DON'T MASK THE ENVIRONMENT: OBSERVATION SUPERVISION CHANGES HOW AGENTS EXPLORE UNDER RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent trajectories record what an agent does and what happens next. Yet standard supervised fine-tuning (SFT) applies loss only to agent-authored action tokens, using environment observations as context but not as prediction targets. We ask whether this convention provides the best initialization for subsequent reinforcement learning. We introduce ActObs, which also supervises the observation tokens already present in each trajectory. Although deployed agents never generate observations, learning to predict them encourages the policy to model action consequences without adding data, parameters, sequence tokens, or forward passes. The methods perform similarly after SFT but diverge after GRPO. On Qwen3-4B, GRPO from ActObs achieves higher pass@k at every evaluated sampling budget than its action-only counterpart on Terminal-Bench 2.0. On Qwen3-8B, it trades some pass@1 reliability for higher pass@k (+3.4 pp at pass@16) and solves more distinct tasks. The advantage extends to cross-domain code editing on aider-polyglot (+4.2 pp at pass@1 at 4B), whose tasks are unseen during SFT and RL. ActObs retains more entropy during RL while requiring less policy movement, leaving the final policy closer to its SFT initialization. Our analysis traces this difference to SFT: action and observation gradients rapidly become orthogonal, while action-only training leaves a large residual observation gradient and degrades environment prediction below the base model. Joint supervision prevents this one-sided specialization, preserving consequence prediction and preparing the policy for downstream exploration.

1 INTRODUCTION

A language-agent trajectory is more than a record of actions. It interleaves decisions with their realized consequences: the agent issues a command, the environment returns a new state, and the agent decides what to do next. Nevertheless, standard trajectory SFT learns from only one side of this interaction. The loss is applied to agent-authored actions, while environment observations remain in the context but are excluded from the prediction targets (Zeng et al., 2024; Chen et al., 2023; 2024). This convention appears natural because the deployed policy produces actions, not terminal output. Masking observations assumes that reading environmental feedback is useful, but learning to predict it is not. That assumption has rarely been tested, especially when SFT is only the initialization for subsequent RL.

We study the counterintuitive alternative: train the policy to predict tokens that it will never emit. As illustrated in Figure 1, ActObs simply unmasks the observations already present in each trajectory and applies the language-modeling loss to both actions and observations. At an observation position, this objective trains $p_\theta(o_t \mid h_t, a_t)$, so the model must represent what the preceding action does to the current environment. Each trajectory therefore serves as both an imitation example and a transition example. Because the two objectives share parameters, learning the action-to-consequence relation can shape the representation used to choose later actions, an intuition shared by predictive world-model approaches (Lin et al., 2024; Guo et al., 2025; Zhang et al., 2025). This additional constraint can preserve consequence modeling and discourage one-sided specialization to action imitation.

We compare ActObs with ActionSFT, the standard action-only objective, and Obs→Act, a timing control that applies observation-only SFT before action-only SFT. The three SFT checkpoints perform similarly on Terminal-Bench 2.0 but diverge after the same GRPO procedure. At 4B, GRPO

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

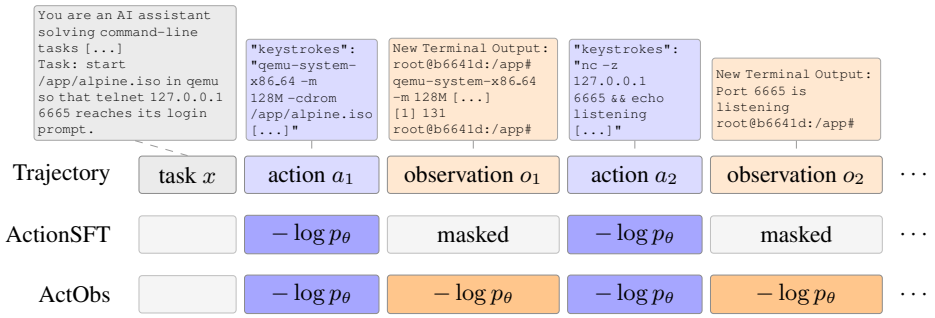

Figure 1: **ActObs learns from the complete interaction.** The top row shows a real agent trajectory (Appendix G): a task, the agent's actions, and the resulting terminal observations. Standard ActionSFT masks the observations from the objective. ActObs trains on the same sequence and exposes those already-available observation tokens to the loss.

from ActObs yields the strongest policy at every evaluated sampling budget, including a 29% relative advantage over ActionSFT at pass@1. At 8B, ActObs gives up some single-attempt reliability but gains 14% at pass@16 and solves 24 tasks rather than 21. The effect is even larger under cross-domain transfer. On aider-polyglot's 225 multilingual code-editing tasks, the 4B ActObs GRPO policy exceeds the ActionSFT GRPO policy by 43% at pass@1 and 24% at pass@4, despite starting from a weaker code-editing checkpoint before RL.

The entropy results point to a structured change in the learned policy. During GRPO, ActObs maintains higher training entropy and ends with higher self-entropy on its own evaluation rollouts, while remaining closer to its SFT initialization. This retained uncertainty can keep a wider range of actions accessible to repeated sampling, helping explain the stronger pass@k at larger k. Increasing the observation-loss weight reveals the tradeoff: stronger observation supervision gradually reduces pass@1 reliability while improving pass@k coverage. Yet raising the action-only policy's inference temperature to match the ActObs entropy does not close the gap in pass@k.

To understand why the same GRPO procedure produces different outcomes, we examine how the SFT objectives shape the initial policies for GRPO. Action and observation gradients are initially aligned but rapidly become nearly orthogonal, so action-only updates no longer approximate the observation update. ActionSFT fits the action objective while leaving a large residual observation gradient, and its ability to predict terminal feedback falls below that of the base model. ActObs continues to optimize the orthogonal component, reaching an SFT checkpoint that fits both objectives well. These distinct initializations shape subsequent GRPO training: the policy initialized from ActObs retains more entropy at positions specifying command arguments, flags, and paths, enabling repeated sampling to explore a wider range of command variants.

