ECHO: Terminal Agents Learn World Models for Free

Vaishnavi Shrivastava* Piero Kauffmann Ahmed Awadallah Dimitris Papailiopoulos
Microsoft Research

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, files, logs, and traces—records the consequences. We argue that this stream is a supervision signal, but standard agent RL discards it: GRPO-style training updates action tokens with sparse outcome-level rewards while ignoring environment responses already in the rollout. Failed rollouts provide little policy-gradient signal despite containing rich evidence about how the environment responds. We introduce **ECHO** (Environment Cross-entropy Hybrid Objective), a hybrid objective that combines the standard policy-gradient loss on action tokens with an auxiliary loss that trains the policy to predict environment observation tokens resulting from its own actions. ECHO reuses the same forward pass as GRPO, requires no additional rollouts, and turns terminal feedback into dense supervision for all rollouts. ECHO doubles GRPO pass@1 on TerminalBench-2.0: Qwen3-8B improves from 2.70% to 5.17%, and Qwen3-14B from 5.17% to 10.79%. ECHO also produces policies that better predict terminal dynamics, even on trajectories they did not generate: across held-out rollouts, it sharply reduces environment-token cross-entropy while GRPO alone barely changes it. From base Qwen3-8B, ECHO matches expert-SFT-then-GRPO performance on held-out terminal tasks without expert demonstrations, and recovers roughly half of the expert-SFT initialization benefit on TerminalBench-2.0. In some settings, the environment prediction loss alone enables verifier-free self-improvement, allowing policies to improve on unseen OOD tasks by learning only from environment interactions. Together, these results suggest that environment observations are not merely context for future actions, but a dense, on-policy supervision signal already present in every rollout.

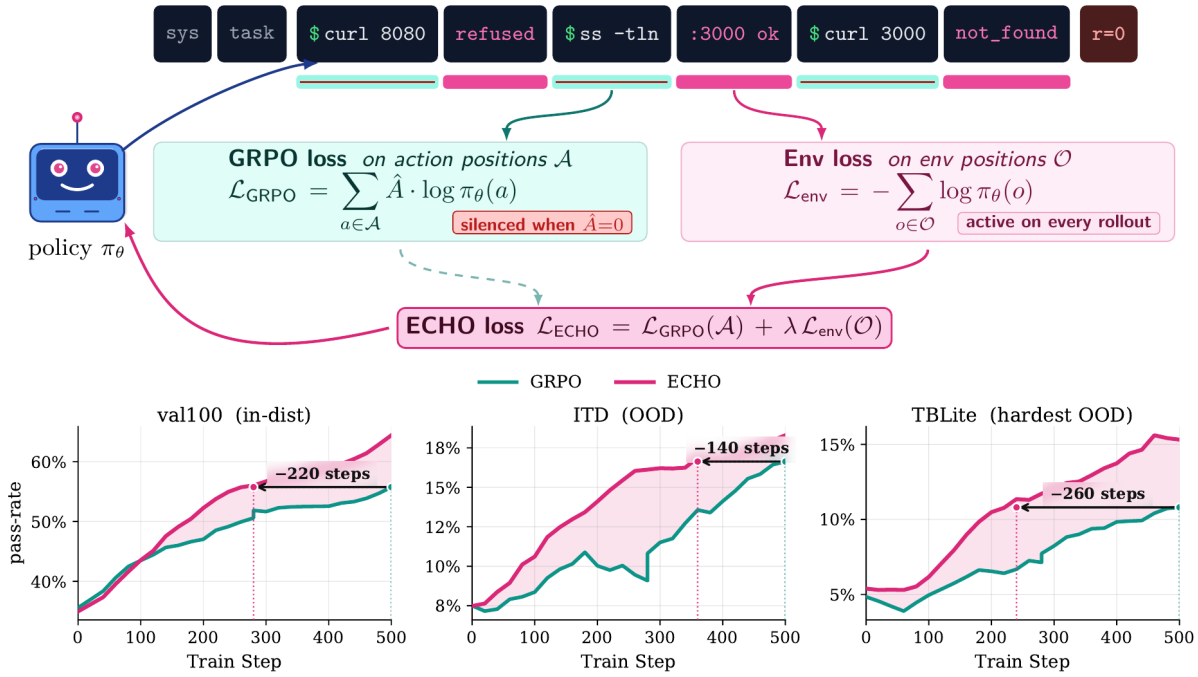

Figure 1: ECHO turns terminal feedback into supervision during agent RL. A terminal-agent rollout interleaves assistant actions, with environment observations. In standard GRPO, previous terminal outputs can inform future commands, but loss is applied only to action-token positions and is driven by sparse outcome-level rewards. ECHO adds a complementary cross-entropy loss on environment-observation tokens, training the same policy to predict the terminal responses caused by its own actions. ECHO reuses the same rollout and forward pass while making every terminal response an additional training target. Bottom: across held-out evaluations, ECHO improves task solve rate relative to matched GRPO-only runs and also reaches the peak GRPO performance substantially earlier in training.

