ECHO: Terminal Agents Learn World Models for Free

Anonymous Author(s)
Affiliation
Address
email

Abstract

CLI agents are the closest thing language models have to an embodied setting: the model emits commands, the terminal executes them, and the returned stream—stdout, errors, file contents—records the consequences of its actions. We argue that this stream is a supervision signal, but standard agent RL largely discards it: GRPO-style training updates action tokens while ignoring environment responses, even though rollouts already contain them. We introduce **ECHO** (Environment Cross-entropy Hybrid Objective), an auxiliary loss that trains the policy to predict observation tokens alongside the policy gradient. ECHO reuses the same forward pass, requires no additional rollouts, and turns terminal feedback into a dense signal that remains informative even on failed trajectories. This objective shapes the policy toward a world model of terminal dynamics, plausibly improving its priors over which actions are likely to succeed. On TerminalBench 2, ECHO nearly doubles pass@1: Qwen3-8B improves from 2.70% to 5.17%, and Qwen3-14B from 5.17% to 10.79%. From base Qwen3-8B, ECHO+GRPO matches expert-SFT-then-GRPO on held-out terminal tasks without the expert demonstrations required for SFT, and closes roughly half this gap on TerminalBench 2. Without verifier rewards, ECHO can lead to self-improvement through exploration alone, suggesting environment observations are not merely context for actions but supervision for the policy itself.

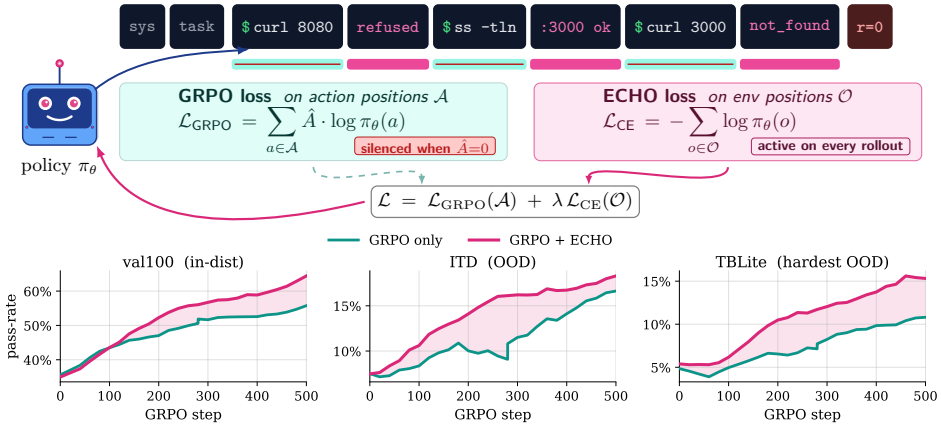

Figure 1: ECHO supervises terminal observation tokens already present in GRPO rollouts. GRPO updates only action tokens; ECHO also trains the policy to predict the terminal responses those actions elicit.

1 Introduction

Language-model agents learn by acting in environments: a terminal agent edits files, runs tests, reads errors, and issues follow-up commands until a verifier declares success or failure. The terminal responds to every action with real environment feedback, but the reward signal that ultimately reaches the trainer is sparse, delayed, and binary.

This sparsity is particularly pronounced in terminal-agent training. As a concrete example, in our Qwen3-8B setting, fewer than 15% of on-policy rollouts solve the task, so under standard GRPO [Shao et al., 2024] the vast majority of interaction yields little policy-gradient signal. These rollouts are far from uninformative.

Even a failed trajectory contains the actual outputs of whatever the agent ran: file listings, training logs, build errors, the contents of a config file, the response from a web request, a stack trace, the result of a `grep`, or anything else a shell can produce. Every token from the terminal enters the model's forward computation, yet none of it enters the loss. The transcript becomes context for the next action and nothing more — and we believe this wastes the most abundant signal in an interactive environment. *We instead train on these tokens directly.*

Why should this help? A long-running intuition in language modeling is that good prediction implies good understanding: predicting the next token well, in Sutskever's phrasing, "means you understand the underlying reality that led to the creation of that token" [Sutskever, 2023]. We borrow this view for agents. To predict what the terminal will return after a command, the model has to internalize what the command actually does. A policy that predicts terminal output well is, in a small but real sense, a policy that understands terminals.

We introduce **ECHO** (Environment Cross-entropy Hybrid Objective), an auxiliary loss for multi-turn agent RL. ECHO adds cross-entropy on environment-observation tokens to the usual GRPO loss on action tokens:

$$\mathcal{L}_{\mathrm{ECHO}}(\theta) = \mathcal{L}_{\mathrm{GRPO}}(\theta; \mathcal{A}) + \lambda\, \mathcal{L}_{\mathrm{CE}}(\theta; \mathcal{O}'), \tag{1}$$

where $\mathcal{A}$ indexes assistant-action positions and $\mathcal{O}'$ indexes terminal-output positions inside the environment observations. The objective requires no teacher model, extra rollouts, or an additional forward pass: it uses the same logits already computed for the policy update, but gathers them at a different set of token positions. In effect, ECHO turns the environment stream into dense supervision, so even failed rollouts can teach the policy how the terminal responds.

We test this hypothesis in TerminalBench-style Docker environments with Qwen3-8B, OpenThinker-Agent-v1-SFT, and Qwen3-14B starting policies. ECHO consistently improves both internal held-out evaluations and the public TerminalBench-2.0 benchmark. On the headline external metric, it raises Qwen3-8B pass@1 from 2.70% to 5.17% and Qwen3-14B from 5.17% to 10.79% — nearly a doubling at both scales. ECHO also reduces dependence on expert demonstrations: from a base Qwen3-8B, GRPO+ECHO matches OpenThinker-SFT+GRPO on internal evaluations without using any of the $\sim$15k expert demonstrations behind the SFT model, and closes about half of the expert-SFT gap on TerminalBench-2.0.

Our contributions:

1. **Environment tokens are supervision.** Stdout, errors, files, and tool traces are dense, on-policy training targets that standard agent RL throws away.
2. **ECHO.** A length-normalized cross-entropy on observation tokens, added to GRPO's loss on action tokens. The two terms share one forward pass, require no teacher model, and add no extra rollouts.
3. **Consistent gains across model families.** ECHO improves Qwen3-8B, OpenThinker-Agent-v1-SFT, and Qwen3-14B on both internal evaluations and TerminalBench-2.0, nearly doubling pass@1 at 8B and 14B. From a base Qwen3-8B, GRPO+ECHO matches SFT-bootstrapped GRPO from $\sim$15k expert demonstrations, without using any of them.
4. **ECHO encourages self-improvement.** Even without a verifier, the policy can improve by predicting what its own actions cause in the environment.

Taken together, these findings suggest agent training has been operating with a supervision source masked out: every policy action has a consequence in the environment, and that consequence is in the rollout already. A simple auxiliary loss is one way to learn from it, including on failed rollouts.

