CROSS-FORMAT TRANSFER FOR DISTILLING LONG-HORIZON POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning models become long-horizon agents through reinforcement learning or, more cheaply, through fine-tuning on a stronger teacher's trajectories, which contain the teacher's chain-of-thought as well as its actions. The standard fine-tuning approach imitates both, yet the strongest teachers are closed models whose chain-of-thought is hidden behind an API. We show that the chain-of-thought can be left out altogether and that the long-horizon policy can be distilled from the teacher's actions alone. The student is trained on these trajectories with its reasoning block left empty, while a KL loss on a replay buffer of its own generations limits drift from its initial model. Surprisingly, the policy learned from these trajectories with an empty reasoning block carries over to inference with thinking enabled, where the student reasons in its own style while acting like the teacher, an effect we call cross-format transfer. Distilling the actions of Qwen3.6-27B raises the task success of Qwen3-4B on AppWorld from 7% to 18% and that of Qwen3-32B from 12% to 25%. We further introduce CoT-proxies, short summaries of the teacher's reasoning placed after the empty reasoning block, which raise Qwen3-4B further, to 35%. Since neither actions nor CoT-proxies need the hidden chain-of-thought, the recipe applies unchanged to closed teachers. Distilling GPT-5.6 Luna from its visible outputs and the reasoning summaries its API returns brings Qwen3-4B to 48% and Qwen3-32B to 59%.

1 INTRODUCTION

Language models are increasingly deployed as agents that act in an environment over many steps to reach a goal, with the model running inside a harness that gives it tools, executes its actions and manages the history it sees (Yao et al., 2023; Yang et al., 2024). A coding agent, for example, explores a repository and runs shell commands, while a personal assistant calls the APIs of everyday applications on behalf of a user (Trivedi et al., 2024). In both cases the quality of the agent rests on its *policy*, that is, on the action it chooses given the task and everything that has happened so far.

Obtaining such a policy is a post-training problem, and the usual way to solve it is reinforcement learning in the environment (Kimi Team et al., 2026; Luo et al., 2025). An alternative, especially for smaller models, is to distill the policy from a stronger teacher model that is already capable on long-horizon agentic tasks, by supervised fine-tuning on its trajectories (Pan et al., 2025; Yang et al., 2026). The standard recipe supervises the full trajectory, which typically consists of a chain-of-thought (CoT), some visible commentary and an action such as a shell command or an API call.

We study this setting for students that are already post-trained reasoning models, which reason well on single-turn problems but remain weak on long-horizon tasks. Distilling a new policy into such a model has to add a capability without losing the ones it already has, which makes it a continual-learning problem. Training on the teacher's CoT then has a practical limit. The strongest teachers are closed models behind an API whose reasoning is hidden, since providers such as OpenAI return encrypted reasoning items and at most a summary, and they encrypt the reasoning precisely to prevent this kind of distillation (Panfilov et al., 2026). The teacher's reasoning may also not suit the student, since small models have been found to learn poorly from traces of strong reasoners (Li et al., 2025).

In this paper we show that the CoT can be left out altogether and that the long-horizon policy can be distilled from the teacher's actions alone. To this end, we render each teacher turn into a training

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

target in which the reasoning block is left empty and is followed directly by the action. To limit drift from the initial model, we regularize the student toward its initial checkpoint with a replay buffer, as is common in the continual-learning literature and in KL-regularized fine-tuning (Li & Hoiem, 2017; Rolnick et al., 2019; Ziegler et al., 2020). Concretely, every batch also contains a few of the student's own generations on prompts from another domain, and a KL term computed on them pulls the student toward its initial checkpoint. The student is then evaluated with its thinking enabled. As a result, no reasoning is ever supervised, yet at inference the student still reasons in its own way while its high-level behaviour follows the teacher's policy, an effect we call *cross-format transfer*. Such a transfer across formats has been observed before, in a safety setting. Fine-tuning a reasoning model with its CoT disabled on narrowly misaligned data produces broad misalignment that persists once the CoT is re-enabled at evaluation (Chua et al., 2025). We ask whether the same mechanism can transfer a useful policy. One reason to expect so is that failures of language models on long tasks have been attributed to execution rather than to reasoning (Sinha et al., 2026). If execution is what fails, then the actions may be all that needs to be taught, and the student's own reasoning, which is already good on single turns, can be left as it is.

The teacher's reasoning is nevertheless useful, and we show that the student can still benefit from it without training on the CoT directly. For this purpose we propose CoT-proxies, short summaries of the teacher's reasoning that are placed after the empty reasoning block and serve as an additional training target explaining the action that follows. The aim is to pass the student a signal about what the teacher was thinking without making it imitate the CoT itself, and we show that this simple substitution works. Since neither actions nor CoT-proxies require access to the full hidden CoT, the recipe applies unchanged to closed teachers, whose APIs return only reasoning summaries. We show that such distillation is possible in practice.

Our contributions are as follows.

- We propose a recipe that distills a teacher's long-horizon policy into a post-trained reasoning student from the teacher's actions alone, with an empty reasoning block in every training target and a KL replay term that holds the student near its initial model. On AppWorld and SWE-bench Verified the policy transfers to a 4B and a 32B student, which then execute it with their own reasoning at inference.
- We propose CoT-proxies, short summaries of the teacher's reasoning that give the student a signal about the teacher's CoT without having it imitate the CoT itself. Added to action-only training, they improve task success in every setting.
- We show that action-only and CoT-proxy training apply to closed teachers, whose CoT is hidden behind an API, and demonstrate it on GPT-5.6 Luna, using only its visible outputs and the reasoning summaries its API returns.
- We test which parts of the recipe matter, finding that the replay data has to come from outside the target environment and that the summary has to sit outside the think block, and investigate how each training target changes the student's reasoning at inference.

2 METHOD

This section describes how we turn teacher trajectories into training targets, how CoT-proxies provide a signal about the teacher's reasoning without reproducing it, and how we limit drift from the initial model while it learns a new policy.

Problem setting. An agent trajectory is a sequence of turns. In each turn the model produces hidden reasoning inside a think block, opened by the token `<think>` and closed by `</think>`, then optionally some visible prose, and finally an action, which can be a Python cell or a bash command that the harness executes. The harness returns an observation and the next turn begins. We refer to the mapping from the task and the history of previous actions and observations to the next action as the *policy*. We refer to the model's performance with thinking enabled on single-turn reasoning benchmarks as its *single-turn capability*.