*Correspondence to: vaishnavi.shrivastava@microsoft.com

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, often 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" (Patel, 2023). We borrow this view for agents. More precisely, terminal output is a lossy textual projection of the container state: it reveals stdout, stderr, exit codes, file contents, traces, and test failures, but not the full filesystem, process tree, or hidden task state. Predicting these observations therefore requires the policy to track the latent consequences of its commands: which files were created, which assumptions failed, which tests broke, and what state is likely to be exposed next. 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), a hybrid loss that adds auxiliary cross-entropy on environment-observation tokens to the usual GRPO loss on action tokens for multi-turn agent RL:

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

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. Because these targets come from the current policy's own rollouts, ECHO is on-policy: as the agent improves and visits new terminal states, the environment produces new responses to predict, creating a self-evolving curriculum. 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 task 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 TerminalBench-2.0, ECHO nearly doubles GRPO's pass@1 rate from 2.70% to 5.17% for Qwen3-8B and from 5.17% to 10.79% for Qwen3-14B. The same checkpoints also become substantially better predictors of terminal behavior: on held-out off-policy trajectories from Qwen3-32B, ECHO sharply lowers environment-token cross-entropy while GRPO alone barely changes it. ECHO also reduces dependence on expert demonstrations: from a base Qwen3-8B, 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 are as follows:

1. **ECHO turns terminal outputs into supervision.** We introduce an on-policy hybrid objective that treats terminal outputs — stdout, errors, files, and tool traces — as dense training targets, adding environment-token cross-entropy to GRPO's action-token loss. The two terms share one forward pass, require no teacher model, and add no extra rollouts. (§3)
2. **Consistent improvements over GRPO.** 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. (§5.1)
3. **ECHO learns terminal dynamics.** On held-out trajectories from a stronger Qwen3-32B policy, ECHO sharply lowers environment-token cross-entropy across model families and evaluation slices, indicating better prediction of terminal behavior, while GRPO alone barely changes it. (§5.2)
4. **Reduced reliance on expert SFT.** From a base Qwen3-8B, ECHO matches SFT-bootstrapped GRPO, without using any expert demonstrations. (§5.3)
5. **Verifier-free adaptation.** Even without a verifier, a policy can sometimes improve just by interacting with the environment and predicting the environment's response to its own actions. (§5.5)

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. ECHO shows that these consequences can be trained on directly, turning even failed interactions into signal for learning how the world responds.

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 $\mathcal{A}$ using the clipped importance ratio $\rho_t^{(i)}$:

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

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 hybrid loss objective combining GRPO's policy gradient loss on action tokens with a supervised next-token objective on observation tokens. Let $p_\theta(\cdot \mid x_{<t})$ denote the model's next-token distribution. The ECHO loss augments GRPO with an Environment-Prediction Loss: length-normalized cross-entropy on a subset $\mathcal{O}' \subseteq \mathcal{O}$ of observation tokens:

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

where $Z = |\mathcal{O}|$ normalizes each sequence by its total observation length. We normalize by the total observation length $|\mathcal{O}|$, rather than $|\mathcal{O}'|$, so runs with different target subsets remain comparable on a per-observation scale. ECHO is the joint objective $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{GRPO}} + \lambda \mathcal{L}_{\text{Env}}$ as in equation 1. Because the observation targets come from the current policy's own rollouts, ECHO is on-policy: as the agent improves and visits new terminal states, the environment-prediction targets evolve with the policy rather than remaining a frozen offline set of trajectories.

The two losses share a single actor forward pass: the same logits feed both, gathered through different masks (assistant-action positions for GRPO, additional observation positions for ECHO's Environment-Prediction loss). ECHO therefore requires no second rollout, teacher model, or second forward pass; the only added work is a masked log-probability sum. The intended effect is representation shaping: by learning which observations follow from which commands, the same policy network can develop better priors over which future commands are likely to expose useful state, repair errors, or advance the task. It composes orthogonally with stabilization techniques aimed at the policy gradient itself, including void-trajectory filtering (Xue et al., 2026), overlong filtering and clip-higher (Yu et al., 2025), and KL-regularized reference updates (Ouyang et al., 2022). Algorithm 1 summarizes the resulting update.

Algorithm 1 ECHO

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.

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

$\log p_t \leftarrow \log \text{softmax}(\ell_t)[x_t]$ for all t

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

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

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

Computationally, ECHO changes the loss mask rather than the rollout or model evaluation. The expensive attention and MLP computations already run over the full rollout to compute action-token log-probabilities for GRPO. ECHO simply gathers the already-computed logits at terminal-output positions and includes their cross-entropy in the same backward pass.