2 Preliminaries

Multi-Turn Rollout Structure. A training sequence interleaves a system prompt, the user task, and a transcript of (assistant action, environment observation) pairs:

$$\underbrace{[\text{sys}]\,[\text{task}]}_{\text{prompt}}\,[\text{action}_1]\,[\text{obs}_1]\,[\text{action}_2]\,[\text{obs}_2]\,\cdots\,[\text{action}_K]\,[\text{obs}_K].$$

At each turn, the policy samples action tokens conditioned on the entire prior transcript; the harness parses these into a bash command, executes it in the container, and appends the resulting terminal output as the next observation. Let $\mathcal{A} \subseteq \{1, \ldots, T\}$ index assistant-action positions and $\mathcal{O} \subseteq \{1, \ldots, T\}$ index environment-observation positions. Trainers compute log-probabilities on the full sequence (every action depends on prior observations), but the policy-gradient loss is applied only on $\mathcal{A}$. Observations in $\mathcal{O}$ are conditioned on, but receive no direct training signal.

Group-Relative Policy Optimization. GRPO [Shao et al., 2024] optimizes a clipped policy-gradient objective with group-normalized advantages and no learned value function. For prompt x, sampled rollouts $\{y^{(i)}\}_{i=1}^{n}$, and binary rewards $r^{(i)} \in \{0, 1\}$, each rollout receives a scalar group-normalized advantage $\hat{A}^{(i)}$, applied uniformly to its action-token positions using the clipped importance ratio $\rho_t^{(i)}$:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{\sum_i |\mathcal{A}^{(i)}|} \sum_i \sum_{t \in \mathcal{A}^{(i)}} \min\Big(\rho_t^{(i)} \hat{A}^{(i)}, \text{clip}(\rho_t^{(i)}, 1-\epsilon, 1+\epsilon)\hat{A}^{(i)}\Big). \tag{2}$$

GRPO optimizes only assistant action tokens. Observation tokens remain in context and affect future actions, but are not policy-gradient targets. With sparse binary rewards, all-zero groups have no reward contrast. In mixed groups, unsuccessful trajectories receive only a trajectory-level negative signal, so learning concentrates on rare successful rollouts.

3 Method

3.1 ECHO Objective

ECHO is a supervised next-token objective on observation positions, added alongside GRPO without modifying it. Let $p_\theta(\cdot \mid x_{<t})$ denote the model's next-token distribution. The ECHO loss is a length-normalized cross-entropy on a subset $\mathcal{O}' \subseteq \mathcal{O}$ of observation tokens:

$$\mathcal{L}_{\text{CE}}(\theta; \mathcal{O}') \;=\; -\frac{1}{Z} \sum_{t \in \mathcal{O}'} \log p_\theta(x_t \mid x_{<t}), \tag{3}$$

where $Z = |\mathcal{O}|$ normalizes each sequence by its total observation length. The total objective is $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{GRPO}} + \lambda \mathcal{L}_{\text{CE}}$ as in equation 1.

The two losses share a single actor forward pass: the same logits feed both, gathered through different masks (assistant-action positions for GRPO, observation positions for ECHO). ECHO therefore requires no second rollout, teacher model, or second forward pass; the only added work is a masked log-probability sum. It composes orthogonally with stabilization techniques aimed at the policy gradient itself, including void-trajectory filtering [Xue et al., 2025], overlong filtering and clip-higher [Yu et al., 2025], and KL-regularized reference updates [Ouyang et al., 2022]. Algorithm 1 summarizes the resulting update.

Computationally, ECHO adds a masked log-probability computation and reduction over existing logits, rather than a separate model evaluation. Its backward pass adds gradients through the same network activations used by the policy objective.

3.2 Choosing Observation Targets

We set $\mathcal{O}'$ to the env tokens only, excluding the harness's warning prefix. An observation message has internal structure

$$\underbrace{\texttt{WARNINGS:\textbackslash n- ...}}_{\text{warning tokens}}\ \underbrace{\texttt{<command_output> ... \ </command_output>}}_{\text{terminal-output (env) tokens}},$$

Algorithm 1 ECHO-Augmented GRPO

Require: rollout sequence $x_{1:T}$; action mask $\mathcal{A}$; env-observation mask $\mathcal{O}'$; full-observation length $|\mathcal{O}|$; advantages $\{\hat{A}_t\}_{t\in\mathcal{A}}$; mixing coefficient λ.

$\boldsymbol{\ell} \leftarrow \texttt{forward}_\theta(x_{1:T})$ ▷ single actor forward; logits at every position

$\log p_t \leftarrow \log\ \mathrm{softmax}(\boldsymbol{\ell}_t)[x_t]$ for all t

$\mathcal{L}_{\text{GRPO}} \leftarrow \texttt{ClippedGRPO}\big(\{\log p_t\}_{t\in\mathcal{A}},\ \{\hat{A}_t\}_{t\in\mathcal{A}}\big)$

$\mathcal{L}_{\text{CE}} \leftarrow -\big(\sum_{t\in\mathcal{O}'} \log p_t\big)/|\mathcal{O}|$

return $\mathcal{L}_{\text{GRPO}} + \lambda\,\mathcal{L}_{\text{CE}}$

where the warning block is a rule-based message emitted when the previous tool call fails parsing or violates format constraints, and the env block carries the actual command output. The reason for excluding warnings is empirical: warning tokens are low-entropy and the model memorizes them within ∼60 training steps, so warn-only configurations quickly lose useful gradient. Terminal-output tokens, by contrast, encode task-specific feedback (file names, test failures, byte counts, error formats) and continue to provide informative gradient throughout training.

3.3 Tuning the Loss Weight

We swept $\lambda \in \{0.001, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2\}$ and found a productive range of 0.01–0.05. Below this range, the auxiliary gradient is too small to shape representations reliably: world loss can fluctuate or increase while the policy objective dominates. Above this range, the observation objective begins to compete with the policy update; at $\lambda = 0.1$ policy quality plateaus or degrades, and at $\lambda = 0.2$ runs can collapse into degenerate rollouts whose terminal outputs are easy to predict but no longer useful. We therefore use a constant $\lambda = 0.05$ in all reported experiments. The constant weight is naturally self-annealing: as the model learns terminal-output statistics, $\mathcal{L}_{\text{CE}}$ falls rapidly, reducing the auxiliary contribution without an explicit schedule.

4 Experimental Setup

Training Task Corpus. We start from 2700 curated terminal tasks: 1977 from Endless Terminals [Gandhi et al., 2026a,b] and 723 from OpenThoughts-Agent-v1-RL [OpenThoughts Team, 2025b], after filtering out analysis/computation, specialized-application, infrastructure/networking, and complex-bash domains. We then generate 6170 additional tasks with a modified Endless Terminals pipeline, covering task specification, Dockerfile generation/validation, and Harbor-format export. We retain only tasks solved by GPT-5 in at least one of 16 attempts, yielding 8870 tasks across data processing, system operations, and development/tooling. We train on 8770 tasks and hold out 100 for in-distribution validation (**val100**).

