) and (d or a or c) and (~c or d or a) and (b or ~a or d) and (c or d or ~b)

## Instructions

Provide your answer as a list of variable assignments, one per line, in the format
"variable_name T" or "variable_name F." For example:
\boxed{
a T
b F
}
This means a=True, b=False.

Another example answer is
\boxed{
a F
b T
}
This means a=False, b=True.

Output and only output the T/F values for the variables that appear in the provided
CNF formula.
```

An example of a generated Sudoku problem:

```
## Sudoku Problem

**Sudoku** is a logic-based number-placement puzzle. The objective is to fill a 9x9 grid with numbers so that each column, each row, and each of the 3x3 sub-grids contains all of the numbers from 1 to 9.
```

```
## The Puzzle

Complete the following 9x9 Sudoku grid (empty cells are marked with '_'):

7 4 2 1 _ 5 8 9 6
1 6 9 2 4 8 3 5 7
8 5 3 _ _ 7 2 1 4
2 _ 8 9 7 1 4 6 5
5 7 6 4 8 2 9 3 _
4 9 1 3 _ 6 _ 8 _
3 1 5 8 2 4 6 7 9
6 8 _ 7 1 _ 5 2 3
_ 2 7 5 6 _ 1 4 8

## Instructions

Provide your answer as a completed 9x9 grid with all numbers filled in, formatted exactly
like the puzzle above but with numbers instead of underscores.

For example, a completed 4x4 grid should look like:
\boxed{
1 2 3 4
3 4 1 2
2 3 4 1
4 1 2 3
}
```

An example of a generated Matrix Multiplication problem:

```
## Matrix Multiplication Problem

**Matrix Multiplication** is a fundamental operation in linear algebra where we compute
the product of two matrices. For two square matrices A and B of size 4x4, the product
C = A x B is computed as:

C[i][j] = Sum(k=0 to 3) A[i][k] x B[k][j]

## The Problem

Compute the product of the following two 4x4 matrices:

**Matrix A:**
0 1 1 4
-1 3 4 4
-2 -5 -5 0
-4 4 5 0

**Matrix B:**
```

```
1 2 0 5
1 -2 0 0
3 -1 -3 -3
2 5 -4 2

## Instructions

Provide your answer as the resulting 4x4 matrix C = A x B, formatted with each row
on a separate line and numbers separated by spaces.

For example, a 2x2 result matrix is formatted like:
\boxed{
1 2
3 4
}
```

D Additional Details on Experimental Setup

We use the following solver prompt for all models:

```
Please reason step by step, and put your final answer within \boxed{{}}.

{question}
```

We use the following verifier prompt for all models:

```
You are a teacher that is evaluating a student's answer to a question.
Your task is to determine whether the answer is correct or incorrect.

Question: {question}

Student's Answer: {response}

Please evaluate the student's answer carefully. Consider:
- Is the answer factually accurate?
- Is the reasoning sound and logical?
- Does it fully address the question asked?

After your evaluation, provide your judgment in the
following format:
- If the answer is correct, write: \boxed{{correct}}.
- If the answer is incorrect, write: \boxed{{incorrect}}.

