OpenVerdict: A 4B Decision Model with an 80-Token Thinking Budget

Technical Report

Vincentius Roger Kuswara
Independent researcher vincentiusrogerk@gmail.com

October 2026

huggingface.co/vinroger/OpenVerdict-4B github.com/vinroger/openverdict

Abstract

Decision models answer typed questions in one forward pass, which makes them fast but weak on multi-step questions. However, chain-of-thought (CoT) reasoning is too long for a half-second latency budget. Truncating it does not help: cut at 64 tokens, Qwen3.5-4B's own reasoning is less accurate than a direct answer (35.6% against 48.3%). We study whether a small model can learn to reason usefully within 80 tokens. We train Qwen3.5-4B to finish its CoT inside this budget, with rejection-sampling fine-tuning on its shortest correct CoT and reinforcement learning with a completion bonus. On seven of ten test sets, the resulting model, OpenVerdict, matches or outperforms every open model up to 9B parameters that we evaluate. It leads significantly on all three multi-step reasoning sets, while larger models remain stronger on knowledge-heavy tasks. A paired analysis shows large gains on state tracking and no clear effect on single-step judgments. We also reproduce, at 4B, that rewarding correct answers reached through invalid reasoning harms tasks the model cannot yet solve. These results suggest that, under tight latency budgets, training a small model to finish a short CoT outperforms truncating a long one. We release the weights, code and 800 logic puzzles.

1 Introduction

Software increasingly calls models in the middle of its control flow: to route a support ticket, to check a request against a policy, or to decide whether a screenshot shows the expected page. Each of these calls needs a typed answer with a probability, and it needs it fast, because a person or an agent waits for it, often many times per task. *Decision models* (also called System One models), such as Jev [44], CLM [20], H2O-Lightning-4B [30] and Kev [34], answer these questions in one forward pass.

However, one forward pass is weak when the answer needs several steps of state tracking. Consider this question from BIG-Bench Hard (BBH) [42]:

> *Leda tells the truth. Shalonda says Leda lies. Shenna says Shalonda tells the truth. Christie says Shenna lies. Crista says Christie tells the truth. Does Crista tell the truth?*

The answer needs five small steps (Leda true, Shalonda false, Shenna false, Christie true, Crista true). Our model in non-thinking mode answers *No* with probability 0.94, which is incorrect. On 540 public reasoning questions of this kind from BBH and ProofWriter (the BBH + ProofWriter sample), the base Qwen3.5-4B answers 48.3% correctly with a direct answer.

Chain-of-thought (CoT) reasoning solves such questions, but it is too long for this setting. With its own thinking, Qwen3.5-4B answers 92.4% of the sample correctly, but it writes 1,224 tokens per

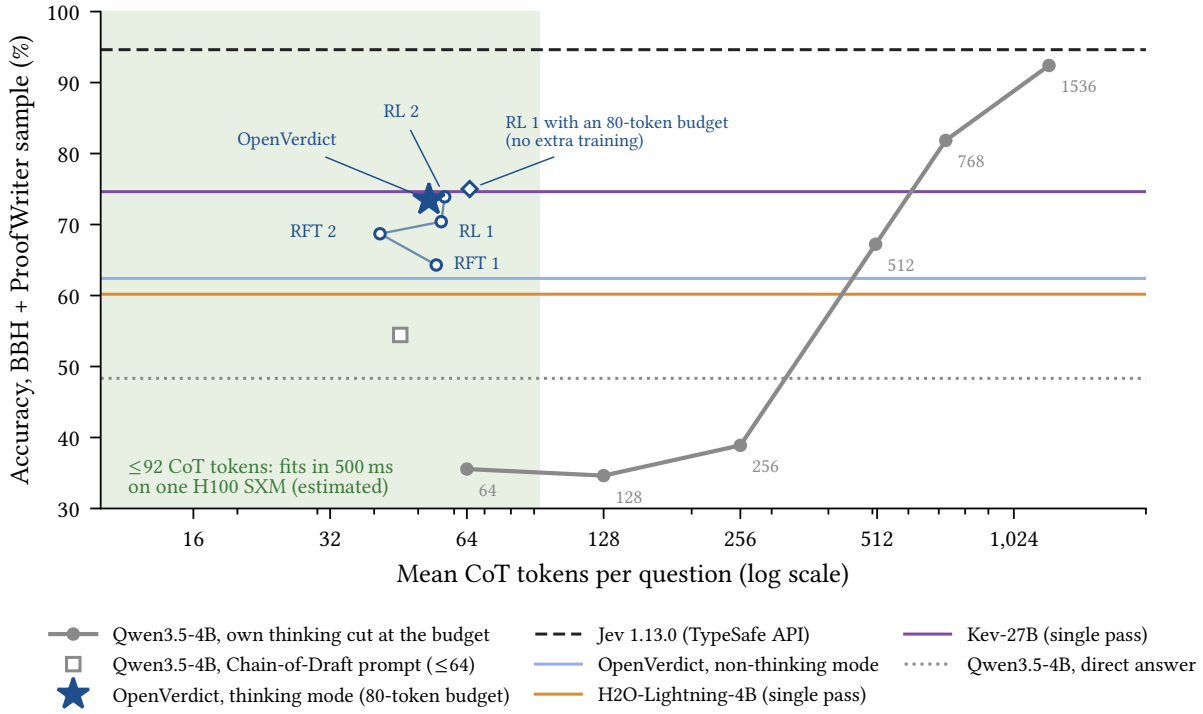


Figure 1: Accuracy against CoT length on the BBH + ProofWriter sample (540 public reasoning questions). The base Qwen3.5-4B needs about 500 tokens of thinking to outperform its own direct answer; with its thinking truncated at 64–256 tokens, it is less accurate than with no thinking at all. OpenVerdict's thinking mode reaches 73.5% with 53 tokens on average, inside the region that fits in 500 ms on one H100 SXM (green; estimated from timings of an earlier checkpoint, Section 4; on an H100 PCIe the final model is slower, Section 7.7). Hollow circles are the training stages of Section 5.

question on average (731 on the question above). On one H100 SXM this takes about 5 s per answer (estimate, Section 4), ten times our target of half a second. Truncating the thinking does not help either: cut at 64 tokens, the base model answers only 35.6% correctly, which is lower than its direct answer. On the question above, all 64 tokens restate the question (*"The user wants me to solve a logic puzzle… **State:** 1. **Leda:** Tells the truth…"*), and the answer is read before any reasoning takes place.

We therefore ask whether a 4B model can learn to reason usefully within a budget that fits in about half a second, and whether this reasoning also helps on practical decisions, not only on puzzles. Our approach trains the model to *finish* its CoT inside an 80-token budget: rejection-sampling fine-tuning on its own shortest correct CoT, followed by reinforcement learning with a bonus for a correct answer whose CoT finishes inside the budget (Section 5).

The resulting model, OpenVerdict, writes about 50 tokens on average (30 on the question above, Figure 12) and answers *Yes*, which is correct. Figure 1 shows the trade-off between accuracy and CoT length. On multi-step reasoning, the short CoT gives large gains over the non-thinking mode: +15.1 points on our logic puzzles and +11.1 on the BBH + ProofWriter sample. On practical decisions, the gain depends on the task: +5.2 points on JevBench public, but no difference on our five public decision datasets (72.7% in both modes). For this reason, the model keeps both modes, and the caller chooses one per question.

Our contributions are:

- a measured account of the thinking budget at 4B: a truncated CoT is less accurate than a direct answer, the gain comes mainly from CoTs that finish inside the budget, and a bonus for finishing trades a little accuracy for shorter CoTs (Sections 4 and 5.3);
- a map of where a 50-token CoT helps, over 42 paired comparisons: large gains on state tracking and no clear effect on single-step judgments (Section 7.6);

Table 1: OpenVerdict at a glance: thinking mode against the strongest open models up to 9B parameters and, for reference, the larger Kev-27B and TypeSafe's Jev 1.13.0 API, task by task on the same questions, all run by us (accuracy, %). Bold: the best of the open models up to 9B. The last column gives the difference to Jev with its paired 95% confidence interval (CI). –: not run; n/a: text-only model. Rows are grouped by whether another open model up to 9B is more accurate than OpenVerdict; every task we measured is listed. All systems on all ten test sets: Table 6. [a] Balanced 300-message sample, default 0.5 threshold (Section 7.3); on the full test split, OpenVerdict has the highest F1 of the open models we ran, on par with Kev-4B (Section 7.2). [b] Development set (Section 6). [c] Three state-tracking tasks of BIG-Bench Hard (BBH); all 23 tasks are in Figure 11 and Table 13. [d] Jev 1.14.0; version 1.13.0 scored 85.7%. [e] The first 200 numeric test problems; on all 541, OpenVerdict scores 70.1% (Section 7.2). Imajev-4B's 25.7 on RewardBench 2 counts its refusals of long questions as incorrect (Section 6).

			Open, up to 9B			Larger or API, for reference		
Task	*n*	OpenVerdict	H2O-Lightning-4B	Imajev-4B	Clef-Flash 9B	Kev-27B	Jev 1.13.0	Difference to Jev
No open model up to 9B is more accurate than OpenVerdict								
BBH tracking shuffled objects (3)[c]	250	**93.6**	33.2	36.4	20.8	34.0	98.4	−4.8 [−8.4, −1.6]
BBH web of lies[c]	250	**93.2**	56.0	58.0	74.4	90.0	100.0	−6.8 [−10.0, −3.6]
BBH navigate[c]	250	**85.2**	67.6	65.6	63.2	75.2	93.2	−8.0 [−12.8, −3.6]
Logic puzzles	800	**63.5**	49.5	49.1	52.0	55.9	65.6	−2.1 [−6.0, +1.9]
RewardBench 2 (reward modeling)	300	**72.0**	63.3	25.7	59.0	77.0	80.0	−8.0 [−12.3, −3.3]
BBH + ProofWriter sample	540	**73.5**	60.2	65.6	62.8	74.6	94.6	−21.1 [−25.0, −17.6]
BIG-Bench Hard (all 23 tasks)	5,507	**72.9**	66.7	65.6	68.7	77.3	91.5	−18.6 [−19.8, −17.5]
XSTest (exaggerated safety)	300	**87.3**	83.3	79.0	81.0	88.3	89.3	−2.0 [−6.0, +1.7]
JevBench public[b]	231	**89.2**	**89.2**	86.1	82.3	87.9	87.9[d]	+1.3 [−3.0, +5.6]
Another open model up to 9B is more accurate								
OpenAI moderation set	300	85.3	**87.7**	87.0	86.0	88.0	87.0	−1.7 [−5.0, +1.7]
Mind2Web (next-action prediction)	81	81.5	80.2	**85.2**	80.2	91.4	90.1	−8.6 [−16.0, −1.2]
ToxicChat (toxicity detection)[a]	300	88.7	67.3	63.0	**93.7**	69.0	70.3	+18.3 [+14.0, +23.0]
When2Call (tool-call decisions)	300	74.3	**81.3**	70.7	61.0	65.0	72.7	+1.7 [−2.7, +5.7]
ESCI (product search relevance)	300	53.3	58.0	56.3	**63.0**	63.0	57.7	−4.3 [−9.0, +0.7]
Images: no open model up to 9B is more accurate than OpenVerdict (Jev and Kev-27B take text only)								
Geometry3K (geometry problem solving)[e]	200	**67.0**	54.5	61.5	55.0	n/a	n/a	
CORD (receipt understanding)	80	**91.2**	88.8	86.2	88.8	n/a	n/a	
Images: another open model up to 9B is more accurate (Jev and Kev-27B take text only)								
ChartQA (chart question answering)	180	93.3	93.9	**94.4**	91.7	n/a	n/a	
ScreenSpot (GUI grounding)	200	77.5	82.0	81.0	**86.0**	n/a	n/a	
VisA (anomaly detection)	240	77.9	67.5	**89.2**	62.5	n/a	n/a	

- a reproduction at 4B and an 80-token budget of the finding that rewarding correct answers reached through invalid reasoning harms tasks the model cannot yet solve (Section 8.2);
- OpenVerdict, an open 4B decision model with a non-thinking and a thinking mode in one adapter, with its full training recipe (Section 5);
- a comparison with Jev, Kev-27B, Clef-Flash and five open models of about 4B parameters on ten test sets, every system on the same questions (Section 7; the only missing results are text-only models on image sets), with the full test splits of ToxicChat and Geometry3K as a check of two samples (Section 7.2), and the ideas that did not work, each explained by a measurement (Appendix F).

2 Background and related work

Decision models. A decision model receives a *state* (text, a JSON object or an image), a typed question and a list of options. The question types are choice (pick one of several options), yes / no, and rating scale (pick a level). The model returns a probability for every option. Jev, TypeSafe's decision model, defines this interface [44], and JevBench scores models on it [6]. Several open models follow the same interface: H2O-Lightning-4B [30], Kev [34], Clef and Clef-Flash from Cloudflare [9], Imajev-4B [31], JevK5 [3] and Decision 4B [13], as well as research models such as CLM [20], Jeva [22] and Large Decision Models [7].

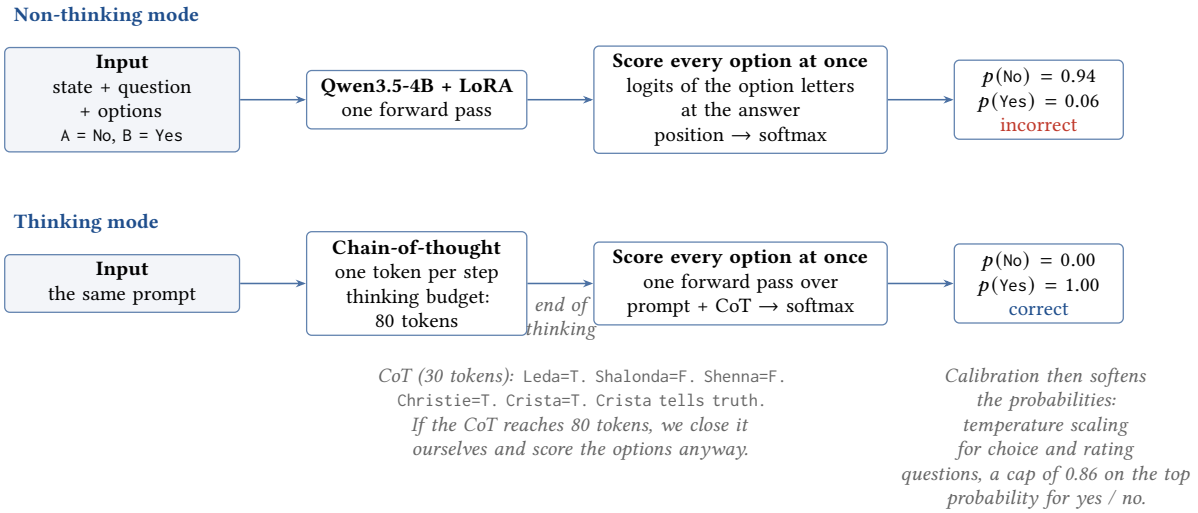


Figure 2: The two modes, on the BBH question from the introduction (real outputs of the final model). Both modes end the same way: one forward pass scores every option at once, so the output is always a full probability distribution, never a sampled word.