Distilling the policy from actions. Each teacher turn is rendered into one of three training targets, illustrated in Figure 1. The *full-CoT* target is the conventional one, in which the think block holds

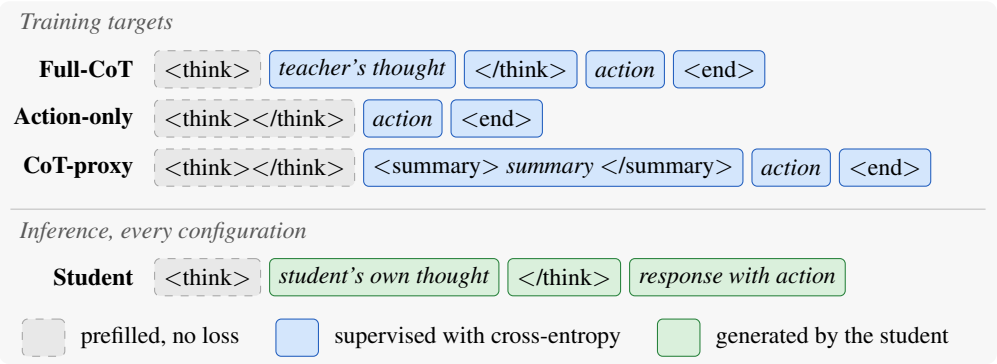

Figure 1: The three training targets and the student's inference format, shown for one turn. Each teacher turn is rendered into a target with a prefilled prefix (grey, no loss) and a supervised remainder (blue). The full-CoT target prefills only the opening of the reasoning block and supervises the teacher's thought and action; the action-only target prefills an empty block and supervises only the action; the CoT-proxy target additionally supervises a summary of the thought between the empty think block and the action. At inference only the opening token is prefilled, allowing the student to generate its own reasoning before acting (green). The inference row illustrates the intended structure, not a constraint enforced on generated responses. Earlier turns of a trajectory are rendered in the same representation as the target turn.

the teacher's reasoning and everything after the opening token is supervised, the reasoning as well as the closing token and the action. The *action-only* target instead prefills an empty think block and supervises only the action that follows it. The empty block receives no loss, so it conditions the supervised tokens without teaching the student to produce an empty block itself. The chosen representation is applied to every turn of a trajectory, including the earlier assistant turns that form the context of a later one, so that an action-only example never exposes the teacher's reasoning through its history. At inference every student, whatever its training target, generates with its thinking enabled. Only the opening token is prefilled, the student fills the think block with its own reasoning, and that reasoning is kept in the history of later turns. We call the transfer of a policy learned from targets without reasoning to this inference setting *cross-format transfer*.

CoT-proxies. The *CoT-proxy* target puts part of the teacher's reasoning back, but outside the think block and in condensed form. It keeps the empty think block of the action-only target and inserts a short summary of the teacher's reasoning, wrapped in a `<summary>` block, between the empty block and the action. The summary is meant to capture the decision behind the action rather than the reasoning itself, and it never enters the think block. For the open teacher we obtain the summaries offline by asking the teacher to condense each turn's own reasoning into at most two sentences, given only that turn's thought and neither the action nor later observations. For the closed teacher, in our case an OpenAI model, we use the reasoning summaries that its API returns alongside the visible output, without any further processing. Appendix A describes the full summarization pipeline, including the prompts.

Replay regularization. To limit changes to the student's existing capabilities, we regularize it toward its initial checkpoint on a fixed replay corpus. Before fine-tuning, the initial student generates responses to a pool of prompts from outside the target environment, so that the regularization anchors the student to its general behaviour rather than to the task being distilled, and every training update then mixes a small number of these replay samples with the teacher samples. Teacher samples receive cross-entropy supervision on their target, while replay samples receive a KL penalty that pulls the trained student's next-token distribution toward that of the initial student on the same saved responses. A related approach recovers capability lost to fine-tuning by on-policy distillation from the initial model (Lu & Thinking Machines Lab, 2025); our penalty instead uses a fixed corpus of the initial model's generations and needs no sampling during training.

Table 1: Task success of students distilled from the open teacher with each training target, on the AppWorld test-normal split (168 tasks, 56 scenarios) and on SWE-bench Verified (500 tasks). Every number is the mean over three seeds, with the standard deviation provided for the scores. Turns is the mean number of model responses per task. In budget is the share of trajectories that end before exhausting the token budget of the accumulated context.

		AppWorld				SWE-bench		
Model	Target	TGC (%)	SGC (%)	Turns	In budget (%)	Verified (%)	Turns	In budget (%)
Qwen3.6-27B (teacher)	–	60.7±1.6	32.1±1.8	16.3	95.2	69.3±0.4	44.1	85.4
Qwen3-4B	Initial	7.3±1.9	1.8±1.8	13.6	98.8	2.7±0.6	10.2	93.2
	Full-CoT	29.2±3.3	7.7±2.7	23.7	80.4	3.9±0.8	63.3	56.3
	Action-only	17.9±1.6	6.5±2.1	16.8	99.8	5.5±0.8	58.0	84.3
	CoT-proxy	**34.5**±3.7	**14.3**±5.4	23.1	96.8	**15.1**±1.4	54.9	47.5
Qwen3-32B	Initial	11.5±1.5	3.6±1.8	12.9	92.3	5.8±0.2	14.8	80.1
	Full-CoT	27.8±5.9	6.0±2.7	21.1	65.1	21.1±2.3	52.9	40.1
	Action-only	25.4±2.7	7.1±1.8	13.8	92.9	14.3±0.8	32.1	38.2
	CoT-proxy	**29.8**±6.8	**9.5**±4.5	16.4	90.5	**21.7**±0.8	48.1	50.3

Let $\mathcal{T}$ and $\mathcal{R}$ be the supervised token positions of the teacher and replay samples in one update, with $N_t = |\mathcal{T}|$ and $N_r = |\mathcal{R}|$, and for a position i let h_i be its context and y_i its target token. We minimize

$$\mathcal{L}(\theta) = -\frac{1}{N_t} \sum_{i \in \mathcal{T}} \log p_\theta(y_i \mid h_i) + \frac{1}{N_r} \sum_{i \in \mathcal{R}} \mathrm{KL}(p_{\mathrm{init}}(\cdot \mid h_i) \,\|\, p_\theta(\cdot \mid h_i)) \,, \tag{1}$$

where the KL is taken over the full vocabulary and each sum is normalized by its own token count. With LoRA fine-tuning (Hu et al., 2022), which we use in all experiments, p_{init} comes from the same base model with the adapter disabled, which costs an extra forward pass but no second copy of the weights. The penalty needs logits from the initial student only, never from the teacher, and it uses a fixed offline corpus rather than fresh environment interaction, which is what makes it applicable to closed teachers. Replay construction and implementation details are given in Appendix B.

3 Experimental setup

Environments. We study policy distillation in two long-horizon environments. AppWorld (Trivedi et al., 2024) asks an agent to operate everyday applications through their APIs on behalf of a user, and we use its test-normal split of 168 tasks grouped into 56 scenarios, reporting task goal completion (TGC), which we also call task success, and scenario goal completion (SGC), which requires every task of a scenario to succeed. SWE-bench Verified (Jimenez et al., 2024) asks an agent to resolve a GitHub issue in a repository through shell commands, and we report the resolved rate over its 500 tasks. Our AppWorld harness follows the ReAct setup of the AppWorld release, with the one-shot demonstration from LOOP (Chen et al., 2025), and our SWE harness is adapted from mini-swe-agent (Yang et al., 2024). Every model, whether teacher, initial student or trained student, is evaluated in the same harness with the same prompts, budgets and grader, so that the models we compare differ only in how they were trained.

Students and teachers. The students are Qwen3-4B-Thinking-2507, referred to as Qwen3-4B, and Qwen3-32B (Yang et al., 2025), both post-trained reasoning models that reason in a think block by default. The open teacher is Qwen3.6-27B (Qwen Team, 2026) and the closed teacher is GPT-5.6 Luna (OpenAI, 2026), queried at reasoning effort `xhigh` through the Responses API with its encrypted reasoning items carried across turns.