Harness and Runtime Environment. At each turn, the policy conditions on its prior reasoning, commands, and command outputs, then emits a thinking block followed by Qwen XML-format bash commands or a task-done signal. A minimal training harness parses the first command or completion signal, executes it, and returns optional format warnings plus stdout/stderr and exit code as the next observation. Episodes run for up to 16 turns in Docker, orchestrated by Harbor [Anthropic and contributors, 2025], with a 16k context window and at most 2048 generated tokens per turn. We verify success with unit tests at episode end, using 10-minute agent and 2-minute verifier timeouts per training task.

Models. We train on three starting policies: **Qwen3-8B** [Qwen Team, 2025], **OpenThinker-Agent-v1-SFT** [OpenThoughts Team, 2025a] (OT-SFT), and **Qwen3-14B** [Qwen Team, 2025]. OT-SFT is a Qwen3-8B model SFT'd on ∼15k expert terminal-agent demonstrations from the GLM-4.6 model.

RL Recipe. All experiments use the same GRPO recipe: $n = 16$ rollouts per prompt, batch size of 16, learning rate 1×10^{-6}, gradient clip 0.2, prompt-level advantage normalization, sequence-level loss aggregation, no KL penalty unless noted, and rollout temperature 0.8. For ECHO-augmented

Model	Setup	val100	ITD	TBLite	TB2 p@1	TB2 p@3	TB2 p@5
Qwen3-8B	Base	34.2	7.0	4.9	1.57	3.71	4.49
	+ GRPO	54.9	16.2	9.5	2.70	6.74	8.99
	+ GRPO + ECHO	**63.7**	**18.9**	**11.4**	**5.17**	**10.45**	**13.48**
OT-SFT	SFT	38.5	10.7	6.0	5.62	10.45	12.36
	+ GRPO	63.5	18.8	11.6	7.64	14.38	17.98
	+ GRPO + ECHO	**73.1**	**22.7**	**12.9**	**7.87**	**13.82**	**17.98**
Qwen3-14B	Base	35.3	12.1	5.7	4.27	8.99	12.36
	+ GRPO	60.3	17.9	9.8	5.17	10.67	13.48
	+ GRPO + ECHO	**65.0**	**19.8**	**15.1**	**10.79**	**16.52**	**19.10**

Table 1: Pass rates before RL, after GRPO, and after GRPO+ECHO for three starting policies. *val100* is an in-distribution 100-task holdout; *ITD* (71 tasks), *TBLite* (OpenThoughts-TBLite), and *TB2* (TerminalBench-2.0, n=89, pass@k over 5 attempts) are out-of-distribution. Bold marks the GRPO+ECHO row in each block. Standard errors and paired Wilcoxon tests appear in App. F–G.

GRPO runs, we use $\lambda = 0.05$ to scale the ECHO loss. We provide a reward of 1 if final tests for a task pass, and a reward of 0 otherwise. We train each model for 500 GRPO steps on 8 B200 GPUs.

Evaluation. We evaluate model performance on **val100**, internal-dev (**ITD**), OpenThoughts-TBLite (**TBLite**) [OpenThoughts Team, 2025d], and **TerminalBench-2.0** (TB2) [Terminal-Bench Team, 2025a]. val100 is a held-out set of 100 tasks from our training corpus. Internal-dev is a set of 71 tasks focusing on data processing, systems operations, and development/tooling, sampled from TerminalBench 1.0 (core and non-core) and OpenThoughts-TB-dev [OpenThoughts Team, 2025c]. OpenThoughts-TBLite is a set of 100 terminal-bench-style tasks calibrated for small-model performance relative to the harder TB2 benchmark. On val100, ITD, and TBLite, we evaluate using our minimal agent harness, with 8 rollouts per task at temperature 0.6. On TB2, we use the Terminus 2 harness [Terminal-Bench Team, 2025b] and perform 5 rollouts at temperature 0.6 with 32k context.

5 Results

ECHO improves every starting policy on every benchmark we test. TerminalBench-2.0 pass@1 nearly doubles at both 8B ($2.70 \rightarrow 5.17$) and 14B ($5.17 \rightarrow 10.79$), and internal pass rates rise on every slice. The same checkpoints become substantially better predictors of terminal feedback. They match GRPO's peak performance in 1.5–$2.3\times$ fewer training steps and waste fewer turns at inference. Starting from base Qwen3-8B, adding ECHO to GRPO fully matches what an expert SFT initialization buys on internal evaluations, and recovers half of its lead on TerminalBench-2.0 — without using any of the $\sim$15k expert demonstrations the SFT model requires.

5.1 ECHO Substantially Boosts GRPO Performance

ECHO consistently improves task success. Adding ECHO to GRPO raises task success on every internal evaluation (val100 and ITD), TBLite, and TerminalBench-2.0. TerminalBench-2.0 pass@1 nearly doubles for Qwen3-8B ($2.70 \rightarrow 5.17$, $\times 1.9$) and Qwen3-14B ($5.17 \rightarrow 10.79$, $\times 2.1$).

Table 1 compares matched GRPO and GRPO+ECHO checkpoints across three starting policies. The setup isolates a single change: whether terminal-output tokens are additionally trained with a cross-entropy objective alongside the standard policy-gradient loss. Across all three starting policies, ECHO improves every internal evaluation metric and consistently boosts performance on TerminalBench-2.0 under the Terminus-2 harness. At 8B, TB2 pass@1 nearly doubles from 2.70 to 5.17; at 14B, it rises from 5.17 to 10.79, with pass@5 increasing from 13.48 to 19.10. The strongest improvements are statistically significant under paired Wilcoxon tests (App. G), including the +5.62 pp TB2 pass@1 gain for Qwen3-14B over matched GRPO.

The 14B result is particularly notable. Although the internal gains at 14B are smaller than at 8B, the improvements on TerminalBench-2.0 are substantially larger. One plausible explanation is that the

