ON THE ELICITATION AND PERFORMANCE OF COMPRESSED CHAIN-OF-THOUGHT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

We study how to compress the reasoning (thinking) that language models produce before they respond, comparing prompting, supervised fine-tuning, and reinforcement learning on reasoning models. For fine-tuning, we train each of three models on telegraphic rewrites of its own reasoning traces and compare it against a control trained on the original traces for the same problems. Fine-tuning reliably elicits compressed reasoning: the share of function words falls from 18–37% to under 2%, and reasoning output becomes 3.5–8.5× shorter. Compressed reasoners are far stronger than truncated reasoning at equal length, but in unconstrained settings they perform worse than the matched control, a gap that 20× more training data narrows but does not close. Prompting with the same specification used to rewrite the source traces does not compress the reasoning of two of the three models, and reinforcement learning with a function-word reward shortens reasoning without changing its style. Overall, we find that compressed reasoning is useful only at very small token budgets; otherwise, disabling reasoning achieves higher accuracy while still using far fewer tokens than full reasoning.

1 INTRODUCTION

Prompting language models to reason in natural language before answering, known as chain of thought (CoT), improves performance on multi-step tasks (Wei et al., 2022; Kojima et al., 2022). Reasoning traces can also be combined with external actions such as tool use (Yao et al., 2023). Recent reasoning models produce extended chains of thought, with reinforcement learning shown to elicit reasoning behavior (DeepSeek-AI, 2025). Generating these traces increases output length and inference cost, motivating methods that trade reasoning length against performance (Chen et al., 2024; Sui et al., 2025). This trade-off has prompted work on reducing explicit reasoning while retaining its benefits (Xu et al., 2025; Han et al., 2024; Adeyemi et al., 2026).

This work studies methods of producing reasoning models with more efficient chain-of-thought, motivated by reports of reasoning chains in unnatural language, such as improper grammar and dropping vestigial words that largely do not affect the meaning of the sequence (Jose, 2025; Trevor050, 2026; aussetg, 2026; htihle, 2026). These reports range from a study of illegible reasoning in RL-trained models to user-posted examples and discussion of terse, fragmentary traces; the latter are anecdotal and their provenance is not independently established. Intuitively, the same meaning can often be conveyed in sentences of very different lengths: two people without a fluent shared language can still communicate by stating subjects, objects and verbs, leaving the listener to infer the rest. In particular, recent attention has been brought to the benefit of prompting to compress chain-of-thought into "caveman" vernacular (Adeyemi et al., 2026), language stripped of function words, to reduce the cost of using a language model. We refer to this style of reasoning as a compressed, or telegraphic, *register*.

We study three approaches to eliciting this register in language models, and evaluate the cost and performance under different conditions. First, we prompt for the register, both with Chain-of-Draft (Xu et al., 2025) and with an explicit specification of the register. We then perform supervised fine-tuning of models on compressed reasoning traces, where the examples are produced by another model compressing each model's chain-of-thought, following work on linguistic and trace compression (Adeyemi et al., 2026; Kang et al., 2025). These fine-tuned models are compared against a control, which is fine-tuned on the uncompressed thinking traces. Finally, we test whether rein-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

forcement learning with length and function-word rewards discovers the register without examples, building on length-regularized reasoning objectives (Aggarwal & Welleck, 2025). The following are our main findings:

- **The register is learnable, at a cost in accuracy.** Fine-tuning makes every model adopt it, with function-word frequency falling below 2% and reasoning becoming 3.5–8.5$\times$ shorter, but it costs 5.4–22.7 points of GSM8K accuracy relative to the matched control. The cost survives crediting every unfinished compressed trace as correct, replicates on a second set of training problems, and shrinks but persists with 20$\times$ more training data.
- **A matched control is necessary.** Fine-tuning on native traces alone changes accuracy by -9.6 to $+1.7$ points depending on the model; comparing against the untrained model would attribute this to compression.
- **Only fine-tuning elicits the register.** Prompting with the register specification leaves the reasoning of two of three models unchanged, and reinforcement learning with a function-word reward shortens reasoning without changing its style.
- **Compressed models win only at the lowest budgets.** At equal reasoning length they beat truncated native reasoning by 35–53 points, but disabling reasoning, Chain-of-Draft prompting on two of three models, and correctness-only reinforcement learning all reach higher accuracy at 1.3–3.5$\times$ the tokens.

2 Related work

Prompting for concise reasoning. A direct way to reduce inference cost is to request shorter reasoning at test time via prompting. Early studies report mixed, task-dependent effects: concise chain-of-thought prompting reduced response length with small average accuracy changes in some settings, while reasoning-step-length experiments found that compressing demonstrations can hurt and that the value of additional steps depends on task difficulty (Renze & Guven, 2024; Jin et al., 2024). Chain-of-Draft asks for a minimal draft at each step (Xu et al., 2025), while token-budget methods such as TALE estimate or specify a per-question budget (Han et al., 2024). CAVEWOMAN studies linguistic constraints at different levels, from telegraphic output to keyword-only responses (Adeyemi et al., 2026). More broadly, a systematic study of compression prompts finds a question-specific threshold below which accuracy falls, motivating adaptive rather than uniform budgets (Lee et al., 2025). Our prompting baselines are Chain-of-Draft and our rewrite specification used directly as a prompt; that specification is a stricter variant of CAVEWOMAN's telegraphic prompt.

Training shorter traces. Training-based approaches aim to learn concise reasoning. They differ in what they preserve and how they represent the shorter trace. TokenSkip uses token-importance estimates to prune a trace and trains a model conditioned on a requested compression ratio (Xia et al., 2025; Pan et al., 2024). CRISP instead uses intrinsic attention signals to identify and prune redundant content (Lan et al., 2026). Generative approaches rewrite or compress the reasoning: C3oT trains on paired long and short traces with a condition that selects the desired mode (Kang et al., 2025), while our method fine-tunes on rewritten traces as the default target and compares against a matched model trained on its native traces. CoT-Valve also targets controllable lengths, learning a parameter-space direction and using progressively shorter traces to tune a model for length control (Ma et al., 2025). Recent controlled analysis emphasizes that the effects of pruning and rewriting depend on granularity, domain, and compression budget, and that concise supervision does not always translate into shorter generations at inference (Lyu et al., 2026). These methods share the goal of reducing explicit reasoning, but differ from our focus on a particular linguistic register.