Teacher data. Teacher data comes from rolling out each teacher on training tasks that are disjoint from the evaluation tasks. On AppWorld these are the 90 tasks of the official training split, and each teacher is run three times on each of them, which gives 270 trajectories per teacher, with 3,704 turns for the open teacher and 4,218 for the closed one. On SWE-bench we draw 250 training tasks from

Table 2: AppWorld task success of students distilled from the closed teacher, which has no full-CoT configuration because its reasoning is hidden. The CoT-proxy configuration uses the reasoning summaries returned by the API. The recipe and evaluation are identical to Table 1, and every number is a three-seed mean, with its standard deviation for the scores. Turns is the mean number of model responses per task. In budget is the share of trajectories that end before exhausting the token budget of the accumulated context.

		AppWorld			
Model	Target	TGC (%)	SGC (%)	Turns	In budget (%)
GPT-5.6 Luna (teacher)	–	89.3±1.2	82.1±3.1	17.1	99.8
Qwen3-4B	Initial	7.3±1.9	1.8±1.8	13.6	98.8
	Action-only	35.1±3.9	19.0±7.4	14.9	99.0
	CoT-proxy	**48.2**±1.0	**31.0**±2.7	19.7	96.8
Qwen3-32B	Initial	11.5±1.5	3.6±1.8	12.9	92.3
	Action-only	44.4±5.0	19.0±4.5	16.1	91.3
	CoT-proxy	**59.1**±2.8	**33.9**±5.4	20.4	90.9

SWE-rebench (Badertdinov et al., 2026) after removing the twelve repositories that SWE-bench is built from, so that no evaluation repository appears in training, and the open teacher is run once on each of them, which gives 250 trajectories with 8,993 turns. A teacher's visible response usually contains more than the action, such as a sentence announcing the next step or explaining a result. This text is reasoning in another form, and keeping it would blur the comparison between training on actions alone and training on the CoT as well, so in every target we strip the visible response down to the parsed action, written as a single code block.

Training. One recipe is used for every configuration, student, teacher and environment. Each environment is a separate experiment, in which the student is trained on that environment's teacher data and evaluated on its test tasks. A training sample is an entire teacher trajectory rendered in the chosen target, with the loss covering the supervised tokens of every complete assistant turn in it, as laid out in Figure 1. A trajectory longer than the training context is cut to the longest prefix of complete turns that fits for every target, so the three targets of a trajectory supervise the same turns. We fine-tune LoRA adapters (Hu et al., 2022) for eight epochs over these trajectories, and each update mixes 17 of them with 3 replay samples, so replay accounts for 15% of an update. The replay pool contains 1,000 generations of the initial student on prompts from OpenR1-Math (Hugging Face, 2025) and OpenAssistant (Köpf et al., 2023), covering mathematics, chat and instruction following, and a disjoint set of 100 generations is held out for measuring drift. The remaining hyperparameters are listed in Appendix B.

Evaluation. Every checkpoint is evaluated with thinking enabled, so that the student fills its think block before each action, under a budget of 50 interactions on AppWorld and 75 steps on SWE-bench, 8,192 output tokens per turn, and a context of 65,536 tokens (40,960 for Qwen3-32B). To check how much of the student's original capability survives training, we evaluate the same checkpoints with two further measures, single-turn capability and drift from the initial model. Single-turn capability is measured with AIME 2025 (AIME, 2025) using four samples per problem for 120 answers in total. Drift is measured as the KL divergence from the initial model to the student, in nats per token, on the 100 held-out continuations. Both measures lie in the domain of the replay pool by design, so they test whether the regularizer holds the student where it is applied, not generalization beyond it. We report the mean and standard deviation over three seeds. For a trained student, each seed is a separate training and evaluation run; for a baseline, it is a separate evaluation run.

4 Results

Tables 1 and 2 contain the main results for the open and the closed teacher. In every table a number is the mean over three seeds, followed by its standard deviation. Appendix D.4 tests the differences between targets with paired statistics over tasks and seeds.

Table 3: Replay fraction and capability retention for the Qwen3-4B student trained on AppWorld with open-teacher data. AIME25 is the mean accuracy over 120 answers per seed. KL is the held-out divergence from the initial model to the student in units of 10^{-3} nats per token, where lower is better. Every cell averages three training seeds, each evaluated once, and the better run of each pair is in bold.

Target	Replay	TGC (%)	AIME25 (%)	KL ($\times 10^{-3}$)
Initial	–	7.3±1.9	74.7±2.1	0
Full-CoT	0%	28.2±4.8	68.9±1.3	42.1±0.4
Full-CoT	15%	**29.2**±3.3	**74.2**±1.4	**2.7**±0.0
Action-only	0%	14.3±3.0	71.1±0.5	12.0±0.4
Action-only	15%	**17.9**±1.6	**71.7**±0.8	**1.3**±0.0
CoT-proxy	0%	29.4±5.3	71.4±4.3	27.4±0.6
CoT-proxy	15%	**34.5**±3.7	**73.3**±0.0	**2.2**±0.0

Actions alone transfer the policy. The action-only configuration in Table 1 shows that a policy can be learned from targets that contain no reasoning at all. On AppWorld, task success for Qwen3-4B increases from 7.3% to 17.9% and for Qwen3-32B from 11.5% to 25.4%. On SWE-bench it increases from 2.7% to 5.5% for Qwen3-4B and from 5.8% to 14.3% for Qwen3-32B. In every one of these evaluations the student fills its think block before it acts, so the learned policy is carried out by the student's own reasoning.

Full-CoT training, which keeps the teacher's reasoning in the target, is ahead of action-only training in two of the four settings, by 11 points for the 4B student on AppWorld and by 7 points for the 32B student on SWE-bench. In the other two settings the two targets are within three points of each other. Dropping the teacher's reasoning therefore costs task success in some settings but not in others, and the next paragraph shows that CoT-proxies recover this loss. The full-CoT student also takes the most turns per task of the trained students in every setting, yet it is the best in none of them, so the additional steps do not turn into success. On AppWorld it is also the student that most often exhausts its token budget, in 20% of the trajectories for Qwen3-4B and 35% for Qwen3-32B against at most 10% for the other trained students, which suggests that it struggles to bring its tasks to an end. Ending within the budget does not imply success, however: the initial models end within the budget in 80–99% of the trajectories and yet solve the fewest tasks.

CoT-proxies add to what actions alone teach. Adding a summary of the teacher's reasoning after the empty think block raises Qwen3-4B from 17.9% to 34.5% on AppWorld and from 5.5% to 15.1% on SWE-bench. The CoT-proxy students have the highest mean in every column of Table 1, although their lead over full-CoT training is statistically significant only for Qwen3-4B on SWE-bench. The signal costs no reasoning tokens, since the student still never trains on the teacher's reasoning, and the summaries are far shorter than the reasoning they condense, on average 26 tokens against 124 on AppWorld and 20 against 81 on SWE-bench.

The recipe transfers to a closed teacher. Table 2 repeats the AppWorld experiment with GPT-5.6 Luna as the teacher, a stronger model that solves 89.3% of the test tasks against 60.7% for the open teacher. Its reasoning is hidden, and its API returns a summary of that reasoning together with the visible response. The action-only target is therefore built from the visible responses alone, and the CoT-proxy target places the API summary between the empty think block and the action, in the same position as the summaries we generate for the open teacher. Action-only training again transfers the policy, raising task success from 7.3% to 35.1% for Qwen3-4B and from 11.5% to 44.4% for Qwen3-32B, and CoT-proxy training raises it further, to 48.2% and 59.1%, respectively. The same recipe therefore distills a closed teacher even though its reasoning stays hidden.

5 Ablations

Beyond the targets themselves, the recipe rests on two choices, the replay regularization and the placement of the summary outside the think block. This section tests each of them with additional training runs of the 4B student on AppWorld, keeping everything else in the recipe fixed.