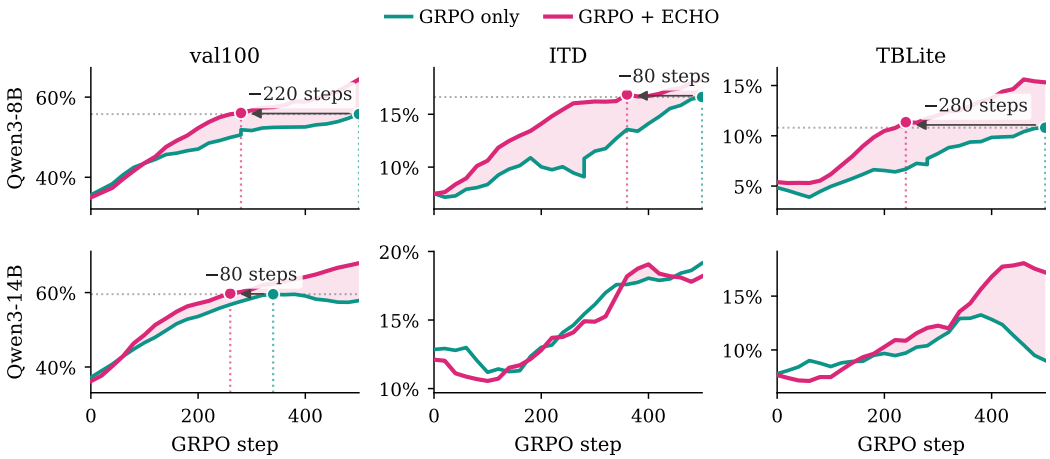

Figure 2: Pass-rate training curves over 500 GRPO steps. Teal shows GRPO only; pink shows GRPO+ECHO, with shading where ECHO exceeds the matched GRPO baseline. Rows are Qwen3-8B base and Qwen3-14B base; columns are val100 (in-distribution), ITD (OOD), and TBLite (hardest OOD). OT-SFT–initialized curves appear in App. E.

larger model can internalize more generalizable terminal dynamics from the observation stream, while at smaller scales the policy and world-modeling objectives compete more directly for limited capacity. Figure 2 shows the corresponding training dynamics: at 8B, ECHO consistently outperforms the GRPO baseline throughout training, while at 14B it reaches a substantially higher final plateau.

5.2 Does ECHO Really Learn Terminal Dynamics?

To test whether ECHO improves the model's understanding of terminal dynamics, we evaluate environment-token cross-entropy on a fixed pool of held-out trajectories. We collect trajectories from our strongest checkpoint (Qwen3-14B + GRPO + ECHO) on val100, ITD, and TBLite, then measure how well each model predicts the resulting terminal-output tokens.

This evaluation is intentionally off-policy: most evaluated models did not generate these trajectories themselves. Low cross-entropy therefore requires predicting the outcomes of another agent's actions, rather than memorizing a model's own rollout distribution. If ECHO learns a useful terminal world model, its checkpoints should outperform matched GRPO-only baselines on this prediction task.

ECHO learns transferable terminal dynamics. On held-out, off-policy trajectories, GRPO+ECHO sharply lowers environment-token cross-entropy across all starting policies and evaluation slices, while GRPO alone barely changes it.

Figure 3 shows exactly this pattern. GRPO alone barely changes environment-token cross-entropy relative to the starting policy, despite improving task success. ECHO, by contrast, sharply lowers prediction error across all starting policies and evaluation distributions. For Qwen3-14B, cross-entropy drops from 0.22→0.08 on val100, 0.40→0.29 on ITD, and 0.22→0.15 on TBLite; for Qwen3-8B, the corresponding drops are 0.27→0.09, 0.46→0.31, and 0.26→0.16. These results support the central mechanism behind ECHO: the auxiliary objective trains the policy to predict how the terminal responds to actions, and the dynamics transfer beyond the model's own trajectories.

5.3 ECHO Reduces Dependence on Expert Demonstrations

Expert SFT primes terminal agents before RL by training them on demonstrations from a stronger policy. In our comparison, OT-SFT is Qwen3-8B SFT'd on ∼15k expert demonstrations from a GLM-4.6 teacher. We ask how much of this expert initialization can be replaced by letting the base model explore on-policy and learn a dense ECHO signal from its own terminal feedback.

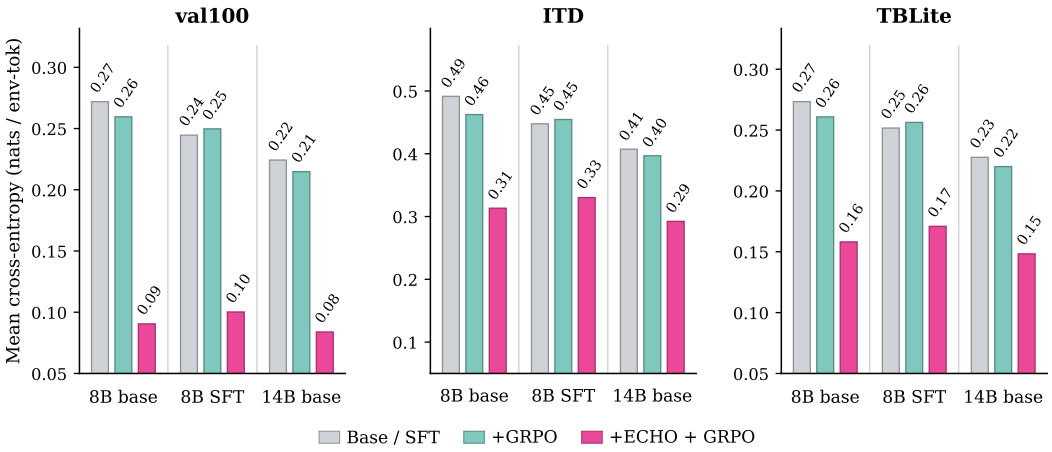

Figure 3: Per-token cross-entropy on terminal output tokens for a fixed reference pool of held-out rollouts. Each panel evaluates a different distribution; within each panel, starting policies are shown before RL, after GRPO, and after GRPO+ECHO. RL alone barely changes env-token CE. ECHO sharply lowers it across all model families and evaluation slices. Lower is better.

We define the expert-SFT gap as the gain from OT-SFT+GRPO over Qwen3-8B+GRPO, and the ECHO lift as the gain from adding ECHO to Qwen3-8B+GRPO. Figure 4 shows that ECHO recovers almost all of the OT-SFT advantage on internal evaluations: 101.6% of the gap on val100, 103.9% on ITD, and 88.9% on TBLite. This suggests that much of what expert SFT provides is environment familiarity—how commands behave, how errors appear, what files and tests reveal, and how terminal state evolves. ECHO learns this signal directly from the model's own rollouts.

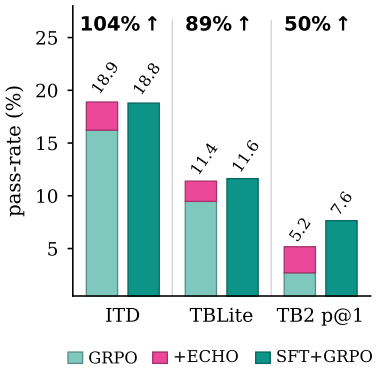

Figure 4: ECHO recovers most of the benefit of expert-SFT initialization on internal evaluations and roughly half on TerminalBench-2.0.

On TerminalBench-2.0, ECHO closes roughly half of the OT-SFT gap: 50.0% on pass@1, 48.6% on pass@3, and 50.0% on pass@5. This is weaker than the internal recovery but still substantial. A likely explanation is that TB2 requires not only terminal familiarity, but also higher-level strategy: which commands to try, how to decompose a task, when to inspect versus edit, and when to stop. ECHO narrows the gap by learning environment dynamics, but expert demonstrations still help with strategic action selection on the hardest tasks.