Reward-based approaches. Reinforcement-learning methods shape reasoning length through reward or budget constraints. L1 uses length-controlled policy optimization to train a model to follow a length budget supplied at inference, while s1 uses budget forcing to manage test-time reasoning (Aggarwal & Welleck, 2025; Muennighoff et al., 2025). Our small-scale experiments instead test whether a correctness-gated length bonus, with and without a function-word-frequency bonus, elicits the target register.

3 METHOD

3.1 DATA CURATION FOR SUPERVISED FINE-TUNING

Models and data. We study Qwen3-1.7B (Qwen3) (Qwen Team, 2025), DeepSeek-R1-Distill-Qwen-7B (DeepSeek-7B) (DeepSeek-AI, 2025) and Gemma-4-E2B-it (Gemma) (Gemma Team, 2026) with its vision and audio towers removed. Each has been trained with explicit native reasoning (`<think>` tags or similar). Training problems come from the GSM8K training split (Cobbe et al., 2021), excluding a 500-problem development split used for configuration selection. Each model generates its own trace for each training problem by greedy decoding with a per-model reasoning cap (750, 500 and 900 tokens for Qwen3, DeepSeek-7B and Gemma), after which an unfinished trace is closed and the model is given 40 tokens to answer. We keep a trace only if its final answer is correct.

Compressing chain-of-thought traces. Qwen2.5-7B-Instruct (Qwen Team, 2024) is used to rewrite each verified trace under a fixed specification for thought compression (Appendix B). The specification is similar to that in CAVEWOMAN (Adeyemi et al., 2026), with some additional instructions to drop verbs in certain cases, vestigial adjectives, and keep every intermediate numeric result in its original order. A rewrite passes if (i) its stated answer matches the original (gold) answer, (ii) at least half of the numbers in the native trace appear in it, and (iii) it has at least five words.

Appendix C shows a native trace and its rewrite. On the training problems, rewriting shortens traces from 433, 92 and 169 words (Qwen3, DeepSeek-7B, Gemma) to 28, 28 and 33, and lowers the frequency of function words from 36.7%, 35.0% and 17.6% to 1–2%.

Problem set compilation. We take 250 problems whose rewrite passed for all three models. Every model trains on the same 250 problems, each with its own native trace (control) or its own rewrite (compressed). Because a problem must be solved and passably rewritten for all three models, this set is easier than GSM8K at large; the selection applies equally to both arms of every model. Notably, we do not expect model performance to improve by training on problems that were already solvable; we are interested in eliciting compressed thinking.

3.2 TRAINING DETAILS

We train on either the native traces (control) or the compressed traces as (question, reasoning+answer) pairs without additional system prompting or instructions, computing the cross-entropy loss over the reasoning and answer. Qwen3 is fully fine-tuned (learning rate 10^{-5}); DeepSeek-7B and Gemma use LoRA (Hu et al., 2022) (rank 16, α=32, learning rate 10^{-4}). All runs use three epochs (Appendix A). The two trained arms of each model share every setting except the reasoning text and, for memory reasons, the split of the effective batch between micro-batch and accumulation.

For each model this gives three arms: *untrained*, *control* and *compressed*. We report the *data effect*, control minus untrained, and the *compression effect*, compressed minus control.

3.3 EVALUATION

We evaluate on the 1,319 GSM8K test set with greedy decoding. The generation budget for each model is set by the longest arm: 4,000 tokens for Qwen3, 1,500 for Gemma, and 1,500 for untrained DeepSeek-7B and 500 for its trained arms, which finish well within 500 tokens. The answer is parsed from the response prioritizing explicit answer markers and falls back to the final number in the response. Additionally, we report three metrics of interest: *mean reasoning tokens* under each model's tokenizer; the *unfinished rate*, the share of problems whose reasoning block never closes within the budget; and the *function-word frequency* of the reasoning, the fraction of words drawn from a fixed list of 73 articles, prepositions, conjunctions, auxiliaries and pronouns.

3.4 REINFORCEMENT LEARNING

We train Qwen3-1.7B and DeepSeek-7B with Group Relative Policy Optimization (GRPO) (Shao et al., 2024) on 60 GSM8K training problems, using six samples per problem. Gemma is excluded

due to budget constraints. The correctness-only control starts from the base model and runs for 180 steps. The length-reward arm similarly starts from the base model and is trained for 60 steps on a similar correctness reward with an added bonus for shorter length completions. The register arm branches from the length-reward checkpoint and runs for 180 additional steps, adding a function-word-frequency bonus. Exact reward definitions and optimization settings are in Appendix F.

4 RESULTS

4.1 SUPERVISED FINE-TUNING ELICITS COMPRESSED THINKING AT THE COST OF PERFORMANCE

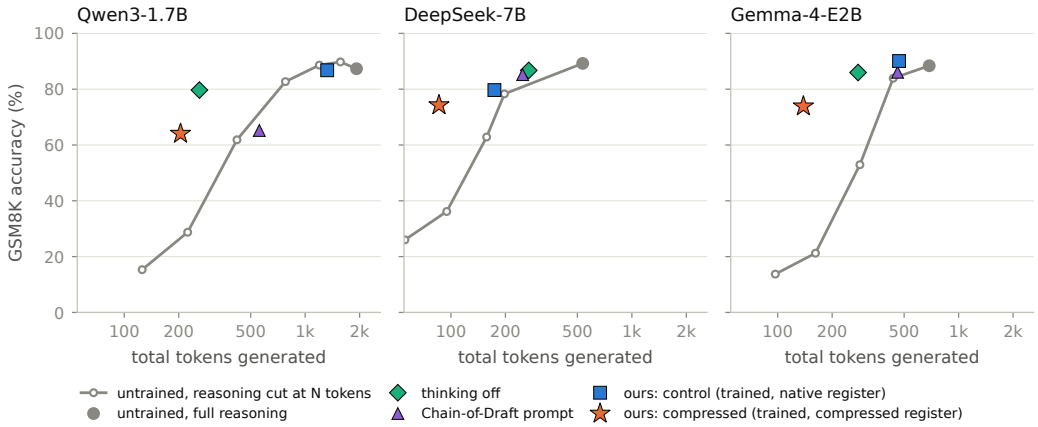

Figure 1: Accuracy against total generated tokens on GSM8K. The gray line shows untrained reasoning cut at different budgets; points compare thinking-off, Chain-of-Draft prompting, and the trained arms. At the lowest token counts, our compressed models are the most accurate.