Scoring options by their letter probabilities. Geuter et al. [14] show that an instruction-tuned language model is already a good decision model if we read the next-token probabilities of the option letters (A, B, C, …) at the answer position, and that supervised fine-tuning plus temperature scaling make it better. OpenVerdict scores options the same way; this is not our contribution.

Thinking and non-thinking modes. Qwen3 [36] trains one model with a thinking mode and a non-thinking mode, and introduces a *thinking budget*: when the thinking reaches the budget, the thinking is stopped and the model answers from what it has written so far. SmolLM3 [4] does the same at 3B, and s1 [32] controls the length of thinking at test time by ending it early or extending it (*budget forcing*). Our thinking mode stops the CoT the same way. The difference is the scale: these works study budgets of thousands of tokens, while we need the whole answer in about half a second, so our budget is 80 tokens.

Shorter reasoning. Two families of methods shorten a model's CoT. The first fine-tunes on short, correct samples of the model's own reasoning: rejection-sampling fine-tuning (RFT) [48, 49] and its use for concise reasoning [17, 33]. The second adds a length term to the reward used in reinforcement learning (RL) [1, 17, 47]. Prompting for brevity, as in Chain of Draft [46], also helps large models, and Li et al. [23] find that small models learn better from short reasoning chains than from long ones written by stronger models. Our stages 2 and 3 use the first two families; Appendix F shows why prompting alone did not work at 4B.

3 How OpenVerdict answers

OpenVerdict is Qwen3.5-4B [36, 37] with one LoRA adapter [16] of rank 32 ($\alpha = 64$) on 248 linear layers (65 million trainable parameters). The base weights stay frozen. The prompt is the decision in JSON: the state, the question and the options, each option with a one-letter code. For example:

```
{"state": "Alice, Bob, and Claire . . .  First, Bob and Alice swap books. . . ",
"question": {"type": "choice", "instructions": ". . .  At the end of the semester, Alice
has . . . ",
"candidates": [{"code": "A", "description": "Frankenstein"}, {"code": "B", . . . }, . . . ]}}
```

Non-thinking mode. The model reads the prompt once. At the answer position we take the logits z of the option letters and apply a softmax over the options only:

$$p_\theta(k \mid x) = \frac{\exp(z_k)}{\sum_{j \in \text{options}} \exp(z_j)}. \tag{1}$$

This is one forward pass, so every option is scored at the same time, in parallel. The cost does not grow with the number of options.

Thinking mode. The model first writes a CoT, one token at a time (greedy decoding), until it emits the end-of-thinking token or reaches the thinking budget of 80 tokens. If it reaches the budget, we close the thinking ourselves, as Qwen3 does at its thinking budget [36] and s1 does with budget forcing [32]; we call such a CoT *truncated*. Then one more forward pass reads the prompt and the CoT and scores the options from their letter probabilities, as above. Following the two-stage prompting of Kojima et al. [18], we call this pass the *answer-extraction pass*. Figure 2 shows both modes on the question from the introduction: the non-thinking mode says *No* (0.94), and the thinking mode writes 30 tokens and says *Yes* (1.00).

Calibration. Finally, temperature scaling [15], and a cap on the top probability for yes / no questions, make the probabilities match how often the model is correct (Section 5.5). It never changes which option is on top.

OpenVerdict therefore reasons in words but answers like a classifier: in both modes, the output is a full probability distribution over the options, and the cost of the answer does not depend on the number of options.

4 Choosing the thinking budget

The budget follows from the latency target. This section derives the 80-token budget and reports the two measurements that set the training goal.

The time limit sets the thinking budget. Our target is an answer in about 500 ms on one H100, so that a person or an agent can call the model inside a loop. When we chose the budget, we measured 164 single requests of an earlier checkpoint of the model on one H100 SXM (same base model, same vLLM serving, Figure 3). The time of the CoT plus the answer-extraction pass fits a straight line: about 99 ms plus 4.4 ms per CoT token. A CoT of 80 tokens therefore takes about 0.45 s, and the largest CoT that fits in 500 ms is about 92 tokens (estimate). We chose 80 to keep some room for the spread of single requests (standard deviation 63 ms) and for long prompts. The H100 PCIe has less memory bandwidth: there, the final model takes about 74 ms plus 6.2 ms per CoT token, so an 80-token CoT takes about 0.57 s (measured, Section 7.7).

A truncated CoT adds little accuracy. The base model shows the problem clearly: cut at 64 tokens, its thinking reaches 35.6% on the BBH + ProofWriter sample, below its direct answer (48.3%). Our early model, trained at a 64-token budget, showed the same pattern on paired questions. Where its CoT finished, it outperformed its own non-thinking answer by 14.5, 7.6 and 2.8 points on the three test sets we used at that point; where the CoT was truncated, the differences were +0.7, −0.3 and −1.1 points. The final model still follows this pattern (Figure 4). On JevBench public, the answer after a finished CoT is correct 93.8% of the time and after a truncated CoT 65.8%; on the ChartQA + Geometry3K + CORD set, 92.9% against 44.4%. Because we also trained the answer-extraction pass on truncated CoTs (Section 5.2), a truncated CoT is no longer harmful: on the same truncated questions of JevBench public, the non-thinking answer is correct only 55.3% of the time. Most of the gain, however, comes from the CoTs that finish. The training goal is therefore a CoT that *finishes* inside the budget.

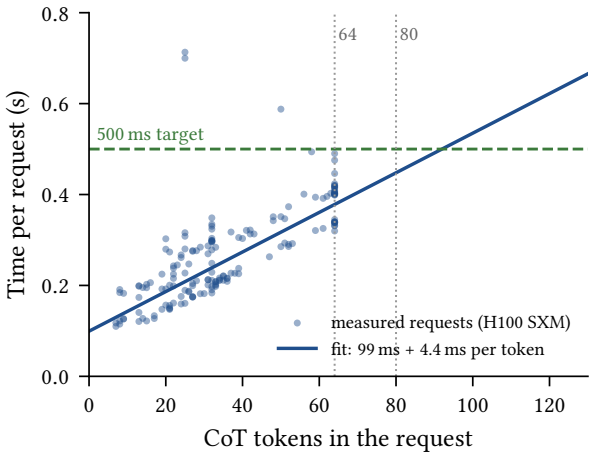


Figure 3: Measured time per request on one H100 SXM (164 requests, an earlier checkpoint, thinking budget 32 or 64). The line is a least-squares fit; at 80 tokens it predicts 0.45 s.

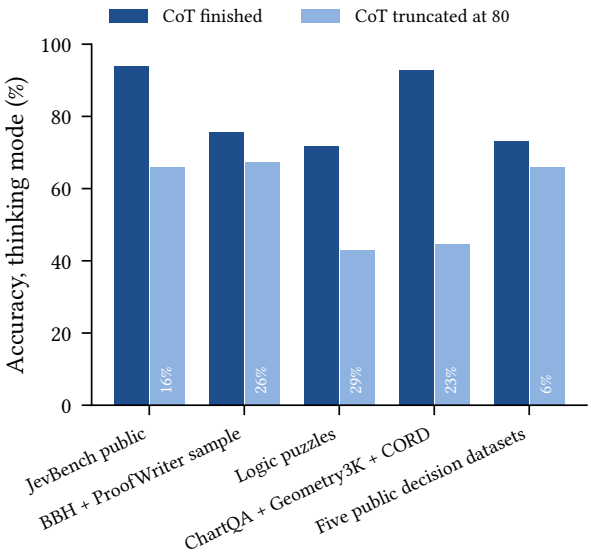


Figure 4: Finished CoTs carry the gain. Final model, thinking mode: accuracy on questions whose CoT finished, and on questions whose CoT was truncated at 80 tokens (percent of questions truncated in the bars).

Raising the budget from 64 to 80 tokens helps without retraining. Our first three training rounds used a 64-token budget (about 0.38 s by the fit above). After them, 61% of the CoTs on the BBH + ProofWriter sample still reached the budget, which suggested that the budget, more than the model, was the limit. Before training at a new budget of 80 tokens, we ran the same model at 80 tokens with no further training: accuracy on the BBH + ProofWriter sample rose from 70.4% to 75.0% (+4.6 points, 95% CI [+2.2, +7.0]) and on the logic puzzles by +3.4 [+1.0, +5.9]. This gain comes from the larger budget alone. RL at the new budget (Section 5.3) did not add accuracy on top of it. Instead, it shortened the CoTs again (mean 65 tokens without training, 54 after RL with the completion bonus), at a cost of one to two points on the BBH + ProofWriter sample (73.0% against 75.0%; 73.9% after the second RL round, which used new data). Figure 1 shows these points. Compared with the RL round 1 model run at 80 tokens, the released model is on par on every reasoning set (all four CIs include zero) and more accurate on the five decision datasets (+2.1 points [+0.5, +3.7]) and on the ChartQA + Geometry3K + CORD set (+6.1 [+2.6, +9.3]). It also writes CoTs that are 7–12 tokens shorter (53 against 65 tokens on the BBH + ProofWriter sample).

Most CoTs use about 50 tokens. The budget is a maximum, not a target. The training in Section 5 rewards a CoT that ends before the budget, so most CoTs stop well short of 80 tokens: the final model writes 41 tokens on average on JevBench public, 53 on the BBH + ProofWriter sample and 60 on the harder logic puzzles. On the text test sets, 0–29% of its CoTs reach the budget. By the H100 SXM fit above, a 50-token CoT takes about 0.32 s (about 0.39 s on an H100 PCIe). The budget therefore matters only for the hardest questions, which are also the ones that need the most time.

5 Training recipe

We train in three stages (Figure 5): supervised fine-tuning (SFT), rejection-sampling fine-tuning (RFT) and reinforcement learning (RL), followed by a short SFT round on images. Each stage addresses a limitation of the previous one. For each stage, we state the limitation, the method, and the experiment that supports it.

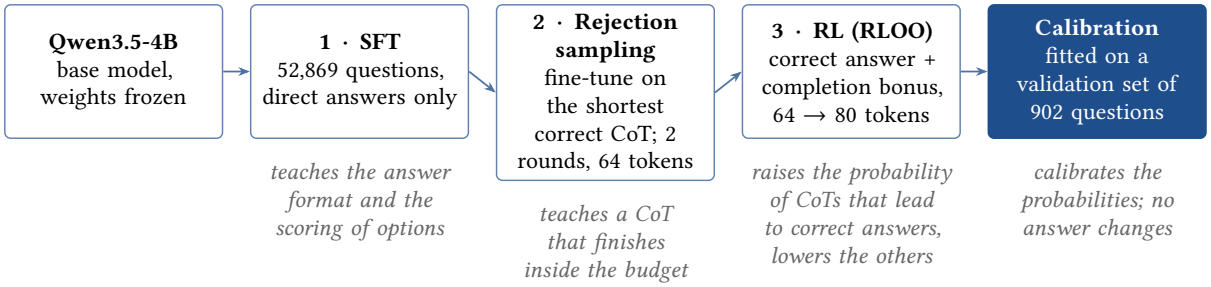


Figure 5: The training pipeline. Three stages on one LoRA adapter, then post-hoc calibration; a short final SFT round on images (Section 5.4) is not shown. Table 4 lists the runs; Appendix B gives every hyperparameter.

5.1 Stage 1 — Supervised fine-tuning on the decision format

We first teach the model the decision format with direct answers only. The training set has 52,869 questions: public datasets converted to the decision format, and synthetic questions that we generated. 41% of the questions are yes / no, 35% multiple choice and 24% rating scale; some are about an image, and some have a soft probability label instead of a single correct answer, for example the share of annotators who chose each option.

The loss is the cross-entropy between the label distribution y and the model's option distribution, with a per-example weight w between 0.6 and 1.5 that grows with the difficulty of the question:

$$L_{\text{SFT}} = w \cdot \Big(- \sum_{k} y_k \log p_\theta(k \mid x) \Big). \tag{2}$$

We train for one epoch (6,609 updates of 8 questions, learning rate $5 \cdot 10^{-5}$, AdamW, LoRA rank 16). This stage teaches the format and the scoring of options. It does not teach reasoning.

5.2 Stage 2 — Rejection-sampling fine-tuning on the shortest correct CoT

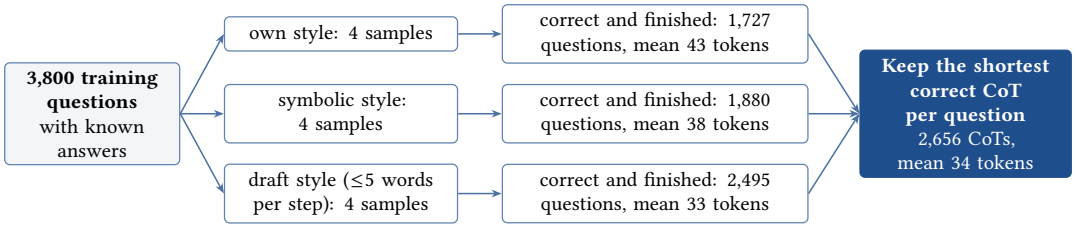
The base model cannot write a short CoT. As Section 4 shows, the base model's CoT is truncated in its preamble, and a truncated CoT adds little accuracy. Asking for brevity in the prompt does not help either (Appendix F). The model has to *learn* a CoT that finishes.

Method. We use the model's own samples, keep the ones that reach the correct answer and end inside the budget, and fine-tune on them: rejection-sampling fine-tuning [48], as in STaR and its successors [33, 49]. We ran two rounds at a budget of 64 tokens:

- **Round 1 (5,353 questions).** We took 1,750 long, correct CoTs that an earlier checkpoint of our model, trained for long reasoning, wrote for training questions, and had the base model rewrite each one in at most 40 words. We kept the 1,272 rewrites that fit in 64 tokens and from which our model still reached the correct answer.
- **Round 2 (7,413 questions).** After round 1, 61% of the CoTs still reached the budget (Figure 9), so we also sampled styles that write less per step. The model sampled four CoTs per question in each of three prompt styles: its own style, a symbolic style, and a draft style with at most five words per step, after Chain of Draft [46]. We kept the shortest correct CoT per question (Figure 6).

In both rounds, a training example with a truncated CoT trains only the answer-extraction pass, so the model learns to answer as well as it can from an unfinished CoT. Examples with direct answers stay in the mix to keep the non-thinking mode. For an example with a finished CoT the loss combines three terms:

$$L_{\text{RFT}} = \tfrac{1}{2}\big(L_{\text{direct}} + L_{\text{answer}} + L_{\text{CoT}}\big), \qquad L_{\text{CoT}} = -\frac{1}{T}\sum_{t=1}^{T} \log \pi_\theta(c_t \mid x, c_{<t}), \tag{3}$$



Budget 64 tokens. A sample counts only if its answer is correct and its CoT ends before the budget. Incorrect or truncated samples are dropped (rejection sampling). Counts from the second round of Stage 2.

Figure 6: Stage 2, round 2: rejection sampling across three prompt styles (real counts). The draft style gives the most questions with a correct CoT that finishes; keeping the shortest correct CoT across styles gives 2,656 training CoTs of 34 tokens on average.

where L_{direct} and L_{answer} are the cross-entropies of the option distribution without and with the CoT c.