ECHO substantially closes expert-SFT gap. On Qwen3-8B, GRPO+ECHO recovers nearly all of the OT-SFT initialization gain on internal evaluations and about half of the TerminalBench-2.0 gain, without using expert demonstrations.

5.4 Training and Inference Efficiency

Because ECHO trains on observation tokens already present in each rollout, it can improve how much learning the policy extracts from a fixed sample budget. We measure this by asking when each ECHO run first exceeds the best validation score reached by its matched GRPO-only run within 500 steps, using the aggregate total score (val100+ITD+TBLite) and TBLite alone.

ECHO often learns faster and uses inference budget more productively. At 8B, ECHO reaches the GRPO-only peak in 1.5–2.3× fewer training steps. At 14B, both runs peak at the same step, but ECHO reaches a higher plateau. At inference, ECHO halves TB2 timeouts for Qwen3-8B and OT-SFT and reduces completion tokens for all three model pairs.

Model	Total			TBLite		
	GRPO	+ECHO	Speedup	GRPO	+ECHO	Speedup
Qwen3-8B	460	240	1.92×	500	220	2.27×
OT-SFT	400	260	1.54×	220	300	0.73×
Qwen3-14B	380	380	1.00×	380	380	1.00×

Table 2: First step at which each run reaches its best internal score within 500 GRPO steps. *Total* aggregates val100, ITD, and TBLite; *TBLite* isolates the hardest OOD slice. Pink speedup cells indicate ECHO reaches the score faster ($> 1\times$); teal indicates GRPO is faster.

Model	Timeout rate (%)			Turns			Tokens		
	GRPO	+ECHO	Δ	GRPO	+ECHO	Δ	GRPO	+ECHO	Δ
Qwen3-8B	19.8	9.0	−55%	24.3	19.8	−18%	43.2k	30.3k	−30%
OT-SFT	45.2	24.7	−45%	66.3	37.7	−43%	35.4k	34.8k	−2%
Qwen3-14B	40.7	43.1	+6%	22.0	23.5	+7%	49.8k	43.1k	−13%

Table 3: TerminalBench-2.0 inference-time trajectory statistics. *Timeout rate*: fraction of TB2 trials that hit the per-task time limit. *Turns*: average agent turns per trial. *Tokens*: average completion tokens per trial. Pink-shaded Δ cells indicate ECHO wins (lower is better on all three columns); teal indicates GRPO wins.

Table 2 shows large 8B speedups on the aggregate score (1.54–1.92×) and for Qwen3-8B on TBLite (2.27×). The exception is OT-SFT on TBLite, where GRPO peaks earlier. At 14B, ECHO and GRPO peak at the same step, but ECHO is higher throughout training (Fig. 2, bottom row), so the benefit appears as a higher plateau rather than faster convergence.

Table 3 shows that ECHO also improves how agents spend their inference budget. For Qwen3-8B, ECHO cuts TB2 timeouts from 19.8% to 9.0% and completion tokens by 30%; for OT-SFT, it cuts timeouts from 45.2% to 24.7% and turns by 43%. Qwen3-14B is the only exception on timeouts and turns, but still uses 13% fewer tokens while achieving the largest TB2 pass@1 gain. Overall, ECHO produces policies that not only solve more tasks, but spend less interaction doing so.

5.5 Verifier-Free Adaptation

The main results use ECHO as an auxiliary objective during RL. We next ask whether the observation loss alone can improve a policy on unseen tasks, without unit-test rewards or any policy-gradient signal. Starting from our strongest 8B GRPO+ECHO checkpoint, we mask out the GRPO term and continue training for 100 steps using only $\mathcal{L}_{\text{CE}}$ on environment tokens. In this setting, the model can improve only by exploring task environments and predicting the terminal outputs of its own actions.

ECHO can drive verifier-free adaptation from interaction alone. On unseen in-distribution tasks, ECHO-only training improves val100 by +3.8%. On harder OOD tasks, after filtering to clean tool-call trajectories: ECHO improves PyTerm by +10.0% and ITD by +5.2% without any reward signal, while preserving val100 within ±1%.

We evaluate verifier-free adaptation on val100, ITD, TBLite, and PyTerm, a held-out set of 928 synthetic terminal tasks emphasizing Python script generation. For PyTerm, we use 828 tasks for ECHO-only adaptation and 100 for evaluation. The starting checkpoint solves ∼11.3% of PyTerm, comparable to its TBLite pass rate.

Table 4 shows that ECHO-only training can improve a policy without verifier rewards, but the usefulness of the signal depends on the quality of exploration. On val100, where the checkpoint is already competent, unfiltered ECHO-only training gives a +3.8 pp lift. On harder OOD tasks, the policy more often enters bad interaction regimes: malformed tool calls, parse errors, and unproductive loops, causing the model to only learn from failure modes. Training on filtered trajectories free of parsing errors and malformed tool calls removes this noise and yields sustained gains on PyTerm (+10%) and ITD (+5.2%).

Target dist.	Rollout filter	Step	Δ target (pp)	Δ val100 (pp)
val100 (in-dist.)	none	70	$+3.8$	$+3.8$
PyTerm (OOD)	parse/tool=1.0	100	$+10.0$	-0.9
ITD (OOD)	parse/tool=1.0	100	$+5.2$	$+0.4$
TBLite (OOD)	parse/tool=1.0	100	-3.9	-0.4

Table 4: ECHO-only adaptation. We mask the GRPO loss term and train only on ECHO loss. We see a $+3.8\%$ lift on val100 and OOD lifts on PyTerm (10%) and ITD (5.2%) on filtered trajectories.

Although TBLite starts at a similar pass rate to PyTerm, the same recipe degrades performance. The difference appears to be observation structure: PyTerm outputs—tracebacks and file contents—are tightly coupled to executed Python code, while TBLite often requires shell orchestration over less visible filesystem and process state. Thus verifier-free ECHO works best when clean exploration exposes predictive feedback. In those settings, the agent can improve from interaction alone.

6 Related Work and Discussion

Related Work. ECHO builds on work showing that environment interaction contains useful supervision beyond sparse reward. Classical world-model methods learn dynamics models for planning, imagination, or search [Ha and Schmidhuber, 2018, Hafner et al., 2020, 2021, 2023, Schrittwieser et al., 2020, Ye et al., 2021]; auxiliary-prediction methods in model-free RL train agents to predict pixels, rewards, forward dynamics, or latent states to improve representations under sparse feedback [Jaderberg et al., 2017, Pathak et al., 2017, Schwarzer et al., 2021]. ECHO follows this auxiliary-prediction view but in a multi-turn LM-agent setting: the targets are textual terminal observations already present in the transcript and already used as context for future actions.