The performance of our supervised fine-tuning arms is compared to several baselines in Figure 1. The compressed models use fewer tokens than their native-trace controls, with lower accuracy, while the untrained truncation curve falls well below the compressed results at similar token budgets. At the compressed models' token counts, neither truncation nor Chain-of-Draft prompting matches their accuracy, although disabling reasoning and Chain-of-Draft are more accurate at higher cost (Section 4.2).

Table 1: Supervised fine-tuning on compressed traces decreases accuracy more than fine-tuning on the model's native traces. GSM8K test accuracy with greedy decoding; trained arms are the mean of three seeds, ± the standard deviation across seeds. "from data" is control minus untrained; "from compression" is compressed minus control.

Model	untrained	control	compressed	from data	from compression
Qwen3-1.7B	87.3%	86.8% ±0.58	64.1% ±2.19	-0.6pp	-22.7pp
DeepSeek-7B	89.2%	79.7% ±1.25	74.3% ±0.59	-9.6pp	-5.4pp
Gemma-4-E2B	88.4%	90.1% ±0.39	73.9% ±0.79	+1.7pp	-16.2pp

Tables 1 and 2 give additional details on the supervised fine-tuning arms, and Figure A2 plots the same decomposition. The compressed arms reach the target register on every model: function-word frequency falls from 0.368, 0.344 and 0.180 from the original language models to under 0.020, while the frequency for the controls doesn't change, as expected. The number of chain-of-thought tokens decreases by factors of 8.5, 3.5 and 3.6, as expected, but interestingly we see a smaller but meaningful decrease in thinking tokens for the Qwen3 and DeepSeek controls as well. While Qwen3 and Gemma control arm performance does not differ much from their original models, the

Table 2: Training on compressed traces leads to highly compressed thinking. Same runs as Table 1.

Model	Untrained tok. (unfinished)	Control tok. (unfinished)	Compressed tok. (unfinished)	Control shrink	Compressed shrink	Function-word frequency untrained → control → compressed
Qwen3-1.7B	1677 (11.3%)	1317 (7.0%)	198 (3.0%)	1.3×	8.5×	0.368 → 0.364 → 0.017
DeepSeek-7B	277 (3.7%)	167 (1.3%)	79 (1.0%)	1.7×	3.5×	0.344 → 0.343 → 0.018
Gemma-4-E2B	464 (1.7%)	461 (2.6%)	131 (5.0%)	1.0×	3.6×	0.180 → 0.178 → 0.012

DeepSeek control arm loses 9.6 points of accuracy from the data. The compressed arm loses 22.7 points on Qwen3 (86.8% to 64.1%), 5.4 on DeepSeek-7B (79.7% to 74.3%) and 16.2 on Gemma (90.1% to 73.9%) compared to the control arm. This drop in performance from compression may appear to be correlated with model size, with larger drops for Qwen3 and Gemma compared to that for DeepSeek, but may also simply be a consequence of the larger decrease in performance from data for DeepSeek. Additionally, we note that Qwen3's native traces are long because the model is eager to re-check its reasoning (Appendix C), so rewriting removes the most from Qwen3's traces (a 15.5× reduction in words, against 5.1× for Gemma and 3.3× for DeepSeek-7B), and Qwen3 also loses the most accuracy.

Unfinished traces cannot close the performance gap. Some arms leave more than 2% of problems unfinished within the budget (Table 2), a limitation of our computational budget; for these arms the accuracy is a lower bound. Most unfinished compressed traces are repetition loops of a short sequence of tokens, which are unlikely to contain correct answers: 89% of them on Qwen3 and 82% on DeepSeek-7B, though only 47% on Gemma (seed 42). Even crediting every unfinished compressed trace as correct, and every unfinished control trace as wrong, leaves compression effects of -19.7, -4.4 and -11.2 points on Qwen3, DeepSeek-7B and Gemma.

Replication and budget sweep. Before fixing the matched problem set, each model was trained on its own pool of verified problems (191, 242 and 260 problems); we call these the *per-pool* checkpoints. With the same recipe and three seeds, their compression effects are -24.6, -3.4 and -19.6 points (Table A2). Sweeping the generation budget from 128 to 4,000 tokens for these checkpoints (Figure A1) shows where compressed reasoning is stronger: at small budgets the compressed arms are ahead because the controls are cut off (at 128 tokens, 99.8% of Qwen3 control traces are unfinished against 12.3% of compressed ones), while from 256 tokens upward on DeepSeek-7B and from 1,024 upward on Qwen3 and Gemma the control is more accurate.

4.2 Comparing prompting-based and trained baselines

Table 3 compares the compressed arms with methods that shorten reasoning without training: budget forcing, disabling reasoning, and prompting.

Budget forcing. Cutting untrained reasoning at the compressed arm's length and forcing an answer gives 28.7% on Qwen3 (200 tokens), 36.2% on DeepSeek-7B (80) and 21.2% on Gemma (130), against 64.1%, 74.3% and 73.9% for the compressed arms: a margin of 35.4, 38.1 and 52.7 points. The compressed arms roughly match forcing at twice their length (61.9%, 62.9% and 52.9%). From these results, it is clear that learning to compress chain-of-thought is superior to naively truncating output (Figure 1).

Reasoning off. Disabling or pre-filling empty reasoning gives scores of 79.7%, 86.7% and 86.0%, higher than every compressed arm, at 260, 269 and 278 total generated tokens against the compressed arms' 205, 86 and 139. Qwen3 without thinking is 15.6 points more accurate than our method while costing 55 more tokens overall, while DeepSeek-7B and Gemma without thinking use 3.1× and 2.0× the tokens of our trained models.

Prompting. Chain-of-Draft prompting gives 65.2%, 85.2% and 86.1% at 558, 249 and 460 total tokens. On Qwen3 it matches the compressed arm's accuracy at 2.7× the tokens; on DeepSeek-7B and Gemma it is 10.9 and 12.2 points more accurate than the compressed arm at roughly three times the tokens. Using the register prompt itself does not lead to compressed chain-of-thought in Qwen3 or DeepSeek-7B (function-word frequency 0.369 and 0.346), and only partially in Gemma's

Table 3: Compressing chain-of-thought without training generally costs more accuracy than using a trained model. We experiment with disabling thinking, budget forcing, and prompting to compress thinking.