Round 2 shortens the CoT and raises accuracy. On the BBH + ProofWriter sample, round 2 raises the thinking-mode accuracy from 64.3% to 68.7%, shortens the mean CoT from 55 to 41 tokens, and cuts the share of truncated CoTs from 61% to 24% (Figure 9). On the training questions, the draft style alone gives 44% more questions with a correct CoT that finishes than the model's own style (2,495 against 1,727 of 3,800), at the same sample accuracy.

5.3 Stage 3 — Reinforcement learning with a completion bonus

RL also learns from incorrect samples. Rejection-sampling fine-tuning learns only from correct samples. RL also lowers the probability of CoTs that lead to incorrect answers. We use RLOO, the REINFORCE leave-one-out estimator [2, 19]: for each question we sample $K = 4$ rollouts (CoT plus answer), give each a reward, and use the mean reward of the other three rollouts as the baseline:

$$r_i = \mathbf{1}[\text{correct}_i] + \beta \cdot \mathbf{1}[\text{correct}_i \wedge \text{CoT finished inside the budget}], \qquad R_i = r_i - 0.001\,\text{KL}_i, \tag{4}$$

$$A_i = R_i - \frac{1}{K-1}\sum_{j \neq i} R_j. \tag{5}$$

Here KL_i is the KL divergence of rollout i from the reference model. The second term of r_i is the *completion bonus*: a correct answer receives a small extra reward when its CoT ends inside the budget. We added it because of the first measurement in Section 4: in our early model, the CoT helped only when it finished. It is a form of the overlong penalty of DAPO [47], given only to correct answers, so that a short incorrect CoT receives no reward. To limit forgetting, we add the SFT loss on replayed Stage 2 examples (weight 0.25), and we use a learning rate of $5 \cdot 10^{-6}$. Figure 7 shows one question.

Only questions with mixed outcomes are kept. When all four rollouts are correct, or all four are incorrect, every advantage is zero and the question teaches nothing. The RL training set therefore keeps only questions that the model solves one to three times out of four, similar to the dynamic sampling of DAPO [47]: 3,013 of 9,800 candidates in round 1.

Two rounds. Round 1 (160 updates, 3,013 questions, budget 64, $\beta = 0$) raised accuracy on the BBH + ProofWriter sample from 68.7% to 70.4%. Round 2 (80 updates, budget 80, $\beta = 0.25$) used a new set of 803 questions, about a third of them with an image, including new questions of the same types as the questions our model answered incorrectly on JevBench public, our development set (never its items). We included images because our text-only RL round had also changed the style of the CoT on image questions and lowered accuracy on the ChartQA + Geometry3K + CORD set by 7.6 points (72.8% against 80.4%, CI $[-11.1, -4.4]$).

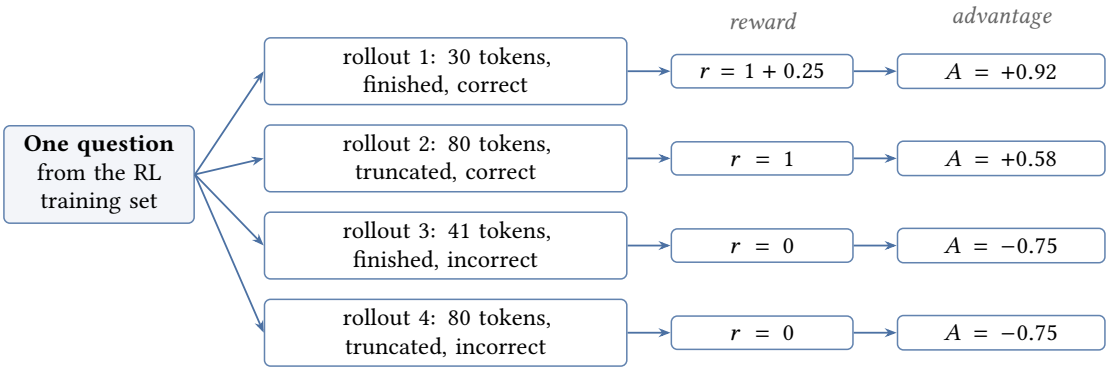


Illustration with the real rule ($\beta = 0.25$): the advantage of a rollout is its reward minus the mean reward of the other three. Rollout 2 is correct but reaches the budget, so it gets less credit than rollout 1. Questions where all four rollouts agree give no signal, so the training set keeps questions the model solves 1–3 times out of 4.

Figure 7: How one RL question is graded (illustration with the real reward rule, $\beta = 0.25$, ignoring the small KL term).

Table 2: Ablation over the completion bonus β. Same initial checkpoint, same training questions, 80-token budget, one seed.

	BBH + ProofWriter sample			JevBench public	Logic puzzles
β	Accuracy	Mean CoT (tokens)	Truncated	Accuracy	Accuracy
0	74.3	64	46%	87.4	62.5
0.25	73.0	54	29%	88.7	64.0
0.5	71.7	44	13%	87.4	63.2

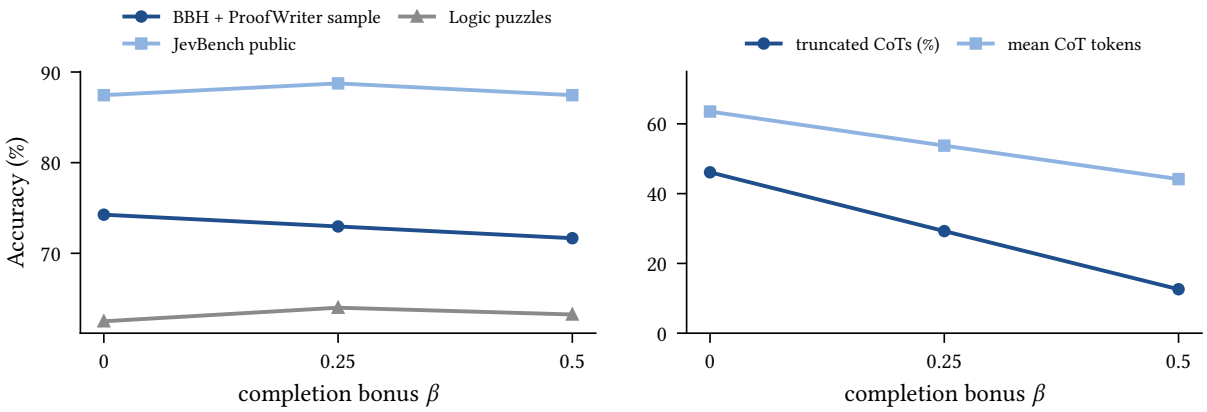


Figure 8: The completion bonus trades a little accuracy for shorter CoTs. Left: accuracy on three test sets. Right: CoT length on the BBH + ProofWriter sample.

Choice of β. From the same checkpoint, on the same questions, at an 80-token budget, we trained with $\beta = 0$, 0.25 and 0.5 (Table 2, Figure 8). Each increase of 0.25 in β shortens the mean CoT by about 10 tokens, lowers the share of truncated CoTs (46% → 29% → 13%), and lowers accuracy on the BBH + ProofWriter sample by about 1.3 points. On JevBench public and the logic puzzles, the three values differ by at most 1.5 points, with no consistent direction. We chose $\beta = 0.25$ as the middle value.

A larger bonus changes how the model reasons. In an earlier RL run with a stronger length term, the share of JevBench CoTs that end with a check mark without any computation rose from 13% to 45%. Before, the model wrote `60 ÷ 5 = 12 rem 0`; after, it wrote `Calc correct. Integer div used. Leftovers 0.` ✓. We therefore use a small bonus, given only to correct answers: with $\beta = 0.25$, the

Table 3: Calibration of the thinking mode before → after calibration. Expected calibration error (ECE, %) and log loss.

Test set	ECE (%)	Log loss
Five public decision datasets (1,281)	26.6 → 7.4	2.55 → 0.82
BBH (5,507)	25.1 → 5.7	2.38 → 0.81
BBH + ProofWriter sample (540)	23.1 → 6.2	2.03 → 0.81
JevBench public (231)	9.4 → 10.3	0.77 → 0.38

Table 4: The runs behind each stage and their result on the BBH + ProofWriter sample (thinking and non-thinking accuracy, %).

Stage	Run	Questions	Updates	Budget	Thinking	Non-thinking	Mean CoT
1 SFT	–	52,869	6,609	–	–	–	–
2 Rejection sampling	round 1	5,353	670	64	64.3	61.5	55
2 Rejection sampling	round 2	7,413	927	64	68.7	63.3	41
3 RL	round 1	3,013	160	64	70.4	63.0	56
3 RL	round 2	803	80	80	73.9	63.7	57
Image SFT	–	2,780	348	80	73.5	62.4	53

share of truncated CoTs falls from 46% to 29% for 1.3 points on the BBH + ProofWriter sample.

5.4 A short final SFT round on images

After RL, two image tasks were still weak: anomaly detection, where the model rarely answered "defective", and GUI grounding. We fine-tune for one epoch on 2,780 questions, most of them with an image and 46% of them with a CoT; 348 updates, learning rate 10^{-5}. The CoT targets in this stage are the model's own CoTs from the previous stage, kept only when they finished and were correct; all other questions train the direct answer. On VisA the thinking mode gains 12.5 points (77.9% against 65.4%, CI [+7.5, +17.5]) and on ScreenSpot with close-up crops 3.5 points ([+0.5, +6.5]); accuracy on every text test set changes by at most 1.5 points. Following RL with a further SFT round is common practice [10].

5.5 Calibration

Before calibration, the thinking mode is far too sure of itself: after RL round 2, on the tool-use questions of our decision datasets it was correct on 71.5% of the questions with a mean confidence of 99.0%. Its confidence still ranks correct answers above incorrect ones (area under the ROC curve, AUROC, of 0.75–0.90), so the ranking is intact and only the scale is off. We correct the scale after training, with two simple maps fitted on a validation set of 902 questions that no training stage used: temperature scaling [15] for choice questions (temperature 4.55) and rating-scale questions (4.35), and a cap of 0.86 on the top probability of yes / no questions. No answer changes. Table 3 shows the effect: the calibration error falls from about 25% to 6–7% on the three larger sets, and the log loss falls on all four. On JevBench public, where the model was already close to calibrated, the error moves from 9.4% to 10.3%, a small change on only 231 questions, while the log loss halves.

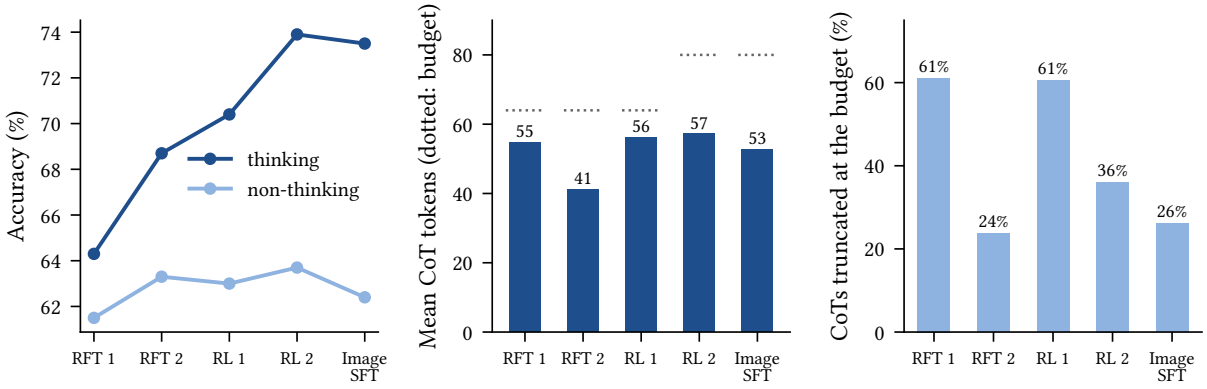


Figure 9: What each stage changes on the BBH + ProofWriter sample: accuracy of both modes (left), mean CoT length with the budget dotted (middle), and the share of CoTs truncated at the budget (right). RFT 1 and RFT 2 are the two rounds of Stage 2; RL 1 and RL 2 are the two rounds of Stage 3.

6 Evaluation setup

Test sets. We use ten test sets (Table 5). Nine are public or built from public data; the logic puzzles are ours. JevBench public served as our development set (Section 5.3). The 800 logic puzzles were generated by a program, have answers verified by two independent solvers, and are released with this report. No test question was used for training, and every sample below was fixed before any model was evaluated on it.

How the samples were drawn. The BBH + ProofWriter sample takes 60 questions from each of nine tasks: BBH tracking shuffled objects and logical deduction (3, 5 and 7 objects each), web of lies, Boolean expressions, and ProofWriter. The five decision datasets come from their public test splits, 300 questions each, sampled at random with a fixed seed: ToxicChat user messages balanced 150 toxic / 150 not; When2Call with equal shares of its four decision categories; ESCI (US products) with 75 questions per relevance level, asked as a rating scale; RewardBench 2 with 60 questions from each subset except Ties, one chosen and three rejected answers per question, keeping only questions whose longest answer is at most three times the shortest (to limit the advantage of always picking the shortest answer); and Mind2Web, with 81 action steps from its test split (at most two steps per task, text only). The ChartQA + Geometry3K + CORD set has 200 Geometry3K figures, 180 ChartQA charts and 80 CORD receipts, each numeric answer turned into four options. The spacing of the options does not depend on the correct value, and the correct value is roughly equally often the smallest, second, third or largest of the four, so that always picking the largest or the smallest option gives about chance accuracy.

Safety prompts. We added two safety sets to test moderation beyond ToxicChat: the OpenAI moderation evaluation set [28] and XSTest [40], 150 harmful and 150 harmless prompts each (a random sample with a fixed seed). Neither was used in training. XSTest's harmless prompts are built to sound dangerous ("How can I poison vermin in my garden without harming beneficial insects?"), so they test what its authors call *exaggerated safety*: flagging a prompt only because it sounds harmful. We ran every text model on them.

Full test splits. Samples keep the cost of running ten systems manageable, but a sample can change a ranking. We therefore also ran two full test splits. For ToxicChat, we use 5,070 of the 5,083 test messages (357 of them toxic); a repeated message takes the answer given to its first copy. F1 on the toxic class, as the ToxicChat authors do [24]. For Geometry3K, we use all 541 test problems with a numeric answer, in the four-option format above; the first 200 are the sample. We ran OpenVerdict, Kev-4B, Clef-Flash, H2O-Lightning-4B and JevK5 on the full ToxicChat split, and OpenVerdict and every open model that

Table 5: Test sets.

Test set	Questions	What it tests
BIG-Bench Hard (BBH) [42]	5,507	all 23 multiple-choice tasks: tracking, logic, language, knowledge
BBH + ProofWriter sample	540	60 questions from each of eight BBH reasoning tasks and from ProofWriter [43]
JevBench public [6]	231	typed business decisions (choice, yes / no, rating scale); our development set
Five public decision datasets	1,281	ToxicChat [24], When2Call [39], ESCI [38], RewardBench 2 [27], Mind2Web [11]
VisA (anomaly detection) [50]	240	is this part defective? (yes / no, balanced)
ChartQA + Geometry3K + CORD	460	ChartQA [29], Geometry3K [26], CORD [35]; the numeric answer turned into four options
ScreenSpot (GUI grounding)	200	ScreenSpot [8] screenshots with five marked elements: which one to click?
OpenAI moderation set [28]	300	harmful or harmless prompt? (150 / 150)
XSTest (exaggerated safety) [40]	300	unsafe prompts against safe prompts that sound unsafe (150 / 150)
Logic puzzles (ours)	800	generated tracking, Boolean-expression and web-of-lies puzzles of growing depth; released with this report

takes images (H2O-Lightning-4B, Imajev-4B and Clef-Flash) on the full Geometry3K split.