First explain your analysis over the student's answer, then provide your final judgment in
the boxed format. Make sure the final judgment is either "correct" or "incorrect" inside
the \boxed{{}}. Do not put anything else in \boxed{{}}. Do not repeat the student's answer
in \boxed{{}}.
```

Figure 8 displays the ratio of filtered solver outputs due to not containing a box for answer extraction, averaged across all datasets.

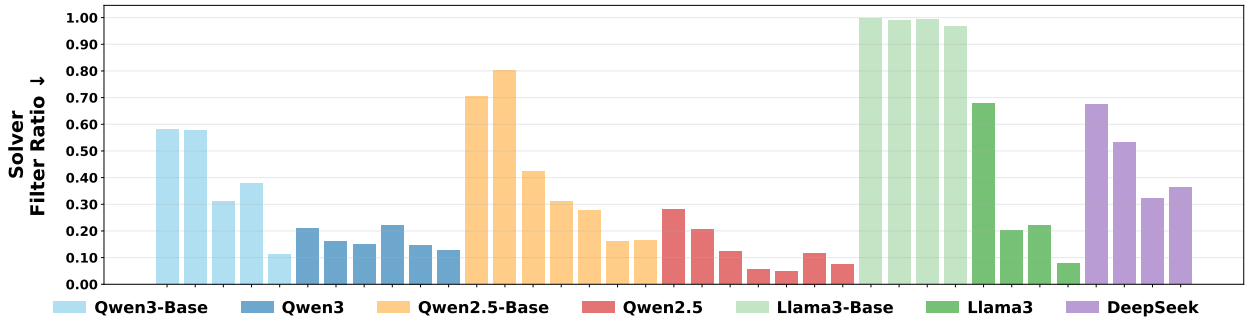


Figure 8: *Average ratio of filtered solver outputs for each model over all datasets. Base model families are suffixed by* ***-Base****. Models within each family are ordered in increasing size.*

E Solver Accuracy by Dataset

Figure 9 shows the solver accuracies of all 37 models on each of our 9 datasets.

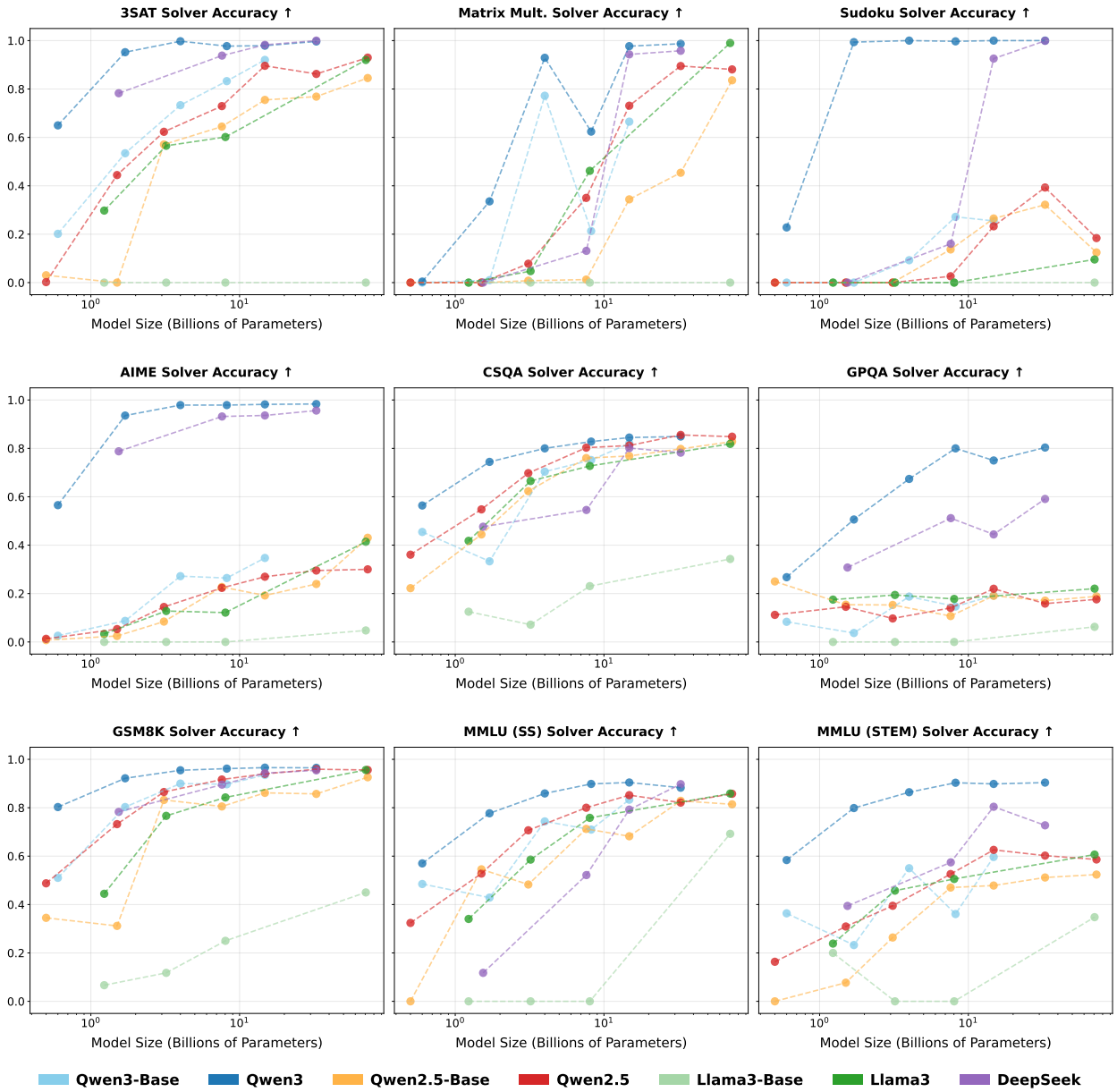


Figure 9: *The solver accuracies of 37 models on each dataset.*

F F1-Score and Precision Visualization

Figure 3 shows the correlation between each model's verification ability and its own solver accuracy for all 21 post-trained models. We additionally display verifier F1-Score and precision in Figure 10.

In comparison to verifier accuracy, while F1-Score also positively correlates with the verifier's own solver accuracy for all verification settings, the slopes decrease from self-verification to intra-family verification, and further decrease for cross-family verification, showing that the increase in false positive rate in Figure 3 has a stronger impact on lowering F1-Score than accuracy.