Our contributions are threefold:

- **Objective.** We introduce ActObs, a simple change to the SFT loss mask that turns environment observations already present in expert trajectories into a predictive target without adding data, parameters, sequence tokens, forward passes, or changes to the RL algorithm.
- **Results.** Despite similar SFT performance, ActObs delivers stronger pass@k performance and solves more distinct tasks after GRPO on Terminal-Bench 2.0, while transferring strongly to a cross-domain multilingual code-editing benchmark, showing that SFT objectives shape downstream learning and exploration beyond immediate performance.
- **Mechanism.** We show that observation supervision changes the SFT solution and subsequent RL dynamics: it prevents one-sided gradient specialization, preserves environment prediction, and allows GRPO to retain more entropy with less policy movement.

2 ACTOBS: LEARNING FROM THE FULL AGENT TRAJECTORY

Every successful agent trajectory contains two aligned sources of supervision: what the expert did and what the environment did next. Standard agent SFT trains on the first while discarding the

second as a target, even though both streams are already present in the training sequence. ActObs recovers this free signal with a single change to the loss mask. The data, model, context, and training procedure remain fixed; only the tokens included in the training loss are changed.

Objective. Let a trajectory be $\tau = (x, a_1, o_1, a_2, o_2, \ldots, a_T, o_T)$, where x is the task prompt, a_t is an assistant turn containing reasoning and commands, and o_t is the response produced by the environment. In our setting, observations are terminal outputs inserted verbatim into the next context. For the tokenized trajectory $y_{1:N}$, let $\mathcal{A}$ and $\mathcal{O}$ denote the action-token and observation-token indices. Standard agent SFT optimizes only $\mathcal{A}$. ActObs instead minimizes (Figure 1)

$$\mathcal{L}_\lambda(\theta) = -\frac{\sum_{i\in\mathcal{A}} \log p_\theta(y_i \mid y_{<i}) + \lambda \sum_{i\in\mathcal{O}} \log p_\theta(y_i \mid y_{<i})}{|\mathcal{A}| + \lambda|\mathcal{O}|}. \tag{1}$$

$\lambda = 0$ recovers action-only SFT (ActionSFT), while the default ActObs uses $\lambda = 1$. Prompts remain masked in both objectives. The denominator preserves the per-example loss scale as λ changes, and at $\lambda = 1$ action and observation tokens receive equal weight. Because observations are already processed as context, ActObs requires no new demonstrations, rollouts, parameters, or forward passes. In implementation, it is only a label-mask change.

Consequence modeling. Although the deployed agent never generates observations, they remain a valuable learning signal. Predicting o_t requires anticipating what a_t will do to the environment. ActObs therefore turns each trajectory into both a policy example and a transition example, training the same representation that the next action must use. Action-only SFT can sharpen the single teacher command while overwriting pretrained knowledge of action consequences, leaving useful alternatives with little probability. ActObs fits decisions and outcomes jointly, preserving task-relevant probability mass for subsequent RL. Appendix B reports controls for sequential, observation-only, and shuffled-observation supervision.

3 Main Results

3.1 Setup

Training. All methods share one corpus: 50k multi-turn terminal trajectories (0.71B tokens) from the synthetic portion of Nemotron-Terminal-Corpus (Pi et al., 2026), generated by DeepSeek-V3.2. Observation tokens are about 45% of the corpus. We fine-tune Qwen3-4B and Qwen3-8B (Team, 2025) for one epoch (781 steps, batch size 64, cosine schedule, peak learning rate 10^{-5}). The primary methods differ only in their loss masks. We then apply GRPO (Shao et al., 2024) for 135 steps on 2,392 containerized tasks from Endless Terminals (Gandhi et al., 2026), disjoint from the SFT corpus, using binary verifiers (16 rollouts per task, batch size 32, learning rate 10^{-6}). Unless otherwise noted, observation supervision is used only during SFT; standard GRPO contains no observation-prediction loss. ECHO (Shrivastava et al., 2026) adds a next-observation loss of weight 0.05 on the policy's own rollouts during GRPO.

Methods. ActionSFT applies loss to actions only. ActObs uses $\lambda = 1$. Obs→Act is a timing control: one epoch of observation-only SFT followed by one epoch of action-only SFT, inspired by work that develops predictive environment models before downstream policy optimization (Zhang et al., 2025; Li et al., 2026b). It receives the same amount of observation supervision as ActObs over two epochs.

Evaluation. All evaluation tasks are disjoint from the SFT corpus and the Endless Terminals RL set. The primary benchmark is Terminal-Bench 2.0 (Merrill et al., 2026), an OOD terminal benchmark with 89 tasks, the terminus-2 reference agent, official wall-clock limits, and one pinned serving configuration for all methods. Headline checkpoints are evaluated with 16 attempts per task. Model failures score zero; infrastructure failures are rerun once. pass@k uses the unbiased estimator of Chen et al. (2021). Uncertainties are one bootstrap standard error, computed by holding the task set fixed and resampling attempts within each task. Appendix A details the serving controls and statistical procedure. Aider-polyglot tests cross-domain generalization on 225 unseen code-editing tasks spanning six programming languages.

Table 1: **Main results.** pass@k in percent on Terminal-Bench 2.0 (89 tasks; 16 attempts per task) and aider-polyglot (225 code-editing tasks; four attempts per task). ECHO combines GRPO with a next-observation prediction loss. Uncertainty is one bootstrap standard error, holding the task set fixed and resampling attempts within each task. Boldface indicates the highest score within each stage.