Model	method	accuracy (unfinished)	think tokens	total tokens	function-word freq.
Qwen3-1.7B	untrained	87.3% (11.3%)	1677	1923	0.368
Qwen3-1.7B	thinking off	79.7% (0.0%)	0	260	—
Qwen3-1.7B	budget forcing, 200 tokens	28.7% (0.0%)	200	224	0.373
Qwen3-1.7B	budget forcing, 400 tokens	61.9% (0.0%)	399	420	0.366
Qwen3-1.7B	Chain-of-Draft prompt	65.2% (0.8%)	512	558	0.347
Qwen3-1.7B	register prompt	81.2% (3.1%)	1362	1374	0.369
Qwen3-1.7B	ours, control (mean of 3 seeds)	86.8% (7.0%)	1317	1323	0.364
Qwen3-1.7B	ours, compressed (mean of 3 seeds)	64.1% (3.0%)	198	205	0.017
DeepSeek-7B	untrained	89.2% (3.7%)	277	535	0.344
DeepSeek-7B	thinking off	86.7% (0.0%)	0	269	—
DeepSeek-7B	budget forcing, 80 tokens	36.2% (0.0%)	80	95	0.375
DeepSeek-7B	budget forcing, 160 tokens	62.9% (0.0%)	142	157	0.355
DeepSeek-7B	Chain-of-Draft prompt	85.2% (0.2%)	141	249	0.340
DeepSeek-7B	register prompt	89.3% (1.4%)	370	481	0.346
DeepSeek-7B	ours, control (mean of 3 seeds)	79.7% (1.3%)	167	174	0.343
DeepSeek-7B	ours, compressed (mean of 3 seeds)	74.3% (1.0%)	79	86	0.018
Gemma-4-E2B	untrained	88.4% (1.7%)	464	688	0.180
Gemma-4-E2B	thinking off	86.0% (0.0%)	0	278	—
Gemma-4-E2B	budget forcing, 130 tokens	21.2% (0.0%)	128	162	0.230
Gemma-4-E2B	budget forcing, 260 tokens	52.9% (0.0%)	251	285	0.195
Gemma-4-E2B	Chain-of-Draft prompt	86.1% (0.4%)	393	460	0.131
Gemma-4-E2B	register prompt	88.2% (5.6%)	766	871	0.104
Gemma-4-E2B	ours, control (mean of 3 seeds)	90.1% (2.6%)	461	469	0.178
Gemma-4-E2B	ours, compressed (mean of 3 seeds)	73.9% (5.0%)	131	139	0.012

(0.104, at 766 reasoning tokens, more than untrained). For these models, prompting does not elicit compressed reasoning; fine-tuning is required.

Table 4: Comparison to a trained-compression baseline: TokenSkip.

Model	method	budget	accuracy (unfinished)	think tokens
Qwen3-1.7B	untrained	4000	87.3% (11.3%)	1677
Qwen3-1.7B	TokenSkip γ=0.5	4000	76.6% (15.2%)	1050
Qwen3-1.7B	TokenSkip γ=0.7	4000	85.8% (5.2%)	974
Qwen3-1.7B	ours, compressed (mean of 3 seeds)	4000	64.1% (3.0%)	198
DeepSeek-7B	untrained	1500	89.2% (3.7%)	277
DeepSeek-7B	TokenSkip γ=0.5	1500	70.7% (0.2%)	95
DeepSeek-7B	TokenSkip γ=0.7	1500	77.1% (0.0%)	125
DeepSeek-7B	ours, compressed (mean of 3 seeds)	500	74.3% (1.0%)	79

TokenSkip. Ratio-conditioned TokenSkip (Xia et al., 2025) (Table 4), trained on each model's native traces for the same 250 problems as Table 1, pruned at six retention ratios (1,500 training examples per model; Appendix G), reaches 76.6% (γ=0.5, 1,050 tokens; 15.2% unfinished) and 85.8% (γ=0.7, 974 tokens; 5.2% unfinished) on Qwen3. However, these traces are much longer than our compressed arm (64.1% at 198 tokens) for Qwen3. On DeepSeek-7B, TokenSkip yields 70.7% (γ=0.5, 95 tokens) and 77.1% (γ=0.7, 125 tokens), compared with our compressed arm's 74.3% at 79 tokens and the control's 79.7% at 167 tokens. Due to budget constraints, we were unable to run this method on Gemma.

4.3 More training data improves performance while maintaining compressed reasoning

Because the native trace is the model's own distribution while the compressed trace is generated by a separate model (Qwen2.5-7B-Instruct), a small training set may penalize the compressed arm more. We therefore train both Qwen3 arms on 200 to 4,000 examples drawn from two pools of native traces: one generated with a 2,048-token reasoning cap, where almost no traces are truncated, and one with a 750-token cap, where most are (Table 5, Figure 2). As data grows, the compressed arm improves from 65.3% to 71.9% while the control stays between 86.6% and 89.5%, and the

Table 5: Scaling training data from 200 to 4000 examples on Qwen3-1.7B. Note that at the 750-token budget there were a maximum of 1,996 valid native traces produced by the model, providing an upper bound for the number of training examples.

training traces	examples	control (unfinished)	compressed (unfinished)	compression
750-token	200	87.0% (6.7%)	64.9% (2.2%)	-22.1pp
750-token	500	84.3% (0.3%)	66.0% (0.8%)	-18.3pp
750-token	1000	83.4% (0.0%)	66.9% (0.8%)	-16.5pp
750-token	1996	83.9% (0.1%)	69.5% (0.5%)	-14.3pp
2048-token	200	88.4% (11.1%)	65.3% (3.4%)	-23.1pp
2048-token	500	86.6% (10.8%)	67.6% (2.1%)	-19.0pp
2048-token	1000	88.1% (9.0%)	67.1% (1.5%)	-21.0pp
2048-token	2000	89.5% (1.7%)	69.8% (1.7%)	-19.7pp
2048-token	4000	88.9% (0.1%)	71.9% (0.6%)	-17.1pp