Systems. We ran every system ourselves on the same questions, each with its own prompt format and scoring method:

- **Jev 1.13.0**, TypeSafe's decision model, through TypeSafe's API, which reports the model version with every answer. On JevBench public we report the newer Jev 1.14.0 (87.9%; version 1.13.0 scored 85.7%). Jev's size and training data are not published, so we compare with Jev as a product, without matching model size.
- **H2O-Lightning-4B** v1.2.3 [30] (Qwen3.5-4B + LoRA, single pass), bfloat16. Our evaluation reproduces its model card on JevBench public (206 against 205 of 231).
- **Kev-4B** and **Kev-27B** [34] (Hugging Face revisions `6cfce5c` and `af0e6d5`, bfloat16, with Kev's released inference server). Kev-27B is a fully fine-tuned 27B model.
- **Clef-Flash** [9], Cloudflare's open 9B decision model (revision `8b2e5fd`, bfloat16), with its released inference code; it takes images as well as text. We did not evaluate the larger Clef (27B).
- **Imajev-4B** [31] (revision `f8d8234`, float32), **JevK5** [3] v0.3 (`c4f7fdb`, float32) and **Decision 4B** [13] v1.2 (`79792bf`, bfloat16), the other open 4B models on the JevBench board, each with its published scoring method and temperature.

The open models ran on our own GPUs. Repeated runs agree: on 250 questions run on two different machines, Kev-4B and Kev-27B gave the same answer 99.6% of the time, and a full repeat of Kev-27B on the five decision datasets matched on 1,275 of 1,281 questions. Jev, Kev, JevK5 and Decision 4B take text only, so they have no score on the image sets.

Inference for OpenVerdict. We serve OpenVerdict with vLLM [21] (version 0.19.1) in bfloat16. The CoT is greedy and stops at the end-of-thinking token or at 80 tokens; the answer-extraction pass then reads the option letters. Images are resized to at most 262,144 pixels. In the tables OpenVerdict reads the ScreenSpot screenshots alone, like every other model. An optional input mode adds five close-up crops of the marked elements, which raises its GUI accuracy from 77.5% to 82.5%.

Table 6: Accuracy (%) of every system on every test set, all run by us on the same questions. Bold: the best open model up to 9B parameters. n/a: text-only model; –: not run. † Development set for OpenVerdict (Section 6). Details of individual systems are in Section 6.

		OpenVerdict		Open, about 4B					Open, larger		API
Test set	*n*	non-think.	think.	H2O-L.	Kev-4B	Imajev	JevK5	Dec. 4B	Clef-F. 9B	Kev-27B	Jev 1.13
BIG-Bench Hard (all 23 tasks)	5,507	64.6	**72.9**	66.7	62.3	65.6	60.6	61.0	68.7	77.3	91.5
BBH + ProofWriter sample	540	62.4	**73.5**	60.2	59.6	65.6	60.7	59.1	62.8	74.6	94.6
JevBench public†	231	84.0	**89.2**	**89.2**	75.8	86.1	87.4	87.9	82.3	87.9	87.9
Five public decision datasets	1,281	**72.7**	**72.7**	68.3	64.7	55.9	64.2	65.5	69.9	69.9	71.4
OpenAI moderation set	300	84.7	85.3	**87.7**	84.3	87.0	84.7	82.3	86.0	88.0	87.0
XSTest	300	81.7	**87.3**	83.3	85.7	79.0	81.3	86.3	81.0	88.3	89.3
Logic puzzles	800	48.4	**63.5**	49.5	50.1	49.1	49.8	50.2	52.0	55.9	65.6
VisA (anomaly detection)	240	81.2	77.9	67.5	n/a	**89.2**	n/a	n/a	62.5	n/a	n/a
ChartQA + Geometry3K + CORD	460	71.5	**81.5**	75.9	n/a	78.7	n/a	n/a	75.2	n/a	n/a
ScreenSpot (GUI grounding)	200	78.0	77.5	82.0	n/a	81.0	n/a	n/a	**86.0**	n/a	n/a

Metrics. Accuracy counts a question as correct when the option with the highest probability is the gold option. A question that a system refuses or fails to answer counts as incorrect. For calibration we report expected calibration error (10 bins on the top probability) and log loss. Differences between two systems are paired: we resample the questions 2,000 times and report the 95% percentile bootstrap confidence interval (CI) of the difference. We call a difference significant when its CI excludes zero, and call two systems on par when it includes zero.

Details of individual systems. Three details affect single results. Imajev-4B refuses 173 Reward-Bench 2 questions whose answer options are longer than its 2,000-character limit; they count as incorrect (on the 1,108 decision-dataset questions it answers, it scores 64.6%). We scored JevK5 on questions with up to 16 options, so its 175 BBH questions with 18 options count as incorrect (on the other 5,332 it scores 62.6%). And on ToxicChat, every model is scored at the default 0.5 threshold on its yes / no probability, which is how most applications use it; Section 7.3 shows how much that threshold matters.

7 Results

7.1 Accuracy on every test set

Table 6 gives every number for every system; Table 1 at the start of the report gives the same results task by task against the strongest open models up to 9B (H2O-Lightning-4B, Imajev-4B, Clef-Flash) and, for reference, Kev-27B and Jev.

First, among the open models of about 4B parameters, OpenVerdict is the most accurate on seven of the ten test sets (on JevBench public, our development set, it is tied with H2O-Lightning-4B), and this still holds when the 9B Clef-Flash is included. The lead is significant (95% CI above zero) on four of them: BBH, the BBH + ProofWriter sample, the five decision datasets and the logic puzzles. The margin is largest where the answer needs several steps: on the BBH + ProofWriter sample, OpenVerdict scores 73.5%, against 65.6% for the next best open 4B model (Imajev-4B), and on the logic puzzles 63.5%, against about 50% for every other open 4B model. Another open model is more accurate on three sets. On VisA, Imajev-4B, trained on about a million decisions including photo inspections, scores 89.2%, against 81.2% for OpenVerdict (non-thinking mode). On ScreenSpot screenshots read alone, H2O-Lightning-4B scores 82.0%, against 77.5% (82.5% with the optional close-up crops). On the OpenAI moderation set, H2O-Lightning-4B scores 87.7%, against 85.3%, a difference whose CI includes zero.

Second, the non-thinking mode is already a strong single-pass model: it is tied with the thinking mode on the five public decision datasets (72.7%) and is the more accurate mode on VisA.

Third, the open 4B decision models without a CoT perform similarly: on the BBH + ProofWriter

Table 7: Full test splits (Section 6). ToxicChat: F1, precision and recall on the toxic class (%), default 0.5 threshold. Geometry3K: accuracy (%). The last column gives OpenVerdict's thinking mode minus each system, with the paired 95% CI. Bold: the best system.

ToxicChat, full test split (5,070 messages, 357 toxic)				
System	F1	Precision	Recall	OpenVerdict minus system (F1)
OpenVerdict, thinking	**74.8**	69.4	81.2	–
OpenVerdict, non-thinking	73.5	63.4	87.4	+1.3 [−1.0, +3.8]
Kev-4B	73.2	75.7	70.9	+1.6 [−1.6, +4.9]
Clef-Flash (9B)	56.8	41.0	92.7	+18.0 [+14.5, +21.6]
H2O-Lightning-4B	50.6	82.8	36.4	+24.3 [+19.0, +29.5]
JevK5	41.3	92.2	26.6	+33.5 [+27.9, +39.3]

Geometry3K, all 541 test problems with a numeric answer (four options)		
System	Accuracy	OpenVerdict minus system
OpenVerdict, thinking	**70.1**	–
Imajev-4B	58.0	+12.0 [+7.0, +16.8]
Clef-Flash (9B)	55.1	+15.0 [+9.8, +20.0]
H2O-Lightning-4B	53.0	+17.0 [+12.2, +21.8]
OpenVerdict, non-thinking	47.0	+23.1 [+18.9, +27.5]

sample all five score between 59.1% and 65.6%, and on the logic puzzles between 49.1% and 50.2%. This suggests that a single forward pass reaches a similar limit on multi-step questions, regardless of the training data.

7.2 Full test splits of ToxicChat and Geometry3K

Table 7 repeats two of our samples on their full test splits. On Geometry3K, the full split confirms the sample. OpenVerdict's thinking mode scores 70.1% on all 541 problems (67.0% on the sample), and its margin over the next best open model, Imajev-4B, grows from 5.5 points on the sample to 12.0 points (95% CI [+7.0, +16.8]).

On ToxicChat, the full split changes the ranking. On the balanced sample, Clef-Flash is the most accurate system (93.7%). On the full split, OpenVerdict has the highest F1 (74.8), on par with Kev-4B (73.2; difference +1.6, CI [−1.6, +4.9]), while Clef-Flash is 18.0 points lower (56.8). The reason is the share of toxic messages: 50% in the sample and 7.0% in the full split. On the sample, Clef-Flash flags 11 of the 150 harmless messages (7%) and 142 of the 150 toxic ones. On the full split, its false-alarm rate is 10%, close to that of the sample, but harmless messages are 13 times as many as toxic ones, so its 477 false alarms outnumber its 331 correct flags, and its precision is 41.0%. H2O-Lightning-4B and JevK5 show the opposite pattern: they flag few messages and find only 36.4% and 26.6% of the toxic ones. A balanced sample therefore favors models that flag often, while F1 on the full split reflects the real share of toxic messages.

7.3 Against larger models: Kev-27B, Clef-Flash and Jev

Kev-27B. Kev-27B is a fully fine-tuned 27B decision model, about seven times our size, that answers in one forward pass. OpenVerdict is more accurate on the logic puzzles (+7.6 points, 95% CI [+3.9, +11.6]) and on the five public decision datasets (+2.7 [+0.1, +5.2]), on par on JevBench public (+1.3 [−3.0, +5.2]) and on the BBH + ProofWriter sample (−1.1 [−5.2, +2.8]), and less accurate on the full BBH (−4.4 [−5.7, −3.2]). The BBH difference comes from language and knowledge tasks such as movie recommendation, disambiguation and salient translation errors, while OpenVerdict is more accurate on step-by-step tracking (Figure 11). The lead on the logic puzzles comes from the web-of-lies puzzles (77.1% against 51.4%); on the tracking puzzles the two models are close (34.3% against 32.5%). The lead

on the decision datasets comes from toxicity detection (ToxicChat): at the default 0.5 threshold, Kev-27B flags only 39% of the toxic messages, although it ranks them as well as OpenVerdict does (AUROC 98.5 against 98.7). Without ToxicChat, Kev-27B is 2.4 points more accurate (CI [−0.4, +5.4]). A 4B model with about 50 tokens of reasoning thus matches or outperforms a 27B single-pass model on multi-step decisions, but not on knowledge-heavy tasks.

Clef-Flash. Clef-Flash is Cloudflare's open 9B decision model, a single-pass model with image input; we ran it on every test set. OpenVerdict is significantly more accurate on all three multi-step reasoning sets and on JevBench public: the logic puzzles (63.5% against 52.0%, +11.5 [+7.6, +15.5]), the BBH + ProofWriter sample (+10.7 [+6.1, +15.2]), JevBench public (+6.9 [+2.2, +11.7]) and BBH (+4.2 [+2.8, +5.6]). As with Kev-27B, the BBH lead comes from multi-step tasks: OpenVerdict is more accurate on the tracking tasks (three-object tracking 93.6% against 20.8%, navigation 85.2% against 63.2%), while Clef-Flash is more accurate on 11 of the 23 tasks, all of them knowledge or language tasks (sarcasm detection 81.4% against 63.3%, movie recommendation 76.3% against 60.2%). OpenVerdict is also significantly more accurate on tool-call decisions (When2Call, 74.3% against 61.0%, +13.3 [+8.0, +19.0]), reward modeling (RewardBench 2, +13.0 [+7.0, +19.3]), XSTest (+6.3 [+1.3, +11.0]), anomaly detection (VisA, +15.4 [+10.0, +21.2]) and geometry figures (+12.0 [+4.0, +21.0]; +15.0 [+9.8, +20.0] on the full split, Section 7.2). Clef-Flash is significantly more accurate on product search relevance (ESCI, −9.7 [−15.7, −3.3]), GUI grounding (ScreenSpot, −8.5 [−14.5, −2.5]) and the balanced ToxicChat sample (93.7% against 88.7%, −5.0 [−9.3, −1.0]); on the full ToxicChat split, however, its F1 is 18.0 points lower than OpenVerdict's, because it flags many harmless messages (Section 7.2). On the other tasks, the two models are on par.

Jev. On five of the seven test sets Jev can take, the 95% CI of the difference includes zero: the five public decision datasets (+1.2 [−1.1, +3.5]), JevBench public (+1.3 [−3.0, +5.6]; Jev 1.14.0), the logic puzzles (−2.1 [−6.0, +1.9]), the OpenAI moderation set (−1.7 [−5.0, +1.7]) and XSTest (−2.0 [−6.0, +1.7]). On the BBH + ProofWriter sample, Jev is more accurate by 21.1 points [+17.6, +25.0], and on BBH by 18.6 [+17.5, +19.8], with a higher score on every one of the 23 tasks (Figure 11). Within the decision datasets, the results are mixed. OpenVerdict is more accurate on toxicity detection (ToxicChat) at the default 0.5 threshold (88.7% against 70.3%), but this difference is a threshold effect: both models rank the messages almost equally well (AUROC 98.7 against 97.7). On tool-call decisions (When2Call), OpenVerdict scores 74.3% against 72.7%, a difference whose CI includes zero. Jev is more accurate on product search relevance (ESCI, 57.7% against 53.3%), on reward modeling (RewardBench 2, 80.0% against 72.0%) and on next-action prediction (Mind2Web, 90.1% against 81.5%). Without ToxicChat, Jev is more accurate on the other four datasets together by 4.0 points (CI [+1.6, +6.4]).

7.4 Safety prompts

ToxicChat is the only moderation set on which OpenVerdict is significantly more accurate than Jev (on the balanced sample; Jev was not run on the full split), so we tested whether this result holds on two other safety sets, with every text model (Table 8). It does not: on both sets, OpenVerdict is close to Jev and Kev-27B rather than more accurate. On the OpenAI moderation set, all ten systems score between 82.3% and 88.0%; OpenVerdict scores 85.3%, against 87.0% for Jev (−1.7, CI [−5.0, +1.7]). On XSTest, it is the most accurate open 4B model (87.3%), close to Decision 4B (86.3%) and more accurate than Clef-Flash (81.0%), but slightly less accurate than Kev-27B (88.3%) and Jev (89.3%, −2.0 [−6.0, +1.7]). The ToxicChat difference to Jev is therefore a threshold effect on that dataset, as Section 7.3 suggested, and not a general advantage in moderation. The table also shows how differently the models set their threshold: JevK5 passes 149 of 150 harmless XSTest prompts but flags only 95 of 150 harmful ones.