While Section 5.2 explains the low verifier gains for self- and intra-family verification through close examination of FPR, we additionally plot verifier precision in Figure 10. However, since precision is the expected performance of verifier-based rejection sampling in the limit of infinite sampling and our main metric "verifier gain" is defined in terms of it (Equation 1), precision does not help explain the differences in verifier gains across verification settings.

G Self-Verification Keywords

To measure the spontaneous self-verification rate of solver outputs (Section 5.2), we scan each solver output for the following case-insensitive keywords (separated by commas):

> *let me check, let me verify, double-check, let me recalculate, wait, hold on, going back, reconsider, let me re-, mistake, i made, i forgot, i missed, that doesn't, that's wrong, that's not, this is wrong*

A solver output is classified as containing spontaneous self-verification if any of these keywords appear in the generated text.

H Generation Log-Likelihood as an Alternative Similarity Metric

As a complement to the cosine similarity analysis in Section 5.3, we compute the average per-token log-likelihood that each verifier assigns to each solver's outputs and use this as an alternative similarity metric. A key limitation of this metric is that it conflates two signals: (a) distributional similarity between the solver and verifier, which is the quantity of interest, and (b) intrinsic predictability of the solver's text, independent of the evaluating model. For example, short, conventionally structured outputs from `Llama3` receive higher likelihood from most verifiers than the long reasoning chains with backtracking produced by `DeepSeek`, regardless of distributional similarity. To isolate (a), we normalize by solver, subtracting each solver's mean log-likelihood across all verifiers.