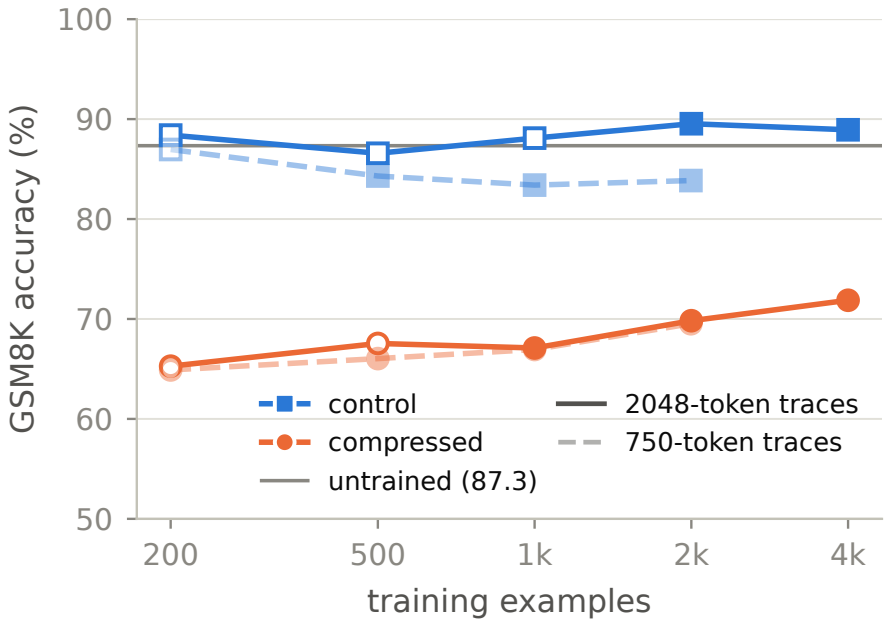

Figure 2: Qwen3 GSM8K accuracy as the training pool grows, for 750- and 2,048-token source-trace budgets. Hollow markers indicate truncation-limited results.

compression effect moves from -23.1 to -17.1 points. The compressed arm keeps its register at every size (function-word frequency 0.016–0.017). In the 750-token pool, most traces were cut off during generation, which caps it at 1,996 usable examples; its gap narrows to -14.3 points, but mainly because its control, trained on truncated reasoning, falls from 87.0% to 83.9%.

4.4 Reinforcement learning compresses thinking but fails to reduce function-word frequency

Table 6 reports small-scale reinforcement learning results. Correctness-only reinforcement learning shortens reasoning substantially for Qwen3 from 1,677 to 697 tokens, which may be related to the training data distribution for Qwen3. The register reward shortens further at small costs relative to correctness-only RL, but barely compresses the chain-of-thought: function-word frequency is 0.329 and 0.332 for Qwen3 and DeepSeek-7B, against 0.349 and 0.342 for correctness only. It appears that the model does not discover compressed reasoning in our small-scale experiments.

Table 6: Reinforcement-learning results under correctness, length and register rewards (GRPO).

Model	budget	arm	accuracy (unfinished)	vs RL control	think tokens	function-word freq.
Qwen3-1.7B	4000	untrained	87.3% (11.3%)		1677	0.368
Qwen3-1.7B	4000	control (correctness only)	88.3% (2.3%)		697	0.349
Qwen3-1.7B	4000	length reward	88.1% (3.9%)	-0.2pp	979	0.358
Qwen3-1.7B	4000	register reward	86.4% (0.9%)	-2.0pp	507	0.329
DeepSeek-7B	1500	untrained	89.2% (3.7%)		277	0.344
DeepSeek-7B	1500	control (correctness only)	88.1% (2.2%)		234	0.342
DeepSeek-7B	1500	length reward	88.6% (2.4%)	+0.5pp	238	0.342
DeepSeek-7B	1500	register reward	87.0% (0.9%)	-1.1pp	176	0.332

4.5 Transfer learning results

To understand transfer learning capabilities, we perform zero-shot evaluation of the GSM8K supervised fine-tuned checkpoints on MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024) in Table 7 and on the ARC-Challenge test set (Clark et al., 2018) (ARC results are in Appendix Table A4).

Table 7: MATH-500 accuracy (unfinished rate) of the GSM8K-trained matched-set checkpoints (seed 42), with no MATH training. Accuracy is a lower bound. The 16,000-token budget was run for Qwen3 only.

Model	budget	untrained	control	compressed	function-word freq. untrained $\rightarrow$ compressed
Qwen3-1.7B	4000	67.2% (39.6%)	70.6% (36.4%)	40.6% (38.4%)	$0.320 \rightarrow 0.071$
Qwen3-1.7B	16000	89.0% (7.2%)	86.0% (7.2%)	43.8% (33.2%)	$0.320 \rightarrow 0.073$
DeepSeek-7B	4000	78.2% (21.4%)	77.0% (20.6%)	67.8% (22.6%)	$0.313 \rightarrow 0.230$
Gemma-4-E2B	4000	61.0% (22.4%)	67.4% (16.8%)	59.0% (29.6%)	$0.216 \rightarrow 0.085$

On MATH-500 at 4,000 tokens, 17–40% of traces in every arm are unfinished, so all accuracies are lower bounds. For Qwen3 we also evaluated at 16,000 tokens, where untrained and control reasoning finish on 92.8% of problems: untrained scores 89.0%, the control 86.0% and the compressed arm 43.8%. A third of the compressed traces never finish, and almost all of those end in a repetition loop, against 36% of the control's far fewer (7.2% vs 33.2%) unfinished traces. While performance is hard to measure, the function-word frequency on MATH-500 is lower for the compressed arms, consistent with the results in Table 2.

On ARC-Challenge (Appendix D) the compression effect is -23.1 points on Qwen3 (with 21.5% of compressed traces unfinished), -2.4 on DeepSeek-7B and -9.0 on Gemma, matching the performance trend seen on GSM8K.

5 Discussion and limitations

Compression studies need a control. After training the DeepSeek-7B, accuracy decreased by 9.6 points while the reasoning was shortened by $1.7\times$ without any clear change of register, as indicated by function-word frequency. Since any compression method trained on a small set of the model's own traces inherits this effect, we recommend the matched native-trace control as a default for evaluating trained compression.

Where telegraphic reasoning is useful. For the models we tested, the compressed register occupies the lowest-cost end of the accuracy-cost curve: 86 total tokens for 74% on DeepSeek-7B, 139 for 74% on Gemma. If a deployment needs reasoning at that length, a compressed reasoner is far better than naive truncation. If it can afford more tokens, switching reasoning off or training with correctness-only RL is more accurate.

Limitations. *Scale and task.* Due to compute scarcity, we fine-tune three models of 1.7–7B parameters on GSM8K. Larger models may lose less, and the MATH-500 results at 4,000 tokens and the ARC-Challenge results are truncation-limited for the same budget reasons. *One compressor, one specification.* Rewrites come from one 7B instruction model under one specification with a loose coverage gate; a stronger rewriter or a gate that preserves checking steps might change the cost.

Easier training problems. The matched set contains only problems all three models solve and all three rewrites pass, which may affect model capacity for learning. *Metric.* Function-word frequency is a proxy for checking if the register has been compressed (beyond looking at output length) and does not measure legibility to a monitor. *Training recipe differences.* Due to budget constraints, Qwen3 is fully fine-tuned and the others use LoRA, and the matched arms differ in micro-batch size at equal effective batch. *Small-scale RL.* RL uses 60 training problems and one run per arm, the register arm received 60 more steps than the correctness-only arm, and neither RL nor TokenSkip was run on Gemma due to budget constraints.