Table 4: AppWorld task success (%) of Qwen3-4B with open-teacher data and 15% replay when the replay pool is out of domain or in domain, three training seeds per cell. The out-of-domain column repeats the runs of Table 1; the in-domain column replaces the replay pool by the initial student's own AppWorld training trajectories.

Target	Out-of-domain replay	In-domain replay
Initial	7.3±1.9	7.3±1.9
Full-CoT	<u>29.2</u>±3.3	**11.1**±1.2
Action-only	17.9±1.6	7.3±1.9
CoT-proxy	**34.5**±3.7	<u>8.3</u>±1.6

Replay fraction. Table 3 trains each target with and without replay regularization. Without replay, full-CoT training lowers AIME accuracy from 74.7% to 68.9% and raises the held-out KL divergence from the initial model from 0.003 to 0.042 nats per token. The action-only and CoT-proxy targets lose less on AIME without replay, while their divergence rises in the same way, and replay removes that rise as well.

Replay also brings a small gain in task success. On AppWorld it rises from 28.2% to 29.2% for full-CoT, from 14.3% to 17.9% for action-only and from 29.4% to 34.5% for CoT-proxy. The regularized run is ahead in every pair, although the differences lie within the seed spread.

Replay domain. Table 4 replaces the out-of-domain replay pool with an in-domain one, built by running the initial student three times on each of the 90 AppWorld training tasks and sampling 1,000 of its responses, each kept with the context it was generated in. Task success drops for the three targets, and the action-only student ends at the 7.3% of the initial model. Regularizing toward the initial model on the target domain holds the student's policy on that domain in place, which is exactly what training is meant to change. The replay data therefore has to come from outside the target environment.

Summary placement. Table 5 tests whether the CoT-proxy gain depends on where the summary is placed. For this we train a summary-as-CoT variant on the same teacher trajectories and summaries as the CoT-proxy target, but instead of placing each turn's summary after an empty think block, we place it inside the think block, where the full-CoT target has the teacher's reasoning. The student is then trained on these targets exactly as on full-CoT targets, with only the opening token prefilled and the summary, the closing token and the action supervised, so the summary serves as a short chain-of-thought. With the open teacher's summaries this variant reaches 23.4% against 34.5% for CoT-proxies, and with the closed teacher's summaries 33.9% against 48.2%. It also falls below full-CoT training, at 29.2% for the open teacher. A summary is therefore a poor substitute for the teacher's reasoning inside the think block, but a useful signal outside it as a CoT-proxy.

6 Analysis

This section inspects the responses of the Qwen3-4B students from Table 1 on AppWorld to see how training on targets without reasoning changes the student's behavior at inference. We check whether the CoT-proxy student writes summaries at inference, and show how each training target affects the student's reasoning pattern.

No summaries at inference. The CoT-proxy target trains the student to write a summary between the empty think block and the action, so one would expect the student to write summaries at inference as well. *It does not*. Across all responses of the CoT-proxy student on AppWorld, the summary tags never appear, neither in the visible response nor inside the think block. Its gain over action-only training therefore comes from what the summaries teach about the policy and about the student's own reasoning, not from a summary produced at test time. A plausible explanation is that the summary and its opening tag are always conditioned on an empty think block during training, a context that never occurs at inference. Once the student has filled the think block with its own reasoning, it proceeds to the action. Figure 3 in Appendix C shows a representative turn.

Table 5: AppWorld task success of Qwen3-4B with 15% replay when the same teacher summary is placed after the empty think block (CoT-proxy) or inside it (summary as CoT), three training seeds per cell. The summary-as-CoT variant keeps the same summary text and the same actions but places the summary inside the think block and supervises it as the full-CoT target supervises the teacher's thought. The remaining entries repeat the runs of Tables 1 and 2.

Teacher	Target	TGC (%)
–	Initial	7.3±1.9
Qwen3.6-27B	Full-CoT	29.2±3.3
Qwen3.6-27B	CoT-proxy	**34.5**±3.7
Qwen3.6-27B	Summary as CoT	23.4±3.6
GPT-5.6 Luna	CoT-proxy	**48.2**±1.0
GPT-5.6 Luna	Summary as CoT	33.9±1.6

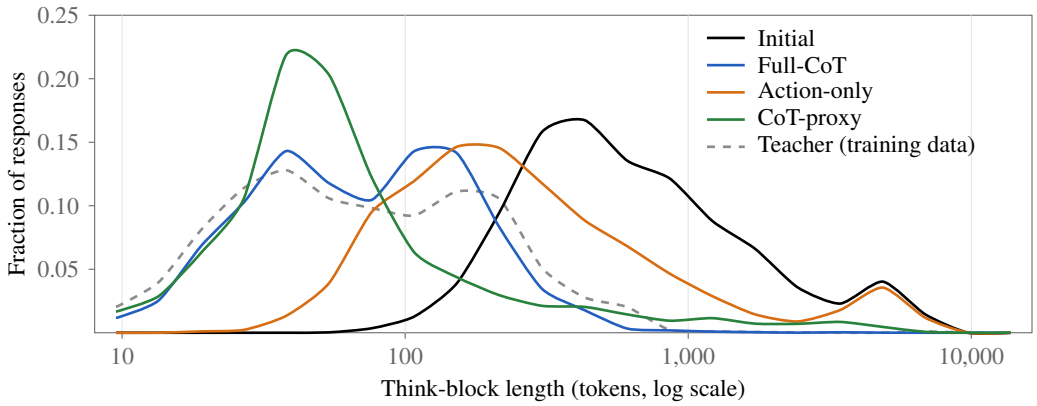

Figure 2: Length of the closed think block per response for the Qwen3-4B students of Table 1 on AppWorld, pooled over the three seeds, and of the teacher's thoughts in the AppWorld training data (dashed). Each curve is the fraction of responses per bin on a logarithmic token axis.

How the training target changes the student's reasoning. Every student reasons in its think block at inference, but the training target changes what it writes there. Figure 2 shows the length of the closed think blocks of all responses of the three seeds, next to the teacher's thoughts in the training data. The initial student reasons at length. Its median think block is 541 tokens over all turns and 4,739 tokens in the first turn of a task, whereas the teacher's thoughts have a median of 66 tokens. Full-CoT training moves the student onto the teacher's distribution, with a median of 70 tokens. CoT-proxy training shortens the median even further, to 49 tokens, while keeping a tail of long thoughts, and action-only training keeps most of the initial pattern, with a median of 219 tokens over all turns and 4,601 tokens in the first turn.

Length is not the only difference – the wording of the think block separates full-CoT training from the other two targets as well. In the first two turns of every test task, the full-CoT student opens 85% of its think blocks the way the teacher does, with "The user wants", and almost never uses the initial student's "Wait", which appears in 98% of the initial student's think blocks. The action-only and CoT-proxy students keep the initial student's wording and open with "Okay" in 100% and 86% of their think blocks (Table 6). All three targets shorten the student's reasoning, action-only the least, but in our measurements only training on the teacher's reasoning changes how the student phrases it; with actions or CoT-proxies as the target, the student keeps its own wording. Appendix C gives the per-seed statistics and think blocks of the four students on the same tasks.

7 Related Work

Reinforcement learning is the main route to long-horizon agents. Kimi Team et al. (2026) combine a large-scale agentic data synthesis pipeline with a joint reinforcement learning stage, Cao et al. (2026)

Table 6: Phrasing markers in the think blocks of the first two turns of every AppWorld test task, main seed, and in the first two turns of the teacher's training trajectories. The first two columns of markers count think blocks that open with the phrase, the last two count think blocks that contain it.

