

LEARNING FROM PARTIAL CHAIN-OF-THOUGHT VIA TRUNCATED-REASONING SELF-DISTILLATION

Gianluigi Silvestri*
 Radboud University
 Nijmegen, the Netherlands
 gianlu.silvestri@gmail.com

Edoardo Cetin
 Sakana AI
 Tokyo, Japan
 edo@sakana.ai

ABSTRACT

Reasoning-oriented language models achieve strong performance by generating long chain-of-thought traces at inference time. However, this capability comes with substantial and often excessive computational cost, which can materialize in redundant or inefficient reasoning. We study this setting and introduce Truncated-Reasoning Self-Distillation (TRSD), a lightweight post-training procedure that encourages models to produce correct predictions from partial reasoning traces. In TRSD, a frozen teacher model first generates a full reasoning trace and evaluates the corresponding answer distribution conditioned on the prompt and the complete reasoning to construct a synthetic training target. A student model with the same architecture is then trained to match the teacher’s answer distribution while being conditioned only on a truncated prefix of its reasoning trace. Across multiple reasoning benchmarks and token budgets, we demonstrate that TRSD improves robustness to truncated inference, with far reduced accuracy tradeoffs when applied to a diverse set of reasoning models. Moreover, although never explicitly regularized for shorter generation during training, we also find that TRSD-trained models inherently output shorter reasoning traces without truncation, significantly reducing inference-time costs even without artificial interventions.

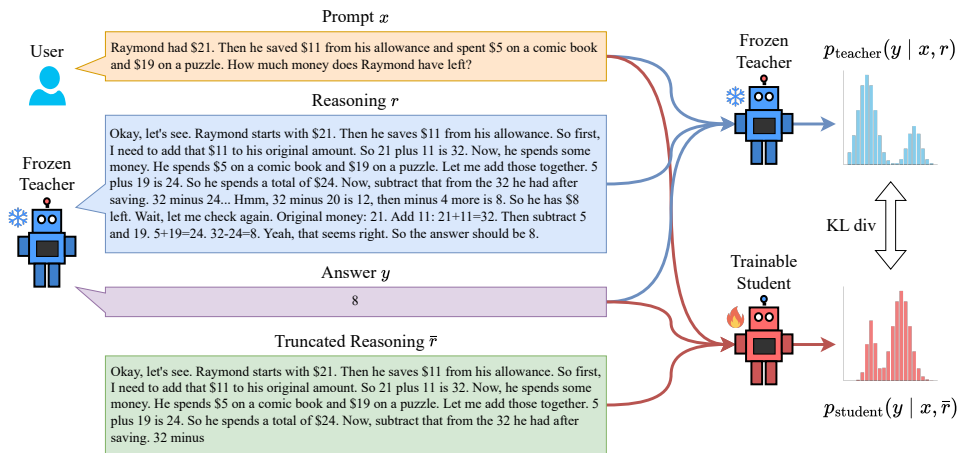


Figure 1: Truncated-Reasoning Self-Distillation (TRSD). Given an input prompt x , a frozen teacher model first generates a full chain-of-thought reasoning trace r and an answer y , and then evaluates the answer-token distribution $p_{\text{teacher}}(y | x, r)$ conditioned on the prompt and the complete reasoning trace. The trainable student model, initialized as a copy of the teacher, is conditioned only on a truncated prefix \bar{r} of the teacher-generated reasoning trace, and evaluates the corresponding answer distribution $p_{\text{student}}(y | x, \bar{r})$. Training minimizes the KL divergence between the teacher and student answer distributions, encouraging the student to recover the same predictions from partial reasoning and to remain accurate when inference-time reasoning is truncated.

*Work done during an internship at Sakana AI.

1 INTRODUCTION

Recent progress in large language models (LLMs) has increasingly focused on solving complex reasoning tasks, moving beyond fluent text generation toward multi-step problem solving in domains such as mathematics and coding (Wei et al., 2022; Guo et al., 2025). A central ingredient behind these gains is the explicit generation of chain-of-thought reasoning, in which models produce long intermediate reasoning traces before emitting a final answer. In modern reasoning models, performance improvements are often achieved by allowing longer such traces at inference time, effectively treating test-time compute as an additional scaling axis alongside model size and training data (Muenighoff et al., 2025; Yang et al., 2025).

However, this reliance on extended inference comes with significant costs. Reasoning models frequently generate thousands of chain-of-thought tokens per query, resulting in high inference latency and substantial computational overhead. More importantly, this additional computation is often inefficient: models tend to overthink, producing verbose or repetitive reasoning that provides little benefit to the final answer. Prior work shows that such behavior can arise even on simple problems, where extended reasoning is unnecessary and does not reliably improve accuracy (Chen et al., 2025b). In more extreme cases, reasoning traces may degenerate into repetitive or looping patterns, consuming large amounts of inference-time compute without meaningful progress (Pipis et al., 2025).

In response to the growing cost of long chain-of-thought reasoning, a range of strategies have been proposed to reduce test-time compute. A first class of approaches acts directly at inference time, guiding the model to produce shorter reasoning traces. This includes prompt-based heuristics, explicit suppression of reflection tokens, and decoding strategies designed to reach a final answer with fewer intermediate reasoning steps (Zhang et al., 2023; Leviathan et al., 2025; Wang et al., 2025; Xu et al.). A second line of work focuses on explicitly compressing or shortening chain-of-thought traces, for instance by skipping, pruning, or selectively retaining reasoning tokens while attempting to preserve final answer accuracy (Yan et al., 2025; Xia et al., 2025; Zhang et al., 2025a;b; Ma et al., 2025; Yuan et al., 2025). Finally, some methods incorporate length control directly into training, encouraging models to produce shorter reasoning traces by design rather than artificial inference interventions (Luo et al., 2025; Chen et al., 2025a).

In this work, we take a different perspective on reducing the cost of reasoning. Rather than modifying decoding procedures or explicitly shortening chain-of-thought traces, we focus on training reasoning models to remain effective when inference-time reasoning is truncated, a setting that naturally arises under latency constraints, compute budgets, or user-facing APIs that allow early stopping. We propose *Truncated-Reasoning Self-Distillation* (TRSD), a post-training self-distillation approach in which a model is trained to match its answer predictions from a truncated prefix of its own reasoning trace to those obtained when conditioned on the full reasoning (see Figure 1). By encouraging robustness to truncated reasoning rather than enforcing shorter traces, TRSD directly targets wasteful inference without constraining how models reason, shifting the optimization objective away from producing shorter reasoning and toward producing answers that rely less on extended or redundant computation.

We evaluate TRSD across multiple reasoning-oriented language model architectures and datasets. Across most settings, TRSD-trained models outperform their corresponding baselines under truncated inference (Figure 2), and in several cases these gains persist even when larger reasoning bud-

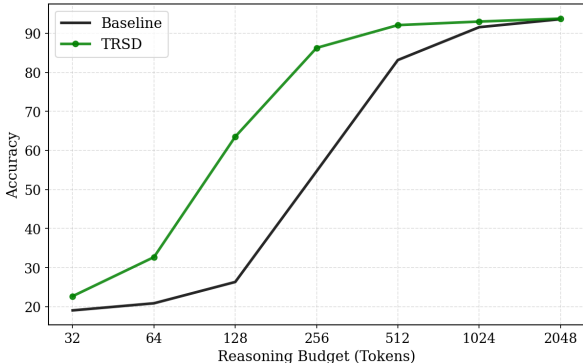


Figure 2: Accuracy as a function of the available reasoning budget for a Qwen3-4B model on GSM8K. Truncated-Reasoning Self-Distillation (TRSD) substantially improves performance in low-budget regimes, enabling accurate predictions with limited reasoning.

gets are available. When reasoning is unconstrained, TRSD-trained models also tend to generate shorter reasoning traces while maintaining or improving accuracy. Importantly, this reduction in reasoning length is not enforced during training, but emerges naturally as a consequence of optimizing robustness to truncated reasoning. Overall, TRSD is a simple and lightweight post-training procedure that improves accuracy under low reasoning budgets and reduces unnecessary inference-time computation in existing reasoning models.

2 BACKGROUND

Reasoning models are language models trained to explicitly generate chain-of-thought reasoning traces during inference, rather than relying solely on implicit internal computation. This paradigm, exemplified by models such as DeepSeek-R1 (Guo et al., 2025), induces language models to decompose complex problems into intermediate steps by generating long thinking traces before producing a final answer. Training reasoning models typically proceeds in two stages. First, the model undergoes a supervised fine-tuning stage, leveraging masked prompts paired with explicit reasoning traces and correct answers to establish a structured reasoning format in its generation patterns (Wei et al., 2022; Guo et al., 2025). Second, the model is directly optimized with reinforcement learning to maximize the correctness of its generated responses on downstream tasks using verifiable rewards (Guo et al., 2025). In this second stage, the thinking trace generated by the language model inherently grows, increasingly trading test-time computation for improved accuracy. To leverage the properties of reasoning while mitigating its inference cost, large reasoning models are often distilled into smaller “students”. In standard distillation pipelines, a “teacher” generates synthetic data consisting of prompts, reasoning traces, and answers, which are used to supervise a smaller model via fine-tuning or distribution matching (Guo et al., 2025; Guha et al., 2025). While distillation enables compact models to acquire strong reasoning performance (Ye et al., 2025; Muennighoff et al., 2025), it can also encourage students to reproduce long and unnecessary reasoning traces, partially inheriting the teacher’s inefficiencies (Chen et al., 2025b; Pipis et al., 2025; Yuan et al., 2025).

3 METHOD

3.1 TRUNCATED-REASONING SELF-DISTILLATION

We propose *Truncated-Reasoning Self-Distillation* (TRSD), a lightweight post-training procedure applied as an additional fine-tuning step on top of already-trained reasoning models to encourage accurate predictions even with cheaper truncated reasoning traces.

The core idea is to train a model to match its own predictions under full reasoning while being conditioned on truncated reasoning during training. Let x denote an input prompt, r a generated chain-of-thought reasoning trace, and y the corresponding answer. As in traditional distillation, a teacher model is prompted with x to generate a complete reasoning trace r followed by an answer y . However, rather than conditioning on the full reasoning trace, the student model is then trained to match the teacher’s answer distribution while being conditioned only on a truncated prefix of r , denoted by \bar{r} . The teacher and student share the same architecture and initialization, but while the student is updated, the teacher remains frozen. Thus, all the supervision signal arises from matching the frozen teacher’s predictions under truncated reasoning rather than expensive, larger models. Our design choice of freezing the teacher serves to provide a stable target distribution and allows the student to learn how to recover accurate predictions from its own truncated reasoning, rather than altering the underlying reasoning process itself.