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

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

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 $\sim$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: environment prediction 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, $\mathcal{L}_{\text{Env}}$ 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., 2026; obiwan96, 2026) and 723 from OpenThoughts-Agent-v1-RL (OpenThoughts-Agent 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 (harbor-framework, 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.

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
	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
	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
	ECHO	**65.0**	**19.8**	**15.1**	**10.79**	**16.52**	**19.10**

Table 1: Pass rates before RL, after GRPO, and after ECHO for three starting policies. For *val100* (100 tasks), *ITD* (71 tasks), *TBLite* (OpenThoughts-TBLite, 100 tasks) we report pass@1 over 8 attempts. For *TB2* (TerminalBench-2.0, 89 tasks) we report pass@k over 5 attempts. Bold marks the best metrics in each block.

Models. We train on three starting policies: **Qwen3-8B** (Yang et al., 2025), **OpenThinker-Agent-v1-SFT** (OT-SFT) (OpenThoughts-Agent Team, 2025a), and **Qwen3-14B** (Yang et al., 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 runs, we use $\lambda = 0.05$ to scale the Environment-Prediction 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-Agent team et al., 2026), and **TerminalBench-2.0** (TB2) (Merrill et al., 2026). 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-Agent 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 (Harbor Framework Team, 2025) 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 → 5.17) and 14B (5.17 → 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× fewer training steps and waste fewer turns at inference. Starting from base Qwen3-8B, ECHO 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 ∼15k expert demonstrations the SFT model requires.

5.1 ECHO Improves Over GRPO Performance

ECHO consistently improves task success. ECHO 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 → 5.17, ×1.9) and Qwen3-14B (5.17 → 10.79, ×2.1).

Table 1 compares matched GRPO and 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.

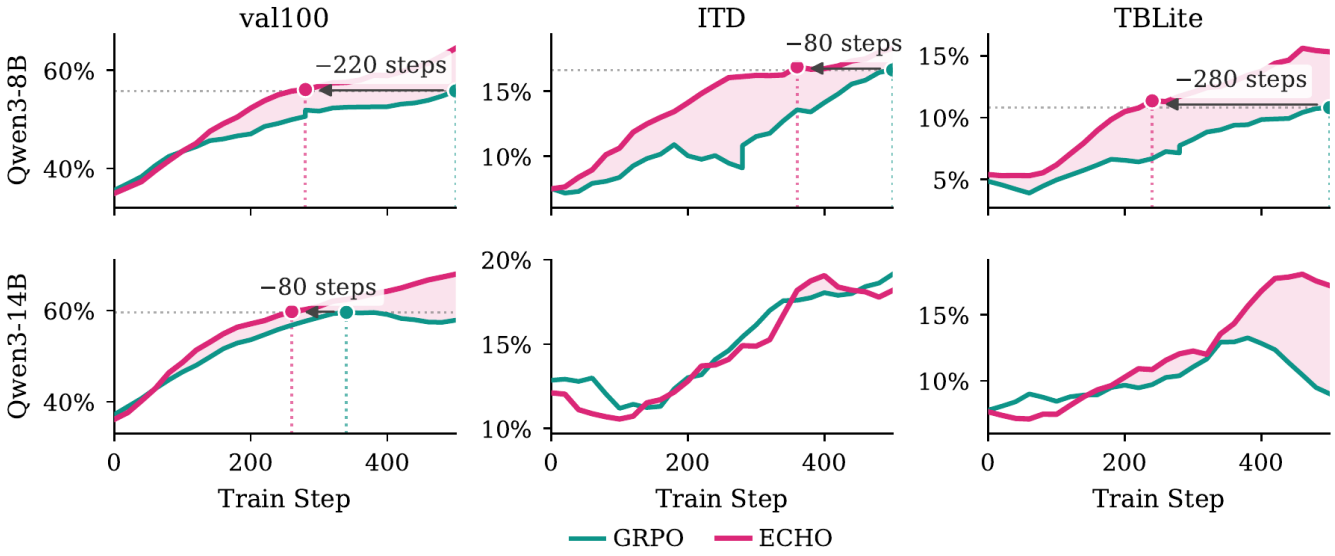

Figure 2: Pass-rate training curves over 500 GRPO steps. Teal shows GRPO only; pink shows ECHO, with shading where ECHO exceeds the matched GRPO baseline. Rows are Qwen3-8B base and Qwen3-14B base; columns are val100 (in-distribution), ITD (OOD), and TBLite (hardest OOD). OT-SFT–initialized curves appear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

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 larger model can internalize more generalizable terminal dynamics from the observation stream, while at smaller scales the policy and environment-prediction 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?

A useful terminal dynamics model should be predictive: given an action, it should be able to simulate the environment's response. We test this directly by measuring environment-token cross-entropy on held-out trajectories: the likelihood the policy assigns to the terminal-output tokens that actually follow each action. We measure how well each model predicts terminal-output tokens on off-policy trajectories generated by a stronger Qwen3-32B model. Across val100, ITD, and TBLite, we evaluate on 8 trajectories per task, totaling 2,168 trajectories.