For LM agents, the closest comparison is CWM [Meta FAIR CodeGen Team, 2025], which trains a large model on observation–action trajectories from Python and Docker environments. ECHO instead injects observation prediction directly into on-policy GRPO, requiring no separate corpus, world-modeling stage, dynamics model, or inference-time simulation. It is also complementary to recent multi-turn agent RL systems such as SkyRL [Cao et al., 2025], SimpleTIR [Xue et al., 2025], DAPO [Yu et al., 2025], and ArCHer [Zhou et al., 2024]: those methods stabilize the policy-gradient update, while ECHO adds a parallel supervised loss on the observation tokens.

Limitations. We evaluate ECHO through downstream task performance and observation-token cross-entropy, but do not directly prove that the policy has internalized terminal dynamics rather than benefiting from a broader representation-shaping effect. Direct probes, counterfactual prediction tests, or state-variable probes would strengthen this claim. ECHO is also not a replacement for verifier rewards: the auxiliary loss alone improves only when the starting policy can explore cleanly and the environment feedback is predictive. Finally, the loss weight must be controlled. At large λ, the policy can optimize for easy-to-predict outputs rather than task success.

Broader Impacts. ECHO lowers the cost of training terminal agents by reusing existing rollouts and logits, which may benefit scientific computing, debugging, and developer assistance. More capable shell agents also raise dual-use risks in software automation. We therefore sandbox all rollouts in ephemeral Docker containers (§4); deployments based on this recipe should retain comparable isolation, auditing, and human-in-the-loop controls.

7 Conclusion

We introduce ECHO, a simple way to turn the environment responses already present in agent rollouts into supervision. ECHO adds cross-entropy on terminal-output tokens to the same logits used for GRPO, requiring no extra rollouts, forward passes, data, or architectural changes. Across different starting policies and model sizes, ECHO improves over RL-only training, learns faster, matches much of the benefit of learning from expert demonstrations, and roughly doubles TerminalBench-2.0 pass@1. This suggests a broader opportunity for agentic RL: trajectories already contain a dense training signal waiting to be used—the observable consequences of the agent's own actions.

References

Anthropic and contributors. Harbor: Container orchestration for coding agents. Documentation, 2025. https://www.harborframework.com/docs.

Shiyi Cao, Sumanth Hegde Mei, Tyler Liu, Yowei Zhuang, Eric Wu, Andy Zhao, et al. SkyRL: A modular full-stack RL library for LLMs. GitHub repository, NovaSky-AI, 2025. https://github.com/NovaSky-AI/SkyRL.

Kanishk Gandhi, Shivam Garg, Noah D. Goodman, and Dimitris Papailiopoulos. Endless Terminals: Scaling RL environments for terminal agents. *arXiv preprint arXiv:2601.16443*, 2026a.

Kanishk Gandhi, Shivam Garg, Noah D. Goodman, and Dimitris Papailiopoulos. Endless Terminals dataset. Hugging Face dataset, 2026b. https://huggingface.co/datasets/obiwan96/endless-terminals.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with discrete world models. In *International Conference on Learning Representations*, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations*, 2017.

Meta FAIR CodeGen Team. CWM: An open-weights LLM for research on code generation with world models, 2025.

OpenThoughts Team. OpenThinker-Agent: An open-source sft recipe for coding agents. Hugging Face model card, 2025a. https://huggingface.co/open-thoughts/OpenThinker-Agent-v1-SFT.

OpenThoughts Team. OpenThoughts-Agent-v1-RL: Reinforcement learning tasks for terminal agents. Hugging Face dataset, 2025b. https://huggingface.co/datasets/open-thoughts/OpenThoughts-Agent-v1-RL.

OpenThoughts Team. OpenThoughts-TB-dev: Terminal-bench development tasks. Hugging Face dataset, 2025c. https://huggingface.co/datasets/open-thoughts/OpenThoughts-TB-dev.

OpenThoughts Team. OpenThoughts-TBLite: Terminal-bench lite tasks. Hugging Face dataset, 2025d. https://huggingface.co/datasets/open-thoughts/OpenThoughts-TBLite.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 2022.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2778–2787. PMLR, 2017.

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

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588: 604–609, 2020.

Max Schwarzer, Ankesh Anand, Rishab Goel, R. Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021.

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

Ilya Sutskever. Ilya sutskever (openai chief scientist) – why next-token prediction could surpass human intelligence. Interview by Dwarkesh Patel, Dwarkesh Podcast, March 2023. URL `https://www.dwarkesh.com/p/ilya-sutskever`. Transcript.

Terminal-Bench Team. TerminalBench: A benchmark for evaluating coding agents on terminal tasks. Stanford CRFM, 2025a. `https://www.tbench.ai/`.

Terminal-Bench Team. Terminus-2: A reference agent for terminal tasks. Software release, 2025b. `https://github.com/laude-institute/terminal-bench`.

Zhenghai Xue, Longtao Yan, Haomin Wang, Yibin Wang, Yiyang Wang, Bo Liu, and Min Yang. SimpleTIR: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. *arXiv preprint arXiv:2509.02479*, 2025.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. In *Advances in Neural Information Processing Systems*, volume 34, pages 25476–25488, 2021.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. *arXiv preprint arXiv:2402.19446*, 2024.

A World Cross-Entropy Trajectories

Figure 5 shows per-token world cross-entropy on warning tokens and on terminal-output (env) tokens over training. Warning CE drops from $\sim$5.6 nats to $<$0.05 nats by step 60—the model memorizes warning structure quickly. Env CE plateaus at 0.05–0.10 nats, the irreducible entropy of real terminal output (variable filenames, byte counts, error formats). A constant λ schedule therefore auto-anneals the warning gradient while preserving the env gradient indefinitely; this motivates the $\mathcal{O}'$=env_only choice in Section 3.2.

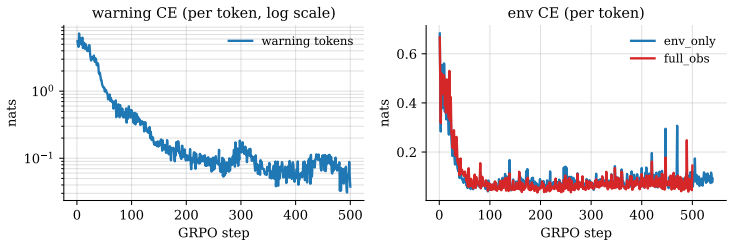

Figure 5: Per-token world cross-entropy by target type. Warning CE drops to near-zero within $\sim$60 steps; env CE plateaus at 0.05–0.10 nats and provides sustained gradient throughout training.

B Hyperparameters and Reproducibility

- Optimizer: AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.01.
- Learning rate: 1×10^{-6} constant (no warmup or decay), gradient clip 0.2.