3.2 REASONING TRUNCATION

In TRSD, reasoning traces are truncated at the token level. For each training example, we sample a truncation ratio $\alpha \sim \mathcal{U}(0, 1)$, corresponding to retaining a prefix containing $\lfloor \alpha \cdot |r| \rfloor$ tokens of the original reasoning trace, where $|r|$ denotes the number of tokens in the reasoning trace. Truncation is applied independently for each example in the batch. This simple uniform sampling strategy exposes the student to a broad range of truncated-reasoning regimes, from near-complete traces to very short prefixes. It also reflects the unpredictability of inference-time truncation in real-world settings, where generation may be stopped at arbitrary points.

3.3 TRAINING OBJECTIVE

Following the canonical DeepSeek R1 formulation (Guo et al., 2025), the teacher model is prompted to generate reasoning within `<think> ...</think>` tags, followed by the final answer within `<answer> ...</answer>` tags. We enforce a maximum reasoning length with answer forcing (Muennighoff et al., 2025): if the model does not emit an answer within a given token budget, we pause generation and append the opening answer tag to elicit immediate answer generation. For each training example, while the teacher generates the answer tokens conditioned on the *full* reasoning trace, the student is trained to match the same answer tokens conditioned on its *truncated* version. To optimize the student, our framework minimizes the KL divergence between the teacher and student answer distributions:

$$\mathcal{L} = \text{KL}(p_{\text{teacher}}(y \mid x, r) \parallel p_{\text{student}}(y \mid x, \bar{r})). \quad (1)$$

The loss is computed only over answer tokens, without backpropagation through the teacher (Algorithm 1). This encourages the student to match the teacher’s answer under truncated reasoning, while discouraging reliance on repetitive or late-stage refinement tokens that are often missing. As a result of our design, the model is optimized to prioritize earlier, more informative reasoning tokens, preserving its final response distribution with a cheaper, truncated reasoning trace.

Algorithm 1 Truncated-Reasoning Self-Distillation

Require: Prompt x , teacher model f_{θ_t} , student model f_{θ_s} , max reasoning length L

Ensure: Updated student parameters θ_s

- 1: $(r, y) \leftarrow f_{\theta_t}(x) \triangleright$ teacher generation (frozen)
 - 2: **if** y not produced within L **then**
 - 3: Force answer generation
 - 4: **end if**
 - 5: Sample $\alpha \sim \mathcal{U}(0, 1)$
 - 6: $\bar{r} \leftarrow \text{prefix}(r, \lfloor \alpha |r| \rfloor)$
 - 7: $p_T \leftarrow f_{\theta_t}(y \mid x, r) \triangleright$ stop gradient
 - 8: $p_S \leftarrow f_{\theta_s}(y \mid x, \bar{r})$
 - 9: $\theta_s \leftarrow \arg \min_{\theta_s} \text{KL}(p_T \parallel p_S)$
-

4 EXPERIMENTAL RESULTS

We evaluate TRSD across a wide range of reasoning-oriented language models and training datasets. Specifically, we consider Qwen3 models at 0.6B, 1.7B, and 4B parameters (Yang et al., 2025), Phi-4-mini-reasoning (4B) (Xu et al., 2025), and OpenThinker-3 (1.5B) (Guha et al., 2025). For each architecture, models are trained using prompts drawn from a single dataset at a time, spanning a range of reasoning difficulty, including Countdown, GSM8K (Cobbe et al., 2021), Dolci Math (Olmo et al., 2025), and competition MATH (Hendrycks et al., 2021). Throughout our analysis, we use the average number of reasoning tokens generated by the baseline model as a coarse proxy for task difficulty, with more challenging tasks typically requiring longer reasoning traces (see, e.g., Table 3). Additional details about training and evaluation datasets are provided in Appendix B. To keep the main discussion focused, we concentrate on the two largest models considered in our study, Qwen3-4B and Phi-4-mini-reasoning, both of which have approximately 4B parameters and exhibit the strongest overall performance across benchmarks. Results for smaller models and additional architectures follow the same qualitative trends and are reported in Appendix C, together with extended tables and per-dataset performance plots.

In all experiments, the student model is initialized as an exact copy of the teacher and trained using the procedure described in Section 3. Unless stated otherwise, the teacher is allowed to generate up to 8192 tokens for reasoning. If this budget is exhausted before an answer is produced, we apply answer forcing by appending the end-of-thinking token and allowing an additional 200 tokens for answer generation. The procedure used to process teacher-generated answers is described in Appendix A.1, and additional training details are provided in Appendix A.2.

At evaluation time, TRSD models autoregressively generate their own reasoning traces without any teacher support. We evaluate performance under a range of fixed reasoning budgets by enforcing a

Table 1: Baseline vs TRSD under truncated inference for in-distribution prompts. Entries report Baseline/TRSD accuracy (higher is better), bold indicates the better entry. When Baseline and TRSD accuracies differ by at most 1 percentage point, both are bold.

Budget / Data	Countdown		Dolci		GSM8K		Math500	
	Baseline	TRSD	Baseline	TRSD	Baseline	TRSD	Baseline	TRSD
<i>Teacher Model: Qwen3-4B</i>								
32	11.72	27.24 (+15.5)	8.85	9.90 (+1.1)	19.03	22.67 (+3.6)	19.00	23.60 (+4.6)
64	13.46	28.06 (+14.6)	8.97	10.54 (+1.6)	20.85	32.68 (+11.8)	18.60	25.40 (+6.8)
128	26.04	35.12 (+9.1)	9.33	11.82 (+2.5)	26.31	63.53 (+37.2)	21.20	25.80 (+4.6)
256	43.94	49.72 (+5.8)	9.94	13.18 (+3.2)	54.66	86.28 (+31.6)	26.00	35.20 (+9.2)
512	60.28	63.22 (+2.9)	11.46	15.46 (+4.0)	83.17	92.12 (+9.0)	45.40	58.00 (+12.6)
1024	71.14	72.54 (+1.4)	13.22	17.19 (+4.0)	91.58	93.03 (+1.4)	67.00	76.00 (+9.0)
2048	76.64	78.30 (+1.7)	17.19	19.87 (+2.7)	93.63	93.78 (+0.2)	80.80	85.60 (+4.8)
4096	81.32	81.20 (-0.1)	23.32	22.72 (-0.6)	94.31	93.86 (-0.5)	88.00	90.00 (+2.0)
8192	84.88	81.30 (-3.6)	34.21	24.16 (-10.1)	94.54	93.63 (-0.9)	91.40	91.20 (-0.2)
<i>Teacher Model: Phi-4-mini-reasoning</i>								
32	4.56	21.76 (+17.2)	7.05	9.17 (+2.1)	17.89	23.28 (+5.4)	13.40	16.00 (+2.6)
64	7.08	22.58 (+15.5)	6.73	9.13 (+2.4)	18.65	35.25 (+16.6)	12.80	17.00 (+4.2)
128	10.42	29.68 (+19.3)	8.33	9.25 (+0.9)	25.63	50.64 (+25.0)	18.00	21.60 (+3.6)
256	32.60	41.64 (+9.0)	9.58	11.18 (+1.6)	44.96	69.29 (+24.3)	21.00	32.60 (+11.6)
512	43.66	51.60 (+7.9)	9.62	12.10 (+2.5)	80.74	85.29 (+4.6)	35.40	44.40 (+9.0)
1024	54.34	57.48 (+3.1)	10.58	14.58 (+4.0)	87.72	86.88 (-0.8)	43.40	52.00 (+8.6)
2048	59.94	59.08 (-0.9)	13.50	16.99 (+3.5)	90.37	87.04 (-3.3)	46.40	60.00 (+13.6)
4096	61.96	59.56 (-2.4)	16.47	19.07 (+2.6)	90.67	87.11 (-3.6)	51.20	66.20 (+15.0)
8192	64.24	60.28 (-4.0)	24.44	20.15 (-4.3)	90.90	87.34 (-3.6)	56.20	69.40 (+13.2)

maximum number of reasoning tokens at inference time, after which generation is interrupted. If a model does not emit an answer within the allotted reasoning budget, we apply the same answer forcing procedure used during training, appending the end-of-thinking token and allowing an additional 200 tokens for answer generation (Muennighoff et al., 2025).

Throughout this section, we distinguish between *in-distribution* and *out-of-distribution* evaluation with respect to the distribution of prompts used during the self-distillation phase. In-distribution results correspond to models evaluated on the same dataset from which self-distillation prompts were drawn, while out-of-distribution results refer to evaluation on reasoning benchmarks not used during fine-tuning.

4.1 IN-DISTRIBUTION PERFORMANCE UNDER TRUNCATED INFERENCE

We begin by evaluating models on validation sets drawn from the same datasets used during self-distillation, in order to characterize robustness to inference-time truncation in a controlled setting. Table 1 reports results for Qwen3-4B and Phi-4-mini-reasoning, respectively, across a range of reasoning budgets.

Across both models, TRSD consistently improves performance in heavily truncated regimes. When the available reasoning budget is small (e.g., below 512 tokens), distilled models reliably outperform their corresponding baselines across all datasets. In contrast, at the largest reasoning budgets, performance differences become increasingly trivial and task-dependent, with TRSD students successfully preserving the baseline’s performance despite the shift in optimization objective. Crucially, these improvements should not be interpreted as standard task-specific fine-tuning effects. During TRSD, the model is not trained on ground-truth labels from the dataset, nor is it optimized to improve full-budget accuracy on that benchmark. Instead, the student is trained only to reproduce the frozen teacher’s answer distribution under truncated reasoning. The observed gains, therefore, suggest a genuine increase in robustness to limited reasoning budgets, rather than adaptation to the data distribution itself.

4.2 OUT-OF-DISTRIBUTION GENERALIZATION ACROSS PROMPT DISTRIBUTIONS

We next evaluate whether the robustness learned through TRSD transfers to unseen prompt distributions. Specifically, we evaluate models on reasoning benchmarks that are not used during self-distillation, assessing whether robustness to truncated reasoning generalizes beyond the datasets

Table 2: Baseline vs TRSD performance under truncated inference on out-of-distribution benchmarks. Entries report Baseline/TRSD accuracy (higher is better), bold indicates the better entry. When Baseline and TRSD accuracies differ by at most 1 percentage point, both are bold. Left: Baseline Qwen3-4B compared to TRSD trained on GSM8K. Right: Baseline Phi-4-mini-reasoning compared to TRSD trained on Countdown.