	Qwen3-4B						Qwen3-8B					
	Terminal-Bench 2.0				aider-polyglot		Terminal-Bench 2.0				aider-polyglot	
Model	@1	@4	@8	@16	@1	@4	@1	@4	@8	@16	@1	@4
After SFT												
Base	1.2±0.2	3.3±0.5	4.7±0.6	5.6±0.7	**3.0**±0.4	**7.6**±0.7	2.7±0.4	7.0±0.6	9.3±0.8	11.2±1.0	2.4±0.4	7.6±0.8
ActionSFT	**4.5**±0.4	**10.4**±0.8	14.0±1.0	16.9±1.2	1.4±0.3	5.8±0.7	**9.2**±0.5	**18.2**±0.8	**21.9**±1.0	**24.7**±1.2	3.9±0.5	10.2±0.8
ActObs	4.4±0.4	9.9±0.9	14.0±1.1	**18.0**±1.4	1.0±0.3	3.6±0.6	7.9±0.5	16.1±0.8	19.7±0.9	22.5±1.2	**4.4**±0.5	**12.0**±0.9
Obs→Act	4.4±0.4	10.1±0.9	**14.2**±1.1	**18.0**±1.4	1.9±0.4	6.7±0.8	8.3±0.5	15.9±0.8	19.8±1.1	**24.7**±1.6	3.2±0.5	11.6±1.0
After GRPO												
ActionSFT→GRPO	5.6±0.4	11.1±0.7	14.0±1.0	18.0±1.4	9.7±0.6	20.4±0.9	**12.3**±0.5	18.7±0.6	21.0±0.8	23.6±1.2	12.7±0.8	28.4±1.3
ActObs→GRPO	**7.2**±0.4	**12.5**±0.7	**15.5**±1.0	**19.1**±1.4	**13.9**±0.7	**25.3**±1.0	11.0±0.5	**19.3**±0.9	**23.6**±1.1	**27.0**±1.3	**14.2**±0.8	**28.9**±1.2
Obs→Act→GRPO	5.3±0.4	11.4±0.8	14.7±0.9	16.9±1.0	10.0±0.7	21.3±0.9	11.9±0.5	**19.3**±0.7	21.7±0.7	23.6±1.0	12.6±0.7	24.4±1.1
After ECHO												
ActionSFT→ECHO	**5.9**±0.4	11.6±0.8	14.7±1.0	18.0±1.3	**10.2**±0.7	20.9±0.9	9.6±0.5	16.5±0.9	20.6±1.2	**25.8**±1.6	**14.8**±0.8	**29.8**±1.4
ActObs→ECHO	**5.9**±0.4	**12.7**±0.8	**16.5**±1.1	**20.2**±1.4	9.1±0.6	**21.8**±1.0	**11.4**±0.5	**18.6**±0.8	**22.0**±0.9	24.7±1.2	14.2±0.7	28.9±1.3

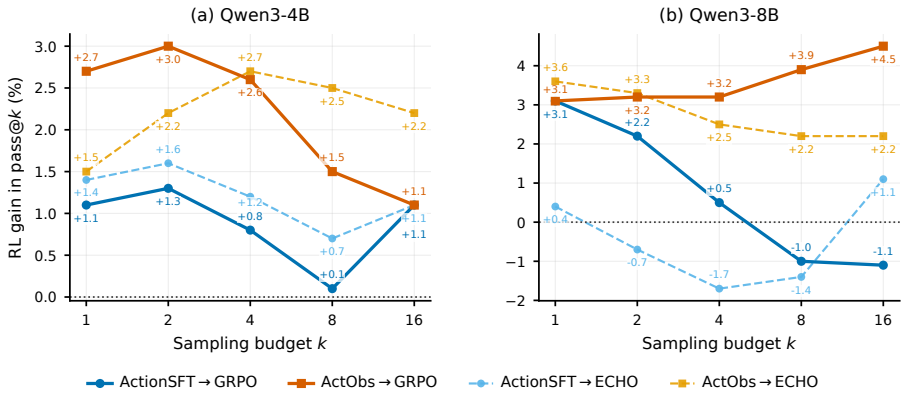

Figure 2: **Performance gain during RL from each SFT initialization.** Each curve reports Δpass@k, the post-RL score minus the pre-RL SFT score for the same initialization. Solid curves use standard GRPO. Dashed curves use ECHO. Positive values indicate improvement during RL.

3.2 Observation supervision strengthens downstream RL

Table 1 summarizes our main results. After SFT, the three initializations remain closely matched. After the same GRPO training, they separate. Figure 2 reports the performance gain during RL. Standard GRPO extracts a larger gain from ActObs at every k at 4B; at 8B, the RL gain from ActObs grows with k, whereas the gain from ActionSFT becomes negative at large k.

At 4B, ActObs produces the strongest post-GRPO policy across the entire sampling curve. It leads ActionSFT→GRPO at pass@1, pass@4, pass@8, and pass@16 by 1.6, 1.4, 1.5, and 1.1 percentage points, corresponding to relative advantages of 29%, 13%, 11%, and 6%. The effect therefore appears in single-attempt reliability and persists as the sampling budget grows. At 8B, the advantage emerges at larger sampling budgets. ActionSFT leads at pass@1, but ActObs overtakes it by pass@4, and its advantage grows from 0.6 points at pass@4 to 2.6 points at pass@8 and 3.4 points at pass@16, relative margins of 3%, 12%, and 14%. ActObs reaches a task coverage of 24, three tasks more than ActionSFT. It also solves three tasks that no SFT policy or ActionSFT→GRPO can solve, showing that RL from the ActObs initialization reaches beyond the SFT-solvable set. The sequential Obs→Act control does not realize this benefit despite receiving the same amount of observation supervision. At pass@16, ActObs leads the sequential control by 2.2 points at 4B and 3.4 points at 8B. Joint action-observation learning, rather than observation exposure alone, creates the stronger RL initialization.

The advantage also survives when RL itself becomes observation-aware. ECHO adds next-observation prediction during RL, yet ActObs leads or ties ActionSFT at seven of the eight reported operating points. Relative to ActionSFT→ECHO, the ActObs lead ranges from 1.1 to 2.2 points between pass@4 and pass@16 at 4B and from 1.4 to 2.1 points through pass@8 at 8B, relative margins as large as 12% and 19%, respectively. Only the 8B pass@16 comparison favors ActionSFT. Observation-aware RL therefore complements rather than replaces the representation established by ActObs during SFT, making ActObs a stronger initialization for ECHO than ActionSFT.