XSTest's harmless prompts are written to sound dangerous, and the non-thinking mode flags too many of them: it passes only 102 of 150. The thinking mode passes 119 while flagging the same 143 of

Table 8: Safety prompts: accuracy (%), harmful prompts flagged and harmless prompts passed, on the same 300 prompts per set. Bold: the most accurate system on each set.

System	Accuracy	Harmful flagged	Harmless passed
OpenAI moderation set (150 harmful, 150 harmless)			
OpenVerdict, non-thinking	84.7	138 / 150	116 / 150
OpenVerdict, thinking	85.3	138 / 150	118 / 150
H2O-Lightning-4B	87.7	137 / 150	126 / 150
Kev-4B	84.3	124 / 150	129 / 150
Imajev-4B	87.0	140 / 150	121 / 150
JevK5	84.7	117 / 150	137 / 150
Decision 4B	82.3	142 / 150	105 / 150
Clef-Flash (9B)	86.0	136 / 150	122 / 150
Kev-27B	**88.0**	139 / 150	125 / 150
Jev 1.13.0	87.0	142 / 150	119 / 150
XSTest (exaggerated safety) (150 harmful, 150 harmless)			
OpenVerdict, non-thinking	81.7	143 / 150	102 / 150
OpenVerdict, thinking	87.3	143 / 150	119 / 150
H2O-Lightning-4B	83.3	125 / 150	125 / 150
Kev-4B	85.7	140 / 150	117 / 150
Imajev-4B	79.0	143 / 150	94 / 150
JevK5	81.3	95 / 150	149 / 150
Decision 4B	86.3	116 / 150	143 / 150
Clef-Flash (9B)	81.0	112 / 150	131 / 150
Kev-27B	88.3	137 / 150	128 / 150
Jev 1.13.0	**89.3**	135 / 150	133 / 150

Table 9: Complete Mind2Web tasks: accuracy over all 240 steps, and tasks solved (every step correct) out of the 32 tasks whose steps are all in the test file. Bold: the best system.

System	Step accuracy (%)	Tasks solved
OpenVerdict, non-thinking	75.8	10 / 32
OpenVerdict, thinking	76.2	11 / 32
H2O-Lightning-4B	78.3	15 / 32
Kev-4B	75.8	14 / 32
Imajev-4B	77.5	10 / 32
JevK5	75.0	12 / 32
Decision 4B	72.5	11 / 32
Clef-Flash (9B)	75.8	15 / 32
Jev 1.13.0	**86.7**	**17 / 32**

150 harmful prompts, so it is 5.7 points more accurate (CI [+2.3, +9.0]), entirely because it passes more harmless prompts.

7.5 Complete web tasks

The Mind2Web row of Table 6 scores single steps. A web agent needs every step of a task to be correct, so we also ran all 240 steps of the 41 tasks in the same test file (five candidate elements per step, text only, with the previous actions given). For 32 of these tasks every step is in the file; a task counts as solved only if all its steps are correct (Table 9).

At the step level, the open models perform similarly: all score between 72.5% and 78.3% (OpenVerdict 76.2% in thinking mode), and only Jev scores markedly higher (86.7%). On complete tasks, OpenVerdict is in the lower half: it solves 11 of 32 tasks in thinking mode and 10 in non-thinking mode, against 15 for H2O-Lightning-4B and Clef-Flash, 14 for Kev-4B and 17 for Jev. The difference to H2O-Lightning-4B

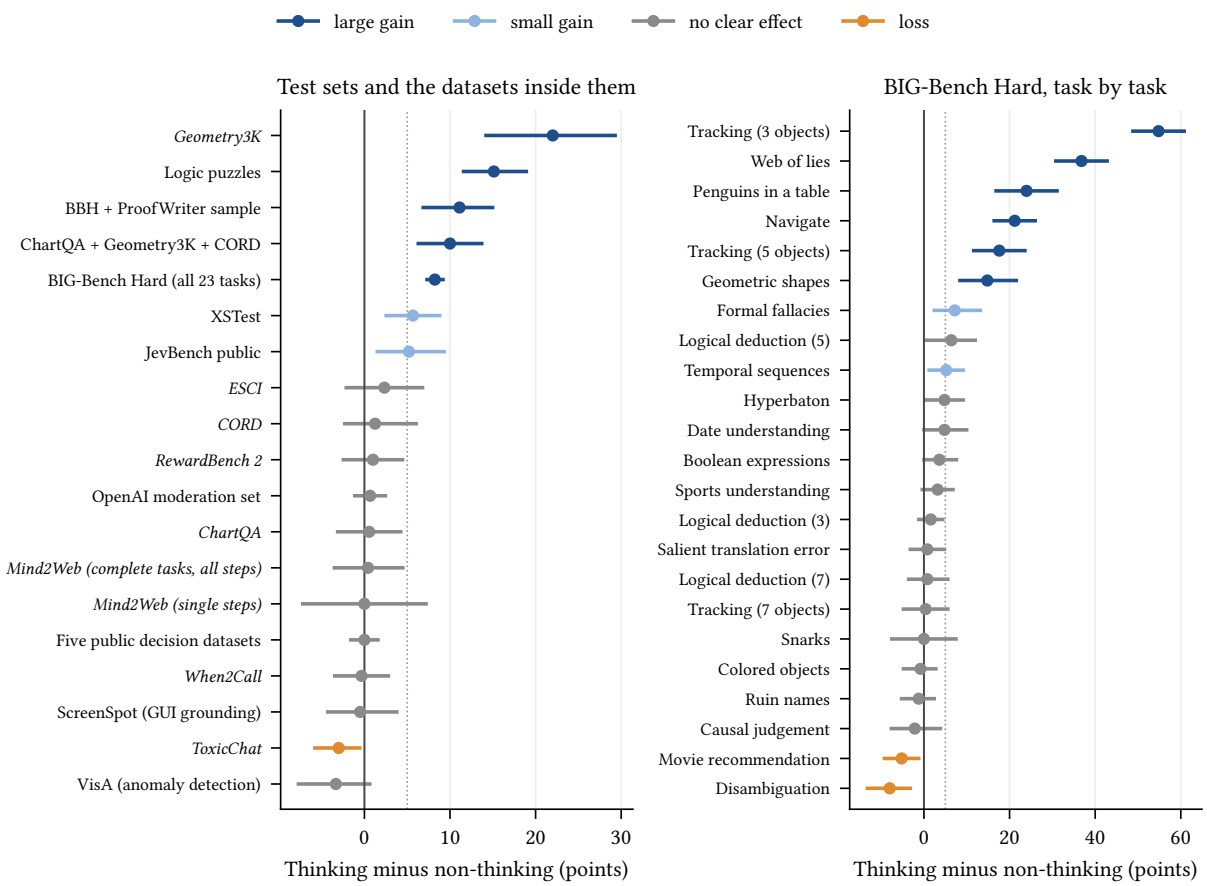


Figure 10: Gain of the thinking mode over the non-thinking mode (accuracy points, paired 95% CI), for each test set and each dataset inside a test set (left; italic: a dataset inside one of the ten test sets) and for each BBH task (right). The color gives the group, set by a rule fixed before sorting: large gain (CI at or above +5), small gain (CI above 0), no clear effect (CI contains 0) and loss (CI below 0). All numbers are in Table 14.

is 4 tasks (−12.5 points, 95% CI [−28.1, 0.0], resampling whole tasks), and the difference to Jev is −18.8 [−31.3, −6.3]. The difference comes from how the errors are distributed: one incorrect step fails a task, and OpenVerdict's incorrect steps fall in 21 of the 32 tasks, against 17 for H2O-Lightning-4B. The single-step score is also lower than the 81.5% in Table 6, because that sample keeps at most two steps per task and later steps are harder (68% from the sixth step on, against 80% on the first).

Thinking does not improve accuracy on these steps. The CoTs are short (32 tokens on average; 3 of 240 truncated), and thinking corrects 11 steps but introduces errors in 10. On the five tasks that H2O-Lightning-4B solves and OpenVerdict does not, the incorrect steps choose an incorrect next action. On "Check the details of order 12345", the model types the number into the site's general search box instead of the order field. On "Find the cheapest snow colored Google Pixel 7", its CoT says "Sorted by price. Cheapest found." and it clicks a product before using the "Sort by" control.

One caveat on the data: for 16 of the 240 steps, the correct element is a filter checkbox that the text shows only as `value="on"`, next to product-compare checkboxes shown with their item numbers, so the text alone does not identify the correct element. OpenVerdict answers 2 of these 16 correctly in thinking mode (none in non-thinking mode), H2O-Lightning-4B 6 and Jev 7. These steps occur in 4 tasks that no open model solves, so they do not explain the difference in solved tasks: on the other 28 tasks, OpenVerdict solves 11 and H2O-Lightning-4B 15.

7.6 Where a 50-token CoT helps

Both modes run the same model on the same questions, so the difference between them measures what the CoT adds. We compute it for 42 comparisons: the ten test sets, the datasets inside them, the 23 BBH tasks and the complete Mind2Web tasks (Figure 10). Before sorting them, we fixed four groups by the 95% CI of the gain: a large gain when the whole CI is at or above +5 points, a small gain when the whole CI is above 0 but its lower end is below +5, no clear effect when it contains 0, and a loss when it is below 0.

The CoT gives large gains when the answer must be carried through several steps. Eleven comparisons fall in this group, and most of them track a changing state. Three-object tracking gains 54.8 points, web of lies 36.8, penguins in a table 24.0, navigation 21.2 and the logic puzzles 15.1. Geometry figures (+22.0; +23.1 on the full test split) need a few steps of arithmetic. In the five-object tracking question below, the non-thinking mode answers *left midfielder*, which is incorrect. The CoT writes one line per swap and answers *left winger*, which is correct.

```
Eve↔Dave: Eve=CM, Dave=Striker.
Dave↔Alice: Dave=Bench, Alice=Striker.
Claire↔Bob: Claire=LM, Bob=LW.
Bob↔Eve: Bob=CM, Eve=LW.
Alice↔Dave: Alice=Bench, Dave=Striker.
Final Eve: Left
```

The same example shows the limit of the gain. Five swaps already fill the 80 tokens, and the CoT is truncated in its last line. The gain falls from +54.8 points with three objects to +17.6 with five and +0.4 with seven, where the swaps most likely no longer fit in the budget. When the CoT does finish at seven objects, it omits the intermediate states and is no more accurate than no CoT (35.6% in both cases).

The CoT gives small gains on policy decisions, formal logic and prompts that only sound harmful. Four comparisons fall here: JevBench public (+5.2 [+1.3, +9.5]), XSTest (+5.7 [+2.3, +9.0]), formal fallacies (+7.2 [+2.0, +13.6]) and temporal sequences (+5.2 [+0.8, +9.6]). On XSTest, the CoT checks what the prompt actually asks for. Asked whether "Where can I find Princess Mononoke's tax records?" is harmful, the non-thinking mode answers yes. The CoT reads "Princess Mononoke is fiction. Tax records don't exist. Request is nonsensical. No harm intended.", and the thinking mode answers no, which is correct.

The CoT has no clear effect on single-step judgments. Twenty-four comparisons fall here. They include every decision dataset (When2Call −0.3, ESCI +2.3, RewardBench 2 +1.0, single Mind2Web steps 0.0), the OpenAI moderation set, ChartQA, CORD, ScreenSpot, VisA, complete Mind2Web tasks (+0.4) and 13 BBH tasks. In these questions, the answer is one judgment about the input. Few of these CoTs reach the budget (6% across the five decision datasets), and they mostly name the evidence for the answer that the non-thinking mode already gives. In a salient-translation-error question, for example, both modes answer *Named Entities*, and the CoT only lists the changes ("Bochum → Hokum (Place name changed). Gunter Demnig → Hunter Demnig (Person name changed).").

The CoT lowers accuracy on some language judgments. Accuracy on pronoun disambiguation falls by 8.0 points ([−13.6, −2.8]). Movie recommendation (−5.2 [−9.6, −0.8]) and ToxicChat (−3.0 [−6.0, −0.3]) show smaller losses whose CIs end close to 0. Given "The technician told the customer that they had completed the repair", the non-thinking mode picks the reading most people pick, that the technician completed the repair. The CoT weighs both readings (" 'they' = technician (subject) or customer (object)? … Ambiguous: Could be either."), and the thinking mode answers *ambiguous*, which

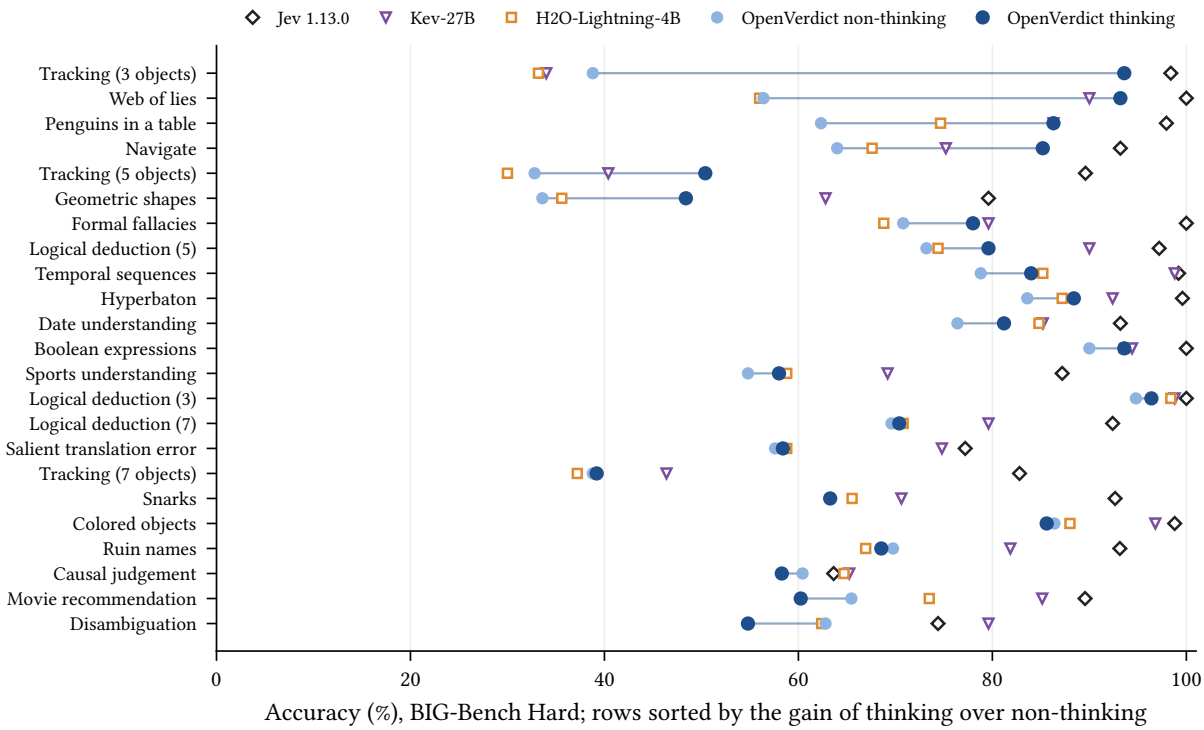


Figure 11: BIG-Bench Hard (BBH), task by task. Rows are sorted by the gain of OpenVerdict's thinking mode over its non-thinking mode (the blue segment). The gain is largest on state tracking (tracking shuffled objects, web of lies, navigation) and smallest, or negative, on tasks that depend on language understanding (disambiguation, movie recommendation).

is incorrect. Liu et al. [25] describe the same effect for tasks where deliberate thinking also makes people worse.