6 Conclusion

We compared three ways of eliciting compressed chain-of-thought in reasoning models. Prompting this leaves the reasoning of two of our three models unchanged, and reinforcement learning with a function-word reward slightly shortens reasoning without strongly affecting the register. Fine-tuning on rewritten traces does elicit the register, but against a control that isolates the effect of fine-tuning it costs 5 to 23 points of GSM8K accuracy. While adding more training data decreases the performance hit, we did not find it to close the gap. Compressed reasoning is therefore worthwhile only at very small token budgets, otherwise one may consider disabling reasoning to reduce token usage at a smaller cost to performance.

AI use statement

We used generative AI tools (Claude Code, Codex, Gemini) to write and debug the experiment code, to launch and monitor training and evaluation runs, and to write the scripts that generate tables and figures from the result files. We also used them to draft and revise parts of the manuscript, to review the manuscript, and to check the reported numbers and citations against the result files and the original sources. The authors reviewed all AI-assisted work: reported numbers are generated from the result files by scripts rather than transcribed, and citations were checked against the original papers. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

References

Morayo Danielle Adeyemi, Ryan A. Rossi, and Franck Dernoncourt. CAVEWOMAN: How large language models behave under linguistic input and output compression. *arXiv preprint arXiv:2606.24083*, 2026.

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.

aussetg. GPT-5.5 trace. GitHub Gist, 2026. URL `https://gist.github.com/aussetg/20747ae00df17992acb4ebdfcd8d8d88`.

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 o1-like LLMs. *arXiv preprint arXiv:2412.21187*, 2024.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Gemma Team. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware LLM reasoning. *arXiv preprint arXiv:2412.18547*, 2024.

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. In *NeurIPS Datasets and Benchmarks Track*, 2021.

htihle. Social media post on reasoning traces. X, 2026. URL https://x.com/htihle/status/2048741770125603304.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. The impact of reasoning step length on large language models. *arXiv preprint arXiv:2401.04925*, 2024. URL https://arxiv.org/abs/2401.04925.

Arun Jose. Reasoning models sometimes output illegible chains of thought. *arXiv preprint arXiv:2510.27338*, 2025. URL https://arxiv.org/abs/2510.27338.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. C3oT: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022. URL https://arxiv.org/abs/2205.11916.

Yangsong Lan, Hongliang Dai, and Piji Li. CRISP: Compressing redundancy in chain-of-thought via intrinsic saliency pruning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 39355–39373, 2026. doi: 10.18653/v1/2026.findings-acl.1961. URL https://aclanthology.org/2026.findings-acl.1961/.

Ayeong Lee, Ethan Che, and Tianyi Peng. How well do LLMs compress their own chain-of-thought? a token complexity approach. *arXiv preprint arXiv:2503.01141*, 2025. URL https://arxiv.org/abs/2503.01141.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

Siyang Lyu, Xinghao Chen, Zhijing Sun, Tong Liu, Dawei Zhu, and Xiaoyu Shen. When compression helps and when it hurts: Condition-aware analysis of chain-of-thought distillation. *arXiv preprint arXiv:2606.21704*, 2026. URL https://arxiv.org/abs/2606.21704. EMNLP 2026 Main.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. CoT-valve: Length-compressible chain-of-thought tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6025–6035, 2025. doi: 10.18653/v1/2025.acl-long.300. URL https://aclanthology.org/2025.acl-long.300/.

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

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Matthew Renze and Erhan Guven. The benefits of a concise chain of thought on problem-solving in large language models. *arXiv preprint arXiv:2401.05618*, 2024. URL https://arxiv.org/abs/2401.05618.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. Stop overthinking: A survey on efficient reasoning for large language models. *Transactions on Machine Learning Research*, 2025.

Trevor050. GPT5.5s CoT keeps leaking in the new codex update. Reddit post, r/OpenAI, 2026. URL https://www.reddit.com/r/OpenAI/comments/1t9pd1m/gpt55s_cot_keeps_leaking_in_the_new_codex_update/.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. TokenSkip: Controllable chain-of-thought compression in LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

A Training and evaluation settings

Table A1: Training and evaluation settings, read from the run manifests.

experiment	model	arm	fine-tuning	lr	epochs	batch × accum	max length	train examples	seeds	eval budget	first pass
matched set	Qwen3-1.7B	control	full	1e-05	3	2×8	1152	250	42, 43, 44	4000	—
matched set	Qwen3-1.7B	compressed	full	1e-05	3	8×2	768	250	42, 43, 44	4000	—
matched set	DeepSeek-7B	control	LoRA r16 32	0.0001	3	2×8	1152	250	42, 43, 44	500	—
matched set	DeepSeek-7B	compressed	LoRA r16 32	0.0001	3	8×2	768	250	42, 43, 44	500	—
matched set	Gemma-4-E2B	control	LoRA r16 32	0.0001	3	2×2	1152	250	42, 43, 44	1500	—
matched set	Gemma-4-E2B	compressed	LoRA r16 32	0.0001	3	4×1	768	250	42, 43, 44	1500	—
per-pool	Qwen3-1.7B	control	full	1e-05	3	2×8	1152	191	42, 43, 44	4000	—
per-pool	Qwen3-1.7B	compressed	full	1e-05	3	8×2	768	191	42, 43, 44	4000	—
per-pool	DeepSeek-7B	control	LoRA r16 32	0.0001	3	2×8	1152	242	42, 43, 44	500	—
per-pool	DeepSeek-7B	compressed	LoRA r16 32	0.0001	3	8×2	768	242	42, 43, 44	500	—
per-pool	Gemma-4-E2B	control	LoRA r16 32	0.0001	3	2×2	1152	260	42, 43, 44	1500	—
per-pool	Gemma-4-E2B	compressed	LoRA r16 32	0.0001	3	4×1	768	260	42, 43, 44	1500	—
scaling, 750-token traces	Qwen3-1.7B	control	full	1e-05	3	2×8	1152	200, 500, 1000, 1996	42	4000	—
scaling, 750-token traces	Qwen3-1.7B	compressed	full	1e-05	3	8×2	768	200, 500, 1000, 1996	42	4000	512
scaling, 2048-token traces	Qwen3-1.7B	control	full	1e-05	3	1×16	2400	200, 500, 1000, 2000, 4000	42	4000	—
scaling, 2048-token traces	Qwen3-1.7B	compressed	full	1e-05	3	8×2	1152	200, 500, 1000, 2000, 4000	42	4000	512
TokenSkip	Qwen3-1.7B	ratio-conditioned	full	1e-05	3	2×8	1152	1194	42	4000	1024
TokenSkip	DeepSeek-7B	ratio-conditioned	LoRA r16 32	0.0001	3	2×8	1152	1512	42	1500	1024

All runs use AdamW, completion-only loss and no packing. Maximum sequence length is 1,152 tokens for control arms and 768 for compressed arms (2,400 and 1,152 for the 2,048-token scaling pool). Some compressed-arm evaluations first generate every problem at a small budget and regenerate only the unfinished ones at the full budget; under greedy decoding this is identical to a single full-budget pass.