After normalization, the same directional findings emerge (Figure 11): higher generation log-likelihood correlates with higher FPR (intra-family slope $= +0.238$, cross-family slope $= +0.459$) and lower verifier gain (intra-family slope $= +0.043$, cross-family slope $= -0.183$), consistent with the cosine similarity results in Figure 5.

I Effect of Reasoning Post-Training on Solver Performance

Figure 12 shows the average improvement in solver accuracies of `Qwen2.5-Base` and `Qwen3-Base` families of models from their respective post-training procedures.

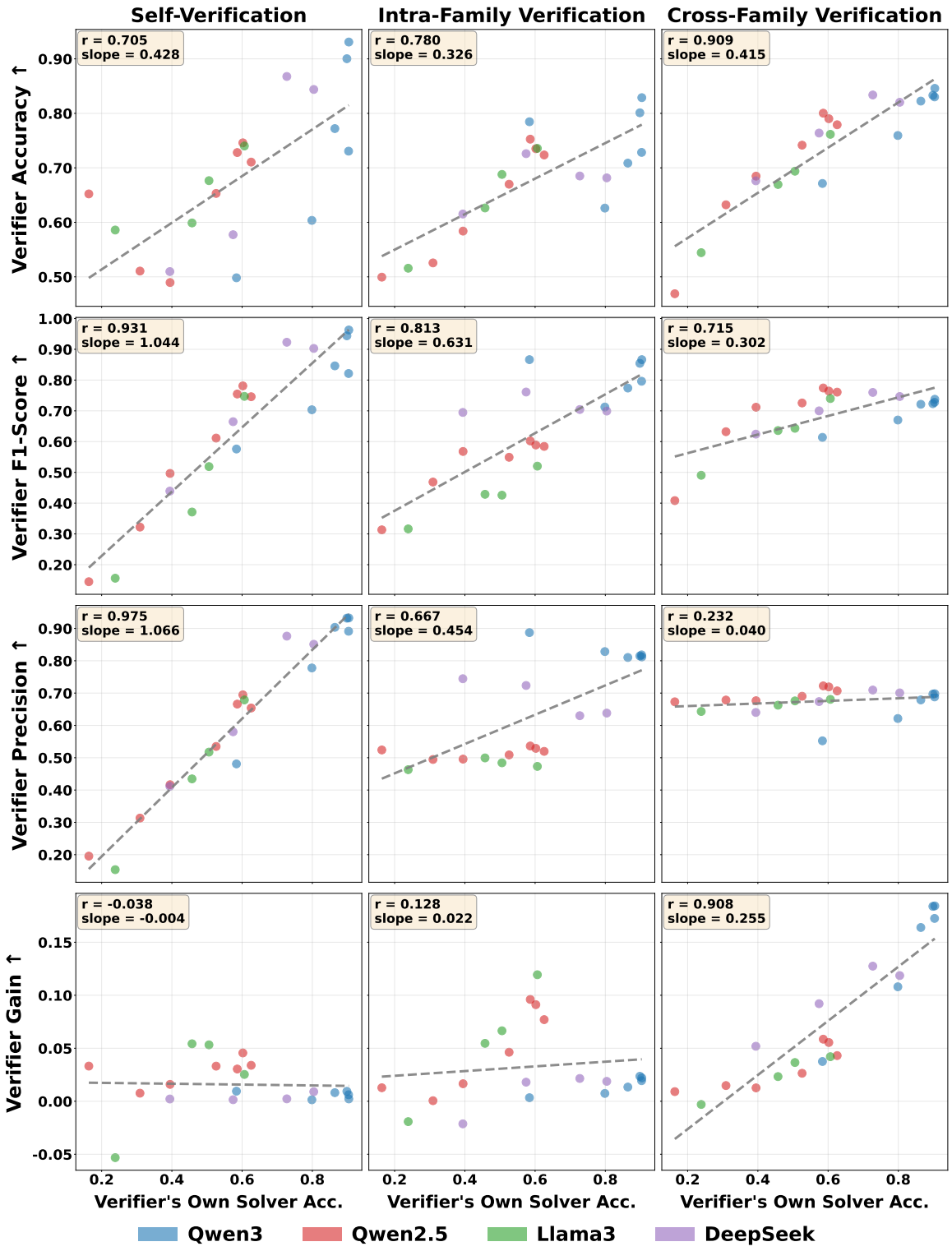


Figure 10: *Correlation between each model's verifier metrics (rows) and its own solver accuracy for all 21 post-trained models, averaged over all datasets. Each verifier metric is computed over three settings (columns): self-verification, intra-family verification, and cross-family verification. We use the same set of post-trained models as the set of solver models.*

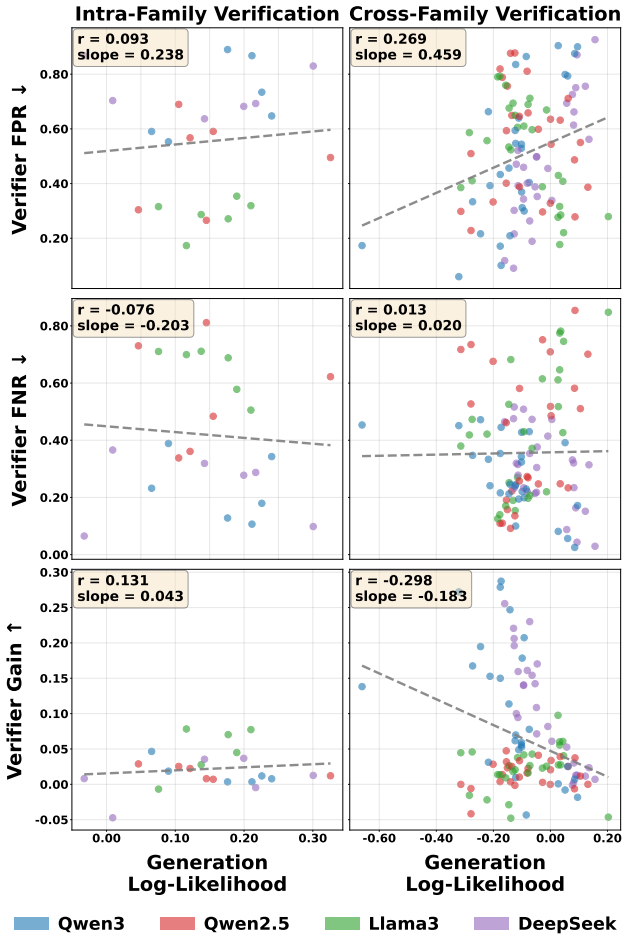


Figure 11: *Correlation between verifier metrics and solver-normalized generation log-likelihood for each solver–verifier pair. Each marker is colored based on the verifier model family. Self-verification is omitted as it yields only one data point per model, too few for reliable correlation analysis.*

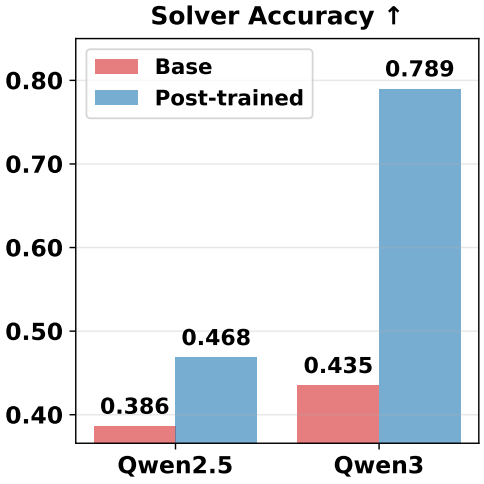


Figure 12: *Improvements in solver accuracies of* `Qwen2.5-Base` *and* `Qwen3-Base` *models from reasoning post-training.*

Table 1: *Complete list of each evaluated model's HuggingFace identifier, family, and size.*

HuggingFace Identifier	Family	Size
Qwen/Qwen3-0.6B-Base	Qwen3-Base	0.6B