This evaluation is intentionally off-policy: the evaluated models did not generate these trajectories themselves. Low cross-entropy therefore requires predicting the outcomes of another stronger agent's actions, rather than memorizing a model's own rollout distribution. In this sense, environment-token cross-entropy provides an operational test of the world-modeling claim: a model that has learned more about terminal dynamics should better simulate terminal responses, even on trajectories it did not generate.

ECHO learns transferable terminal dynamics. On held-out, off-policy trajectories from Qwen3-32B, 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 0.24→0.07 on val100, 0.39→0.31 on ITD, and 0.30→0.23 on TBLite; for Qwen3-8B, the corresponding drops are 0.29→0.07, 0.46→0.32, and 0.35→0.25. The larger reduction on val100 is expected: val100 is drawn from the same task distribution as training, whereas ITD and TBLite are out-of-distribution evaluations, so successful transfer requires predicting terminal behavior under less familiar task structure. These results support the central mechanism behind ECHO: the environment-prediction objective improves the policy's ability to simulate terminal responses, and this ability transfers beyond the model's own trajectories.

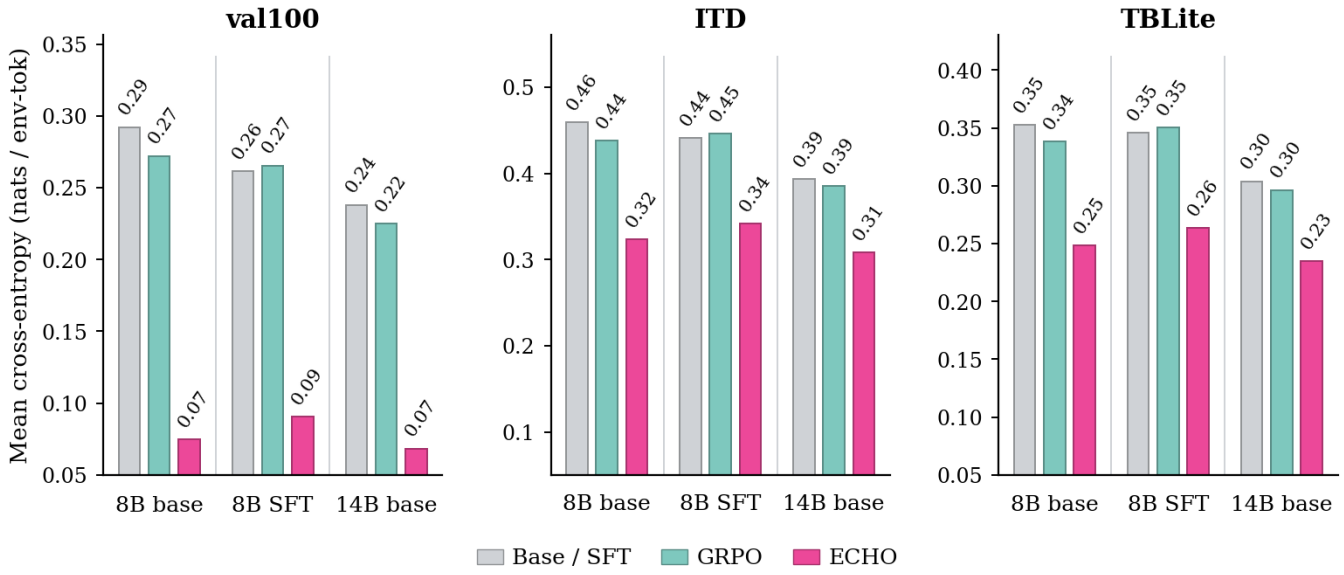

Figure 3: Per-token cross-entropy on terminal-output tokens for trajectories from a stronger model, Qwen3-32B. Each panel evaluates a different distribution; within each panel, starting policies are shown before RL, after GRPO, and after ECHO. GRPO barely changes env-token CE relative to the starting policy, whereas ECHO sharply lowers it across all model families and evaluation slices. Lower is better.

5.3 ECHO Reduces Dependence on Expert Demonstrations

Expert SFT primes terminal agents before RL by behavior-cloning 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 and learn from its own terminal interactions. 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 ECHO over 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 an interaction prior: how terminal agents inspect files, run tests, encounter errors, follow tracebacks, and expose useful state. ECHO does not imitate the expert's action choices; instead, it learns from the terminal consequences of the base model's own actions. This can recover the interaction-modeling component of expert SFT without behavior-cloning the teacher.