Teacher Model: Qwen3-4B, TRSD: GSM8K						
Budget / Data	Countdown		Dolci		Math500	
	Baseline	TRSD	Baseline	TRSD	Baseline	TRSD
32	11.72	12.62 (+0.9)	8.85	12.18 (+3.3)	19.00	23.20 (+4.2)
64	13.46	15.50 (+2.0)	8.97	12.14 (+3.2)	18.60	24.40 (+5.8)
128	26.04	24.88 (-1.2)	9.33	12.10 (+2.8)	21.20	27.40 (+6.2)
256	43.94	44.36 (+0.4)	9.94	13.46 (+3.5)	26.00	39.00 (+13.0)
512	60.28	60.82 (+0.5)	11.46	14.78 (+3.3)	45.40	58.80 (+13.4)
1024	71.14	70.54 (-0.6)	13.22	17.95 (+4.7)	67.00	74.00 (+7.0)
2048	76.64	77.20 (+0.6)	17.19	21.55 (+4.4)	80.80	80.60 (-0.2)
4096	81.32	81.44 (+0.1)	23.32	29.13 (+5.8)	88.00	84.20 (-3.8)
8192	84.88	82.86 (-2.0)	34.21	39.34 (+5.1)	91.40	86.20 (-5.2)

Teacher Model: Phi-4-mini-reasoning, TRSD: Countdown						
Budget / Data	Dolci		GSM8K		Math500	
	Baseline	TRSD	Baseline	TRSD	Baseline	TRSD
32	7.05	9.54 (+2.5)	17.89	20.70 (+2.8)	13.40	15.00 (+1.6)
64	6.73	9.98 (+3.3)	18.65	26.00 (+7.4)	12.80	16.80 (+4.0)
128	8.33	10.10 (+1.8)	25.63	35.56 (+9.9)	18.00	20.00 (+2.0)
256	9.58	10.58 (+1.0)	44.96	60.58 (+15.6)	21.00	26.60 (+5.6)
512	9.62	11.70 (+2.1)	80.74	84.91 (+4.2)	35.40	46.80 (+11.4)
1024	10.58	14.02 (+3.4)	87.72	90.60 (+2.9)	43.40	67.00 (+23.6)
2048	13.50	16.23 (+2.7)	90.37	91.96 (+1.6)	46.40	77.80 (+31.4)
4096	16.47	19.03 (+2.6)	90.67	92.42 (+1.8)	51.20	83.40 (+32.2)
8192	24.44	24.24 (-0.2)	90.90	92.42 (+1.5)	56.20	87.80 (+31.6)

seen during TRSD. For each architecture, we select the best-performing model across the evaluation datasets, as described in Appendix A.3. Results are summarized in Table 2, with per-dataset performance curves reported in Figure 3 for Qwen3-4B and in Figure 4 for Phi-4-mini-reasoning.

Overall, TRSD generalizes well across prompt distributions. Under truncated inference, TRSD-trained models consistently outperform the corresponding baselines on out-of-distribution benchmarks, and in several cases match or exceed in-distribution performance. Performance gains are most pronounced in the low-budget regime and remain stable across datasets, even when the evaluation task differs from the self-distillation source. At larger reasoning budgets, performance typically approaches that of the baseline, with only trivial and task-dependent differences, mirroring the behavior observed in the in-distribution setting. Together, these results indicate that TRSD encourages a general ability to recover correct predictions from truncated reasoning, rather than adapting models to dataset-specific reasoning patterns.

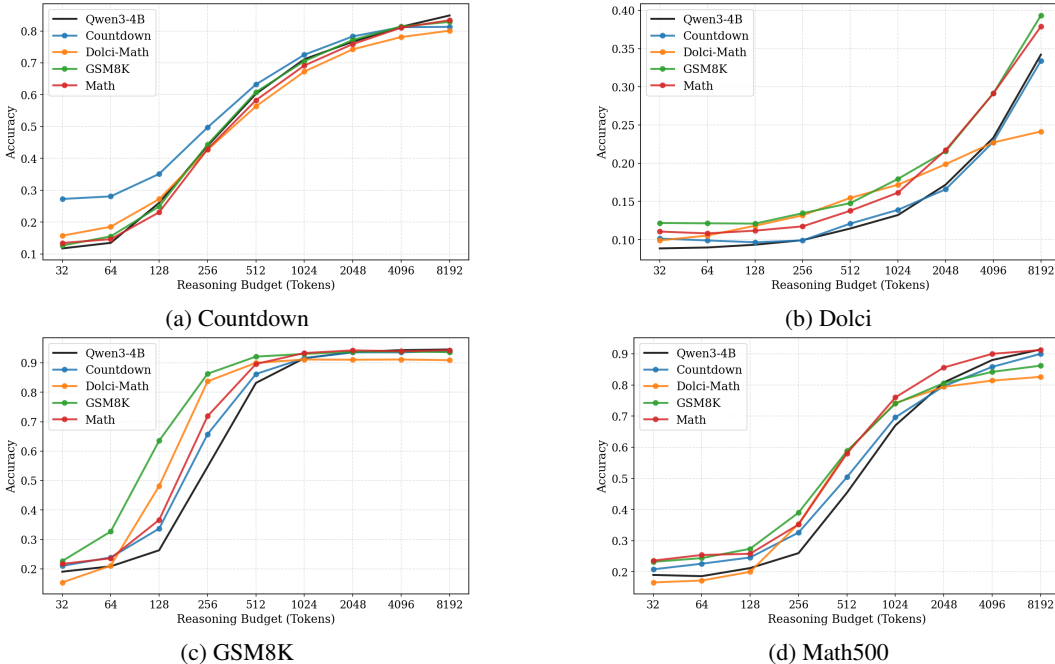


Figure 3: Per-dataset accuracy as a function of the reasoning budget for Qwen3-4B. The evaluation dataset is specified below the respective plot.

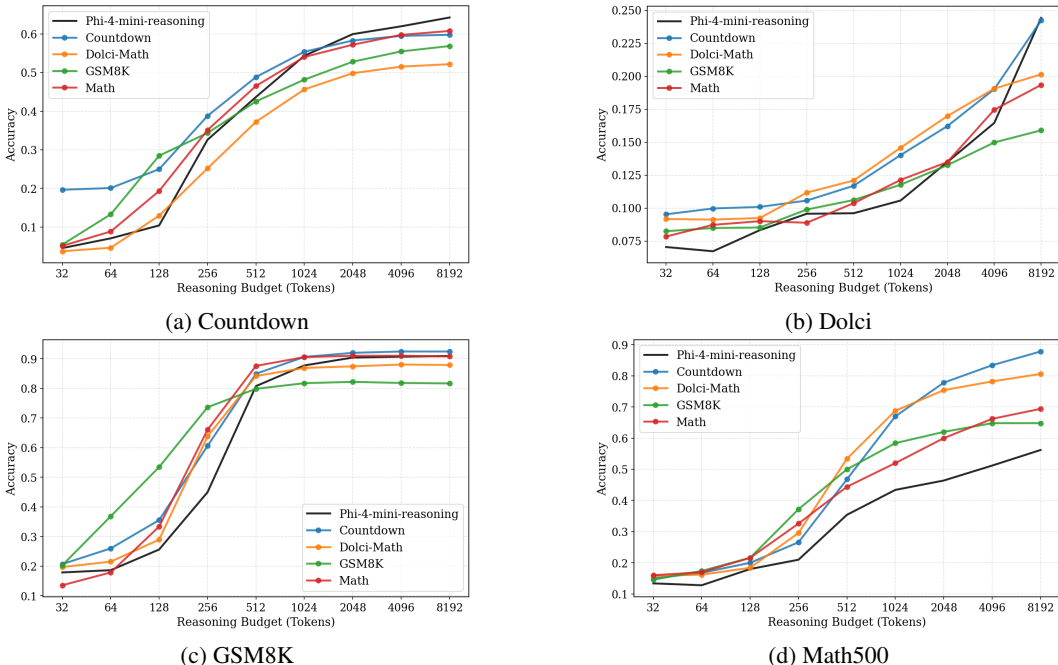


Figure 4: Per-dataset accuracy as a function of the reasoning budget for Phi-4-mini-reasoning. The evaluation dataset is specified below the respective plot.

4.3 EMERGENT REDUCTION IN REASONING LENGTH

While TRSD does not impose any explicit constraint on the length of the reasoning trace produced at inference time, its training objective discourages reliance on late-stage reasoning tokens that may be absent under truncation. As a result, TRSD-trained models consistently exhibit shorter reasoning traces even when inference is unconstrained.

We analyze this effect in detail by reporting the average number of reasoning tokens generated at the maximum reasoning budget ($t_{\max} = 8192$), conditioned on whether the final prediction is correct or incorrect, in Table 3. TRSD consistently reduces reasoning length for both correct and incorrect predictions. This behavior is observed both for in-distribution evaluation and out-of-distribution. The reduction in reasoning length is particularly pronounced for Qwen3-4B. For Phi-4-mini-reasoning under out-of-distribution evaluation, reasoning length on GSM8K and MATH is comparable to, or slightly longer than, the baseline; however, this is accompanied by improved performance on both datasets.

4.4 QUALITATIVE ANALYSIS OF REASONING TRACES

Beyond aggregate accuracy and token-level statistics, we examine how TRSD affects the structure and length of model reasoning. We qualitatively compare reasoning traces produced by baseline and TRSD-trained models on representative examples where both models produce the correct answer. Figure 5 shows an example from GSM8K using Qwen3-4B. While both models arrive at the same final answer, the baseline produces a longer reasoning trace that repeatedly restates intermediate quantities and re-verifies simple arithmetic. In contrast, the TRSD-trained model reaches the solution using a more compact reasoning prefix that focuses on the essential computations. We provide more examples in Appendix D evidencing this same pattern: TRSD prunes unnecessary tokens and computation from a model’s natural thinking trace, complementing the reasoning paradigm by successfully tackling its canonical inefficiencies.

Table 3: Average number of reasoning tokens conditioned on answer correctness under unconstrained inference ($r_{\max} = 8192$). Top block: in-distribution TRSD (trained and evaluated on the same prompt distribution). Bottom block: out-of-distribution TRSD using a single fixed checkpoint per model (Qwen3-4B trained on GSM8K; Phi-4-mini-reasoning trained on Countdown). Bold indicates the configuration (Baseline or TRSD) with lower average reasoning length.