3.3 Cross-domain transfer to code-editing tasks

We evaluate aider-polyglot as a cross-domain benchmark: none of its 225 code-editing tasks appears in either the terminal SFT corpus or the Endless Terminals RL set. Its six programming languages and single-file editing objectives, evaluated by unit tests, define a task family that is distinct from the environments used for training. ActObs produces the strongest plain-GRPO policy at both model scales. At 4B, it outperforms ActionSFT→GRPO by 4.2 percentage points at pass@1 and 4.9 points at pass@4, relative advantages of 43% and 24%. It also exceeds Obs→Act→GRPO by 3.9 and 4.0 points. This advantage cannot be attributed to a stronger code-editing policy before RL: the 4B ActObs SFT checkpoint trails both alternatives on aider-polyglot. GRPO instead raises ActObs by 12.9 points at pass@1 and 21.7 points at pass@4. At 8B, ActObs leads ActionSFT→GRPO by 1.5 points at pass@1, a 12% relative advantage, and retains a 0.5-point lead at pass@4. Among the plain-GRPO policies, it therefore has the strongest pass@1 and pass@4. Taken together, Terminal-Bench 2.0 and aider-polyglot show that observation-aware SFT can strengthen downstream RL on unseen terminal tasks and under cross-domain transfer to multilingual code editing.

4 ActObs preserves exploration during RL

4.1 Training dynamics during GRPO

Figure 3 follows the quantities recorded online during GRPO. The entropy in Figure 3(b) is *training entropy*: the token-level policy entropy on the current on-policy rollout batch at each update. The 4B runs begin with similar reward and training entropy, but their trajectories separate. ActObs reward rises more smoothly and finishes higher, while its training entropy continues to rise late in RL as ActionSFT entropy falls and only partially recovers. Rollout length separates as well (Figure 3(c)). After the early phase, ActObs rollouts become progressively shorter and remain shorter at the end, whereas ActionSFT rollout length turns upward late in training. Thus, the higher late-stage entropy of ActObs does not come from generating longer trajectories.

The same qualitative entropy separation appears at 8B: ActObs maintains higher training entropy after the curves diverge, even though the runs converge in training reward (Figure 8). Across scales, the initialization changes how the policy evolves under the same GRPO objective rather than simply providing more reward headroom.

4.2 Self-entropy at the final checkpoints

The vertical axis of Figure 4(a) reports *self-entropy*. To compute it, we freeze each RL endpoint, select 200 rollouts from that policy's Terminal-Bench 2.0 evaluation, and average its next-token entropy over the assistant tokens. Self-entropy therefore measures the final policy's uncertainty on states that the policy itself visits. This differs from the *training entropy* in Figure 3(b), which is recorded on the newly sampled on-policy training batch at every GRPO update. Appendix D also reports *fixed-state entropy*: every checkpoint scores the same 200 rollout traces, holding the inputs fixed to control for state visitation.

ActObs retains more post-RL self-entropy than ActionSFT while reaching higher pass@16 (Figure 4(a)). The fixed-state probe in Appendix D shows the same ordering. ActObs therefore leaves non-negligible probability on a wider set of plausible command continuations. With multiple rollouts, these commands are more likely to be sampled, raising pass@k as k grows.

Higher entropy by itself is not sufficient. ECHO produces the highest endpoint entropy but does not match ActObs→GRPO at pass@16. We also raise the sampling temperature of ActionSFT→GRPO

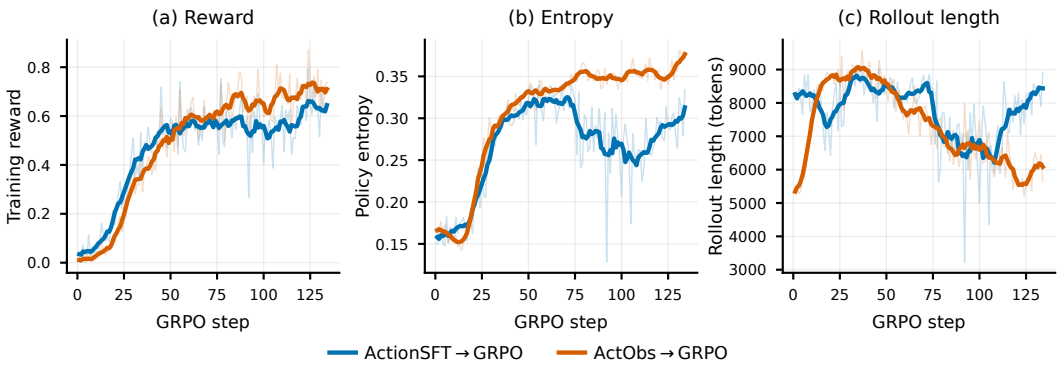

Figure 3: **GRPO training dynamics at 4B.** (a) Training reward, (b) on-policy training entropy, and (c) rollout length. The two runs begin with similar reward and entropy, then diverge under the same GRPO procedure.

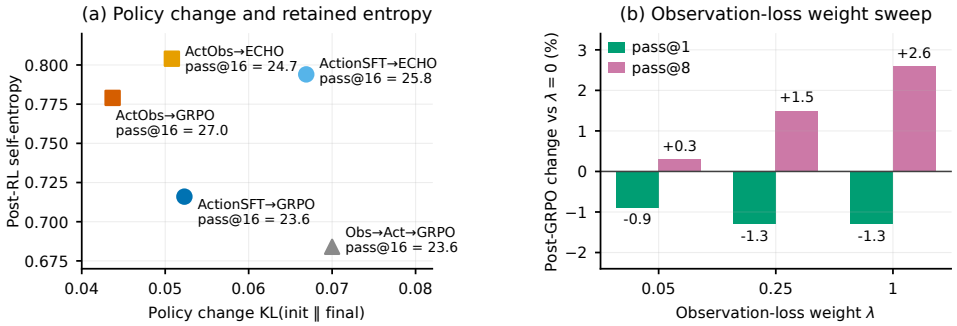

Figure 4: **Observation supervision controls the post-RL policy at 8B.** (a) Endpoint KL versus self-entropy for each post-RL policy. KL is measured from its SFT initialization on 200 shared Terminal-Bench 2.0 traces, self-entropy on 200 of the policy's own Terminal-Bench 2.0 evaluation rollouts, and labels report pass@16. (b) Post-GRPO pass@1 and pass@8 relative to the action-only endpoint ($\lambda = 0$) as the observation-loss weight increases. Stronger observation supervision progressively shifts performance from pass@1 toward pass@8.

until its self-entropy matches that of ActObs→GRPO under the same self-entropy probe used above ($T^* = 0.64$ instead of 0.6). Across 16 attempts, temperature matching changes pass@1, pass@4, pass@8, and pass@16 by at most 0.7 points, solves no additional task, and leaves the pass@16 gap intact. The high-k advantage therefore comes from the distribution learned by the policy, not from injecting more randomness at inference time.