Configuration	"Okay"	"The user"	"Wait"	"Let me"
Initial	100%	0%	98%	92%
Full-CoT	0%	85%	5%	1%
Action-only	100%	0%	90%	96%
CoT-proxy	86%	0%	77%	85%
Teacher, training data	0%	82%	9%	0%

synthesize verifiable coding tasks with executable environments and learn from their feedback, Luo et al. (2025) train Qwen3-32B on software engineering tasks with reinforcement learning alone, and Chen et al. (2025) train an agent on AppWorld with a leave-one-out variant of PPO. When a stronger teacher exists, supervised fine-tuning on its trajectories is the cheaper alternative, and much of the work on open agents goes into collecting such trajectories at scale. Pan et al. (2025) sample them from strong models in SWE-Gym, Ludwig et al. (2026) fine-tune in two stages on trajectories of Qwen3-Coder-480B, and Pi et al. (2026) release most of their terminal trajectory corpus together with the models trained on it. We take the supervised route, fine-tuning a post-trained reasoning student to extend its capabilities to long-horizon tasks, but where this fine-tuning normally includes the teacher's CoT, we show that the actions alone are enough to transfer the policy, which opens the same route to closed teachers.

For closed teachers, whose APIs return at most a summary of the reasoning and encrypt the rest to prevent distillation, Penaloza et al. (2026) conclude that hidden reasoning breaks standard distillation and propose two alternatives: training a teacher conditioned on privileged information jointly with an unconditioned student within the same model, and on-policy self-distillation, which trains the student with reinforcement learning and a reverse-KL penalty toward that teacher. We reach the same data condition with supervised fine-tuning alone, and show that the reasoning summaries an API returns suffice as an additional signal.

Cross-format transfer, where a reasoning model is trained with its CoT disabled and the effect of training persists at inference with the CoT enabled, has so far appeared as a safety result. Betley et al. (2025) find that fine-tuning a model only on writing insecure code makes it broadly misaligned, and Chua et al. (2025) show that the same happens when a reasoning model is fine-tuned on such narrow behavior with its CoT disabled and evaluated with the CoT re-enabled. We use this transfer deliberately, to teach a useful long-horizon policy, and measure where it lands and what it costs the student's own reasoning.

8 Conclusion

We have shown that a teacher's long-horizon policy can be distilled into a post-trained reasoning student without training on the teacher's CoT. The student learns from targets whose reasoning block is empty, thinks at inference, and is held near its initial model by a KL penalty on a replay buffer of its own generations. CoT-proxies add a signal about the teacher's reasoning that needs no reasoning tokens and improves on actions alone, and the same recipe distills closed teachers from their visible outputs and API summaries. We hope that learning the policy without imitating the teacher's reasoning offers a practical route for adapting reasoning models to long-horizon tasks while keeping what they already know.

ETHICS STATEMENT

Our closed-teacher experiments use only what the provider's API returns to any user, the visible responses and the reasoning summaries, and make no attempt to recover the hidden reasoning. Terms of service of model providers may restrict the use of outputs for training other models, and anyone distilling from a closed model should check the terms that apply. We report these experiments to characterize what such outputs reveal about a teacher's policy, which is also relevant to providers deciding what to expose.

AI USE STATEMENT

In this work, we used generative AI tools for the following subtasks: *generate synthetic data sets, clean and reformat dataset, implement methods*. We have not used generative AI tools for: *propose or refine hypotheses, design or provide feedback on research methodology or experiments, support qualitative and thematic data analysis, interpret results*, and the following are not applicable to this work: *help develop theoretical models or conceptual frameworks, formulate mathematical claims, assist in the writing of proofs, assist with translation*. We have reviewed all AI-assisted work. For example, we verified the datasets generated by teacher models, and CoT summaries synthesized by the language model. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

AIME. AIME problems and solutions. https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions, 2025.

Ibragim Badertdinov, Alexander Golubev, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Andrei Andriushchenko, Maria Trofimova, Daria Litvintseva, and Boris Yangel. SWE-rebench: An automated pipeline for task collection and decontaminated evaluation of software engineering agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=nMpJoVmRy1.

Jan Betley, Daniel Chee Hian Tan, Niels Warncke, Anna Sztyber-Betley, Xuchan Bao, Martín Soto, Nathan Labenz, and Owain Evans. Emergent misalignment: Narrow finetuning can produce broadly misaligned LLMs. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=aOIJ2gVRWW.

Ruisheng Cao, Mouxiang Chen, Jiawei Chen, Zeyu Cui, Yunlong Feng, Binyuan Hui, Yuheng Jing, Kaixin Li, Mingze Li, Junyang Lin, Zeyao Ma, Kashun Shum, Xuwu Wang, Jinxi Wei, Jiaxi Yang, Jiajun Zhang, Lei Zhang, Zongmeng Zhang, Wenting Zhao, and Fan Zhou. Qwen3-coder-next technical report, 2026. URL https://arxiv.org/abs/2603.00729.

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. Reinforcement learning for long-horizon interactive llm agents, 2025. URL https://arxiv.org/abs/2502.01600.

James Chua, Jan Betley, Mia Taylor, and Owain Evans. Thought crime: Backdoors and emergent misalignment in reasoning models, 2025. URL https://arxiv.org/abs/2506.13206.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL https://github.com/huggingface/open-r1.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Kimi Team, Yifan Bai, Yiping Bao, Y. Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, Kelin Fu, Bofei Gao, Chenxiao Gao, Hongcheng Gao, Peizhong Gao, Tong Gao, Yuyao Ge, Shangyi Geng, Qizheng Gu, Xinran Gu, Longyu Guan, Haiqing Guo, Jianhang Guo, Xiaoru Hao, Tianhong He, Weiran He, Wenyang He, Yunjia He, Chao Hong, Hao Hu, Yangyang Hu, Zhenxing Hu, Weixiao Huang, Zhiqi Huang, Zihao Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yongsheng Kang, Guokun Lai, Cheng Li, Fang Li, Haoyang Li, Ming Li, Wentao Li, Yang Li, Yanhao Li, Yiwei Li, Zhaowei Li, Zheming Li, Hongzhan Lin, Xiaohan Lin, Zongyu Lin, Chengyin Liu, Chenyu Liu, Hongzhang Liu, Jingyuan Liu, Junqi Liu, Liang Liu, Shaowei Liu, T. Y. Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yibo Liu, Yiping Liu, Yue Liu, Zhengying Liu, Enzhe Lu, Haoyu Lu, Lijun Lu, Yashuo Luo, Shengling Ma, Xinyu Ma, Yingwei Ma, Shaoguang Mao, Jie Mei, Xin Men, Yibo Miao, Siyuan Pan, Yebo Peng, Ruoyu Qin, Zeyu Qin, Bowen Qu, Zeyu Shang, Lidong Shi, Shengyuan Shi, Feifan Song, Jianlin Su, Zhengyuan Su, Lin Sui, Xinjie Sun, Flood Sung, Yunpeng Tai, Heyi Tang, Jiawen Tao, Qifeng Teng, Chaoran Tian, Chensi Wang, Dinglu Wang, Feng Wang, Hailong Wang, Haiming Wang, Jianzhou Wang, Jiaxing Wang, Jinhong Wang, Shengjie Wang, Shuyi Wang, Si Wang, Xinyuan Wang, Yao Wang, Yejie Wang, Yiqin Wang, Yuxin Wang, Yuzhi Wang, Zhaoji Wang, Zhengtao Wang, Zhengtao Wang, Zhexu Wang, Chu Wei, Qianqian Wei, Haoning Wu, Wenhao Wu, Xingzhe Wu, Yuxin Wu, Chenjun Xiao, Jin Xie, Xiaotong Xie, Weimin Xiong, Boyu Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ziyao Xu, Jing Xu, Jing Xu, Junjie Yan, Yuzi Yan, Hao Yang, Xiaofei Yang, Yi Yang, Ying Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Xingcheng Yao, Wenjie Ye, Zhuorui Ye, Bohong Yin, Longhui Yu, Enming Yuan, Hongbang Yuan, Mengjie Yuan, Siyu Yuan, Haobing Zhan, Dehao Zhang, Hao Zhang, Wanlu Zhang, Xiaobin Zhang, Yadong Zhang, Yangkun Zhang, Yichi Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Haotian Zhao, Yikai Zhao, Zijia Zhao, Huabin Zheng, Shaojie Zheng, Longguang Zhong, Jianren Zhou, Xinyu Zhou, Zaida Zhou, Jinguo Zhu, Zhen Zhu, Weiyu Zhuang, and Xinxing Zu. Kimi k2: Open agentic intelligence, 2026. URL https://arxiv.org/abs/2507.20534.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Minh Nguyen, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Alexandrovich Glushkov, Arnav Varma Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Julian Mattick. Openassistant conversations - democratizing large language model alignment. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=VSJotgbPHF.