- GRPO: $n = 16$ rollouts per prompt, batch size 16, no KL penalty, prompt-level advantage normalization, $\epsilon_{\text{lo}} = 0.2$, $\epsilon_{\text{hi}} = 0.28$.
- Sampling: training temperature 0.8; evaluation temperature 0.6.
- World loss: $\lambda \in \{0.02, 0.05\}$ (SFT vs. base); $\mathcal{O}'$ = terminal-output (env) tokens; per-sequence normalization by total observation length.
- Run length: 500–1000 GRPO steps on 8 GPUs (A100/B200 mix), $\sim$24–48 h wallclock per run.

Internal-Eval Reproducibility. 8 rollouts/task, sampling temperature 0.6, 16-turn budget. Per-task pass-rate variance at $n = 8$ is ± 0.05; mean-of-100-task variance is approximately ± 0.025 (Wilson interval). We treat trends across ≥ 3 consecutive eval points as more informative than single-checkpoint deltas.

TerminalBench-2.0 Reproducibility. Terminus-2 reference agent at temperature 0.6, $n_{\text{attempts}} = 5$, 1200 s agent and verifier timeouts, sampling seed 42. Standard error on pass@1 at $n = 5$ is $\sim$1.5 pp.

C Per-Step Training Wallclock

Stack	min/step (steady state)	500 steps cumulative
Qwen3-8B + RL	$\sim$5.4	$\sim$42 h
Qwen3-8B + RL + ECHO	$\sim$4.4	$\sim$37 h
OT-SFT + RL	$\sim$4.7	$\sim$40 h
OT-SFT + RL + ECHO	$\sim$4.5	$\sim$38 h
Qwen3-14B + RL	$\sim$5.9	$\sim$50 h
Qwen3-14B + RL + ECHO	$\sim$5.6	$\sim$47 h

World-loss runs are slightly *faster* per step than matched RL-only runs, plausibly because parse-error rates are lower and rollouts are therefore shorter on average.

D Expert-SFT Gap

Table 5 reports the absolute pass-rates for the three Qwen3-8B configurations compared in §5.3, together with the per-metric SFT gap (SFT+RL minus base RL), the lift from ECHO (base RL+ECHO minus base RL), and the fraction of the SFT gap recovered by ECHO without expert demonstrations. Values are in pass-rate units (%); internal columns (val100, ITD, TBLite) are mean pass-rate, TB2 columns are pass@k.

Table 5: Full breakdown of the expert-SFT gap closed by ECHO on Qwen3-8B base. Rows are absolute pass-rates for the three matched configurations, the SFT initialization gap, the lift from adding ECHO without expert SFT, and the fraction of that gap closed.

Configuration	val100	ITD	TBLite	TB2 p@1	TB2 p@3	TB2 p@5
q8b RL	54.94	16.22	9.47	2.70	6.74	8.99
q8b RL + ECHO	63.66	18.89	11.39	5.17	10.45	13.48
q8b SFT + RL	63.52	18.79	11.63	7.64	14.38	17.98
SFT gap (SFT+RL − RL)	8.58	2.57	2.16	4.94	7.64	8.99
ECHO lift (RL+ECHO − RL)	8.72	2.67	1.92	2.47	3.71	4.49
% SFT gap closed by ECHO	**101.6%**	**103.9%**	88.9%	50.0%	48.6%	50.0%

E OT-SFT Training Curves

Figure 6 shows training curves for the OpenThinker-Agent-v1-SFT-initialized Qwen3-8B runs. Headline tables and Figure 2 use the env-only-fresh ECHO variant for consistency with the other configurations; the full-observation variant achieves a lower env CE (Section 5.2) but a slightly weaker tblite-val accuracy.

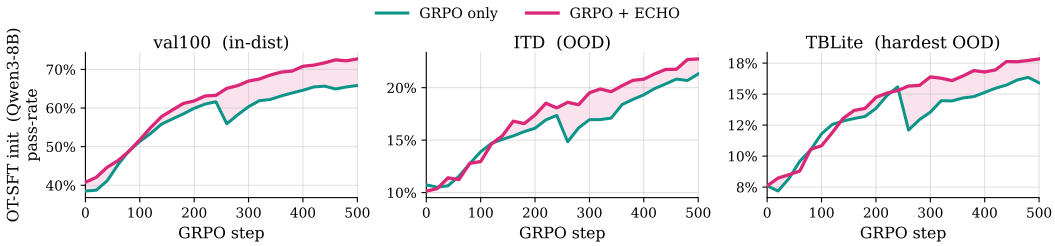

Figure 6: OT-SFT-init Qwen3-8B training curves. ECHO (env-only fresh) is the run reported in main-text Table 1, with checkpoint at step 440.

F Headline Results with Standard Errors

Table 6 reports the same configurations as main-text Table 1 with ± 1 standard error. Standard errors for TBLite are computed across the n=100 questions (8 rollouts each at T=0.6, SE = std/$\sqrt{n}$); standard errors for TB2 pass@k use the unbiased pass@k estimator on per-task success counts across the n=89 tasks.

Table 6: Headline results with ± 1 standard error. Internal columns are mean pass-rate (%) over n questions $\times$ 8 rollouts at temperature 0.6. TB2 columns are pass@k (%) over 5 attempts on 89 tasks with the Terminus-2 reference agent.

Configuration	**TBLite**	**TB2 p@1**	**TB2 p@3**	**TB2 p@5**
Qwen3-8B	4.9 ± 1.4	1.57 ± 0.80	3.71 ± 1.85	4.49 ± 2.21
Qwen3-8B RL	9.5 ± 2.1	2.70 ± 0.97	6.74 ± 2.34	8.99 ± 3.05
Qwen3-8B RL+ECHO	$\mathbf{11.4 \pm 2.3}$	$\mathbf{5.17 \pm 1.68}$	$\mathbf{10.45 \pm 2.91}$	$\mathbf{13.48 \pm 3.64}$
OT-SFT	6.0 ± 1.6	5.62 ± 1.84	10.45 ± 3.02	12.36 ± 3.51
OT-SFT RL	11.6 ± 2.4	7.64 ± 2.08	14.38 ± 3.37	17.98 ± 4.09
OT-SFT RL+ECHO	$\mathbf{12.9 \pm 2.7}$	$\mathbf{7.87 \pm 2.06}$	$\mathbf{13.82 \pm 3.43}$	$\mathbf{17.98 \pm 3.88}$
Qwen3-14B	5.7 ± 1.4	4.27 ± 1.48	8.99 ± 2.63	12.36 ± 3.51
Qwen3-14B RL	9.8 ± 2.1	5.17 ± 1.65	10.67 ± 2.95	13.48 ± 3.64
Qwen3-14B RL+ECHO	$\mathbf{15.1 \pm 2.6}$	$\mathbf{10.79 \pm 2.77}$	$\mathbf{16.52 \pm 3.71}$	$\mathbf{19.10 \pm 4.19}$

G Paired Wilcoxon Tests

To test the strongest claims supporting the main-text paragraphs, we report paired Wilcoxon signed-rank p-values on per-question (resp. per-task) pass rates. Pairing is on `question_idx` for internal evaluations (the same parquet is loaded by every configuration) and on `task_name` for TerminalBench-2.0; the metric scale is unchanged. Significance levels are uncorrected; the strongest claims (best vs. base, q14b RL+ECHO vs. q14b RL) clear $p < 0.001$ on multiple metrics and remain significant under Bonferroni at our claim count.