4.3 Less policy movement

The horizontal axis of Figure 4(a) compares each final RL policy with the SFT checkpoint from which it was initialized. For every pair, we score both policies at action-token positions on the same fixed set of 200 Terminal-Bench 2.0 traces and average $\mathrm{KL}(\pi_{\text{init}} \,\|\, \pi_{\text{final}})$. The states are therefore identical across all pairs, and the value measures endpoint displacement. ActObs undergoes the smallest endpoint change, retains more self-entropy, and attains the strongest pass@16. ActionSFT moves farther and retains less entropy, while Obs→Act moves farthest and retains the least.

This joint ordering explains how smaller policy movement can preserve higher entropy. Because the ActObs endpoint remains closer to its own initialization, GRPO reallocates less probability mass and leaves more of the starting distribution within reach of repeated sampling. Prior work similarly finds that RLVR can improve pass@1 by concentrating probability on rewarded paths while reducing the set of problems solved at large k (Yue et al., 2025; Wu et al., 2025). ActObs limits this contraction, preserving more of the initial policy support for repeated sampling. ECHO marks the boundary of this explanation: it produces still higher entropy but does not match ActObs at pass@16. Therefore, high entropy alone is not enough. ActObs combines limited policy movement with high retained entropy and higher pass@k at larger k.

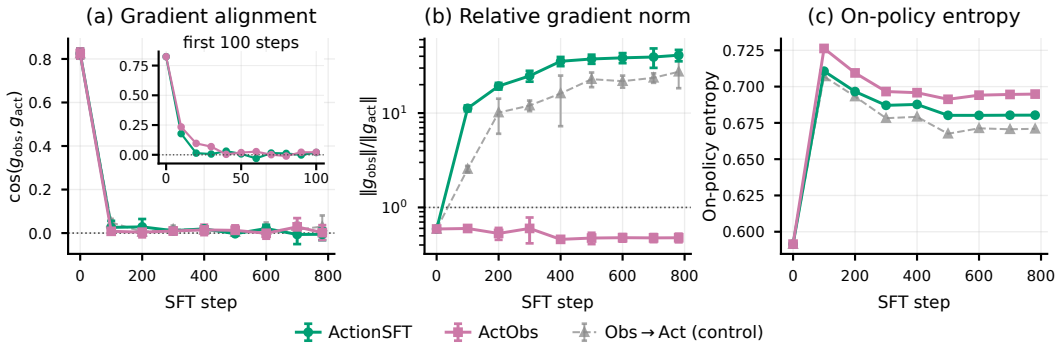

Figure 5: **Joint supervision prevents one-sided gradient specialization.** At each saved SFT checkpoint, action and observation gradients are computed separately on the same 256 held-out trajectories. Error bars show standard deviation across four equal partitions. (a) Cosine similarity c starts at 0.83 and falls to the empirical noise floor within 10 to 20 SFT steps. (b) Norm ratio r grows to about 41 under ActionSFT but stays near 0.5 under ActObs. (c) On-policy entropy measured on 200 Terminal-Bench 2.0 evaluation rollouts per method. The separation appears by step 100 and persists through the rest of SFT.

4.4 Varying observation supervision strength

Recall that λ in Eq. 1 controls the weight on observation-token prediction: $\lambda = 0$ is action-only SFT, and $\lambda = 1$ is the default ActObs objective. Figure 4(b) varies this weight while holding the data and downstream GRPO recipe fixed. As λ increases, the post-GRPO pass@8 advantage over action-only SFT grows monotonically, while pass@1 moves in the opposite direction. The sweep therefore reveals a pass@1-to-pass@8 tradeoff rather than a uniform shift in performance. Lower observation weight favors success from a single rollout; higher observation weight sacrifices some pass@1 but increases the chance that at least one of several rollouts succeeds. Because the SFT checkpoints remain similar in benchmark performance, λ controls how the initialization allocates probability mass for later RL, with its benefit emerging at larger sampling budgets.

5 Why joint supervision is a stronger initialization

5.1 Gradient dynamics during SFT

The observation loss cannot directly constrain standard GRPO because it is absent from the RL objective. Its effect is instead carried by the SFT checkpoint delivered to GRPO. Action-only SFT fits the demonstrated decisions while allowing the model's prediction of their environmental consequences to deteriorate. Joint supervision keeps both signals active through the SFT-to-RL handoff and therefore presents RL with a different policy to refine. We measure how this separation develops using checkpoints saved throughout the ActionSFT and ActObs SFT runs. At every checkpoint, we separately compute the action-token and observation-token gradients on the same 256 held-out trajectories. Thus both gradients are measured for both methods. For ActionSFT, the observation gradient is a diagnostic only; it was masked during training and is never applied to the checkpoints.

Let g_{act} and g_{obs} denote the summed action-token and observation-token gradients. We use the two quantities plotted in Figure 5 throughout: their cosine similarity $c = \cos(g_{\text{obs}}, g_{\text{act}})$ and norm ratio $r = \|g_{\text{obs}}\|/\|g_{\text{act}}\|$. Let $\hat{g}_{\text{act}} = g_{\text{act}}/\|g_{\text{act}}\|$ be the unit vector along the action-gradient direction, and let $\hat{g}^{\perp}_{\text{act}}$ be the unit vector along the component of g_{obs} orthogonal to g_{act}. We can then write

$$g_{\text{obs}} = r\|g_{\text{act}}\| \left(c\hat{g}_{\text{act}} + \sqrt{1 - c^2}\, \hat{g}^{\perp}_{\text{act}} \right). \tag{2}$$

This form maps directly onto the figure. Figure 5(a) reports c, which controls how the observation gradient divides between the parallel and orthogonal directions. Figure 5(b) reports its overall scale r. When $c \approx 0$, the parallel fraction disappears, $\sqrt{1 - c^2} \approx 1$, and r directly approximates the size of the orthogonal observation component relative to the action gradient.