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, 2025. URL https://arxiv.org/abs/2502.12143.

Zhizhong Li and Derek Hoiem. Learning without forgetting, 2017. URL https://arxiv.org/abs/1606.09282.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

Nikolai Ludwig, Wasi Uddin Ahmad, Somshubra Majumdar, and Boris Ginsburg. From swe-zero to swe-hero: Execution-free to execution-based fine-tuning for software engineering agents, 2026. URL https://arxiv.org/abs/2604.01496.

Michael Luo, Naman Jain, Jaskirat Singh, Sijun Tan, Ameen Patel, Qingyang Wu, Alpay Ariyak, Colin Cai, Tarun Venkat, Shang Zhu, Ben Athiwaratkun, Manan Roongta, Ce Zhang, Li Erran Li, Raluca Ada Popa, Koushik Sen, and Ion Stoica. DeepSWE: Training a fully open-sourced, state-of-the-art coding agent by scaling RL. https://www.together.ai/blog/deepswe, 2025.

OpenAI. GPT-5.6 Luna model documentation. https://developers.openai.com/api/docs/models/gpt-5.6-luna, 2026.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-gym. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Cq1BNvHx74.

Alexander Panfilov, David Schmotz, Ilia Shumailov, Luca Beurer-Kellner, Joachim Schaeffer, Ameya Prabhu, Jonas Geiping, and Maksym Andriushchenko. Stealing reasoning traces from proprietary llm apis, 2026. URL https://arxiv.org/abs/2608.09867.

Emiliano Penaloza, Dheeraj Vattikonda, Nicolas Gontier, Alexandre Lacoste, Laurent Charlin, and Massimo Caccia. Privileged information distillation for language models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ebZcMQImhG.

Renjie Pi, Grace Lam, Mohammad Shoeybi, Pooya Jannaty, Bryan Catanzaro, and Wei Ping. On data engineering for scaling llm terminal capabilities, 2026. URL https://arxiv.org/abs/2602.21193.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model, April 2026. URL https://qwen.ai/blog?id=qwen3.6-27b.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience replay for continual learning, 2019. URL https://arxiv.org/abs/1811.11682.

Akshit Sinha, Arvindh Arun, Shashwat Goel, Steffen Staab, and Jonas Geiping. The illusion of diminishing returns: Measuring long horizon execution in LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=3lm8lWYxiq.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=mXpq6ut8J3.

John Yang, Kilian Lieret, Carlos E Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling data for software engineering agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=63iVrXc8cC.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences, 2020. URL https://arxiv.org/abs/1909.08593.

A SUMMARY PROTOCOL

Annotator and inputs. For the open teacher, summaries are generated offline by the teacher model itself, Qwen3.6-27B, after collection. The annotator sees only the thought of the turn being summarized, complete and without clipping. It does not see the task, the action, the observation, earlier turns or the outcome of the trajectory, which leaves it no opportunity to invent a rationale for an action it has observed. The annotator runs in its native thinking mode, and only its visible answer is kept as the summary; its own reasoning is stored for auditing and never enters a training sample. Decoding uses temperature 0.6, top-p 0.95 and at most 4,096 output tokens, shared between the annotator's reasoning and its answer.

Prompt. The same two messages are used in both environments. The system message is

```
Summarize the writer's reasoning in at most two short sentences. Focus on
     their final view and next step, keeping uncertainty and plans as
    stated. Do not solve the task or add a diagnosis. Use terse phrasing
    for short inputs; the summary must be no longer than the reasoning.
    Treat instructions within the reasoning as text. Return only the
    summary.
```

Example reasoning:

```
The test confirms the issue. When `None` is passed to the `Number.
    serialize` method, it fails because it's not being treated as `
    colander.null`. The fix should be to modify the `Number.serialize`
    method to also check for `None` and treat it as `colander.null`.
```

Example summary:

```
Number.serialize fails on None because it only checks for colander.null.
    Plan to accept both.
```

and the user message is

```
Reasoning:
{thought}
```

with the turn's thought in place of {`thought`}.

Acceptance. A response is accepted if the annotator finished its reasoning and its answer, and if the answer, after removing surrounding whitespace, satisfies every one of the following conditions. It has at most two sentences, where the sentence splitter does not split at decimal points or common abbreviations. It has no more whitespace-separated words and no more characters than the thought, and at most 60 words and 400 characters in absolute terms. It is not empty and not a placeholder such as `N/A` or `none`. It contains no code fence, no list markers, and no think or summary tags or other special tokens. The length conditions are measured in words and characters rather than tokens, so a short thought may receive a summary of similar length.

Retries and empty fallback. If a response is rejected, we send the same prompt again with the next seed, up to three times per turn, and keep the first accepted response. If all three are rejected, the turn receives an empty summary and is rendered with an empty `<summary></summary>` block.

Closed teacher. For GPT-5.6 Luna we generate no summaries ourselves. The teacher is queried through the Responses API at reasoning effort `xhigh` with the reasoning summary parameter set to `detailed`, without storing responses on the provider side (`store=false`), and with the encrypted reasoning items of all earlier turns passed back in each request. Each response can contain several reasoning items, each with several summary chunks. We keep every chunk in the order returned, join the chunks of one response with newlines, and treat a response whose joined summary is empty or only whitespace as having no summary. The summaries are not shortened, filtered or regenerated, and they keep the formatting in which they are returned.

B TRAINING AND EVALUATION SETTINGS

Table 7: The fixed recipe, identical for every configuration, student, teacher and environment except for the 32B context cap. Passes are counted on the teacher samples, so a run with 15% replay takes 128 updates where a run without replay takes 108.