Single-seed limitation. All training runs use a single seed per configuration; pass@k standard errors above reflect rollout sampling and per-question / per-task variance, not training stochasticity. We do not cross-seed RL training and do not claim our point estimates are stable across re-trains.

Table 7: Paired Wilcoxon signed-rank tests for the headline claims. "Δ" is mean$(B)-$mean(A); signs are $A < B$ on metrics where higher is better. P-values are uncorrected.

Claim	Strongest evidence
Best run (Q14b RL+ECHO) > Q8b base	$p < 0.001$ across all 6 metrics ($\Delta_{\text{TB2 p@1}} = +9.21$ pp)
Best Q14b > Q14b base	$p < 0.01$ on TB2 p@1 ($\Delta = +6.52$ pp); $p < 10^{-4}$ on val100/ITD/TBLite
ECHO helps over RL (Q14b)	$p < 0.01$ on TB2 p@1 ($\Delta = +5.62$ pp); $p < 0.005$ on val100/ITD/TBLite
ECHO helps over RL (Q8b base)	$p < 10^{-4}$ on val100; marginal ($p \sim 0.09$) on TB2 p@1/3/5
RL+ECHO matches SFT+RL on TB2 (Q8b)	no significant difference on TB2 ($p > 0.7$ on p@1/3/5); RL+ECHO significantly exceeds SFT+RL on val100/ITD/TBLite ($p < 0.012$)
OT-SFT RL > q8b RL (gap claim)	$p < 0.01$ on TB2 p@1 ($\Delta = +4.94$ pp); not significant on ITD / TBLite

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state three concrete contributions—near-doubling of TerminalBench-2.0 pass@1 at 8B and 14B (verified in §5.1), recovery of the expert-SFT initialization gap (§5.3), and ECHO-only adaptation under a competence-and-predictable-dynamics scope condition (§5.5).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Justification: §6 includes an explicit Limitations paragraph discussing the main scope conditions of the method: the indirect nature of our evidence for learned terminal dynamics, the dependence of verifier-free adaptation on clean exploration and predictive feedback, and the need to keep the auxiliary loss weight small enough not to compete with task success. Additional statistical significance results and training-run limitations, including standard errors and single-seed training, are reported in App. G.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors

should reflect on how these assumptions might be violated in practice and what the implications would be.

- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper proposes an empirical training objective and reports experiments; it does not state any theorems or formal theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: §3 fully specifies the ECHO objective (Eq. 3) and Alg. 1; §4 describes the training corpus, harness, RL recipe, and evaluation protocol; App. B lists optimizer, learning rate, batch size, λ, GRPO clipping, and sampling temperatures; App. C provides per-step wallclock. The objective adds only a masked log-probability term to standard GRPO and can be reimplemented on any open RL trainer.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.

- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: Code and the synthetic task corpus are not included with the submission; we plan to release them upon de-anonymization at camera-ready. The submission supplies a complete algorithmic description (Alg. 1), full training setup (§4), and hyperparameters (App. B) sufficient to reimplement ECHO on top of any open RL trainer.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: §4 specifies the training corpus and val100/ITD/TBLite/TB2 evaluation splits, the harness and 16k context, GRPO settings, and sampling temperatures. App. B reports optimizer (AdamW), learning rate (1×10^{-6}), gradient clip, GRPO clip ranges, λ, and per-eval sampling protocol. The λ sweep that selected λ=0.05 is described in §3.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: App. F reports ±1 standard error on TBLite and TB2 pass@k for every configuration; TBLite SEs are computed across n=100 questions (8 rollouts each at T=0.6), and TB2 SEs use the unbiased pass@k estimator on per-task success counts across n=89 tasks. App. G reports paired Wilcoxon signed-rank p-values for the strongest claims, with the strongest (best 14B vs. base; q14b RL+ECHO vs. q14b RL on TB2 p@1) clearing $p<0.001$. Single-seed training stochasticity is explicitly flagged as a limitation in §6 and in App. G.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: §4 states that each run uses 8 B200 GPUs for 500 GRPO steps. App. C reports per-step wallclock minutes and 500-step cumulative hours for every (model, configuration) pair, including both RL-only and RL+ECHO at 8B base, OT-SFT, and 14B base. The world-loss adds no additional rollouts or forward passes, and is observed to be slightly faster per step than the matched RL-only baseline (App. C).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The work involves no human subjects, no scraped personal data, and no model deployment. Training tasks are synthetic terminal tasks generated programmatically (§4), and all evaluation benchmarks are publicly released or constructed in-house from synthetic specifications. The authors have reviewed the NeurIPS Code of Ethics and submission preserves anonymity.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: A short Broader Impacts paragraph appears in §6, discussing the potential of more capable terminal agents (positive: scientific computing and developer productivity; negative: dual-use risk in software-engineering automation and the importance of sandboxed evaluation).

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: We do not release pre-trained models, scraped datasets, or generative outputs at submission time. The synthetic terminal tasks are constructed from solvability-filtered task specifications and contain no personal data. Any future code or data release will follow the licenses and safeguards of the underlying base models (Qwen3, OpenThinker-Agent-v1-SFT) and the source benchmarks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All third-party assets are cited in §4: Qwen3 base models [Qwen Team, 2025], OpenThinker-Agent-v1-SFT [OpenThoughts Team, 2025a], Endless Terminals tasks [Gandhi et al., 2026a,b], OpenThoughts-Agent-v1-RL and OpenThoughts-TBLite [OpenThoughts Team, 2025b,d], the Harbor harness [Anthropic and contributors, 2025], the Terminus-2 reference agent [Terminal-Bench Team, 2025b], and TerminalBench-2.0 [Terminal-Bench Team, 2025a]. We use each within its respective license and terms of use; specific license attributions and version pins will accompany code release at camera-ready.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [No]

Justification: We construct a synthetic 8770-task corpus (8770 train + 100 val100; §4) and a held-out 928-task PyTerm split for ECHO-only adaptation (§5.5). These datasets are not released as part of this submission. The construction pipeline is described in §4 (modified Endless Terminals format with solvability filtering); we plan to release datasets at camera-ready alongside license and documentation.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper involves no crowdsourcing or human-subjects research. All training tasks are synthetic, and verifiers are programmatic unit tests run inside Docker (§4).

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper involves no human-subjects research and therefore no IRB review was required.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: GPT-5 is used as a solvability filter when generating synthetic training tasks (§4); OpenThinker-Agent-v1-SFT, one of three starting policies, was distilled from GLM-4.6 demonstrations [OpenThoughts Team, 2025a]. LLMs are not components of the ECHO objective itself or the analysis pipeline.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.