B PROMPTS AND SPECIFICATIONS

Rewriting specification (system prompt of the rewriter).

```
Answer the question under a TELEGRAPHIC constraint,
tightened beyond the standard version.
Rules:
- DO NOT use any function words.  No articles (the, a, an).
No conjunctions (and, but, or, so).  No prepositions (of,
in, to, for, at, with, from, by, on, per).
- DO NOT use a light or stative verb (has, is, are, gets,
receives, makes, costs) when a bare "Subject Number Noun"
fragment already states the same fact without it - e.g.
"Claire makes 3 eggs" becomes "Claire 3 eggs", "Marco has
81 points" becomes "Marco 81 points".  KEEP a verb that
names an actual operation or event the arithmetic depends
on (buys, sells, gives, spends, splits, doubles, divides) -
e.g.  keep "sells 15" or "splits evenly", don't reduce it to
a bare number.
- DO NOT use an adjective that doesn't affect any
calculation (drop decorative adjectives like "beautiful",
"young", "nice", "old").  KEEP an adjective that
distinguishes between two quantities the trace must not
confuse (keep "original" vs.  "sale" price, "first" vs.
"second" week, "regular" vs.  "overtime" pay).
- DO use nouns, numbers, and standard symbols (+, -, *, /,
=).
- Show each reasoning step.  Every intermediate numeric
result in the original must still appear, in the same order
- never delete or skip a computational step, only strip
grammar.
- End with a line:  'Answer:  <answer>'.
Example:
Before tightening:  "Claire makes 3 egg omelet every
morning.  4 weeks = 7 * 4 = 28 days."
After tightening:   "Claire 3 egg omelet morning.  4 weeks =
7 * 4 = 28 days."
```

Rewriting request (user turn).

```
Question:  {question}
Original step-by-step reasoning:  """{trace}"""
Rewrite the reasoning above following the rules exactly.
Keep every number and every arithmetic step from the
original - do not skip, merge, or invent any calculation,
only restyle the language.
```

Prompting baselines. Both are placed in the user turn before the question. Chain-of-Draft (Xu et al., 2025): `Think step by step, but only keep a minimum draft for each thinking step, with 5 words at most. Return the answer at the end of the response after a separator ####.` Register prompt: the rewriting specification above, followed by `Apply these rules to your reasoning (your thinking) as well as to your answer.`

C EXAMPLE TRACE

Problem. Lexie and Tom went apple picking. Lexie picked 12 apples and Tom picked twice as many apples. How many apples did they collect altogether?

Qwen3-1.7B native trace (first 1,200 characters of 433 words, matched set). *Okay, let's see. Lexie and Tom went apple picking. Lexie picked 12 apples, and Tom picked twice as many as Lexie. The question is asking how many apples they collected altogether. Hmm, let me break this down*

*step by step. First, I know that Lexie picked 12 apples. That part is straightforward. Now, Tom picked twice as many as Lexie. So, if Lexie has 12, then Tom's apples would be 2 times 12. Let me write that out: 2 * 12. Let me calculate that. 2 times 12 is 24. So Tom picked 24 apples. Now, the question is asking for the total number of apples they collected altogether. That means I need to add Lexie's apples and Tom's apples together. So, 12 (Lexie) + 24 (Tom). Let me do that addition. 12 plus 24... 12 + 20 is 32, and then plus 4 more is 36. So, 36 apples in total. Wait, let me check again to make sure I didn't make a mistake. Lexie has 12, Tom has twice as many. Twice of 12 is 24. 12 + 24 is indeed 36. Hmm, that seems right. But let me think if there's another way to approach this. Alternatively, maybe I can think of it as Tom's apples being 2 times Lexie's...*

Telegraphic rewrite (training target). `Lexie 12 apples. Tom twice 12 = 2 * 12 = 24 apples. Altogether 12 + 24 = 36 apples. Answer: 36`

D UNMATCHED TRAINING POOLS

Table A2: The decomposition of Table 1 for the per-pool checkpoints, each trained on its own pool of verified problems (191 / 242 / 260 problems; three seeds).

Model	untrained	control	compressed	from data	from compression
Qwen3-1.7B	87.3%	87.8%	63.2%	+0.4pp	-24.6pp
DeepSeek-7B	89.2%	80.1%	76.6%	-9.2pp	-3.4pp
Gemma-4-E2B	88.4%	90.6%	71.0%	+2.2pp	-19.6pp

These per-pool checkpoints are used for the budget sweep (Figure A1) and ARC-Challenge (Table A4).

E BUDGET FORCING

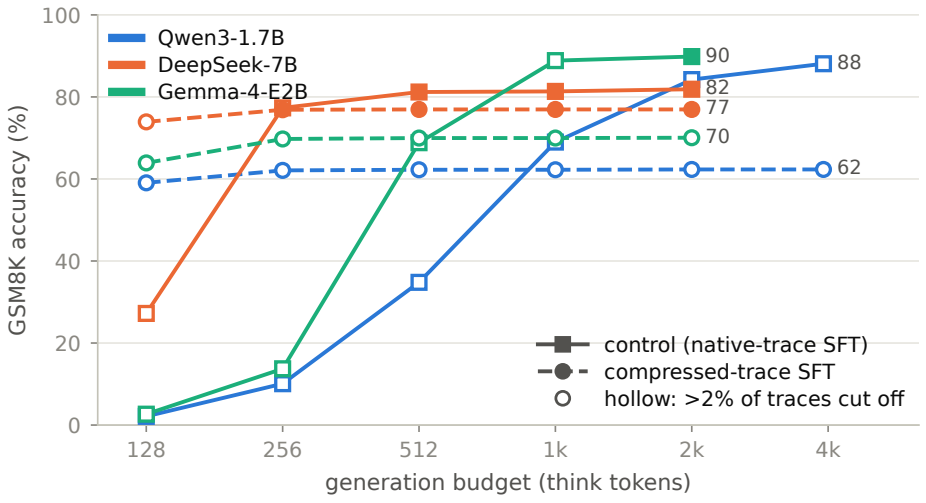

Figure A1: Accuracy against the generation budget for the per-pool control and compressed checkpoints. Hollow markers indicate truncation-limited results.