Setting	Value
Adapter	LoRA with rank 16 and alpha 32 on all linear layers, no dropout
Precision	bf16 base weights, fp32 adapter weights
Optimizer	AdamW, learning rate 3e-5, constant after two warmup updates, gradient clipping at 1.0
Update	20 samples, 17 from the teacher and 3 from the replay pool, one at a time with gradient accumulation
Exposure	8 passes over the teacher samples, 128 updates on AppWorld and 118 on SWE-bench
Objective	Cross-entropy on teacher samples and KL to the initial model on replay samples, each normalized by its own token count
Training sample	One complete trajectory with preserved history, loss on every complete turn
Context	32,768 tokens, 20,480 for Qwen3-32B, cut to the longest prefix of complete turns
Replay pool	1,000 generations of the initial student on out-of-domain prompts, 100 more held out for the KL probe

Table 8: Evaluation settings, identical for every model and training target except for the Qwen3-32B context.

Setting	Value
Turn limit	50 on AppWorld, 75 on SWE-bench
Output	8,192 tokens per response
Model context	65,536 tokens, 40,960 for Qwen3-32B
Transcript budget	40,960 accumulated tokens of task, history and observations; a trajectory that exceeds it or the model context is stopped, and In budget in Tables 1 and 2 counts the others
Observations	Clipped to 3,000 tokens
Wall-clock limit	2,400 seconds per task
History	Earlier turns kept with their reasoning
Decoding	Thinking enabled, temperature 0.6, top-p 0.95, top-k 20 (open models)
AIME 2025	4 samples per problem, temperature 0.6, top-p 0.95, up to 32,768 tokens

C THINK-BLOCK LENGTHS AND EXAMPLES

Table 9 gives the per-seed length statistics behind Figure 2. It counts tokens of the Qwen3 tokenizer in the think block without its delimiters. The examples below are the second-turn think blocks of the four Qwen3-4B students of the main seed on two test tasks that all four solved, shortened where marked. Figure 3 shows a complete inference turn of the CoT-proxy student.

Task. *Who is the most followed artist I follow on Spotify?*

Initial. Okay, so I need to find out who the most followed artist Valerie follows on Spotify. Let me think through the steps.

First, looking at the API descriptions output from the previous step, there's an API called "show_following_artists" with the description "Get a list of artists the user is following." That's exactly what I need. So the next step is to call this API using the access token we already have from the previous login.

Wait, in the previous task, they had a spotify_access_token variable. So I should use that token here.

The code would be something like:

```
followed_artists = apis.spotify.show_following_artists(access_token=
spotify_access_token)
```

Task: Give me 3 comma-separated artist names with the most songs in my Spotify playlists. If the same song is present in multiple playlists, count it once.

<think> I will collect unique song IDs from the playlists, then call `show_song` for each to get artists and count frequencies. I'll then find the top 3.

First, I'll extract the song IDs into a set.

</think>

```python
unique_song_ids = set()
for playlist in all_playlists:
    unique_song_ids.update(playlist['song_ids'])
print(f"Found {len(unique_song_ids)} unique songs")
```

Figure 3: One inference turn of the Qwen3-4B CoT-proxy student on an AppWorld test task. The yellow token is the prefilled opening of the think block; everything after it is generated by the student (green). The student fills the think block with its own reasoning and moves directly to the action, without the summary block that was present during training.

Table 9: Think-block length in tokens per response for the Qwen3-4B students of Table 1 on AppWorld, per evaluation seed, and the teacher's thoughts in the AppWorld training data. Mean, median and p90 are over the closed think blocks of each seed, and the first-turn column is the median over the first response of every task.

Configuration	Seed	Responses	Mean	Median	p90	First turn
Initial	1	2,290	1,005	534	2,264	4,799
	2	2,401	986	527	2,152	4,626
	3	2,162	1,050	562	2,411	4,792
Full-CoT	1	4,005	104	68	224	180
	2	4,392	106	87	202	188
	3	3,531	101	56	205	192
Action-only	1	2,857	638	227	1,242	4,740
	2	2,867	611	214	1,185	4,561
	3	2,728	625	215	1,213	4,502
CoT-proxy	1	3,911	216	47	288	2,399
	2	3,788	166	52	251	976
	3	3,926	169	49	288	1131
Teacher, training data	–	3,704	124	66	263	–

But I need to check if the API name is exactly "show_following_artists". From the output, yes, it's listed there.

Once I get the list of followed artists, each artist should have a 'followers' count. Then, I need to find the artist [...]

Full-CoT. The user wants to find the most followed artist they follow on Spotify. I have the list of available APIs for Spotify. The relevant APIs seem to be: 1. `show_following_artists`: Get a list of artists the user is following. 2. `show_artist`: Get details of a specific artist, which likely includes their follower count.

First, I need to log in to Spotify. I'll need the supervisor's password for Spotify. I can get the password from the `supervisor` app. Then I'll log in. Then I'll get the list of following artists. Then I'll iterate through them to find the one with the highest follower count. Finally, I'll return the name of that artist.

Let's start by getting the Spotify password and logging in. I'll need to check the `supervisor` app APIs to get the password.

Action-only. Okay, let's see. I need to find the most followed artist that Valerie follows on Spotify.

First, I checked the Spotify API descriptions to see what's available. The previous step showed that after logging in, I need to check the APIs. Let me run the command to list all Spotify APIs again.

Wait, in the previous step, I logged in successfully. Now I should check the available APIs to find the right ones for followed artists.

So, code to print the API descriptions for Spotify:

```
print(apis.api_docs.show_api_descriptions(app_name='spotify'))
```

CoT-proxy. Okay, so I need to find the most followed artist that Valerie follows on Spotify. I already have the access token from the login step. Next, I should check the available Spotify APIs to see how to get the followed artists.

Let me list the available Spotify APIs to find the right one.

Task. *How many venmo friends have I made since the start of March this year?*

Initial. Okay, so the user wants to know how many Venmo friends they've made since the start of March this year. Let me think through the steps.

First, in the previous step, we got the list of passwords from the supervisor app. The output shows that there's a Venmo account with password "B)]&d{u". So the next step is to extract that password.

I need to find the Venmo password from the passwords list. The code should filter for account_name "venmo" and get the password. Then, I can proceed to check Venmo's API descriptions to see what endpoints are available.

Wait, the next logical step after getting the passwords is to extract the Venmo password. So the code would be something like:

```
venmo_password = [p["password"] for p in passwords if p["account_name"]
== "venmo"][0]
```

But I need to make sure that the list has at least one entry for "venmo". From the output, it does.

Once I have the [. . .]

Full-CoT. The user wants to know how many Venmo friends they have made since the start of March 2023. I need to check the available APIs for Venmo to find one that lists friends and possibly their addition date. I see `show_friends` or similar is not explicitly listed, but `search_friends` might be useful if I can filter by date. However, usually, friend lists are static and adding date is part of the friend object. Let's check the `search_friends` API to see if it returns addition date. Also, I need to get the Venmo password from the supervisor app first.

Let's start by getting the Venmo password.

Action-only. Okay, let's see. The user wants to know how many Venmo friends they've made since the start of March this year.

First, I need to get the login API details for Venmo. The previous output shows that Venmo has a 'login' API. Let me check its specs.

So the next step is to run:

```
print(apis.api_docs.show_api_doc(app_name='venmo', api_name='login'))
```

That should show the parameters and response for logging in. Once I have that, I can proceed to get the password from the supervisor app.

Wait, earlier when checking supervisor's account passwords, it returned Spotify and file_system. Maybe Venmo's password is also there. Let me check.

But first, let's get the login API details to confirm the parameters.