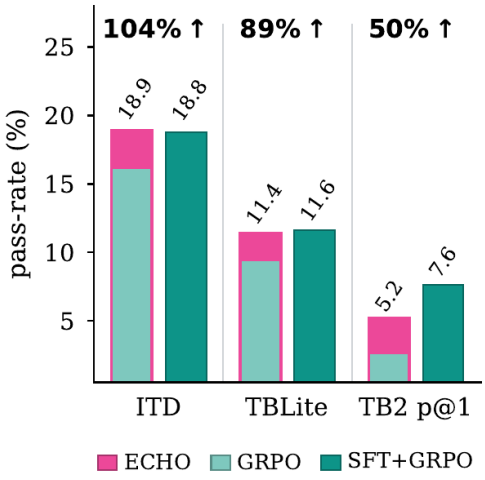

Figure 4: ECHO recovers most of the benefit of expert-SFT initialization on ITD and TBLite and roughly half on TerminalBench-2.0.

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. ECHO narrows the gap by learning the interaction prior directly from environment feedback, but expert demonstrations still help with the strategy prior: which commands to try first, how to decompose a task, when to inspect versus edit, and when to stop. Thus, ECHO does not make expert demonstrations obsolete. Rather, it suggests that one component of their value—familiarity with terminal feedback and state evolution—can be learned directly from interaction.

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

	Total			TBLite		
Model	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.

	Timeout rate (%)			Turns			Tokens		
Model	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 of 1200 seconds. *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.

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.

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 a joint objective that adds environment-observation prediction 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 ECHO checkpoint, we mask out the GRPO term and continue training for 100 steps using only $\mathcal{L}_{\text{Env}}$ on environment tokens. The model still acts in the environment, observes the terminal response, and updates only by predicting the terminal-output tokens caused by its own actions. Since there is no label indicating if a trajectory was correct or an action was good, any task improvement must arise indirectly: predicting terminal feedback must reshape the model states from which future actions are sampled.

Environment prediction can improve agents without verifier. On unseen in-distribution tasks, env-only adaptation improves val100 by +3.8pp. On harder OOD tasks, after filtering to clean tool-call trajectories, it improves PyTerm by +10.0pp and ITD by +5.2pp without any reward signal, while preserving val100 within ±1pp.

Target dist.	Rollout filter	Step	Δ target (pp)	Δ val100 (pp)
val100 (in-dist.)	none	70	+3.8	+3.8
PyTerm (OOD)	clean tool calls	100	+10.0	−0.9
ITD (OOD)	clean tool calls	100	+5.2	+0.4
TBLite (OOD)	clean tool calls	100	−3.9	−0.4

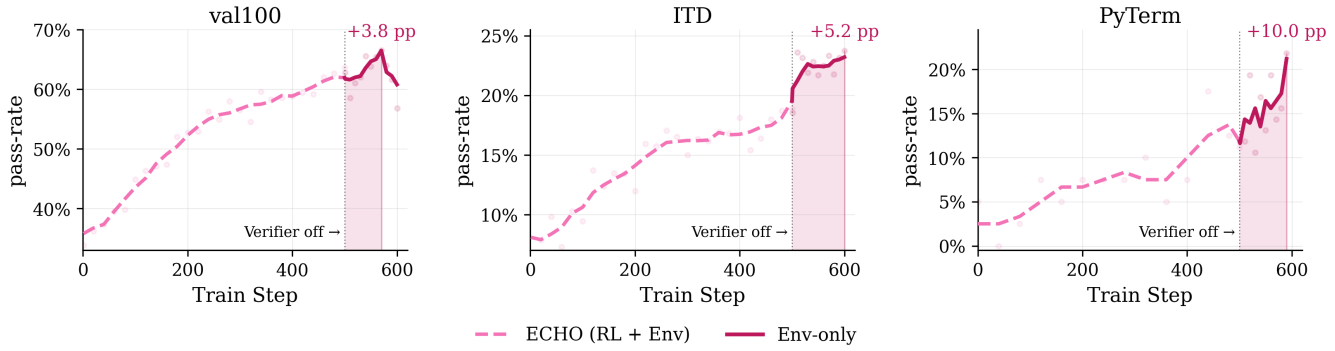

Figure 5: Verifier-free adaptation from environment prediction alone. Starting from the strongest Qwen3-8B ECHO checkpoint, we turn the verifier off at step 500 and continue training only on environment-token cross-entropy. The table reports the best post-adaptation change for each target distribution: "clean tool calls" means retaining only rollouts where every tool call is parseable and valid. Env-only adaptation improves val100 without filtering (+3.8 pp), and improves ITD (+5.2 pp) and PyTerm (+10.0 pp) after clean-rollout filtering, while preserving val100 within roughly ±1 pp during OOD adaptation. TBLite does not improve under the same recipe, likely because its feedback is less directly action-linked.

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 env-only adaptation and 100 for evaluation. The starting checkpoint solves ∼11.3% of PyTerm, comparable to its TBLite pass rate. In all cases, adaptation uses no task rewards, verifier outputs, or action-token loss.