Qwen/Qwen3-1.7B-Base	Qwen3-Base	1.7B
Qwen/Qwen3-4B-Base	Qwen3-Base	4B
Qwen/Qwen3-8B-Base	Qwen3-Base	8B
Qwen/Qwen3-14B-Base	Qwen3-Base	14B
Qwen/Qwen3-0.6B	Qwen3	0.6B
Qwen/Qwen3-1.7B	Qwen3	1.7B
Qwen/Qwen3-4B	Qwen3	4B
Qwen/Qwen3-8B	Qwen3	8B
Qwen/Qwen3-14B	Qwen3	14B
Qwen/Qwen3-32B	Qwen3	32B
Qwen/Qwen2.5-0.5B	Qwen2.5-Base	0.5B
Qwen/Qwen2.5-1.5B	Qwen2.5-Base	1.5B
Qwen/Qwen2.5-3B	Qwen2.5-Base	3B
Qwen/Qwen2.5-7B	Qwen2.5-Base	7B
Qwen/Qwen2.5-14B	Qwen2.5-Base	14B
Qwen/Qwen2.5-32B	Qwen2.5-Base	32B
Qwen/Qwen2.5-72B	Qwen2.5-Base	72B
Qwen/Qwen2.5-0.5B-Instruct	Qwen2.5	0.5B
Qwen/Qwen2.5-1.5B-Instruct	Qwen2.5	1.5B
Qwen/Qwen2.5-3B-Instruct	Qwen2.5	3B
Qwen/Qwen2.5-7B-Instruct	Qwen2.5	7B
Qwen/Qwen2.5-14B-Instruct	Qwen2.5	14B
Qwen/Qwen2.5-32B-Instruct	Qwen2.5	32B
Qwen/Qwen2.5-72B-Instruct	Qwen2.5	72B
meta-llama/Llama-3.2-1B	Llama3-Base	1B
meta-llama/Llama-3.2-3B	Llama3-Base	3B
meta-llama/Llama-3.1-8B	Llama3-Base	8B
meta-llama/Llama-3.1-70B	Llama3-Base	70B
meta-llama/Llama-3.2-1B-Instruct	Llama3	1B
meta-llama/Llama-3.2-3B-Instruct	Llama3	3B
meta-llama/Llama-3.1-8B-Instruct	Llama3	8B
meta-llama/Llama-3.1-70B-Instruct	Llama3	70B
deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B	DeepSeek	1.5B
deepseek-ai/DeepSeek-R1-Distill-Qwen-7B	DeepSeek	7B
deepseek-ai/DeepSeek-R1-Distill-Qwen-14B	DeepSeek	14B
deepseek-ai/DeepSeek-R1-Distill-Qwen-32B	DeepSeek	32B

Table 2: *HuggingFace information and sizes of real-world datasets.*

Dataset Name	HuggingFace Identifier	HuggingFace Split	Size
GSM8K	openai/gsm8k	test	1319
AIME	TianHongZXY/aime-1983-2025	test	963
MMLU (STEM)	cais/mmlu	test	316
MMLU (Social Sciences)	cais/mmlu	test	308
CSQA	tau/commonsense_qa	validation	2442
GPQA	Idavidrein/gpqa	train	198