At the pretrained checkpoint in Figure 5(a), the two token streams initially favor a largely shared update. Within 10 to 20 SFT steps, c falls to the noise floor and remains there. The parallel fraction

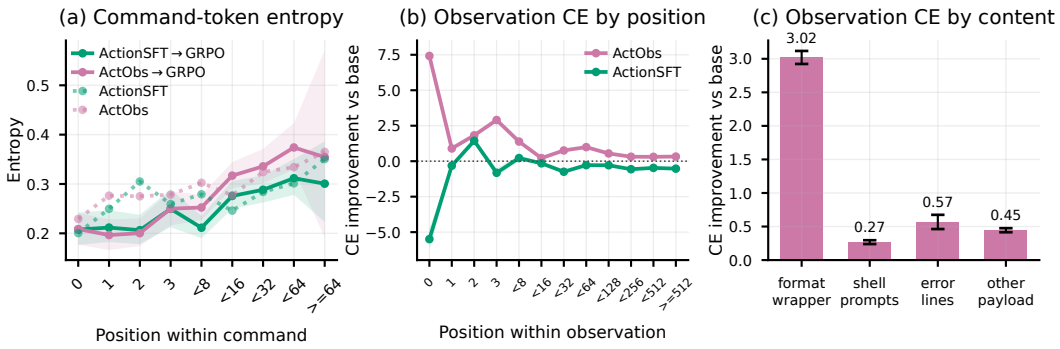

Figure 6: **Where observation supervision changes the policy and the prediction target.** (a) Command-token entropy by position on 200 of each checkpoint's own Terminal-Bench 2.0 evaluation rollouts (solid: RL checkpoints; dotted: SFT checkpoints; shading: bootstrap 95% intervals over traces). After GRPO, ActObs and ActionSFT are similarly sharp at the first command tokens, but ActObs retains more entropy later in the command. (b,c) Teacher-forced observation cross-entropy on a shared set of 300 held-out trajectories from the Nemotron-Terminal-Corpus validation split, reported as improvement over the base model. (b) groups tokens by position, and (c) groups tokens by content type.

of the observation gradient has therefore disappeared, leaving an almost entirely orthogonal signal. Action-only training cannot supply the orthogonal direction, while ActObs continues to receive this component through its observation loss. Figure 5(b) shows the magnitude of that orthogonal signal. Under ActionSFT, r rises to about 41 because fitting the demonstrated actions makes g_{act} small while the unfitted g_{obs} remains large. Under ActObs, both signals are optimized and r stays near 0.5 through the final checkpoint. This orthogonal update is what prevents one-sided specialization: after the shared direction disappears, ActObs continues to reduce the observation residual along a direction that action-only training never follows. The two objectives therefore end SFT in different regions of the joint loss landscape. ActionSFT is nearly stationary for the action loss but remains steep in an orthogonal observation direction; ActObs keeps the two signals balanced. Functionally, ActObs preserves the base model's observation prediction, while action-only fitting erodes it.

Figure 5(c) tracks on-policy entropy across SFT checkpoints, measured on 200 Terminal-Bench 2.0 evaluation rollouts per method. The entropy curves separate by step 100 of 781 and remain apart through the end of SFT. Joint supervision thus establishes the policy difference early and maintains it after the initial gradient alignment has disappeared.

5.2 Token-level effects

Every terminal observation in the SFT data begins with the fixed wrapper "New Terminal Output:", whose short repeated prefix occupies the first few observation positions. This regular prefix is an unusually easy prediction target and must be distinguished from the variable terminal payload. The scalar entropy measurements in Section 4.2 do not reveal where the additional uncertainty lies. Figure 6(a) shows that the difference is highly structured. We freeze each SFT or RL checkpoint, score its next-token distributions on 200 of its own Terminal-Bench 2.0 evaluation rollouts, and bucket command-token entropy by position. After GRPO, the first few command tokens are similarly sharp under ActObs and ActionSFT, and the separation opens later in the command, where tokens commonly encode arguments, flags, paths, and other choices. ActObs therefore does not simply make the agent less decisive at the beginning of a tool call. It preserves more variation in how a selected command is instantiated. A representative ActObs-only success uses this freedom to sweep QEMU launch parameters one at a time until it finds a working memory configuration (Appendix G).

Figures 6(b) and 6(c) locate the training signal that accompanies this policy difference. Here the base model, ActionSFT, and ActObs are teacher-forced on a shared set of 300 held-out Nemotron-Terminal-Corpus validation trajectories, and each ground-truth observation token is scored after its preceding trajectory context. The plotted quantity is the reduction in cross-entropy relative to the base model, so positive values mean better prediction. As expected, the fixed wrapper produces the largest early cross-entropy gain, but it is not the whole effect: ActObs also improves at later positions and on shell prompts, error lines, and other payload text, whereas ActionSFT is worse than

the base model across most of the sequence. Observation supervision therefore preserves prediction of environmental feedback following actions beyond memorizing the response header. By contrast, ActionSFT degrades observation prediction below the base model, suggesting that its weaker downstream RL performance may partly stem from losing this capacity.

6 Related work

Trajectory supervision and agent RL. Language-agent SFT distills multi-turn interaction traces from human or stronger-model demonstrations (Zeng et al., 2024; Chen et al., 2023; 2024; Pan et al., 2025). Although these traces interleave agent actions with environment feedback, representative pipelines compute loss only on assistant outputs, leaving observations as context rather than prediction targets (Zeng et al., 2024; Chen et al., 2024). Prior work mainly improves how trajectories are collected, filtered, and formatted, including at the scale of terminal-agent corpora (Pi et al., 2026). ActObs targets this overlooked choice: holding the teacher, trajectories, and token sequence fixed, it extends supervision from action tokens to both action and observation tokens. Agent RL instead learns directly from interaction rewards, with recent work improving optimization for multi-turn agents and building scalable environments with verifiable feedback (Wang et al., 2025b; Da et al., 2025; Gandhi et al., 2026; Ivison et al., 2026). We keep the GRPO recipe unchanged and isolate the initialization supplied by SFT.