Figure 5 shows that env-only adaptation can improve a policy without verifier rewards, but only when the interaction data provides useful prediction targets. On val100, where the checkpoint is already competent, unfiltered env-only adaptation 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. In that regime, the environment loss can become a model of failure modes rather than useful task dynamics. Filtering to clean tool-call trajectories removes this noise and yields sustained gains on PyTerm (+10.0 pp) and ITD (+5.2 pp).

Although TBLite starts at a similar pass rate to PyTerm, the same recipe degrades performance. We suspect the difference is observation structure. PyTerm provides dense, action-linked feedback: code produces tracebacks, printed values, and file contents that often point directly to what should change next. TBLite often requires broader shell orchestration over less visible filesystem, configuration, and process state, so the observed terminal tokens are less directly tied to the action choices that would solve the task. Thus verifier-free env-only adaptation works best when clean exploration exposes predictive, action-linked feedback. In those settings, a competent agent can continue improving from consequences alone: acting, observing what comes back, and updating only on the prediction loss.

6 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 (Schmidhuber, 1990; Ha and Schmidhuber, 2018; Hafner et al., 2020; 2021; 2025; Schrittwieser et al., 2020; Ye et al., 2021); recent embodied-agent work similarly uses world or action models to improve planning and control (Wang et al., 2025; Ye et al., 2026). Auxiliary-prediction methods in model-free RL train agents to predict pixels, rewards, forward dynamics, future observations, or latent states to improve representations under sparse feedback (Jaderberg et al., 2017; Pathak et al., 2017; Schwarzer et al., 2021; Kwon et al., 2024). 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 (FAIR CodeGen team et al., 2025), which trains a large model on observation–action trajectories from Python and Docker environments. Related recent work also uses agent experience or rich textual feedback to densify learning beyond scalar rewards (Zhang et al., 2025; Song et al., 2026; Hübotter et al., 2026). ECHO instead injects observation prediction directly into on-policy GRPO, requiring no separate corpus, world-modeling stage, feedback generator, dynamics model, or inference-time simulation. Some recent work also identifies the next-state/environment signal as discarded supervision. RLTF (Song et al., 2026) predicts judge-generated critiques as an auxiliary loss; OpenClaw-RL (Wang et al., 2026) recovers next-state signals either as scalar process rewards via a judge, or as token-level distillation targets via judge-extracted hints. ECHO differs from both: we predict the raw environment-observation tokens directly via an auxiliary cross-entropy loss, with no judge, no critique, and no distillation — the supervision is the environment's literal response. It is also complementary to recent multi-turn agent RL systems such as SkyRL (Cao et al., 2025), SimpleTIR (Xue et al., 2026), 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.

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 expert demonstrations, and roughly doubles TerminalBench-2.0 pass@1. Every action elicits an environment response, and every environment response is already in the agent rollout. This suggests a broader opportunity for agentic RL: between expert demonstrations and sparse outcome rewards there exists a dense training signal waiting to be used—the observable consequences of the agent's own actions.

Acknowledgments

We thank MSR AI Frontiers for supporting this work. We are especially grateful to Ahmed Ghoneim for developing the data generation pipelines and datasets that supported these experiments. We greatly appreciate Vidhisha Balachandran, Sahaj Agarwal, Abhishek Goswami, and Mojan Javaheripi for helping build the agentic RL training and evaluation stacks in our internal codebase. We also thank John Langford for helpful early discussions.

References

Shiyi Cao, Dacheng Li, Fangzhou Zhao, Shuo Yuan, Sumanth R. Hegde, Connor Chen, Charlie Ruan, Tyler Griggs, Shu Liu, Eric Tang, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. SkyRL-Agent: Efficient RL Training for Multi-turn LLM Agent, 2025. URL https://arxiv.org/abs/2511.16108.

FAIR CodeGen team, Jade Copet, Quentin Carbonneaux, Gal Cohen, Jonas Gehring, Jacob Kahn, Jannik Kossen, Felix Kreuk, Emily McMilin, Michel Meyer, Yuxiang Wei, David Zhang, Kunhao Zheng, Jordi Armengol-Estapé, Pedram Bashiri, Maximilian Beck, Pierre Chambon, Abhishek Charnalia, Chris Cummins, Juliette Decugis, Zacharias V. Fisches, François Fleuret, Fabian Gloeckle, Alex Gu, Michael Hassid, Daniel Haziza, Badr Youbi Idrissi, Christian Keller, Rahul Kindi, Hugh Leather, Gallil Maimon, Aram Markosyan, Francisco Massa, Pierre-Emmanuel Mazaré, Vegard Mella, Naila Murray, Keyur Muzumdar, Peter O'Hearn, Matteo Pagliardini, Dmitrii Pedchenko, Tal Remez, Volker Seeker, Marco Selvi, Oren Sultan, Sida Wang, Luca Wehrstedt, Ori Yoran, Lingming Zhang, Taco Cohen, Yossi Adi, and Gabriel Synnaeve. CWM: An Open-Weights LLM for Research on Code Generation with World Models, 2025. URL https://arxiv.org/abs/2510.02387.