At the budget, the reasoning block is closed and the answer forced with `The final answer is \boxed{`, with 24 tokens to complete it; a trace that closes earlier is cut at its close tag and answered in the same way.

Table A3: Budget forcing: untrained reasoning cut at N tokens, with the untrained and compressed arms of Table 1 for reference.

Model	reasoning	accuracy	cut mid-reasoning	think tokens
Qwen3-1.7B	forced at 100	15.3%	100%	100
Qwen3-1.7B	forced at 200	28.7%	100%	200
Qwen3-1.7B	forced at 400	61.9%	98%	399
Qwen3-1.7B	forced at 800	82.7%	77%	759
Qwen3-1.7B	forced at 1600	88.6%	36%	1181
Qwen3-1.7B	forced at 3200	89.8%	15%	1551
Qwen3-1.7B	untrained, unforced	87.3% (11.3%)		1677
Qwen3-1.7B	ours, compressed (mean of 3 seeds)	64.1%		198
DeepSeek-7B	forced at 40	26.0%	100%	40
DeepSeek-7B	forced at 80	36.2%	97%	80
DeepSeek-7B	forced at 160	62.9%	54%	142
DeepSeek-7B	forced at 320	78.3%	14%	182
DeepSeek-7B	untrained, unforced	89.2% (3.7%)		277
DeepSeek-7B	ours, compressed (mean of 3 seeds)	74.3%		79
Gemma-4-E2B	forced at 65	13.7%	100%	63
Gemma-4-E2B	forced at 130	21.2%	100%	128
Gemma-4-E2B	forced at 260	52.9%	90%	251
Gemma-4-E2B	forced at 520	83.9%	32%	401
Gemma-4-E2B	untrained, unforced	88.4% (1.7%)		464
Gemma-4-E2B	ours, compressed (mean of 3 seeds)	73.9%		131

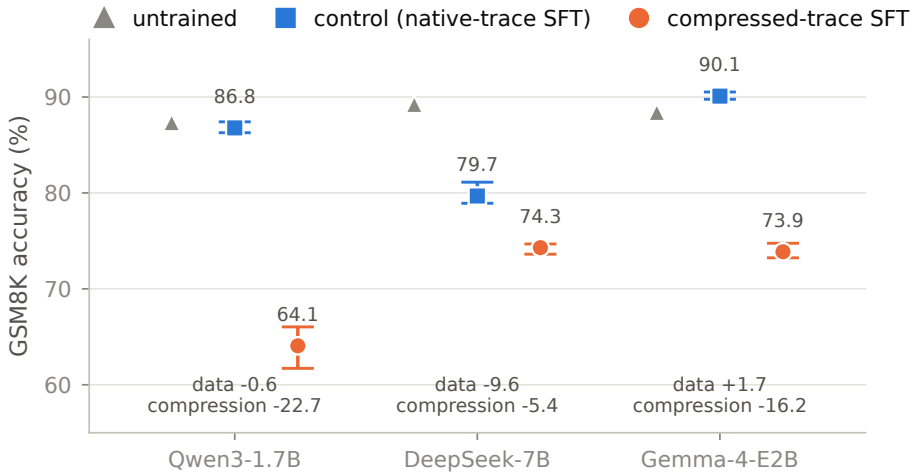

Figure A2: Matched-control decomposition of accuracy across untrained, native-target control and compressed-target arms.

F Reinforcement learning details

We use Group Relative Policy Optimization (GRPO) (Shao et al., 2024) with six samples per prompt on 60 GSM8K training problems (a different sample per arm), maximum completion length 800 tokens for Qwen3 and 1,024 for DeepSeek-7B, learning rate 2×10^{-6} with full fine-tuning for Qwen3. DeepSeek-7B is trained with 4-bit quantised LoRA (rank 16, α=32, learning rate 2×10^{-5}), and the length adapter is merged before the register stage. The reward for a completion with reasoning block t and answer section a is -1 if the answer is wrong or the reasoning block never closes. Otherwise it is

$$1 + 0.6\,\mathrm{clip}\left(\tfrac{200-w(t)}{200}\right) + \beta\,\mathrm{clip}\left(\tfrac{0.35-f(t)}{0.35}\right) - 0.5\mathbb{1}[w(a) > 30],$$

where w counts words, f is function-word frequency, clip clips to $[0, 1]$, and $\beta = 0.6$ for the register reward and 0 otherwise. The correctness-only reward is $+1$ for any correct, closed completion.

Both bonuses are computed on the reasoning block only, and the penalty on long answer sections prevents the reasoning from moving past the close tag. The correctness-only control starts from the base model and is evaluated after 180 steps. The length-reward arm also starts from base and is evaluated after 60 steps; the register arm branches from that checkpoint and is evaluated after 180 additional steps (240 cumulative steps from base).

G TOKENSKIP

We follow Xia et al. (2025): each native trace is pruned by LLMLingua-2 token importance (`llmlingua-2-xlm-roberta-large-meetingbank`) at retention ratios $\gamma \in \{0.5, 0.6, \ldots, 1.0\}$, with no force-kept tokens, and one checkpoint is trained on all ratios with γ stated in the user turn (`Compression ratio: 0.5`). We prune only the reasoning block. Training data are each model's native traces for the 250 matched problems of Table 1, pruned at the six ratios (1,500 examples per model).

H ARC-CHALLENGE

Table A4: ARC-Challenge accuracy (unfinished rate) of the GSM8K-trained per-pool checkpoints (one seed, 3,000-token budget).

Model	untrained	control	compressed	from data	from compression	function-word freq. untrained → compressed
Qwen3-1.7B	87.6% (3.2%)	86.3% (2.8%)	63.1% (21.5%)	-1.4pp	-23.1pp	0.429 → 0.223
DeepSeek-7B	81.5% (2.9%)	79.3% (2.2%)	76.9% (3.4%)	-2.2pp	-2.4pp	0.391 → 0.364
Gemma-4-E2B	89.0% (1.4%)	88.7% (2.0%)	79.6% (5.3%)	-0.3pp	-9.0pp	0.310 → 0.239