World modeling for language agents. Predictive objectives and world models have long supported representation learning and imagined control (Jaderberg et al., 2017; Pathak et al., 2017; Ha & Schmidhuber, 2018). For language agents, predicted outcomes guide planning (Lin et al., 2024; Qiao et al., 2024; Gu et al., 2025; Chae et al., 2025) or self-verification (Guo et al., 2025). During RL, ECHO (Shrivastava et al., 2026), PaW (Lu et al., 2026), and TAPO (Li et al., 2026a) reuse on-policy transitions for observation supervision, while RWML rewards agreement between predicted and realized states (Yu et al., 2026); AAWM uses agent-authored, decision-oriented targets instead of literal next observations (Cai et al., 2026). Closest to our setting, Early Experience collects new transitions before sequential world-model SFT, imitation, and GRPO (Zhang et al., 2025); SPA inserts explicit state descriptions through self-play (Chen et al., 2025a); and From Word to World (Li et al., 2026b) and Qwen-AgentWorld (Zuo et al., 2026) use world-model training as a separate policy warm-up. ActObs requires none of these extra transitions, targets, stages, or inference-time simulators. It only unmasks raw environment observations already present in the same expert traces.

Initialization and exploration in RLVR. SFT can stabilize the behavior needed for RL, yet excessive SFT can reduce plasticity at the handoff (DeepSeek-AI, 2025; Chu et al., 2025; Liu et al., 2026). Exploration-aware fine-tuning and posterior behavioral cloning further show that similar pre-RL performance can conceal different initial policy distributions or action support (Mu et al., 2026; Wagenmaker et al., 2025). During RLVR, entropy often falls as reward rises, and higher pass@1 can come with fewer rare solutions under repeated sampling (Cui et al., 2025; Yu et al., 2025; Yue et al., 2025; Wu et al., 2025); ProRL shows that prolonged, KL-controlled training can instead expand this boundary (Liu et al., 2025). Existing methods mainly intervene during RL through clipping, entropy-aware updates, or pass@k objectives (Yu et al., 2025; Wang et al., 2025a; Cheng et al., 2026; Chen et al., 2025b). ActObs intervenes earlier by changing the checkpoint handed to otherwise unchanged RL.

7 Conclusion

ActObs learns from both the teacher's actions and the environment's responses, using information already present in every trajectory. Predicting the feedback that follows an action encourages the shared model to represent how that action changes the environment, shaping the policy that later chooses actions. Despite similar benchmark performance after SFT, ActObs provides a stronger starting point for GRPO. At 4B, ActObs achieves higher pass@k at every evaluated sampling budget after GRPO. At 8B, it trades some pass@1 reliability for higher pass@k. It also transfers strongly to a cross-domain multilingual code-editing benchmark, showing that the benefit extends beyond terminal tasks. Our analyses link these gains to joint supervision preserving consequence prediction during SFT and supporting higher entropy with less policy movement during RL.

AI USE STATEMENT

In this work, we used generative AI tools to assist with implementing methods. We did not use generative AI tools to generate synthetic data sets, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, clean or reformat datasets, support qualitative or thematic data analysis, or interpret results. The remaining tasks requiring disclosure are not applicable to this work.

Additionally, we used generative AI tools to create or modify scientific figures, create or edit software code, source or search for information, and edit the paper to improve readability. We reviewed all AI-assisted work. We manually surveyed the relevant literature to verify the originality of the research ideas and ensure appropriate attribution. We reviewed and tested AI-assisted code, checked figure-generation scripts and figures against the underlying results. All authors reviewed and approved the final manuscript. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3.1 and Appendix A specify the corpus, training recipes, serving configuration, evaluation convention, and statistical procedures; Appendix B reports the full method matrix with per-method attempt counts. The loss-mask change is fully specified by Eq. 1. Upon acceptance, we will release the code, data, per-task evaluation outcomes, training and evaluation configurations, and analysis artifacts needed to reproduce the reported results.

REFERENCES

Guangfeng Cai, Kaibing Yang, Shuo He, Yu Li, Shengtian Yang, Jiaqi Lv, and Lei Feng. Beyond next-observation prediction: Agent-authored world modeling for sequential decision making. *CoRR*, abs/2606.25421, 2026. doi: 10.48550/ARXIV.2606.25421. URL https://doi.org/10.48550/arXiv.2606.25421.

Hyungjoo Chae, Namyoung Kim, Kai Tzu-iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. Web agents with world models: Learning and leveraging environment dynamics in web navigation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=moWiYJuSGF.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning. *CoRR*, abs/2310.05915, 2023. doi: 10.48550/ARXIV.2310.05915. URL https://doi.org/10.48550/arXiv.2310.05915.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

Shiqi Chen, Tongyao Zhu, Zian Wang, Jinghan Zhang, Kangrui Wang, Siyang Gao, Teng Xiao, Yee Whye Teh, Junxian He, and Manling Li. Internalizing world models via self-play finetuning for agentic RL. *CoRR*, abs/2510.15047, 2025a. doi: 10.48550/ARXIV.2510.15047. URL https://doi.org/10.48550/arXiv.2510.15047.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. Agent-flan: Designing data and methods of effective agent tuning for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of *Findings of ACL*, pp. 9354–9366. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.557. URL https://doi.org/10.18653/v1/2024.findings-acl.557.

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. Pass@k training for adaptively balancing exploration and exploitation of large reasoning models. *CoRR*, abs/2508.10751, 2025b. doi: 10.48550/ARXIV.2508.10751. URL https://doi.org/10.48550/arXiv.2508.10751.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Xin Zhao, Zhenliang Zhang, and Furu Wei. Reasoning with exploration: An entropy perspective. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pp. 30377–30385. AAAI Press, 2026. doi: 10.1609/AAAI.V40I36.40290. URL https://doi.org/10.1609/aaai.v40i36.40290.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/chu25c.html.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The entropy mechanism of reinforcement learning for reasoning language models. *CoRR*, abs/2505.22617, 2025. doi: 10.48550/ARXIV.2505.22617. URL https://doi.org/10.48550/arXiv.2505.22617.

Jeff Da, Clinton Wang, Xiang Deng, Yuntao Ma, Nikhil Barhate, and Sean Hendryx. Agent-rlvr: Training software engineering agents via guidance and environment rewards. *CoRR*, abs/2506.11425, 2025. doi: 10.48550/ARXIV.2506.11425. URL https://doi.org/10.48550/arXiv.2506.11425.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025. doi: 10.48550/ARXIV.2501.12948. URL https://doi.org/10.48550/arXiv.2501.12948.