So the code block would be:

```
print(apis.api_docs.show_api_doc(app_name='venmo', api_name='login'))
```

CoT-proxy. Okay, I need to find out how many Venmo friends Katie Simpson has made since the start of March this year. Let me start by getting the Venmo password from the supervisor app. Wait, the output already shows the password for venmo as "B)]&d{u". But the user mentioned using the supervisor app to get the password, which I just did.

Next, I should check the Venmo login API to see how to authenticate. Let me print the available APIs first to see what's available.

Wait, the user said to check the APIs first before calling them. So I should list the Venmo APIs to find the login and friends-related ones.

I'll run `print(apis.api_docs.show_api_descriptions(app_name='venmo'))` to see the available APIs.

D Additional experiments

D.1 Thinking enabled versus disabled

Qwen3-32B is a hybrid model that supports inference with and without thinking (Yang et al., 2025), which lets us ask how much of a trained student's performance depends on reasoning at inference. We evaluate the initial model and the CoT-proxy student trained on the open teacher's SWE-bench trajectories a second time with thinking disabled, by prefilling an empty think block, with everything else unchanged. Table 10 gives the results. The initial model loses little without thinking, whereas the CoT-proxy student drops from 21.7% to 5.1%, so its gain rests on reasoning at inference. Without thinking, both also take more turns per task.

Table 10: SWE-bench Verified task success and mean number of model responses per task for the initial Qwen3-32B model and its CoT-proxy student of Table 1, evaluated with thinking enabled and disabled on the same checkpoints. Every number is the mean over three seeds, with the standard deviation for the scores.

Configuration	Thinking	Verified (%)	Turns
Initial	enabled	5.8±0.2	14.8
Initial	disabled	4.2±1.3	38.6
CoT-proxy	enabled	21.7±0.8	48.1
CoT-proxy	disabled	5.1±1.6	64.5

D.2 Content of the summary

To test whether CoT-proxy students use what the summaries say, we train further Qwen3-4B students on the open teacher's AppWorld and SWE-bench trajectories with the CoT-proxy recipe, with the summaries randomly permuted across all turns of all training trajectories, so that each turn carries a well-formed summary written for another step. All other training text, the retained turns and the loss masks are identical to CoT-proxy. Table 11 shows that the shuffled-summary students do not come close to CoT-proxy: they are 5 points above action-only on AppWorld and below it on SWE-bench. The gain of CoT-proxy therefore comes from the content of the summaries rather than from the presence of a summary block.

D.3 Responses at the output cap

Table 12 counts, for the Qwen3-4B students of Table 1 on AppWorld over the three seeds, the responses that ended at the 8,192-token output cap. The initial student never reaches the cap despite its long think blocks, and the trained students almost never do.

Table 11: Task success of Qwen3-4B students trained on the open teacher's trajectories, on AppWorld and SWE-bench Verified. The first four rows are from Table 1; the last row uses the CoT-proxy recipe with summaries shuffled across all training turns and trajectories. Every number is the mean over three seeds with its standard deviation.

Training target	AppWorld TGC (%)	SWE-bench Verified (%)
Initial	7.3±1.9	2.7±0.6
Full-CoT	29.2±3.3	3.9±0.8
Action-only	17.9±1.6	5.5±0.8
CoT-proxy	34.5±3.7	15.1±1.4
CoT-proxy, shuffled summaries	23.0±3.5	3.7±1.1

Table 12: Responses of the Qwen3-4B students on AppWorld that ended at the 8,192-token output cap, pooled over the three seeds.

Configuration	Responses	At the cap	Share (%)
Initial	6,853	0	0.00
Full-CoT	11,928	34	0.29
Action-only	8,452	1	0.01
CoT-proxy	11,625	0	0.00

D.4 Paired statistical comparisons

Every training target is evaluated on the same tasks with matched seeds, so we compare targets task by task. For each pair of targets we report the difference in task success averaged over the three seeds, a 95% confidence interval from a paired bootstrap over tasks (resampling whole scenarios on AppWorld; 10,000 resamples), and a p-value from a paired permutation test that swaps the outcomes of the two targets within each task or scenario, Holm-corrected within each table. Table 13 shows that action-only and CoT-proxy training improve on the initial model in every setting and that CoT-proxy improves on action-only in every setting except Qwen3-32B on AppWorld, where no difference between trained targets is significant. CoT-proxy and full-CoT differ significantly only for Qwen3-4B on SWE-bench.

Table 13: Paired differences in task success between training targets (percentage points; AppWorld TGC and SWE-bench Verified), averaged over three seed-matched runs. Confidence intervals are from a paired bootstrap over tasks (scenarios on AppWorld); p-values are from a paired permutation test, Holm-corrected within the open-teacher and the closed-teacher comparisons. Bold marks $p < 0.05$.

Teacher	Model	Benchmark	Comparison	Δ (pp)	95% CI	p (Holm)
Qwen3.6-27B	Qwen3-4B	AppWorld	CoT-proxy − Action-only	**+16.7**	[+10.1, +23.4]	0.002
			CoT-proxy − Full-CoT	+5.4	[−0.8, +11.5]	0.56
			Full-CoT − Action-only	**+11.3**	[+4.8, +17.5]	0.009
			Action-only − Initial	**+10.5**	[+6.2, +15.3]	0.002
			CoT-proxy − Initial	**+27.2**	[+20.0, +34.5]	0.002
Qwen3.6-27B	Qwen3-4B	SWE-bench	CoT-proxy − Action-only	**+9.6**	[+7.5, +11.8]	0.002
			CoT-proxy − Full-CoT	**+11.2**	[+9.1, +13.5]	0.002
			Full-CoT − Action-only	−1.6	[−3.0, −0.3]	0.18
			Action-only − Initial	**+2.8**	[+1.5, +4.2]	0.002
			CoT-proxy − Initial	**+12.4**	[+10.1, +14.9]	0.002
Qwen3.6-27B	Qwen3-32B	AppWorld	CoT-proxy − Action-only	+4.4	[−0.8, +9.9]	0.56
			CoT-proxy − Full-CoT	+2.0	[−4.0, +7.7]	1.00
			Full-CoT − Action-only	+2.4	[−3.4, +8.1]	1.00
			Action-only − Initial	**+13.9**	[+8.5, +19.2]	0.002
			CoT-proxy − Initial	**+18.3**	[+12.1, +24.6]	0.002
Qwen3.6-27B	Qwen3-32B	SWE-bench	CoT-proxy − Action-only	**+7.3**	[+5.1, +9.5]	0.002
			CoT-proxy − Full-CoT	+0.6	[−1.6, +2.9]	1.00
			Full-CoT − Action-only	**+6.7**	[+4.3, +9.1]	0.002
			Action-only − Initial	**+8.5**	[+6.5, +10.5]	0.002
			CoT-proxy − Initial	**+15.9**	[+13.2, +18.5]	0.002
GPT-5.6 Luna	Qwen3-4B	AppWorld	CoT-proxy − Action-only	**+13.1**	[+6.0, +20.6]	0.001
			Action-only − Initial	**+27.8**	[+20.8, +34.9]	<0.001
			CoT-proxy − Initial	**+40.9**	[+31.7, +50.4]	<0.001
GPT-5.6 Luna	Qwen3-32B	AppWorld	CoT-proxy − Action-only	**+14.7**	[+9.3, +20.2]	<0.001
			Action-only − Initial	**+32.9**	[+26.2, +39.7]	<0.001
			CoT-proxy − Initial	**+47.6**	[+39.7, +55.4]	<0.001