Model	Outcome	Countdown		Dolci		GSM8K		Math500	
		Baseline	TRSD	Baseline	TRSD	Baseline	TRSD	Baseline	TRSD
<i>In-distribution evaluation</i>									
Qwen3-4B	Correct	1209	660	6915	3328	783	247	2810	1570
	Wrong	7169	3725	7913	4458	2546	661	5770	5243
Phi-4-mini-reasoning	Correct	1350	1095	6272	4346	683	513	1859	1831
	Wrong	4418	3799	7585	6477	1983	1537	3523	3079
<i>Out-of-distribution evaluation (fixed TRSD checkpoint)</i>									
Qwen3-4B	Correct	1209	824	6915	5624	783	247	2810	1216
	Wrong	7169	6307	7913	7141	2546	661	5770	3135
Phi-4-mini-reasoning	Correct	1350	1064	6272	5625	683	680	1859	1893
	Wrong	4418	3602	7585	7484	1983	2024	3523	5957

Qualitative comparison of reasoning traces (Qwen3-4B, GSM8K)	
<p>Prompt. Ray had 25 lollipops. He kept 5 lollipops and shared the remaining equally with his four friends. How many lollipops did each of his friends receive?</p>	
<p>Baseline (Qwen3-4B)</p> <p>Think: Okay, let's see. Ray had 25 lollipops. He kept 5 for himself, so the remaining ones are shared equally with his four friends. Hmm, first I need to figure out how many lollipops were left after he kept 5. So 25 total minus 5 he kept equals 20. Then he shares those 20 equally with his four friends. So I need to divide 20 by 4. Let me check that again. 25 minus 5 is 20, right? Then 20 divided by 4 is 5. So each friend gets 5 lollipops. Wait, that seems straightforward. Did I miss anything? The problem says "shared the remaining equally with his four friends." So yeah, 20 divided by 4 is definitely 5. I think that's the answer.</p> <p>Answer: 5</p>	<p>TRSD (GSM8K)</p> <p>Think: Okay, let's see. Ray starts with 25 lollipops. He keeps 5 for himself, so we subtract those first. That leaves 20 lollipops to share with his four friends. To find out how many each friend gets, I divide 20 by 4. 20 divided by 4 is 5. So each friend gets 5 lollipops.</p> <p>Answer: 5</p>

Figure 5: Example where both models answer correctly, but the TRSD-trained model uses a substantially shorter reasoning trace. The example is taken verbatim from the Qwen3-4B GSM8K evaluation set.

5 RELATED WORK

A common approach to reducing the cost of chain-of-thought reasoning is to explicitly shorten or prune reasoning traces. TokenSkip (Xia et al., 2025) skips intermediate reasoning tokens during generation based on learned importance scores, while Conditional Token Selection (Yuan et al., 2025) trains models to selectively retain only a subset of reasoning tokens. TokenSqueeze (Zhang et al., 2025b) compresses reasoning traces using length-aware training objectives that encourage more compact outputs. Related work, such as O1-Pruner (Luo et al., 2025) and CoT-Valve (Ma et al., 2025), incorporates length control directly into training, encouraging models to generate shorter reasoning traces through budget constraints or controllable reasoning length. Prompting-based methods such as Chain of Draft (Xu et al.) pursue a similar goal by instructing models to produce minimal intermediate reasoning at inference time. All these approaches explicitly optimize reasoning length or structure, either by deciding which tokens to keep, compress, or suppress, or by enforcing a target reasoning budget. In contrast, we do not enforce brevity, modify decoding, or alter the reasoning process itself. Instead, we train models to remain accurate when reasoning is arbitrarily truncated, allowing inference to stop early without explicitly selecting or pruning reasoning

tokens. Any reduction in reasoning length observed at inference time emerges as a consequence of optimizing robustness to partial reasoning, rather than as an explicit training objective.

The work of Chen et al. (2025a) analyzes which parts of reasoning traces provide effective supervision during distillation, showing that supervising selected portions of a teacher’s reasoning can retain strong performance. Adaptive Prefix Alignment (Liu et al., 2026) further explores how prefixes of reasoning traces can be used during distillation to reduce noise from later reasoning steps. These approaches typically distill from a larger teacher into a smaller student and rely on carefully selecting, weighting, or aligning specific parts of the teacher’s reasoning trace. In contrast, TRSD uses self-distillation, where the teacher and student share the same architecture and initialization, and does not assume that any particular portion of the reasoning is inherently more informative. By randomly truncating reasoning prefixes across the full range of possible truncation points, we directly optimize robustness of the answer distribution to partial reasoning, rather than optimizing for a fixed cutoff, compressed trace, or predefined notion of “important” reasoning tokens.

6 DISCUSSION AND FUTURE WORK

In this section, we summarize the main empirical findings of TRSD, discuss its limitations, and outline directions for future work.

Across all evaluated architectures and datasets, TRSD consistently improves robustness to truncated inference, with the largest gains observed in low-budget regimes where baseline models frequently fail to complete a useful reasoning trajectory. At the same time, we note that the observed magnitude of the gains can be sensitive to the interaction between model architecture and the dataset used to generate self-distillation prompts. Different models achieve their largest improvements when trained on different prompt distributions, and no single dataset is uniformly optimal across architectures. While we do not identify a single underlying cause for this behavior, a plausible explanation is that models respond differently to prompt-induced reasoning patterns depending on their prior training, which shapes how reasoning traces are generated.

6.1 LIMITATIONS AND FAILURE CASES

While TRSD consistently improves robustness under truncated inference, its gains naturally diminish as the available reasoning budget increases. In addition, improvements obtained through TRSD do not always translate uniformly across out-of-distribution prompt distributions. This sensitivity reflects the fact that TRSD optimizes robustness with respect to the reasoning patterns induced by the self-distillation prompts, and that different prompt distributions emphasize different reasoning structures. As a result, identifying suitable self-distillation prompts is an important factor in maximizing the benefits of the TRSD step, which we believe this work has only begun to explore.

6.2 FUTURE DIRECTIONS

Several directions could further improve the effectiveness and applicability of TRSD. First, alternative training strategies could be explored to mitigate the occasional performance inconsistencies observed at large reasoning budgets, for example, by adapting the distillation objective based on truncation level or teacher confidence. Such approaches may help better balance robustness to partial reasoning with performance under full-length inference. Second, our results indicate that the choice of self-distillation prompts influences downstream performance. Developing curated prompt datasets that elicit reasoning patterns that transfer reliably across tasks could improve the consistency of TRSD gains, particularly in out-of-distribution settings. Finally, while TRSD is applied here as a post-training step, robustness to truncated reasoning could be incorporated directly into the distillation process from larger teacher models, allowing robustness to partial reasoning to be learned jointly with standard distillation objectives rather than added as a separate training stage.

Overall, this work shows that robustness to truncated reasoning can be learned as a lightweight post-training property of existing reasoning models through self-distillation alone. By improving accuracy under limited reasoning budgets without introducing additional supervision or architectural changes, TRSD provides a practical approach for deploying reasoning models in settings where inference-time computation is constrained.

REFERENCES

- Wei-Rui Chen, Vignesh Kothapalli, Ata Fatahibaarzi, Hejian Sang, Shao Tang, Qingquan Song, Zhipeng Wang, and Muhammad Abdul-Mageed. Distilling the essence: Efficient reasoning distillation via sequence truncation. *arXiv preprint arXiv:2512.21002*, 2025a.
- Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT think that much for $2+3=?$ on the overthinking of long reasoning models. In *Forty-second International Conference on Machine Learning*, 2025b. URL <https://openreview.net/forum?id=MSbU3L7V00>.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*, 2025.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.
- Yaniv Leviathan, Matan Kalman, and Yossi Matias. Prompt repetition improves non-reasoning llms. *arXiv preprint arXiv:2512.14982*, 2025.
- Zhenghao Liu, Zhuoyang Wu, Xinze Li, Yukun Yan, Shuo Wang, Zulong Chen, Yu Gu, Ge Yu, and Maosong Sun. Long-chain reasoning distillation via adaptive prefix alignment. *arXiv preprint arXiv:2601.10064*, 2026.
- Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*, 2025.
- Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. Cot-valve: Length-compressible chain-of-thought tuning, 2025. URL <https://arxiv.org/abs/2502.9601>, 2025.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.
- Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2025. URL <https://arxiv.org/abs/2512.13961>.

- Charilaos Pipis, Shivam Garg, Vasilis Kontonis, Vaishnavi Shrivastava, Akshay Krishnamurthy, and Dimitris Papailiopoulos. Wait, wait, wait... why do reasoning models loop? *arXiv preprint arXiv:2512.12895*, 2025.
- Chenlong Wang, Yuanning Feng, Dongping Chen, Zhaoyang Chu, Ranjay Krishna, and Tianyi Zhou. Wait, we don't need to" wait"! removing thinking tokens improves reasoning efficiency. *arXiv preprint arXiv:2506.08343*, 2025.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*, 2025.
- Haoran Xu, Baolin Peng, Hany Awadalla, Dongdong Chen, Yen-Chun Chen, Mei Gao, Young Jin Kim, Yunsheng Li, Liliang Ren, Yelong Shen, et al. Phi-4-mini-reasoning: Exploring the limits of small reasoning language models in math. *arXiv preprint arXiv:2504.21233*, 2025.
- Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less, 2025a. URL <https://arxiv.org/abs/2502.18600>.
- Jianzhi Yan, Le Liu, Youcheng Pan, Shiwei Chen, Zike Yuan, Yang Xiang, and Buzhou Tang. From long to lean: Performance-aware and adaptive chain-of-thought compression via multi-round refinement. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12290–12306, 2025.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.
- Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.
- Hang Yuan, Bin Yu, Haotian Li, Shijun Yang, Christina Dan Wang, Zhou Yu, Xueyin Xu, Weizhen Qi, and Kai Chen. Not all tokens are what you need in thinking. *arXiv preprint arXiv:2505.17827*, 2025.
- Hongxuan Zhang, Zhining Liu, Yao Zhao, Jiaqi Zheng, Chenyi Zhuang, Jinjie Gu, and Guihai Chen. Fast chain-of-thought: A glance of future from parallel decoding leads to answers faster. *arXiv preprint arXiv:2311.08263*, 2023.
- Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. Lightthinker: Thinking step-by-step compression. *arXiv preprint arXiv:2502.15589*, 2025a.
- Yuxiang Zhang, Zhengxu Yu, Weihang Pan, Zhongming Jin, Qiang Fu, Deng Cai, Binbin Lin, and Jieping Ye. Tokensqueeze: Performance-preserving compression for reasoning LLMs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL <https://openreview.net/forum?id=Wc1VZ2bVJn>.

A IMPLEMENTATION DETAILS

A.1 TEACHER ANSWER PROCESSING

When generating outputs, the teacher model does not always strictly follow the prescribed format, namely a reasoning trace enclosed in `<think></think>` tags followed by a final answer enclosed in `<answer></answer>` tags. In addition, generation may terminate early due to exhaustion of the reasoning token budget. We therefore apply a deterministic post-processing procedure to sanitize teacher outputs before they are used for distillation. First, if the generated output does not begin with a `<think>` tag, we prepend one. If a closing `</think>` tag is missing, we append it at the end

of the generated text and mark the example for answer forcing. We then inspect the text following the final `</think>` tag. If no `<answer>` tag is present, we discard any trailing content and prompt the model to generate an answer by appending `<answer>` to the sanitized prefix, following the answer forcing procedure of Muennighoff et al. (2025). If an `<answer>` tag is present but the corresponding closing `</answer>` tag is missing, we similarly continue generation until the answer is completed. In all answer-forcing cases, we allocate a maximum budget of 200 tokens for answer generation. If multiple `<answer></answer>` pairs are present in the output, only the first pair is retained and all subsequent content is discarded. This procedure ensures that each training example contains a well-formed reasoning block and a single, clearly delimited answer segment.