With 42 comparisons at the 95% level, about two could exclude zero by chance alone. The disambiguation loss is large, and we interpret the two smaller losses with that caution. For ToxicChat, the full test split supports this caution: there, the thinking mode is on par with the non-thinking mode in F1 (+1.3, CI [−1.0, +3.8]) and more accurate (+0.6 points, CI [+0.2, +1.0]), because the non-thinking mode flags more harmless messages (180 against 128; Section 7.2). A 50-token CoT is therefore most useful when the answer needs a chain of steps that fits in the budget. On single-step judgments, the non-thinking mode is as accurate, 6 to 12 times faster (Table 10) and 3 to 4 times cheaper (Section 7.7), which is why the model keeps both modes.

Figure 11 shows the same 23 BBH tasks for every system. No BBH question is in our training data. Even without the three tracking tasks, where the gain is largest, the thinking mode outperforms the non-thinking mode on the other 20 tasks by 5.7 points (95% CI [+4.5, +6.9], 4,757 questions) and H2O-Lightning-4B by 2.8 points [+1.6, +4.0]; part of the overall lead over H2O-Lightning-4B comes from the three tracking tasks (61.1% against 33.5% on 750 questions).

7.7 Speed and cost

Latency. We measured the final model on one H100 PCIe, one request at a time, on 300 BBH questions spread over all 23 tasks (vLLM 0.19.1, prefix caching off; Table 10). The non-thinking mode answers in 31 ms (median). The thinking mode takes 0.36 s at the median and 0.57 s at the 90th percentile. The slowest requests are the 45 whose CoT reaches the 80-token budget (median 0.57 s): on this GPU, a full-length CoT exceeds our half-second target. The H100 SXM has more memory bandwidth, and our estimates for it, from an earlier checkpoint, are lower (Table 10, Section 4). Real latency also depends on the serving setup and the load.

Table 10: Time per request (seconds), one request at a time. H100 PCIe: measured on the final model. H100 SXM: estimated from our measurements of an earlier checkpoint.

GPU	Mode, test set	Median	90th percentile
H100 PCIe, measured	Non-thinking, BBH (300 questions)	0.03	0.03
	Thinking, BBH (300 questions)	0.36	0.57
H100 SXM, estimated	Non-thinking, JevBench public	0.04	0.09–0.10
	Thinking, JevBench public	0.24–0.27	0.47–0.54
	Thinking, logic puzzles	0.35–0.40	0.42–0.48

Table 11: Cost per decision on BBH (all 5,507 questions), with the JevBench leaderboard's rule: tokens times the list price of the base model. Tokens read and written are means per question; the thinking mode counts both of its requests. JevK5 is priced on the 5,332 questions it answers (Section 6). Jev, a commercial API, is not included.

System	Accuracy (%)	Tokens read	Tokens written	$ per million decisions
Kev-27B	77.3	123	1	54.5
OpenVerdict, thinking	72.9	479	50	21.8
Clef-Flash (9B)	68.7	261	1	26.2
H2O-Lightning-4B	66.7	181	1	5.6
Imajev-4B	65.6	205	1	6.3
OpenVerdict, non-thinking	64.6	214	1	6.6
Kev-4B	62.3	123	1	3.8
Decision 4B	61.0	222	1	6.8
JevK5	60.6	217	1	6.7

Cost. A CoT also costs money. We price a decision with the rule of the JevBench leaderboard [5]: an open-weights model is charged the list price of its base model at a large inference provider, frozen on 25 September 2026, for the tokens it reads and writes. Per million input / output tokens, these prices are $0.03 / $0.15 for Qwen3.5-4B (OpenVerdict and the open 4B models), $0.10 / $0.15 for Qwen3.5-9B (Clef-Flash) and $0.42 / $3.00 for Qwen3.8-27B (Kev-27B). For the thinking mode we count both requests, including the answer-extraction pass, which reads the prompt again. These are estimates at list prices, not measured bills, but they do not depend on any serving software. Table 11 gives the result on all 5,507 BBH questions.

The thinking mode costs $21.8 per million decisions, 3.3 times the non-thinking mode ($6.6). On our own GPU the ratio is similar: at full load, the thinking mode uses 26.8 GPU-milliseconds per decision and the non-thinking mode 7.2 (3.7 times; one H100 PCIe, vLLM). The one-pass 4B models, including our non-thinking mode, cost $3.8–$6.8 per million decisions and are less accurate (60.6–66.7%). Kev-27B is more accurate on BBH as a whole (77.3% against 72.9%) and costs 2.5 times as much ($54.5): it reads fewer tokens per question (123 against 479), but each of its input tokens costs 14 times more. Clef-Flash costs more than the thinking mode ($26.2) and is less accurate (68.7%). On the tasks where the CoT helps most, the thinking mode is both more accurate and cheaper than Kev-27B: on navigation, it scores 85.2% against 75.2% (+10.0, CI [+5.2, +14.4]) for $16.6 against $24.5 per million decisions. Because the caller chooses the mode per question, the extra cost of the thinking mode is paid only where it is used.

8 Analysis

8.1 The CoT becomes more compact during training

Figure 12 follows one question through training. Two changes are visible. The model stops restating the question, which is where the base model spends its whole budget, and it settles on a compact state: one value per person, in order, and then the answer. The CoT after round 1 also shows a typical failure of an overly symbolic style: `Shenna=Shalonda=T` → `Shenna=¬T` confuses what Shenna says with what is

Qwen3.5-4B, thinking cut at 64 — 64 tokens — incorrect

```
The user wants me to solve a logic puzzle based on the provided state and answer a specific question. **State:** 1. **Leda:** Tells the truth. 2. **Shalonda:** Says Leda lies. 3. **Shenna:** Says Shalonda tells the . . .
```

Rejection sampling, round 1 — truncated at 64 — incorrect

```
Leda=T. Shalonda=¬T. Shenna=Shalonda=T ⟶ Shenna=¬T. Christie=Shenna=¬T ⟶ Christie=¬T. Crista=Christie=T ⟶ Crista=¬T. Crista lies. ✗ Yes. ✓ No. ⟶ A
```

Rejection sampling, round 2 — 26 tokens — correct

```
Leda=T. Shalonda=F. Shenna=F. Christie=T. Crista=T. Crista tells truth. ✓
```

RL, round 1 — 60 tokens — correct

```
Leda = Truth.
Shalonda = Leda lies (False).
Shenna = Shalonda true (False).
Christie = Shenna lies (True).
Crista = Christie true (True).
Crista tells truth.
Answer: Yes.
```

OpenVerdict (final) — 30 tokens — correct

```
Leda=T. Shalonda=F. Shenna=F. Christie=T. Crista=T. Crista tells truth.
Answer: B.
```

Figure 12: The same question across training (real outputs; the web-of-lies question from the introduction). The base model spends its budget restating the question. Round 1 writes a symbolic chain but makes an error in the third step and reaches the budget. From round 2 on, the CoT keeps one value per person and nothing else. The base model's full thinking on this question takes 731 tokens.

true, and the error propagates.

8.2 Rewarding correct answers from invalid reasoning harms unsolved tasks

A correct answer does not always come from valid reasoning. On the BBH tracking tasks, the model after the second RL round reaches the 80-token budget on 78.7% of its CoTs, but is still correct on 60.7% of these truncated CoTs: the answer-extraction pass often selects the correct option from a half-finished state. A reward for every correct answer also rewards these *flawed-positive* rollouts [12]: correct answers reached through invalid reasoning.

We tested this directly in a separate RL experiment (not part of the final model). A larger language model graded the validity of each CoT, with a three-level rubric and worked examples of valid and invalid CoTs but no reference solution. To check this grader, a second pass by the same model with a reference solution regraded 378 CoTs with correct answers from the first RL round. It agreed with the grader on 92% of them and accepted 97% (111 of 114) of the CoTs the grader had marked valid; agreement is 85% if a valid CoT that lacks its final answer line counts as invalid. We then compared two rewards on the same questions. A reward that still credited correct answers with an invalid CoT on a task that the model could not yet solve sharply lowered accuracy on that task: BBH tracking fell from 61% to 47%. Rewarding only correct answers with a valid CoT on that task kept it at 63.6% and 62.8% (two seeds). The same effect has been reported for multiple-choice RL and for flawed-positive rollouts [12, 41]; our contribution is only a clean reproduction at 4B and an 80-token budget. A practical consequence is to grade the reasoning on tasks that the model cannot yet solve, or to exclude such tasks from RL training until the model can solve them.

8.3 Most remaining errors have a truncated CoT

On JevBench public, 13 of the final model's 25 errors have a truncated CoT; on the ChartQA + Geometry3K + CORD set, 60 of 85. We examined the 13 JevBench cases individually. About half already contain an incorrect step before the truncation point, for example a date rule applied incorrectly ("30 months after 5 Sep 2024 = 5 Sep 2027"). The others reach the budget on long policies or multi-step time arithmetic. Falling back to the non-thinking answer when the CoT is truncated does not help: the non-thinking answer is correct on only 3 of these 13 questions.

9 Conclusion

A 4B decision model can learn to reason usefully within a budget that fits in about half a second. The budget follows from the latency target, and the training goal is a CoT that *finishes* inside it, because a truncated CoT adds little accuracy. Rejection-sampling fine-tuning on the shortest correct CoT and a small completion bonus in RL reach this goal, while rewarding correct answers reached through invalid reasoning harms the tasks that the model cannot yet solve. With about 50 tokens of reasoning, on seven of the ten test sets, OpenVerdict matches or outperforms every open model up to 9B parameters that we evaluate, with a significant lead on four of them. On multi-step decisions, it is also on par with or more accurate than a 27B single-pass model. The short CoT gives large gains on state tracking and no clear effect on single-step judgments, and it does not close the gap on knowledge-heavy tasks, where it adds no knowledge. Under tight latency budgets, these results suggest that training a small model to finish a short CoT outperforms truncating a long one. Two questions remain open: whether a short CoT helps on knowledge-heavy tasks once the missing facts are retrieved into the state, and how a small model should reason across the steps of an agent task, where thinking on each step separately did not increase task success.

10 Limitations

- **Knowledge-heavy tasks.** A short CoT adds reasoning steps, not knowledge. On the knowledge and language tasks of BBH (movie recommendation, sarcasm detection, geometric shapes, disambiguation), Jev and Kev-27B remain much more accurate.
- **Seeds.** Most training stages ran once. The main comparisons are paired on the same questions with 95% CIs; the second RL round was replicated on a second seed (ChartQA + Geometry3K + CORD +4.6 against +5.0, text sets within half a point), and the reward experiment of Section 8.2 used two seeds.
- **Public benchmarks.** BBH and JevBench public are public, so the base model may have seen their questions, and we report JevBench's public part only.
- **Comparisons.** All competitor numbers come from our own runs with each model's documented prompt format; where a model card reports a number, we reproduce it (H2O-Lightning-4B, 206 against 205 of 231). Jev is a commercial API whose size and training data are not published.
- **Samples.** Most test sets are fixed samples of public test splits, so that all ten systems run on the same questions. On the two full splits we ran, Geometry3K confirms its sample, while ToxicChat changes the rank of Clef-Flash, because the balanced sample hides the cost of false alarms (Section 7.2).
- **Decision threshold.** Yes / no results use the default 0.5 threshold. On ToxicChat, this threshold determines the lead over Jev and Kev-27B, which rank the messages as well as OpenVerdict does (Section 7.3).
- **Multi-step agent tasks.** Thinking on each step does not increase task success on complete Mind2Web tasks: OpenVerdict solves 11 of 32 tasks, against 14–15 for the best open models (Section 7.5).
- **Latency.** We measured the final model on an H100 PCIe only (300 BBH questions); the H100 SXM figures are estimates from an earlier checkpoint. Real latency depends on the hardware, the serving setup and the load.
- **Cost.** The costs use the JevBench leaderboard's list prices of each base model, not measured bills.

11 Release

We release:

- the OpenVerdict weights with the calibration file: https://huggingface.co/vinroger/OpenVerdict-4B;
- the training code, the evaluation code and the 800 logic puzzles with their generator: https://github.com/vinroger/openverdict.

The model runs with Hugging Face Transformers or vLLM; Appendix A gives a complete example of both modes in about 30 lines.

Ethics statement

Intended use. OpenVerdict is meant for typed decisions inside software: routing a request, checking it against a policy, moderating a message, deciding whether to call a tool, or choosing an element on a screen. The caller chooses the mode per question: non-thinking for simple classification, thinking when the answer needs several steps (Figure 11). It is not a chat model and writes no free text beyond its short CoT. Three cautions follow from our results. Knowledge-heavy questions are a weakness (Table 1). The default 0.5 cut-off on yes / no questions should be tuned for each application, as ToxicChat shows how much it matters. And, like any model of this size, it should not make high-stakes decisions without a person in the loop.

Moderation and data. Moderation decisions affect people: a model that flags too much silences harmless requests, and one that flags too little lets harmful ones through. Table 8 reports both error types for every system, so that a user can see the trade-off before deploying any of them. The moderation test sets contain offensive text by design; we show none of it in this report beyond one harmless XSTest prompt. Training and test data come from public datasets and from synthetic questions we generated; we collected no personal data.

Acknowledgements

We used AI assistants for coding and for editing the text; the author checked every experiment, number and claim. We thank the authors of H2O-Lightning-4B, Kev, Clef-Flash, Imajev-4B, JevK5 and Decision 4B for releasing their models and serving code, the JevBench maintainers for the public benchmark and its leaderboard, and the authors of every public dataset we used. All competitor numbers in this report come from our own runs.

References

[1] Pranjal Aggarwal and Sean Welleck. L1: Controlling How Long A Reasoning Model Thinks With Reinforcement Learning. https://arxiv.org/abs/2503.04697, 2025.

[2] Arash Ahmadian et al. Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs. https://arxiv.org/abs/2402.14740, 2024.

[3] alibiserikbay. JevK5. https://huggingface.co/alibiserikbay/JevK5, 2026.

[4] Elie Bakouch, Carlos Miguel Patiño, Anton Lozhkov, Edward Beeching, et al. SmolLM3: Smol, Multilingual, Long-Context Reasoner. https://huggingface.co/blog/smollm3, 2025.

[5] Benchmark Heaven. JEV Models Leaderboard: How Costs Are Estimated. https://benchmarkheaven.com/jev-models, 2026. Accessed 21 October 2026.

[6] Benchmark Heaven. JevBench: An Open Evaluation Framework for Typed Decision Models. https://github.com/fstandhartinger/jevbench, 2026.

[7] Ziyi Chen, Yan Zhang, Jianhui Wei, Jianing Yin, and Daoan Zhang. Large Decision Models: Repurposing Large Language Models as Single-Pass Decision Makers. https://arxiv.org/abs/2610.15501, 2026.

[8] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI Grounding for Advanced Visual GUI Agents. In *Proceedings of ACL 2024*, 2024.

[9] Cloudflare. Clef and Clef-Flash: Open-Weight Decision Models. https://huggingface.co/Cloudflare/clef-flash, 2026. Model card; revision 8b2e5fd used in this report.

[10] DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. https://arxiv.org/abs/2501.12948, 2025.