Kanishk Gandhi, Shivam Garg, Noah D. Goodman, and Dimitris Papailiopoulos. Endless Terminals: Scaling RL Environments for Terminal Agents, 2026. URL https://arxiv.org/abs/2601.16443.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by

latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with discrete world models. In *International Conference on Learning Representations*. OpenReview.net, 2021. URL https://openreview.net/pdf?id=0oabwyZbOu.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, April 2025. doi: 10.1038/s41586-025-08744-2. URL https://doi.org/10.1038/s41586-025-08744-2.

harbor-framework. Harbor: A framework for evaluating and optimizing agents and models in container environments. Software, August 2025. URL https://github.com/harbor-framework/harbor.

Harbor Framework Team. Terminus-2: Harbor's high-performance reference agent implementation. Harbor documentation, 2025. URL https://www.harborframework.com/docs/agents/terminus-2. Accessed 2026-05-18.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement Learning via Self-Distillation, 2026. URL https://arxiv.org/abs/2601.20802.

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. URL https://openreview.net/forum?id=SJ6yPD5xg.

Jeongyeol Kwon, Liu Yang, Robert Nowak, and Josiah Hanna. An empirical study on the power of future prediction in partially observable environments, 2024. URL https://arxiv.org/abs/2402.07102.

Mike A Merrill, Alexander Glenn Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Kumar Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Kwesi Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Jenia Jitsev, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=a7Qa4CcHak.

obiwan96. Endless Terminals dataset. Hugging Face dataset, 2026. URL https://huggingface.co/datasets/obiwan96/endless-terminals. Associated with Gandhi et al., "Endless Terminals: Scaling RL Environments for Terminal Agents"; accessed 2026-05-18.

OpenThoughts-Agent Team. OpenThinker-Agent-v1-SFT. Hugging Face model card, December 2025a. URL https://huggingface.co/open-thoughts/OpenThinker-Agent-v1-SFT. Accessed 2026-05-18.

OpenThoughts-Agent Team. OpenThoughts-Agent-v1-RL. Hugging Face dataset, December 2025b. URL https://huggingface.co/datasets/open-thoughts/OpenThoughts-Agent-v1-RL. Accessed 2026-05-18.

OpenThoughts-Agent Team. OpenThoughts-TB-Dev. Hugging Face dataset, 2025c. URL https://huggingface.co/datasets/open-thoughts/OpenThoughts-TB-dev. Accessed 2026-05-18.

OpenThoughts-Agent team, Snorkel AI, and Bespoke Labs. OpenThoughts-TBLite: A high-signal benchmark for iterating on terminal agents. Hugging Face dataset, February 2026. URL https://huggingface.co/datasets/open-thoughts/OpenThoughts-TBLite. Accessed 2026-05-18.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–

27744. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

Dwarkesh Patel. 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, accessed 2026-05-18.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017. URL https://proceedings.mlr.press/v70/pathak17a.html.

Jürgen Schmidhuber. Making the world differentiable: On using self-supervised fully recurrent neural networks for dynamic reinforcement learning and planning in non-stationary environments. Technical Report FKI-126-90, Technische Universität München, 1990. URL https://people.idsia.ch/~juergen/FKI-126-90_%28revised%29bw_ocr.pdf.

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(7839):604–609, 2020. doi: 10.1038/s41586-020-03051-4. URL https://doi.org/10.1038/s41586-020-03051-4.

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. URL https://arxiv.org/abs/2007.05929.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, 2024. URL https://arxiv.org/abs/2402.03300.

Yuda Song, Lili Chen, Fahim Tajwar, Remi Munos, Deepak Pathak, J. Andrew Bagnell, Aarti Singh, and Andrea Zanette. Expanding the capabilities of reinforcement learning via text feedback, 2026. URL https://arxiv.org/abs/2602.02482.

Siyin Wang, Zhaoye Fei, Qinyuan Cheng, Shiduo Zhang, Panpan Cai, Jinlan Fu, and Xipeng Qiu. World modeling makes a better planner: Dual preference optimization for embodied task planning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21518–21537, 2025. URL https://aclanthology.org/2025.acl-long.1044/.

Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. OpenClaw-RL: Train Any Agent Simply by Talking, 2026. URL https://arxiv.org/abs/2603.10165.

Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun MA, and Bo An. SimpleTIR: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=EplNy91Xqh.