A.2 TRAINING DETAILS

All models are fine-tuned with TRSD for 2000 optimization steps using the AdamW optimizer, with learning rate 3×10^{-6} , batch size 32, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, no weight decay, and gradient norm clipping at 1.0. We save intermediate checkpoints at steps 250, 500, 1000, and 2000 and evaluate each checkpoint using the protocol described in Section A.3. We use temperature 1 to generate reasoning and answers during training, and 0.8 at evaluation.

A.3 CHECKPOINT SELECTION

We use different checkpoint selection rules for in-distribution and out-of-distribution evaluation.

In-distribution. For each model and each training dataset, we select the checkpoint that achieves the best performance on the corresponding evaluation set of that same dataset (i.e., the dataset used to draw prompts during self-distillation). These checkpoints are used for all in-distribution results.

Out-of-distribution. For each model, we select a single checkpoint to be used across all out-of-distribution evaluations. Concretely, among the four candidate checkpoints (steps 250/500/1000/2000), we select the checkpoint that performs best on average across all four evaluation datasets, including the in-distribution dataset. These checkpoints are used for all out-of-distribution results.

Table 4 summarizes the checkpoints selected for in-distribution and out-of-distribution evaluation.

Table 4: Checkpoint selection summary. Columns corresponding to a specific dataset report the checkpoints achieving the best in-distribution performance for each (model, dataset) pair. The final column reports the checkpoint used for out-of-distribution evaluation, selected by best average performance across all four datasets; the dataset used for self-distillation is shown in parentheses. All entries denote the training step of the selected checkpoint.

Model	Countdown	Dolci	GSM8K	MATH500	OOD checkpoint
Qwen3-0.6B	500	2000	250	250	500 (Countdown)
Qwen3-1.7B	500	1000	250	250	250 (GSM8K)
Qwen3-4B	250	2000	250	250	250 (GSM8K)
Phi-4-mini-reasoning	1000	1000	500	500	500 (Countdown)
OpenThinker3-1.5B	1000	250	2000	1000	2000 (GSM8K)

B DATASETS

We evaluate TRSD on four reasoning benchmarks spanning arithmetic search, grade-school math word problems, and competition-style mathematics. For all datasets, prompts are formatted to request reasoning inside `<think></think>` tags and a final answer inside `<answer></answer>` tags (see Appendix A.1 for how we sanitize teacher outputs when this format is not followed).

Countdown. We use a preconstructed version of the Countdown arithmetic-construction task from the Jiayi-Pan/Countdown-Tasks-3to4 dataset on Hugging Face. Each example provides

a multiset of integers and a target value, and the model must construct an equation that reaches the target using basic arithmetic operations while using each number at most once. We subsample the dataset and split it into 45000 training prompts and 5000 test prompts. A typical prompt is:

```
Using the numbers 33, 5, 68, 29, create an equation that
equals 67. You can use basic arithmetic operations (+,
-, *, /) one or multiple times but each number can only be
used once. Show your work in the <think> </think> tags and
return the final equation in the <answer> </answer> tags,
for example <answer> (1 + 2) / 3 </answer>. Think step by
step inside <think> tags.
```

MATH. For competition-style mathematics, we use the MATH dataset of Hendrycks et al. (2021) via the EleutherAI/hendrycks_math release on Hugging Face. We draw the 7500 training prompts from the official training split and evaluate on the standard MATH500 subset (500 problems), following common practice for efficient evaluation. Prompts follow a standard instruction format:

```
Solve the math problem. Think step-by-step inside <think>
tags, then put only the final answer inside <answer> tags.
Problem: The point (a,b) lies on the line with the equation
3x + 2y = 12. When a = 4, what is the value of b?
```

GSM8K. We use GSM8K (Cobbe et al., 2021) through the openai/gsm8k Hugging Face dataset. GSM8K consists of grade-school math word problems requiring multi-step reasoning. The dataset is composed by 7470 training prompts and 1320 evaluation prompts. A typical prompt is:

```
Solve the math word problem. Think step-by-step inside
<think> tags, then put only the final integer answer inside
<answer> tags. Question: Darrell and Allen's ages are in
the ratio of 7:11. If their total age now is 162, calculate
Allen's age 10 years from now.
```

Dolci (math subset). We use the math portion of the allenai/Dolci-Think-RL-7B dataset, a collection of prompts designed to elicit deliberate reasoning (Olmo et al., 2025). We split the 24951 available examples into 22455 training and 2496 test prompts. An example prompt is:

```
user: What is the probability of such event happening:
Form a word by randomly choosing 2 letters from the multiset
x: 3, l: 4, shuffle the letters in the word, what is the
probability of no letter 'x' occupy any of their original
positions? If the probability can be written as the form
m/n, where m and n are relatively prime integers, find m + n.
Show your work in the <think> </think> tags and return the
final equation in the <answer> </answer> tags.
```

C EXTENDED RESULTS

We report here extended experimental results that complement the main findings in Section 4. In particular, we provide full in-distribution and out-of-distribution tables for additional model sizes and architectures, together with per-dataset plots that visualize performance as a function of the available reasoning budget. All results follow the same evaluation protocol and checkpoint selection strategy described in Section 4 and Appendix A.3.

C.1 EXTENDED TABLES

We first report extended numerical results for additional model families omitted from the main text. In-distribution performance under truncated inference for Qwen3-0.6B, Qwen3-1.7B, and OpenThinker3-1.5B is shown in Table 5.

Table 5: Baseline vs TRSD accuracy under truncated inference for smaller and mid-size models (in-distribution prompts). Entries report Baseline/TRSD accuracy (higher is better). Colored values report the absolute change in accuracy (TRSD – Baseline, percentage points). Bold indicates the better entry; when Baseline and TRSD differ by at most 1 percentage point, both are bold.

Budget / Data	Countdown		Dolci		GSM8K		Math500	
	Baseline	TRSD	Baseline	TRSD	Baseline	TRSD	Baseline	TRSD
<i>Teacher Model: Qwen3-0.6B</i>								
32	14.06	22.14 (+8.1)	3.37	4.45 (+1.1)	3.03	8.34 (+5.3)	7.20	11.20 (+4.0)
64	13.84	22.12 (+8.3)	3.89	4.69 (+0.8)	3.41	10.24 (+6.8)	6.40	11.00 (+4.6)
128	17.18	29.86 (+12.7)	3.37	5.17 (+1.8)	4.47	25.85 (+21.4)	7.40	11.60 (+4.2)
256	25.82	42.46 (+16.6)	3.93	5.57 (+1.6)	23.96	54.59 (+30.6)	12.40	22.60 (+10.2)
512	40.24	53.56 (+13.3)	4.81	5.69 (+0.9)	59.14	67.55 (+8.4)	31.80	37.00 (+5.2)
1024	52.54	61.22 (+8.7)	5.05	5.89 (+0.8)	70.81	72.71 (+1.9)	47.40	50.00 (+2.6)
2048	61.76	66.54 (+4.8)	6.85	5.97 (-0.9)	74.60	74.45 (-0.2)	60.80	54.40 (-6.4)
4096	68.44	68.58 (+0.1)	7.29	6.29 (-1.0)	76.35	75.36 (-1.0)	67.00	55.60 (-11.4)
8192	70.30	68.74 (-1.6)	6.77	6.13 (-0.6)	76.50	74.91 (-1.6)	70.40	56.60 (-13.8)
<i>Teacher Model: Qwen3-1.7B</i>								
32	10.68	25.98 (+15.3)	4.37	6.21 (+1.8)	6.82	15.09 (+8.3)	11.60	14.80 (+3.2)
64	13.50	26.70 (+13.2)	3.89	6.93 (+3.0)	6.90	17.36 (+10.5)	11.20	16.40 (+5.2)
128	19.40	30.20 (+10.8)	3.81	6.77 (+3.0)	9.40	32.75 (+23.4)	12.20	20.20 (+8.0)
256	39.24	42.94 (+3.7)	4.29	6.81 (+2.5)	33.06	65.88 (+32.8)	17.80	24.00 (+6.2)
512	55.60	57.38 (+1.8)	6.57	6.77 (+0.2)	69.67	81.65 (+12.0)	38.00	44.60 (+6.6)
1024	66.10	66.42 (+0.3)	8.17	8.21 (+0.0)	84.53	87.72 (+3.2)	60.00	64.60 (+4.6)
2048	72.60	71.76 (-0.8)	9.82	9.78 (-0.0)	88.25	88.70 (+0.5)	74.20	74.80 (+0.6)
4096	77.54	73.28 (-4.3)	12.78	10.66 (-2.1)	89.61	88.86 (-0.8)	80.40	79.20 (-1.2)
8192	80.16	73.32 (-6.8)	17.23	12.26 (-5.0)	89.92	89.16 (-0.8)	84.60	82.20 (-2.4)
<i>Teacher Model: OpenThinker3-1.5B</i>								
32	0.26	12.96 (+12.7)	3.73	4.81 (+1.1)	2.58	7.28 (+4.7)	3.40	13.20 (+9.8)
64	0.46	15.64 (+15.2)	4.49	4.89 (+0.4)	3.34	7.28 (+3.9)	6.00	13.20 (+7.2)
128	0.82	17.30 (+16.5)	4.37	4.77 (+0.4)	4.02	8.26 (+4.2)	6.20	15.00 (+8.8)
256	4.84	26.36 (+21.5)	3.37	5.49 (+2.1)	9.33	22.59 (+13.3)	11.20	19.00 (+7.8)
512	16.64	36.08 (+19.4)	4.65	5.93 (+1.3)	35.25	54.81 (+19.6)	24.80	29.00 (+4.2)
1024	28.28	44.42 (+16.1)	4.53	7.21 (+2.7)	57.47	66.87 (+9.4)	51.00	35.80 (-15.2)
2048	41.16	45.02 (+3.9)	6.49	8.93 (+2.4)	62.62	69.83 (+7.2)	60.40	37.60 (-22.8)
4096	43.16	43.28 (+0.1)	8.33	10.22 (+1.9)	53.90	71.04 (+17.1)	50.80	41.00 (-9.8)
8192	39.98	43.72 (+3.7)	12.22	12.98 (+0.8)	41.70	70.89 (+29.2)	38.00	43.20 (+5.2)

Across all three architectures, the same qualitative pattern observed in Section 4 holds. TRSD yields the largest gains in the heavily truncated regime, where only a small number of reasoning tokens are available and baseline models often fail to complete a useful reasoning trace. As the reasoning budget increases, the performance gap typically narrows, and in some cases reverses slightly at the largest budgets.