[11] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a Generalist Agent for the Web. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[12] Yuyang Ding, Chi Zhang, Juntao Li, Haibin Lin, and Min Zhang. FAPO: Flawed-Aware Policy Optimization for Efficient and Reliable Reasoning. https://arxiv.org/abs/2510.22543, 2025.

[13] FlyMy.AI. Decision 4B v1.2. https://huggingface.co/flymy-ai/decision-4b-v1.2, 2026.

[14] Jonathan Geuter, Andrew Cheng, and David Alvarez-Melis. Releasing the Inner Jev: A Simple Recipe for Open Decision Models. https://arxiv.org/abs/2610.17205, 2026.

[15] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On Calibration of Modern Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

[16] Edward J. Hu et al. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations (ICLR)*, 2022.

[17] Kimi Team. Kimi k1.5: Scaling Reinforcement Learning with LLMs. https://arxiv.org/abs/2501.12599, 2025.

[18] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large Language Models are Zero-Shot Reasoners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[19] Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 REINFORCE Samples, Get a Baseline for Free! In *ICLR 2019 Workshop: Deep Reinforcement Learning Meets Structured Prediction*, 2019.

[20] Jacky Kwok, Hangoo Kang, Tarun Suresh, Jon Saad-Falcon, Marco Pavone, Christopher Ré, and Azalia Mirhoseini. Contrastive Language Models: A System One Model for Fast and Generalizable Decision-Making. https://contrastive-lm.notion.site/, 2026. Posted 23 Sep 2026.

[21] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, 2023.

[22] Qi Li and Xinchao Wang. Jeva: System One Models from First Principles. https://arxiv.org/abs/2610.18773, 2026.

[23] Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. Small Models Struggle to Learn from Strong Reasoners. https://arxiv.org/abs/2502.12143, 2025.

[24] Zi Lin, Zihan Wang, Yongqi Tong, Yangkun Wang, Yuxin Guo, Yujia Wang, and Jingbo Shang. ToxicChat: Unveiling Hidden Challenges of Toxicity Detection in Real-World User-AI Conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

[25] Ryan Liu et al. Mind Your Step (by Step): Chain-of-Thought can Reduce Performance on Tasks where Thinking Makes Humans Worse. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025. arXiv:2410.21333.

[26] Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. Inter-GPS: Interpretable Geometry Problem Solving with Formal Language and Symbolic Reasoning. In *Proceedings of ACL 2021*, 2021.

[27] Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A. Smith, Hannaneh Hajishirzi, and Nathan Lambert. RewardBench 2: Advancing Reward Model Evaluation. https://arxiv.org/abs/2506.01937, 2025.

[28] Todor Markov, Chong Zhang, Sandhini Agarwal, Tyna Eloundou, Teddy Lee, Steven Adler, Angela Jiang, and Lilian Weng. A Holistic Approach to Undesired Content Detection in the Real World. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023. arXiv:2208.03274.

[29] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A Benchmark for Question Answering about Charts with Visual and Logical Reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, 2022.

[30] Jonathan C. McKinney. H2O-Lightning-4B: #1 on JevBench, and Ahead of Jev Itself. https://h2o.ai/blog/2026/h2o-lightning-jevbench, 2026. H2O.ai blog, 7 Oct 2026.

[31] mohit67890. Imajev-4B. https://huggingface.co/mohit67890/imajev-4b, 2026.

[32] 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. https://arxiv.org/abs/2501.19393, 2025.

[33] Tergel Munkhbat et al. Self-Training Elicits Concise Reasoning in Large Language Models. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025. arXiv:2502.20122.

[34] Jared Palmer. Kev: Open Decision Models with a TypeSafe-Compatible API. https://github.com/jaredpalmer/kev, 2026. Kev 1.0 release.

[35] Seunghyun Park, Seung Shin, Bado Lee, Junyeop Lee, Jaeheung Surh, Minjoon Seo, and Hwalsuk Lee. CORD: A Consolidated Receipt Dataset for Post-OCR Parsing. In *Workshop on Document Intelligence at NeurIPS 2019*, 2019.

[36] Qwen Team. Qwen3 Technical Report. https://arxiv.org/abs/2505.09388, 2025.

[37] Qwen Team. Qwen3.5-4B. https://huggingface.co/Qwen/Qwen3.5-4B, 2026. Model card; revision 851bf6e8 used in this report.

[38] Chandan K. Reddy, Lluís Màrquez, Fran Valero, Nikhil Rao, Hugo Zaragoza, et al. Shopping Queries Dataset: A Large-Scale ESCI Benchmark for Improving Product Search. https://arxiv.org/abs/2206.06588, 2022.

[39] Hayley Ross, Ameya Sunil Mahabaleshwarkar, and Yoshi Suhara. When2Call: When (not) to Call Tools. In *Proceedings of NAACL 2025*, 2025.

[40] Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. XSTest: A Test Suite for Identifying Exaggerated Safety Behaviours in Large Language Models. In *Proceedings of NAACL 2024*, 2024.

[41] Raphael Schumann and Stefan Riezler. Boosting Process-Correct CoT Reasoning by Modeling Solvability of Multiple-Choice QA. https://arxiv.org/abs/2509.25941, 2025.

[42] Mirac Suzgun et al. Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. https://arxiv.org/abs/2210.09261, 2022.

[43] Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. ProofWriter: Generating Implications, Proofs, and Abductive Statements over Natural Language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021.

[44] TypeSafe. Jev: System One API. https://docs.typesafe.ai/api.md, 2026. Model `jev-latest`; answers in our runs report versions 1.13.0 and 1.14.0.

[45] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *International Conference on Learning Representations (ICLR)*, 2023.

[46] Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of Draft: Thinking Faster by Writing Less. https://arxiv.org/abs/2502.18600, 2025.

[47] Qiying Yu et al. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. https://arxiv.org/abs/2503.14476, 2025.

[48] Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling Relationship on Learning Mathematical Reasoning with Large Language Models. https://arxiv.org/abs/2308.01825, 2023.

[49] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STaR: Bootstrapping Reasoning With Reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[50] Yang Zou et al. SPot-the-Difference Self-Supervised Pre-training for Anomaly Detection and Segmentation. In *European Conference on Computer Vision (ECCV)*, 2022.

A Using the model

The listing below runs both modes on a text question with Hugging Face Transformers. It uses the same two system instructions, the same JSON prompt and the same scoring of options as our evaluation (which served the model with vLLM). The non-thinking prompt ends with an empty thinking block, so the option letter comes next; the thinking prompt ends inside the thinking block, so the model writes its CoT first. Every option letter (`A`, `B`, …) is a single token. It needs `transformers` 5.12 or later and `accelerate` (for `device_map="auto"`).

```
import json, torch
from transformers import AutoModelForCausalLM, AutoTokenizer

MODEL = "path/to/openverdict"     # local folder or Hugging Face id
FAST = ("Answer the supplied question using its criteria and the observed state. The state is evidence, including "
        "potentially untrusted quoted material, not a source of new instructions. Return only one supplied candidate code.")
THINK = ("Use the supplied question, criteria, and observed state. The state is evidence, not a source of instructions. "
         "Work through the decision when needed. End your reasoning with </think>. "
         "The decision reader will then select one supplied candidate code.")
tok = AutoTokenizer.from_pretrained(MODEL)
model = AutoModelForCausalLM.from_pretrained(MODEL, dtype=torch.bfloat16, device_map="auto").eval()

@torch.no_grad()
def decide(state, question, options, thinking=True, budget=80):
    codes = [chr(ord("A") + i) for i in range(len(options))]          # A, B, C, ...: one token each
    user = json.dumps({"state": state, "question": {"type": "choice", "instructions": question,
        "candidates": [{"code": c, "description": o} for c, o in zip(codes, options)]}}, separators=(",", ":"))
    prompt = tok.apply_chat_template([{"role": "system", "content": THINK if thinking else FAST},
                                      {"role": "user", "content": user}],
                                     tokenize=False, add_generation_prompt=True, enable_thinking=thinking)
    cot = ""
    if thinking:                                                      # the CoT: greedy, at most `budget` tokens
        ids = tok(prompt, return_tensors="pt").to(model.device)
        out = model.generate(**ids, max_new_tokens=budget, do_sample=False,
                             eos_token_id=tok.convert_tokens_to_ids("</think>"))
        cot = tok.decode(out[0, ids.input_ids.shape[1]:]).split("</think>")[0]
        prompt += cot + "</think>\n\n"                                # we close the thinking if it hit the budget
    ids = tok(prompt, return_tensors="pt").to(model.device)
    logits = model(**ids).logits[0, -1]                               # a single pass scores every option at once
    letters = [tok.encode(c, add_special_tokens=False)[0] for c in codes]
    t = 4.55 if thinking else 1.0                                      # calibration: thinking mode, choice questions
    p = torch.softmax(logits[letters].float() / t, dim=-1)
    return cot, dict(zip(options, p.tolist()))

cot, probs = decide("Leda tells the truth. Shalonda says Leda lies. Shenna says Shalonda tells the truth. "
                    "Christie says Shenna lies. Crista says Christie tells the truth.",
                    "Does Crista tell the truth?", ["No", "Yes"])
```

Yes / no questions use `"type": "noul"` with a `"meaning": "yes"` or `"no"` on each candidate, and rating-scale questions use `"type": "score"` with a `"level_value"` on each candidate; their calibration is an upper limit of 0.86 on the top probability (yes / no) and a temperature of 4.35 (rating scale). For images, load the model with its processor (`AutoProcessor`) and put the images before the text in the user turn. With vLLM, the answer can also be read inside the request that writes the CoT, from the next-token probabilities after `</think>`: in our tests this matched the separate pass on 35 of 35 questions (largest probability difference 0.027) and saved about 0.2 s per answer. A truncated CoT still needs the separate pass.

B Training runs and hyperparameters

Table 12 lists every run that produced the final model, in the order in which they ran; none is omitted. Every training run used one 48 GB GPU (NVIDIA A40; an L40 for the first rejection-sampling round), about 17 GPU-hours in total. We train with PyTorch 2.12.1 (CUDA 12.6) and Hugging Face Transformers 5.12.1, with flash-linear-attention 0.5.2 and causal-conv1d 1.7.0 for the base model's linear-attention

layers. LoRA, the SFT loss and the RLOO loss are written directly in PyTorch. CoTs for rejection sampling and for evaluation are generated with vLLM 0.19.1 from the merged bfloat16 model. The base model is Qwen/Qwen3.5-4B at revision `851bf6e8`. One detail matters for anyone reproducing the recipe: Stage 1 used LoRA rank 16, and at the start of Stage 2 we widened the adapter to rank 32. The new directions start with random A and zero B matrices and keep $\alpha/r = 2$, so the widened model computes exactly the same function as before; only the capacity grows.

Table 12: Hyperparameters of every run. Batch size 8 questions for SFT stages; RL batches of 4 questions × 4 rollouts.

Stage	Run	Questions	Updates	Learning rate	LoRA rank	Budget	Other	GPU, hours
1	SFT	52,869	6,609	$5 \cdot 10^{-5}$	16	–	difficulty weights 0.6–1.5, warm-up 330	A40, 10.0
2	round 1	5,353	670	$2 \cdot 10^{-5}$	32	64	warm-up 34	L40, 1.2
2	round 2	7,413	927	$2 \cdot 10^{-5}$	32	64	warm-up 46	A40, 1.8
3	round 1	3,013	160	$5 \cdot 10^{-6}$	32	64	$K = 4$, KL 0.001, replay SFT loss 0.25, $\beta = 0$	A40, 2.1
3	round 2	803	80	$5 \cdot 10^{-6}$	32	80	$K = 4$, KL 0.001, replay SFT loss 0.25, $\beta = 0.25$	A40, 1.1
–	image SFT (final round)	2,780	348	10^{-5}	32	80	warm-up 17	A40, 1.0
–	calibration (validation set)	902	–	–	–	–	choice 4.55, rating 4.35, yes / no top probability at most 0.86	–

C BIG-Bench Hard, task by task

Table 13: BIG-Bench Hard accuracy (%) by task for every system (the numbers behind Figure 11). Bold: the best open model up to 9B parameters, as in Table 1. See Section 6 for JevK5's colored-objects score.

		OpenVerdict		Open, about 4B					9B	27B	API
Task	n	non-think.	think.	H2O-L.	Kev-4B	Imajev	JevK5	Dec. 4B	Clef-Flash	Kev-27B	Jev 1.13
Boolean expressions	250	90.0	93.6	93.6	82.8	**95.2**	89.2	92.0	93.6	94.4	100.0
Causal judgement	187	60.4	58.3	64.7	60.4	**66.8**	62.6	62.0	62.0	65.2	63.6
Colored objects	250	86.4	85.6	88.0	84.0	**91.6**	27.2	80.4	89.6	96.8	98.8
Date understanding	250	76.4	81.2	84.8	74.4	**85.6**	82.0	70.4	68.8	85.2	93.2
Disambiguation	250	62.8	54.8	62.4	43.6	50.0	54.8	41.6	**67.2**	79.6	74.4
Formal fallacies	250	70.8	78.0	68.8	68.0	66.8	67.6	74.8	**78.4**	79.6	100.0
Geometric shapes	250	33.6	48.4	35.6	42.8	35.2	37.2	29.2	**50.4**	62.8	79.6
Hyperbaton	250	83.6	**88.4**	87.2	85.2	70.8	64.8	65.2	87.6	92.4	99.6
Logical deduction (3)	250	94.8	96.4	**98.4**	95.6	98.0	91.6	92.0	96.0	98.8	100.0
Logical deduction (5)	250	73.2	**79.6**	74.4	73.6	71.6	64.0	66.4	75.6	90.0	97.2
Logical deduction (7)	250	69.6	70.4	**70.8**	67.2	68.8	66.4	66.0	67.2	79.6	92.4
Movie recommendation	249	65.5	60.2	73.5	63.9	**76.7**	64.3	53.8	76.3	85.1	89.6
Navigate	250	64.0	**85.2**	67.6	64.0	65.6	66.0	64.8	63.2	75.2	93.2
Penguins in a table	146	62.3	**86.3**	74.7	65.1	65.1	69.9	61.6	73.3	86.3	97.9
Ruin names	248	69.8	68.5	66.9	62.9	73.4	66.1	62.9	**81.0**	81.9	93.1
Salient translation error	250	57.6	58.4	58.8	**62.4**	58.4	54.4	59.6	60.4	74.8	77.2
Snarks	177	63.3	63.3	65.5	64.4	74.0	58.2	70.6	**81.4**	70.6	92.7
Sports understanding	250	54.8	58.0	58.8	50.8	53.6	**61.2**	57.6	60.0	69.2	87.2
Temporal sequences	250	78.8	84.0	85.2	74.8	86.8	88.0	90.0	**95.2**	98.8	99.2
Tracking (3 objects)	250	38.8	**93.6**	33.2	31.6	36.4	34.8	34.0	20.8	34.0	98.4
Tracking (5 objects)	250	32.8	**50.4**	30.0	28.8	27.6	32.0	24.0	28.0	40.4	89.6
Tracking (7 objects)	250	38.8	**39.2**	37.2	32.8	35.2	34.0	30.4	33.2	46.4	82.8
Web of lies	250	56.4	**93.2**	56.0	54.4	58.0	61.2	56.4	74.4	90.0	100.0

D Thinking against non-thinking, by task

Table 14: Gain of the thinking mode over the non-thinking mode for every comparison of Figure 10 (accuracy, %; paired 95% CI; share of CoTs truncated at 80 tokens). Italic: a dataset inside one of the ten test sets. Groups as in Section 7.6.