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.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering Atari games with limited data. *Advances in neural information processing systems*, 34:25476–25488, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/d5eca8dc3820cad9fe56a3bafda65ca1-Paper.pdf.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, Jian Xie, Yuxuan Sun, Boyu Gou, Qi Qi, Zihang Meng, Jianwei Yang, Ning Zhang, Xian Li, Ashish Shah, Dat Huynh, Hengduo Li, Zi Yang, Sara Cao, Lawrence Jang, Shuyan Zhou, Jiacheng Zhu, Huan Sun, Jason Weston, Yu Su, and Yifan Wu. Agent learning via early experience, 2025. URL https://arxiv.org/abs/2510.08558.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL, 2024. URL https://arxiv.org/abs/2402.19446.

Appendix

A Environment-Token Cross-Entropy Trajectories

Figure 6 shows per-token environment cross-entropy on warning tokens and on terminal-output (env) tokens over training. Warning CE drops from ∼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.

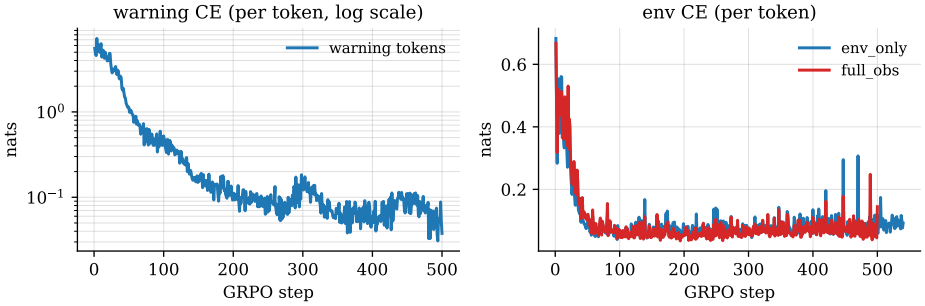

Figure 6: Per-token environment cross-entropy by target type. Warning CE drops to near-zero within ∼60 steps; env CE plateaus at 0.05–0.10 nats and provides sustained gradient throughout training.

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.
- Environment-prediction 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), ∼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 Expert-SFT Gap

Table 4 reports the absolute pass-rates for the three Qwen3-8B configurations compared in §5.3, together with the per-metric SFT gap (SFT+GRPO minus GRPO), the lift from ECHO (ECHO - GRPO), and the fraction of the SFT gap recovered by ECHO without using expert demonstrations. Values are in pass-rate units (%); internal columns (val100, ITD, TBLite) are mean pass-rate, TB2 columns are pass@k.

Configuration	val100	ITD	TBLite	TB2 p@1	TB2 p@3	TB2 p@5
Qwen3-8B GRPO	54.94	16.22	9.47	2.70	6.74	8.99
Qwen3-8B ECHO	63.66	18.89	11.39	5.17	10.45	13.48
Qwen3-8B SFT+GRPO	63.52	18.79	11.63	7.64	14.38	17.98
SFT gap (SFT+GRPO − GRPO)	8.58	2.57	2.16	4.94	7.64	8.99
ECHO lift (ECHO − GRPO)	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%**

Table 4: 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.

D OT-SFT Training Curves

Figure 7 shows training curves for OpenThinker-Agent-v1-SFT Qwen3-8B. The curves follow a similar trend as Qwen3-8B and Qwen3-14B with ECHO quickly surpassing GRPO and remaining consistently ahead during training.

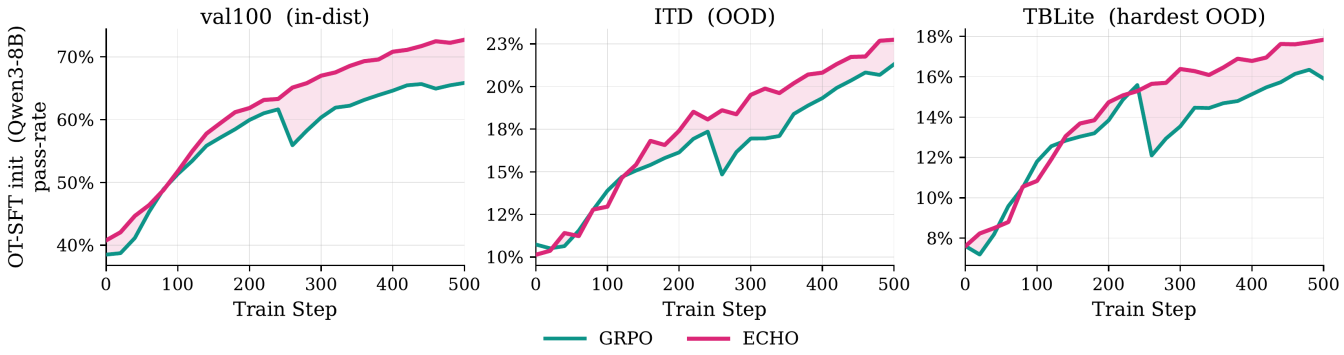

Figure 7: OT-SFT-init Qwen3-8B training curves.