Out-of-distribution generalization results for the same models are reported in Table 6. Following the checkpoint selection procedure described in Appendix A.3, a single checkpoint per model is used for all out-of-distribution evaluations. Despite this restriction, TRSD-trained models retain most of their gains under truncated inference, indicating that robustness learned from partial reasoning transfers across prompt distributions.

Table 6: Baseline vs TRSD accuracy under truncated inference on out-of-distribution benchmarks (smaller/mid-size models). Each block reports a fixed TRSD source dataset (shown in the block header) and evaluates on the remaining benchmarks. Entries report Baseline/TRSD accuracy (higher is better). Colored values report the absolute change in accuracy (TRSD – Baseline, percentage points). Bold indicates the better entry; when Baseline and TRSD accuracies differ by at most 1 percentage point, both are bold.

<i>Teacher Model: Qwen3-0.6B — TRSD: Countdown</i>						
Budget / Data	Dolci		GSM8K		Math500	
	Baseline	TRSD	Baseline	TRSD	Baseline	TRSD
32	3.37	4.01 (+0.6)	3.03	5.23 (+2.2)	7.20	11.40 (+4.2)
64	3.89	4.69 (+0.8)	3.41	5.46 (+2.1)	6.40	11.80 (+5.4)
128	3.37	4.77 (+1.4)	4.47	12.36 (+7.9)	7.40	13.80 (+6.4)
256	3.93	5.09 (+1.2)	23.96	41.17 (+17.2)	12.40	21.60 (+9.2)
512	4.81	5.61 (+0.8)	59.14	64.22 (+5.1)	31.80	38.60 (+6.8)
1024	5.05	7.05 (+2.0)	70.81	70.36 (-0.5)	47.40	52.40 (+5.0)
2048	6.85	8.09 (+1.2)	74.60	72.78 (-1.8)	60.80	59.00 (-1.8)
4096	7.29	7.57 (+0.3)	76.35	73.16 (-3.2)	67.00	63.40 (-3.6)
8192	6.77	7.97 (+1.2)	76.50	73.01 (-3.5)	70.40	65.20 (-5.2)

<i>Teacher Model: Qwen3-1.7B — TRSD: GSM8K</i>						
Budget / Data	Countdown		Dolci		Math500	
	Baseline	TRSD	Baseline	TRSD	Baseline	TRSD
32	10.68	7.14 (-3.5)	4.37	8.77 (+4.4)	11.60	15.80 (+4.2)
64	13.50	9.64 (-3.9)	3.89	8.93 (+5.0)	11.20	16.20 (+5.0)
128	19.40	18.20 (-1.2)	3.81	9.29 (+5.5)	12.20	17.80 (+5.6)
256	39.24	39.64 (+0.4)	4.29	9.78 (+5.5)	17.80	25.80 (+8.0)
512	55.60	55.14 (-0.5)	6.57	10.98 (+4.4)	38.00	46.00 (+8.0)
1024	66.10	65.12 (-1.0)	8.17	12.94 (+4.8)	60.00	64.20 (+4.2)
2048	72.60	70.20 (-2.4)	9.82	15.30 (+5.5)	74.20	72.60 (-1.6)
4096	77.54	73.90 (-3.6)	12.78	19.91 (+7.1)	80.40	79.80 (-0.6)
8192	80.16	74.70 (-5.5)	17.23	25.40 (+8.2)	84.60	81.20 (-3.4)

<i>Teacher Model: OpenThinker3-1.5B — TRSD: GSM8K</i>						
Budget / Data	Countdown		Dolci		Math500	
	Baseline	TRSD	Baseline	TRSD	Baseline	TRSD
32	0.26	2.34 (+2.1)	3.73	4.93 (+1.2)	3.40	10.20 (+6.8)
64	0.46	2.90 (+2.4)	4.49	5.97 (+1.5)	6.00	10.00 (+4.0)
128	0.82	2.44 (+1.6)	4.37	6.33 (+2.0)	6.20	12.40 (+6.2)
256	4.84	7.78 (+2.9)	3.37	6.25 (+2.9)	11.20	15.80 (+4.6)
512	16.64	18.86 (+2.2)	4.65	7.25 (+2.6)	24.80	32.80 (+8.0)
1024	28.28	31.02 (+2.7)	4.53	9.33 (+4.8)	51.00	50.20 (-0.8)
2048	41.16	32.64 (-8.5)	6.49	10.78 (+4.3)	60.40	55.20 (-5.2)
4096	43.16	32.00 (-11.2)	8.33	13.10 (+4.8)	50.80	59.20 (+8.4)
8192	39.98	33.34 (-6.6)	12.22	16.59 (+4.4)	38.00	59.80 (+21.8)

Table 7: Average number of reasoning tokens conditioned on answer correctness for smaller models. Top block: in-distribution TRSD using per-dataset checkpoints selected as in Table A.3. Bottom block: out-of-distribution-style TRSD using a single fixed checkpoint per model (Countdown step 500 for Qwen3-0.6B, GSM8K step 250 for Qwen3-1.7B, GSM8K step 2000 for OpenThinker3-1.5B). Bold indicates the configuration (Baseline or TRSD) with lower average reasoning length.

Model	Outcome	Countdown		Dolci		GSM8K		Math500	
		Baseline	TRSD	Baseline	TRSD	Baseline	TRSD	Baseline	TRSD
<i>In-distribution evaluation</i>									
Qwen3-0.6B	Correct	1169	724	5655	3765	848	575	2460	1575
	Wrong	4719	3230	7295	5268	2183	1888	5606	4506
Qwen3-1.7B	Correct	1098	747	6633	4590	1005	710	2879	2227
	Wrong	6379	2482	7777	6071	3364	2501	5829	5633
OpenThinker3-1.5B	Correct	4814	2641	7765	7375	3679	1543	4142	3625
	Wrong	6517	5951	7999	7788	4720	3143	4692	3571
<i>Out-of-distribution evaluation (fixed TRSD checkpoint)</i>									
Qwen3-0.6B	Correct	1169	725	5655	3833	848	555	2460	1738
	Wrong	4719	3230	7295	5750	2183	1308	5606	4371
Qwen3-1.7B	Correct	1098	915	6633	6362	1005	710	2879	2283
	Wrong	6379	4518	7777	7512	3364	2501	5829	5055
OpenThinker3-1.5B	Correct	4814	2580	7765	6806	3679	1543	4142	2876
	Wrong	6517	5017	7999	7642	4720	3143	4692	3951

C.2 REASONING LENGTH ANALYSIS FOR SMALLER MODELS

We complement the main analysis in Section 4.3 by examining how TRSD affects inference-time reasoning length for smaller model architectures. Table 7 reports the average number of reasoning tokens generated under unconstrained inference ($r_{\max} = 8192$), conditioned on answer correctness, for Qwen3-0.6B, Qwen3-1.7B, and OpenThinker3-1.5B.

We report in the top part results using per-dataset in-distribution checkpoints, following the same selection procedure as in the main experiments, while in the bottom part we fix a single TRSD checkpoint per model across all datasets, mirroring the out-of-distribution evaluation protocol. Across all three models and datasets, the same qualitative pattern observed for larger architectures persists, where TRSD reduces reasoning length for both correct and incorrect predictions across nearly all settings.

C.3 PER-DATASET PERFORMANCE PLOTS

To complement the aggregate tables, Figures 6, 7 and 8 visualize per-dataset performance as a function of the available reasoning budget. These plots make two consistent behaviors particularly clear. First, the advantage of TRSD is concentrated in the low-budget regime. Across datasets, TRSD curves typically rise earlier or degrade more slowly as the budget decreases, indicating that distilled models can recover correct predictions from much shorter prefixes of the reasoning trace. In contrast, baseline models often exhibit sharp transitions in accuracy as the reasoning budget increases. Second, the plots highlight that TRSD does not uniformly improve performance across all budgets. At moderate and large budgets, TRSD-trained models generally approach the baseline, and in some cases plateau below it at the largest budget.

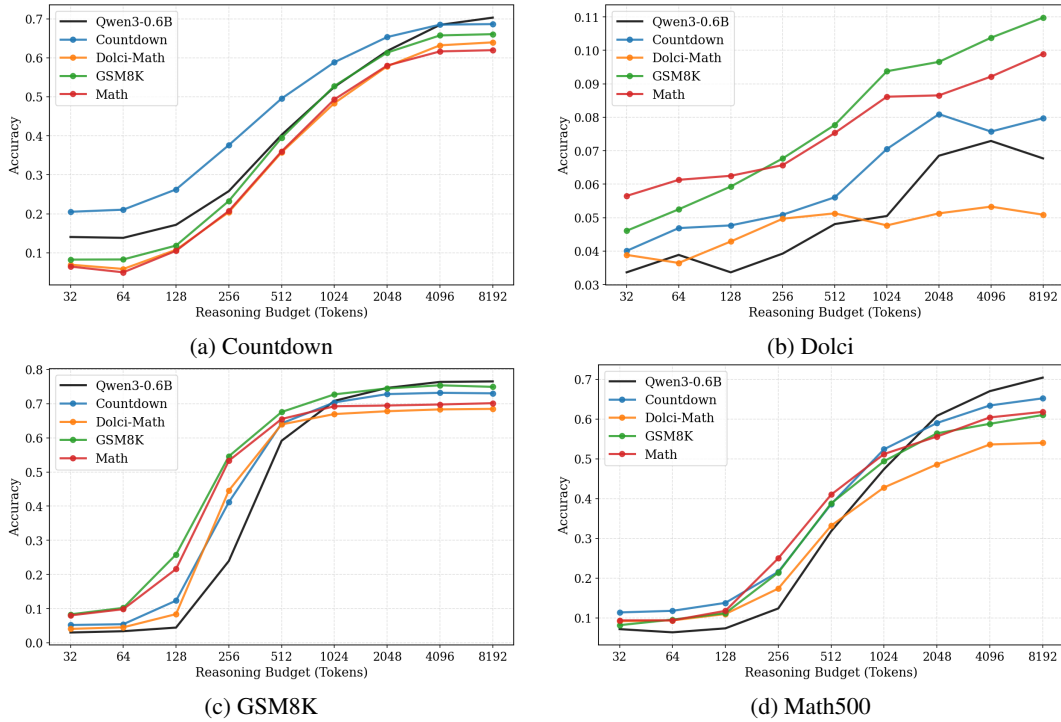


Figure 6: Per-dataset accuracy as a function of the reasoning budget for Qwen3-0.6B. The evaluation dataset is specified below the respective plot.