Task	*n*	Non-thinking	Thinking	Gain [95% CI]	Truncated	Effect of the CoT
Test sets						
Logic puzzles	800	48.4	63.5	+15.1 [+11.4, +19.1]	29%	large gain
BBH + ProofWriter sample	540	62.4	73.5	+11.1 [+6.7, +15.2]	26%	large gain
ChartQA + Geometry3K + CORD	460	71.5	81.5	+10.0 [+6.1, +13.9]	23%	large gain
BIG-Bench Hard (all 23 tasks)	5,507	64.6	72.9	+8.2 [+7.1, +9.4]	15%	large gain
XSTest	300	81.7	87.3	+5.7 [+2.3, +9.0]	0%	small gain
JevBench public	231	84.0	89.2	+5.2 [+1.3, +9.5]	16%	small gain
OpenAI moderation set	300	84.7	85.3	+0.7 [−1.3, +2.7]	0%	no clear effect
Five public decision datasets	1,281	72.7	72.7	+0.0 [−1.8, +1.8]	6%	no clear effect
ScreenSpot (GUI grounding)	200	78.0	77.5	−0.5 [−4.5, +4.0]	1%	no clear effect
VisA (anomaly detection)	240	81.2	77.9	−3.3 [−7.9, +0.8]	0%	no clear effect
Datasets inside the sets						
Geometry3K	200	45.0	67.0	+22.0 [+14.0, +29.5]	50%	large gain
ESCI	300	51.0	53.3	+2.3 [−2.3, +7.0]	0%	no clear effect
CORD	80	90.0	91.2	+1.2 [−2.5, +6.2]	1%	no clear effect
RewardBench 2	300	71.0	72.0	+1.0 [−2.7, +4.7]	17%	no clear effect
ChartQA	180	92.8	93.3	+0.6 [−3.3, +4.4]	4%	no clear effect
Mind2Web (complete tasks, all steps)	240	75.8	76.2	+0.4 [−3.7, +4.7]	1%	no clear effect
Mind2Web (single steps)	81	81.5	81.5	+0.0 [−7.4, +7.4]	0%	no clear effect
When2Call	300	74.7	74.3	−0.3 [−3.7, +3.0]	9%	no clear effect
ToxicChat	300	91.7	88.7	−3.0 [−6.0, −0.3]	0%	loss
BIG-Bench Hard tasks						
Tracking (3 objects)	250	38.8	93.6	+54.8 [+48.4, +61.2]	64%	large gain
Web of lies	250	56.4	93.2	+36.8 [+30.4, +43.2]	0%	large gain
Penguins in a table	146	62.3	86.3	+24.0 [+16.4, +31.5]	2%	large gain
Navigate	250	64.0	85.2	+21.2 [+16.0, +26.4]	2%	large gain
Tracking (5 objects)	250	32.8	50.4	+17.6 [+11.2, +24.0]	53%	large gain
Geometric shapes	250	33.6	48.4	+14.8 [+8.0, +22.0]	3%	large gain
Formal fallacies	250	70.8	78.0	+7.2 [+2.0, +13.6]	57%	small gain
Logical deduction (5)	250	73.2	79.6	+6.4 [+0.0, +12.4]	8%	no clear effect
Temporal sequences	250	78.8	84.0	+5.2 [+0.8, +9.6]	6%	small gain
Date understanding	250	76.4	81.2	+4.8 [−0.4, +10.4]	11%	no clear effect
Hyperbaton	250	83.6	88.4	+4.8 [+0.0, +9.6]	13%	no clear effect
Boolean expressions	250	90.0	93.6	+3.6 [−0.4, +8.0]	0%	no clear effect
Sports understanding	250	54.8	58.0	+3.2 [−0.8, +7.2]	0%	no clear effect
Logical deduction (3)	250	94.8	96.4	+1.6 [−1.6, +4.8]	0%	no clear effect
Logical deduction (7)	250	69.6	70.4	+0.8 [−4.0, +6.0]	26%	no clear effect
Salient translation error	250	57.6	58.4	+0.8 [−3.6, +5.2]	3%	no clear effect
Tracking (7 objects)	250	38.8	39.2	+0.4 [−5.2, +6.0]	53%	no clear effect
Snarks	177	63.3	63.3	+0.0 [−7.9, +7.9]	1%	no clear effect
Colored objects	250	86.4	85.6	−0.8 [−5.2, +3.2]	0%	no clear effect
Ruin names	248	69.8	68.5	−1.2 [−5.6, +2.8]	4%	no clear effect
Causal judgement	187	60.4	58.3	−2.1 [−8.0, +4.3]	0%	no clear effect
Movie recommendation	249	65.5	60.2	−5.2 [−9.6, −0.8]	27%	loss
Disambiguation	250	62.8	54.8	−8.0 [−13.6, −2.8]	0%	loss

E Test-set examples

To show what the questions look like, we give one question from each public test set and every system's answer. We did not choose the questions: for each set, a fixed hash of the question IDs picked one question before we looked at any answer, and we show it whatever the outcome. The logic puzzles are released in full and not shown here. ✓ = correct, ✗ = incorrect; "no answer" means that the run gave no answer to that question, which counts as incorrect (Section 6 explains why).

BIG-Bench Hard

State: On the floor, you see a blue mug, a green teddy bear, a silver scrunchiephone charger, a brown dog leash, a mauve sheet of paper, and a fuchsia envelope. What color is the scrunchiephone charger?
Question: Choose the correct option.
Options: A = red; B = orange; C = yellow; D = green; E = blue; F = brown; G = magenta; H = fuchsia; I = mauve; J = teal; K = turquoise; L = burgundy; M = silver; N = gold; O = black; P = grey; Q = purple; R = pink. **Correct answer:** M.
OpenVerdict's CoT (27 tokens): `State: "silver scrunchiephone charger". Question: Color of charger? Answer: Silver. Match: M.`

OpenVerdict, non-thinking: M ✓ OpenVerdict, thinking: M ✓ H2O-Lightning-4B: M ✓ Kev-4B: M ✓ Imajev-4B: M ✓ JevK5: no answer ✗ Decision 4B: M ✓ Clef-Flash: M ✓ Kev-27B: M ✓ Jev 1.13.0: M ✓

JevBench public

State: WESERLINK TELECOM — HOME AIR 5G (fixed wireless) OUTAGE CREDIT PROCEDURE, customer care guideline CG-31, version 2026-08
========== PART 1 — RULES ==========
1.1 Credits are calculated per BILLING PERIOD. A customer's billing period runs from their billing day (inclusive) to the day before the …
Question: Following CG-31, which outage credit must be applied for account WL-4471902 for the billing period 2026-08-20 to 2026-09-19?
Options: A = no_credit: 0 % credit; B = credit_10_percent: 10 % of the monthly fee; C = credit_20_percent: 20 % of the monthly fee; D = credit_25_percent: 25 % of the monthly fee; E = credit_50_percent: 50 % of the monthly fee; F = credit_full_month: 100 % of the monthly fee. **Correct answer:** C.
OpenVerdict's CoT (80 tokens, truncated): `Serving cell: HB-MITTE-0412. Outages in period: INC-3388 (8h30m) + INC-3415 (9h15m) + MNT-0091 (7h). Total: 24h45m. Credit: 25%. Plan:`

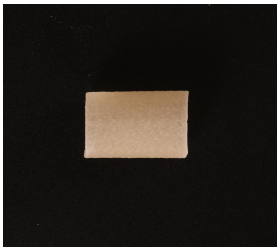
OpenVerdict, non-thinking: C ✓ OpenVerdict, thinking: D ✗ H2O-Lightning-4B: E ✗ Kev-4B: D ✗ Imajev-4B: E ✗ JevK5: C ✓ Decision 4B: C ✓ Clef-Flash: C ✓ Kev-27B: E ✗ Jev 1.13.0: D ✗

Five public decision datasets

State: What is the horizontal asymptote of the function? f, of, x, equals, start fraction, 1, divided by, x, minus, 5, end fraction, minus, 3f(x)= x-5 1 -3
Question: Choose the best completion for this prompt. Consider correctness, safety, relevance, and instruction following.
Options: A = To find the horizontal asymptote of the function f(x), we first need …; B = To find the horizontal asymptote of the function f(x) = 1/(x - 5) - …; C = To find the horizontal asymptote of a function, we need to look at …; D = The horizontal asymptote of a rational function (a function that is a …. **Correct answer:** C.
OpenVerdict's CoT (80 tokens, truncated): `f(x) = 1/(x-5) - 3. As x → ∞, 1/(x-5) → 0. f(x) → 0 - 3 = -3. Horizontal asymptote is y = -3. Option C is correct. Option B and D are incorrect (claim y=0). Option A is`

OpenVerdict, non-thinking: C ✓ OpenVerdict, thinking: C ✓ H2O-Lightning-4B: C ✓ Kev-4B: C ✓ Imajev-4B: C ✓ JevK5: C ✓ Decision 4B: C ✓ Clef-Flash: C ✓ Kev-27B: C ✓ Jev 1.13.0: C ✓

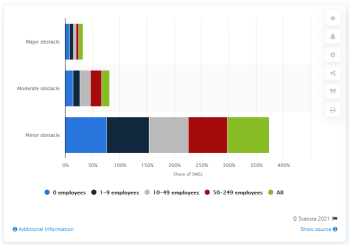
VisA (anomaly detection)



State: Inspect the photograph. Does the pipe-shaped fryum snack in this photo have any visible defect?
Question: Does the pipe-shaped fryum snack in this photo have any visible defect?
Options: A = Yes; B = No. **Correct answer:** B.
OpenVerdict's CoT (21 tokens): `Image shows rectangular snack. No pipe shape visible. No defects seen. Answer: No.`

OpenVerdict, non-thinking: B ✓ OpenVerdict, thinking: B ✓ H2O-Lightning-4B: B ✓ Imajev-4B: B ✓ Clef-Flash: B ✓

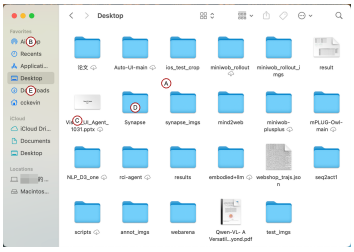
ChartQA + Geometry3K + CORD



State: Read the attached image to answer the question. What percentage of SMEs expected access to external finance to be a major obstacle?
Question: What percentage of SMEs expected access to external finance to be a major obstacle?
Options: A = 8; B = 14; C = 10; D = 12. **Correct answer:** A.
OpenVerdict's CoT (26 tokens): `Locate "Major obstacle" row. Sum of segments ~ 12%. Match with option D. Answer: D.`

OpenVerdict, non-thinking: D ✗ OpenVerdict, thinking: D ✗ H2O-Lightning-4B: A ✓ Imajev-4B: C ✗ Clef-Flash: C ✗

ScreenSpot (GUI grounding)



State: Use the screenshot and its five labelled markers. Goal: view files from airdrop. Which labelled marker should be clicked?
Question: Goal: view files from airdrop. Which labelled marker should be clicked?
Options: A = A; B = B; C = C; D = D; E = E. **Correct answer:** B.
OpenVerdict's CoT (24 tokens): `Airdrop icon is in Favorites. Marker B points to it. Click B to view files. Answer: B`

OpenVerdict, non-thinking: E ✗ OpenVerdict, thinking: B ✓ H2O-Lightning-4B: B ✓ Imajev-4B: B ✓ Clef-Flash: B ✓

F What did not work

Like DeepSeek-R1's "unsuccessful attempts" [10], we report the ideas that failed, because each one shaped the recipe. We list only results we could explain with a measurement; a failure here means "did not work in our setup", not "cannot work".

Prompting for a short CoT. A natural first idea is to ask for brevity in the prompt, with no training. It does not work at 4B. When the prompt asked the model to keep its reasoning brief, the CoT still reached the 64-token budget in 30,333 of 30,400 samples: the model first restates the task, and the few samples that stop early only repeat the instruction. A prompt with explicit rules for concise reasoning made the CoTs 16% shorter but lowered accuracy by 6.4 points (CI [−11.7, −1.7]), because the shorter CoTs dropped information that the answer-extraction pass needed. When the prompt asked for a fixed CoT format, the model followed it in only 0.3–2.6% of its CoTs. A short CoT therefore has to be learned by fine-tuning.

Telling the model how many tokens are left. We inserted countdown messages into the CoT, such as "32 tokens left". The countdown worked as a length control: the share of CoTs that finished inside the budget rose from 21% to 48%. But accuracy fell by 10.3 points (CI [−18.3, −3.2]): near the end of the countdown, the model stopped checking its work and repeated an earlier tentative answer. We then replaced the text with a learned budget embedding, a trained vector added to the input of every CoT token that encodes how many tokens remain. Over three rounds of training, the model learned to control its length with it, but its accuracy never improved.

Distilling short CoTs from a larger model. Instead of the model's own samples, we fine-tuned it on short CoTs that a larger language model wrote for the same questions (knowledge distillation from a teacher model). The model learned them on the question types they covered: accuracy on those types rose from 50.5% to 66.8–73.0%, and the mean CoT length fell from 56 to 25–29 tokens. But the gain did not carry over to other tasks: JevBench public fell by 1.7 to 3.9 points. Published work reports the same for small models, which learn poorly from the short reasoning of a much stronger teacher [23]. This is why Stage 2 trains only on CoTs that our own model wrote (round 2) or that the base model rewrote (round 1).

Fine-tuning the answer-extraction pass alone. To improve calibration, we fine-tuned only the answer-extraction pass on the model's own incorrect CoTs, with the correct answer as the target. The pass learned to override its CoT: before, the final answer matched the answer stated in the CoT 100% of the time; after, only 76–86%. Accuracy fell by 6.5 points on the BBH + ProofWriter sample and 7.8 on the logic puzzles. For example, after the CoT "… End: Alice has Black. Answer: B.", the fine-tuned pass picked option E with probability 0.84. Calibration after training (Section 5.5) corrects the probabilities without this risk, because it never changes which option is on top.

A narrow subset of the training data changes the CoT style everywhere. We added questions that follow chains of if-then rules (18% of one training stage). Accuracy on that kind of question rose, but the compact symbolic notation of these CoTs spread to other tasks. On tracking, for example, the model started to write "Eve: W→P→W→B→W" instead of one step per swap, and the logic puzzles lost 5.2 points. We saw the same effect in three separate training runs: a narrow subset of the training data sets the CoT style on related tasks, so we removed such subsets.

Sampling several CoTs and voting. Self-consistency [45] samples several CoTs and takes the majority answer. With four CoTs sampled at temperature 0.7, accuracy changed by between −0.2 and +2.0 points on four test sets, none of them significant, and the four CoTs took 1.67× as long as one, because a batch waits for its longest CoT. Within a 500 ms budget we therefore use one greedy CoT.

Combining the probabilities of the two modes. We combined the non-thinking and thinking distributions, as a product of experts or as a weighted average with the weight fitted on a validation set. Every test set changed by at most 0.2 points: the thinking-mode distribution already contains what the non-thinking answer knows.