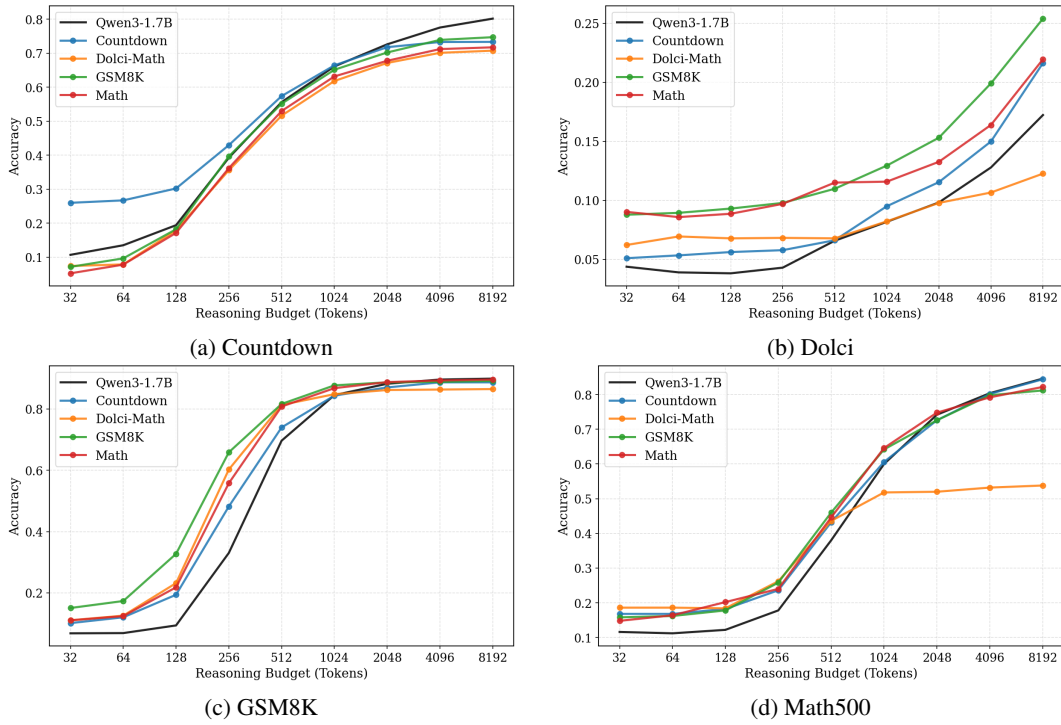


Figure 7: Per-dataset accuracy as a function of the reasoning budget for Qwen3-1.7B. The evaluation dataset is specified below the respective plot.

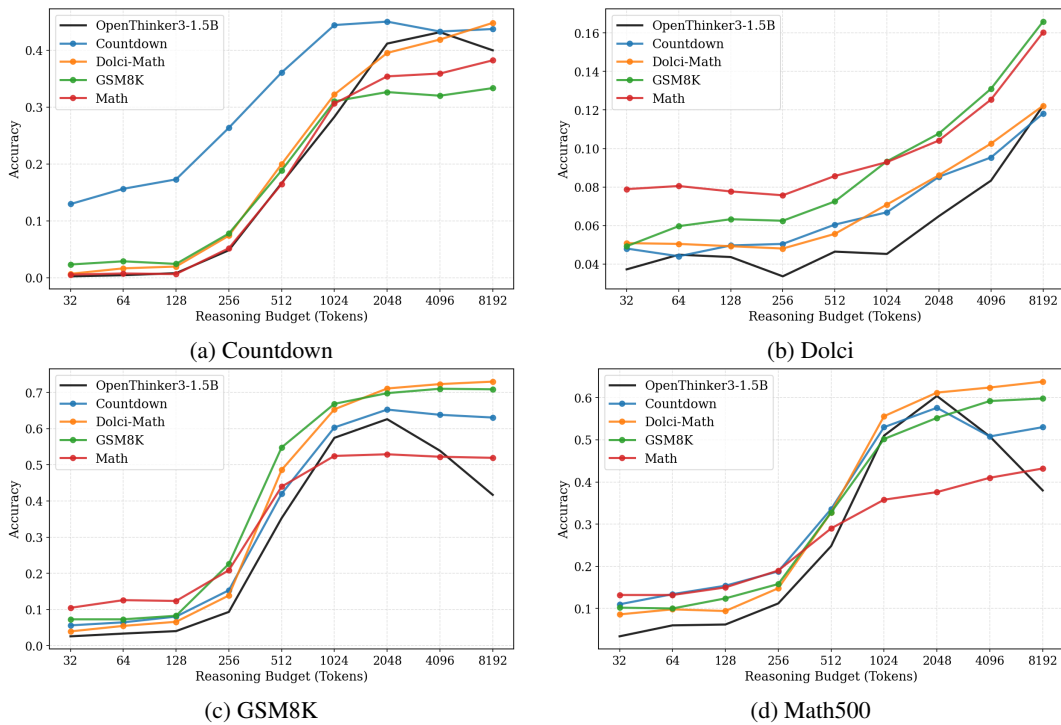


Figure 8: Per-dataset accuracy as a function of the reasoning budget for OpenThinker3-1.5B. The evaluation dataset is specified below the respective plot.

D ADDITIONAL QUALITATIVE EXAMPLES

We provide additional qualitative comparisons between baseline and TRSD-trained models. We consider two complementary regimes: cases where *both models answer correctly*, and cases where *both models fail on the same prompt*. These examples are intended to illustrate differences in reasoning behavior and representative shared failure modes, rather than to suggest systematic correctness improvements beyond those reported in the main paper. In both cases, we use Qwen3-4B as baseline and the TRSD version trained with prompts from GSM8K.

Both models correct. Figure 9 shows a representative example where both the baseline and TRSD-trained models produce the correct answer on the Countdown dataset. While correctness is preserved, the TRSD-trained model typically reaches the solution using a shorter and more focused reasoning trace, whereas the baseline exhibits additional intermediate steps or redundant self-verification. This behavior mirrors the quantitative trends reported in Section 4, where TRSD reduces reasoning length without degrading accuracy.

Both models incorrect. Figure 10 presents a case where both models fail on the same prompt from GSM8K. These examples illustrate shared failure modes, such as semantic ambiguity or invalid implicit assumptions, that are not directly addressed by reasoning truncation.

Qualitative comparison of reasoning traces (Qwen3-4B, G8M8K)

Prompt. Using the numbers 59, 92, 24, 62, create an equation that equals 53. You can use basic arithmetic operations (+, -, *, /) one or multiple times but each number can only be used once.

Baseline (Qwen3-4B)**Think:**

Okay, let's see. I need to use the numbers 59, 92, 24, and 62 exactly once each, and combine them with basic arithmetic operations to make 53. Hmm. Let me start by thinking about possible operations.

First, maybe I can subtract some numbers. Let's see, 92 is a big number. If I subtract something from 92, maybe that could help. Let me try 92 minus something. For example, 92 minus 59 is 33. But then I have 24 and 62 left. Not sure how to combine those. Maybe 24 plus 62 is 86. 33 and 86... that's way over. Not helpful.

Alternatively, maybe 62 minus something. Let's see. If I take 62 and subtract 24, that's 38. Then maybe 92 minus 39 is 33. 38 and 33... not helpful. Hmm.

Wait, maybe I need to use division or multiplication. Let me think. For example, 59 minus something. If I can get 59 minus a number that equals 53, but that would require subtracting 6, but I don't have a 6. Alternatively, maybe combining numbers through operations. What if I do 92 minus 62? That's 30. Then maybe 59 minus 24 is 35. 30 and 35... not helpful. Or 30 plus 35 is 65. Not 53.

Another approach: Maybe multiply two numbers and then subtract others. Let's see. 24 times something. 24 times 2 is 48, but I don't have a 2. Alternatively, 24 times 59? That's way too big. 24*59 is 1416. Not helpful.

How about division? Let's see. Maybe 92 divided by something. 92 divided by 2 is 46, but again, no 2. Wait, 62 divided by something. Maybe 62 divided by 24? That's around 2.58. Not sure.

Alternatively, maybe 59 minus (something). Let's think. If I can get 59 minus (something) equals 53, that something would be 6. But how to get 6 from 92, 24, and 62? Let me try 92 minus 62 is 30. Then 30 minus 24 is 6. Oh! Wait, that's 92 - 62 = 30, then 30 - 24 = 6. So then 59 - 6 = 53. But wait, that uses 92, 62, 24, and 59. Each number is used once. Let me check:

So $59 - ((92 - 62) - 24) = 59 - (30 - 24) = 59 - 6 = 53$. Yes! That works. Let me verify the operations step by step.

First, compute $92 - 62 = 30$. Then subtract 24 from that: $30 - 24 = 6$. Then subtract that result from 59: $59 - 6 = 53$. Yes! So that uses all four numbers once each. Let me write that as an equation.

So the equation would be $59 - ((92 - 62) - 24) = 53$. Alternatively, simplifying the parentheses, it's $59 - 92 + 62 - 24$. Wait, let me check that. Because $(92 - 62)$ is 30, then subtract 24 gives 6. So $59 - 6$ is 53.

Alternatively, expanding the parentheses: $59 - 92 + 62 - 24$. Let me compute that: $59 - 92$ is -33, plus 62 is 29, minus 24 is 5. Wait, that's not 53. Wait, that can't be right. Wait, what's wrong here?

Wait, no. The original expression is $59 - ((92 - 62) - 24)$. Which is $59 - (92 - 62 - 24)$. Let's compute the inner part first. $92 - 62$ is 30. Then subtract 24: $30 - 24$ is 6. Then $59 - 6$ is 53. So that's correct. But if I expand the parentheses, it's $59 - 92 + 62 - 24$? Wait, no. Because $(92 - 62) - 24$ is $92 - 62 - 24$. So the entire expression is $59 - (92 - 62 - 24) = 59 - 92 + 62 + 24$? Wait, no. Wait, subtracting a sum is the same as adding the negatives. So $59 - (92 - 62 - 24) = 59 - 92 + 62 + 24$. Let me compute that: $59 - 92$ is -33, plus 62 is 29, plus 24 is 53. Oh right! So that's correct. So the equation can be written as $59 - 92 + 62 + 24$. But wait, that would be $59 - 92$ is -33, plus 62 is 29, plus 24 is 53. Yes. So that's another way to write it. But in that case, the original expression is $59 - ((92 - 62) - 24)$. Which simplifies to $59 - 92 + 62 + 24$. But that uses all four numbers once. So that's valid.

Alternatively, maybe the user wants the equation to be written with parentheses. So the answer would be $59 - ((92 - 62) - 24) = 53$. But I need to check if that's allowed. The problem says each number can be used once. So yes, 59, 92, 62, 24 are all used once. So that works.

Alternatively, maybe there's another way. Let me check. For example, $62 - 24 = 38$. $92 - 59 = 33$. $38 + 33 = 71$. Not helpful. Or $92 - 59 = 33$. $62 - 24 = 38$. $33 + 38 = 71$. Still no. How about $92 - (62 + 24) = 92 - 86 = 6$. Then $59 - 6 = 53$. Oh! That's another way. So that would be $59 - (92 - (62 + 24)) = 53$. Let's check: $62 + 24 = 86$. $92 - 86 = 6$. $59 - 6 = 53$. So that's another valid equation. Which one is better? Both are correct. Either way.

So the equation can be written as $59 - (92 - (62 + 24)) = 53$. Or $59 - ((92 - 62) - 24) = 53$. Either is acceptable.

Alternatively, maybe using multiplication or division. Let me think. For example, 24 divided by something. $24 / (\text{something})$ to get a number. But I don't see a straightforward way. Let me try. Suppose I do $92 - (59 + 24) = 92 - 83 = 9$. Then $62 - 9 = 53$. But that would be $62 - (92 - (59 + 24)) = 53$. Let me check: $59 + 24 = 83$. $92 - 83 = 9$. $62 - 9 = 53$. So that's another way. But that uses all numbers once. So $62 - (92 - (59 + 24)) = 53$. That's another valid equation.

So there are multiple correct answers. The problem just requires one. So any of these would work. Let me pick one. The first one I thought of was $59 - ((92 - 62) - 24) = 53$. Alternatively, the second one $59 - (92 - (62 + 24)) = 53$. Either is correct.

But let me check if the numbers are all used once. In the first case: 59, 92, 62, 24. Yes. In the second case, same numbers. So either is correct. The answer can be written in any of these forms. So perhaps the simplest way is to write $59 - (92 - (62 + 24)) = 53$. Wait, but that would be $59 - (92 - 62 - 24)$. Let's compute that. $92 - 62$ is 30, minus 24 is 6. $59 - 6$ is 53. So that's the same as before.

Alternatively, $59 - 92 + 62 + 24$. Which is also correct. But the problem says to use the numbers once each. So that's acceptable.

But maybe the problem expects parentheses to show the order of operations. So perhaps the answer is written as $59 - (92 - 62 - 24) = 53$. Or maybe $59 - ((92 - 62) - 24) = 53$. Either is fine.

I think that's the correct approach. So the final answer is $59 - (92 - 62 - 24) = 53$. Let me check again:

Compute $92 - 62 = 30$. Then $30 - 24 = 6$. Then $59 - 6 = 53$. Correct.

Alternatively, $92 - 62 - 24 = 6$. $59 - 6 = 53$. Yes. So that's valid. So the equation is $59 - (92 - 62 - 24) = 53$. Or written with parentheses as needed.

So the answer would be that equation. Let me make sure I didn't miss any other possible ways. Another way could be $62 - (92 - (59 + 24)) = 53$. Let me check: $59 + 24 = 83$. $92 - 83 = 9$. $62 - 9 = 53$. Yes. So that's another valid equation. But the problem didn't specify any preference, so any of these is acceptable.

But the first one I found is probably easier. So I think the answer is $59 - (92 - 62 - 24) = 53$.

Answer: $59 - (92 - 62 - 24)$

TRSD (G8M8K)**Think:**

Okay, let's see. I need to use the numbers 59, 92, 24, and 62 exactly once each with basic arithmetic operations to make 53. Hmm. Let me start by thinking about possible operations.

First, maybe subtract some numbers. Let's see, 92 minus 59 is 33. Then 62 minus 24 is 38. Not sure how that helps. Wait, maybe combining them differently. Let me try adding some numbers and subtracting others.

What if I take 92 and subtract 62? That gives 30. Then maybe add 24? $30 + 24 = 54$. Close to 53, but one off. Oh, but I still have 59 left. Wait, but I can't use numbers more than once. Wait, I already used 92, 62, and 24. So 59 is left. Maybe that's not the right path.

Alternatively, let's try 59 minus something. Let's see, $59 - (92 - 62) = 59 - 30 = 29$. Then adding 24? $29 + 24 = 53$. Wait, but that would use 59, 92, 62, and 24. Each exactly once. Let me check that.

So $92 - 62$ is 30. Then $59 - 30$ is 29. Then $29 + 24$ is 53. Wait, but how to structure that into an equation using all numbers once. Let me write it step by step:

First, $92 - 62 = 30$

Then $59 - 30 = 29$

Then $29 + 24 = 53$.

But how to combine these steps into one equation. Let's see:

$(59 - (92 - 62)) + 24 = 53$.

Yes, that uses all numbers once. Let me verify:

Compute inside the parentheses first: $92 - 62$ is 30. Then $59 - 30$ is 29. Then add 24: $29 + 24 = 53$. Perfect. So the equation would be $(59 - (92 - 62)) + 24 = 53$.

Alternatively, maybe there's another way, but this seems to work. Let me check if each number is used once: 59, 92, 62, 24. Yes, all four numbers are used once each. The operations are subtraction and addition. So that's valid.

Answer: $(59 - (92 - 62)) + 24$

Figure 9: Example where both models answer correctly on a prompt from the Countdown dataset.

Qualitative comparison of failure cases (Qwen3-4B, GSM8K)

Prompt. Susan orders 3 magazines that send 12 issues a year. She has 1 magazine that sends 6 issues a year. Her last magazine sends her 4 times the amount of the 6 issue magazine. How many magazines does she get every year?

Correct answer: 66

Baseline (Qwen3-4B)

Think:

Okay, let's see. Susan has three magazines. Wait, the problem says she orders 3 magazines that send 12 issues a year. Then there's another one that sends 6 issues a year. And her last magazine sends 4 times the amount of the 6-issue magazine. Hmm, need to figure out how many magazines she gets every year.

Wait, first, let me parse the question again. The first part says Susan orders 3 magazines that send 12 issues a year. So that's three magazines, each with 12 issues. Then she has 1 magazine that sends 6 issues a year. So that's another magazine. Then her last magazine sends her 4 times the amount of the 6-issue magazine. So the last magazine is 4 times 6, which is 24 issues.

Wait, but the question is asking how many magazines does she get every year? Wait, maybe I need to check if the question is asking for the total number of magazines or the total number of issues? Wait, the wording says "how many magazines does she get every year?" So maybe it's the total number of magazines she has? Let me check again. Original question: "How many magazines does she get every year?" So the answer is the total number of magazines she has. Let me check the problem again. Susan orders 3 magazines that send 12 issues a year. Then she has 1 magazine that sends 6 issues a year. Then her last magazine sends 4 times the amount of the 6-issue magazine. So, the total number of magazines would be $3 + 1 + 1 = 5$? Because the first three are 12 issues each, then one is 6, and another one is 4 times 6, which is 24. So total magazines: $3 + 1 + 1 = 5$. But wait, maybe the wording is tricky. Let me check again.

Wait, the problem says she orders 3 magazines that send 12 issues a year. Then she has 1 magazine that sends 6 issues. Then her last magazine sends 4 times the amount of the 6-issue magazine. So the last magazine is another one. So total number of magazines is $3 + 1 + 1 = 5$. But maybe the problem is asking for the total number of issues? Wait, but the question says "how many magazines does she get every year?" So the answer is 5. But let me check again.

Alternatively, maybe the problem is phrased in a way that the three magazines are part of the total. Let me re-express:

Original problem:

Susan orders 3 magazines that send 12 issues a year. She has 1 magazine that sends 6 issues a year. Her last magazine sends her 4 times the amount of the 6 issue magazine. How many magazines does she get every year?

So, the first part: 3 magazines (each 12 issues). Then she has 1 magazine (6 issues). Then her last magazine is 4 times the 6-issue magazine. So that's $4 * 6 = 24$ issues. So the last magazine is another one. So total number of magazines is $3 + 1 + 1 = 5$. Therefore, the answer is 5. But let me check again if I'm misinterpreting the problem.

Alternatively, maybe the "last magazine" is part of the three she ordered? Wait, no. The problem says she orders 3 magazines that send 12 issues a year. Then she has 1 magazine that sends 6 issues. Then her last magazine sends 4 times the 6-issue. So that's an additional magazine. So total is $3 + 1 + 1 = 5$. Therefore, the answer is 5. But maybe the problem is asking for the total number of issues? Wait, the question says "how many magazines does she get every year?" So it's the number of magazines, not issues. So answer is 5. But let me check again.

Wait, maybe the problem is in the wording. The first sentence says she orders 3 magazines that send 12 issues a year. Then she has 1 magazine that sends 6 issues. Then her last magazine sends 4 times the amount of the 6 issue magazine. So total magazines: $3 + 1 + 1 = 5$. So the answer is 5. So I think that's correct. But maybe the problem is trying to trick me? Let me check again.

Alternatively, maybe the "last magazine" is part of the three she ordered? But the problem says she has 1 magazine that sends 6 issues. So that's separate. Then the last magazine is another one. So total of $3 + 1 + 1 = 5$. Yes. Therefore, the answer is 5.

Answer: 5

TRSD (GSM8K)

Think:

Okay, let's break this down. Susan has three types of magazines. The first part says she orders 3 magazines that send 12 issues a year each. Then there's one magazine that sends 6 issues a year. The last magazine sends 4 times the amount of the 6-issue magazine.

So, the first three magazines: 3 magazines * 12 issues = 36 issues. The next one is 6 issues. The last one is 4 times 6, which is 24 issues. Adding those together: $36 + 6 + 24 = 66$. Wait, but the question asks how many magazines she gets every year. Wait, the total number of magazines is $3 + 1 + 1 = 5$? Wait, no. Wait, the problem says "how many magazines does she get every year?" Wait, the issues per magazine, but the question is about the number of magazines. Wait, the problem says she orders 3 magazines that send 12 issues each, then 1 that sends 6, and one that sends 24 (4 times 6). So total number of magazines is $3 + 1 + 1 = 5$. Wait, but the answer is 5? But the problem says "every year" so maybe the answer is 5. But let me check again.

The first part: 3 magazines that send 12 issues a year. Then one magazine that sends 6, and another that sends 4 times 6 (which is 24). So the total number of magazines is $3 + 1 + 1 = 5$. So the answer is 5.

Answer: 5

Figure 10: Example where both baseline and TRSD-trained models fail on the same prompt, illustrating a shared failure mode.