Kanishk Gandhi, Shivam Garg, Noah D. Goodman, and Dimitris Papailiopoulos. Endless terminals: Scaling RL environments for terminal agents. *CoRR*, abs/2601.16443, 2026. doi: 10.48550/ARXIV.2601.16443. URL https://doi.org/10.48550/arXiv.2601.16443.

Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. Is your LLM secretly a world model of the internet? model-based planning for web agents. *Trans. Mach. Learn. Res.*, 2025, 2025. URL https://openreview.net/forum?id=c6l7yA0HSq.

Shangmin Guo, Omar Darwiche Domingues, Raphaël Avalos, Aaron C. Courville, and Florian Strub. Sample, predict, then proceed: Self-verification sampling for tool use of llms. *CoRR*, abs/2506.02918, 2025. doi: 10.48550/ARXIV.2506.02918. URL https://doi.org/10.48550/arXiv.2506.02918.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 2455–2467, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html.

Hamish Ivison, Junjie Oscar Yin, Rulin Shao, Teng Xiao, Nathan Lambert, and Hannaneh Hajishirzi. Tmax: A simple recipe for terminal agents. *CoRR*, abs/2606.23321, 2026. doi: 10.48550/ARXIV.2606.23321. URL https://doi.org/10.48550/arXiv.2606.23321.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJ6yPD5xg.

Cong Li, Peixi Peng, Yisen Zhao, Xinyu Hu, Shudong Liu, Zhan Su, and Zhuojian Li. TAPO: transition-aware policy optimization for LLM agents. *CoRR*, abs/2607.27973, 2026a. doi: 10.48550/ARXIV.2607.27973. URL https://doi.org/10.48550/arXiv.2607.27973.

Yixia Li, Hongru Wang, Jiahao Qiu, Zhenfei Yin, Dongdong Zhang, Cheng Qian, Zeping Li, Xiaoteng Ma, Guanhua Chen, and Heng Ji. From word to world: Can large language models be implicit text-based world models? In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 8084–8111. Association for Computational Linguistics, 2026b. doi: 10.18653/V1/2026.ACL-LONG.366. URL https://doi.org/10.18653/v1/2026.acl-long.366.

Jessy Lin, Yuqing Du, Olivia Watkins, Danijar Hafner, Pieter Abbeel, Dan Klein, and Anca D. Dragan. Learning to model the world with language. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 29992–30017. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/lin24g.html.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/1a22b912945fb7c0bdd079e792b31b6f-Abstract-Conference.html.

Runze Liu, Jiashun Liu, Xu Wan, Yuqian Fu, and Ling Pan. When RL fails after SFT: rejuvenating model plasticity for robust sft-to-rl handoff. *CoRR*, abs/2606.09932, 2026. doi: 10.48550/ARXIV.2606.09932. URL https://doi.org/10.48550/arXiv.2606.09932.

Ning Lu, Baijiong Lin, Shengcai Liu, Jiahao Wu, Haoze Lv, Yanbin Wei, Lingting Zhu, Shengju Qian, Xin Wang, Ying-Cong Chen, Qi Wang, and Ke Tang. Policy and world modeling co-training for language agents. *CoRR*, abs/2606.02388, 2026. doi: 10.48550/ARXIV.2606.02388. URL https://doi.org/10.48550/arXiv.2606.02388.

Mike A. Merrill, Alexander Glenn Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, Estefany Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, 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 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, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *CoRR*, abs/2601.11868, 2026. doi: 10.48550/ARXIV.2601.11868. URL https://doi.org/10.48550/arXiv.2601.11868.

Yongyu Mu, Jiali Zeng, Fandong Meng, JingBo Zhu, and Tong Xiao. Offline exploration-aware fine-tuning for long-chain mathematical reasoning. *CoRR*, abs/2603.16206, 2026. doi: 10.48550/ARXIV.2603.16206. URL https://doi.org/10.48550/arXiv.2603.16206.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with swe-gym. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/pan25g.html.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 488–489. IEEE Computer Society, 2017. doi: 10.1109/CVPRW.2017.70. URL https://doi.org/10.1109/CVPRW.2017.70.

Renjie Pi, Grace Lam, Mohammad Shoeybi, Pooya Jannaty, Bryan Catanzaro, and Wei Ping. On data engineering for scaling LLM terminal capabilities. *CoRR*, abs/2602.21193, 2026. doi: 10.48550/ARXIV.2602.21193. URL https://doi.org/10.48550/arXiv.2602.21193.

Shuofei Qiao, Runnan Fang, Ningyu Zhang, Yuqi Zhu, Xiang Chen, Shumin Deng, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Agent planning with world knowledge model. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/d032263772946dd5026e7f3cd22bce5b-Abstract-Conference.html.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

Vaishnavi Shrivastava, Piero Kauffmann, Ahmed Awadallah, and Dimitris Papailiopoulos. ECHO: terminal agents learn world models for free. *CoRR*, abs/2605.24517, 2026. doi: 10.48550/ARXIV.2605.24517. URL https://doi.org/10.48550/arXiv.2605.24517.

Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

Andrew Wagenmaker, Perry Dong, Raymond Tsao, Chelsea Finn, and Sergey Levine. Posterior behavioral cloning: Pretraining BC policies for efficient RL finetuning. *CoRR*, abs/2512.16911, 2025. doi: 10.48550/ARXIV.2512.16911. URL https://doi.org/10.48550/arXiv.2512.16911.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025a. URL http://papers.nips.cc/paper_files/paper/2025/hash/a797c2d2e0c1fdabf4d1ab8cd0b465c6-Abstract-Conference.html.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: understanding self-evolution in LLM agents via multi-turn reinforcement learning. *CoRR*, abs/2504.20073, 2025b. doi: 10.48550/ARXIV.2504.20073. URL https://doi.org/10.48550/arXiv.2504.20073.

Fang Wu, Weihao Xuan, Ximing Lu, Zaïd Harchaoui, and Yejin Choi. The invisible leash: Why RLVR may not escape its origin. *CoRR*, abs/2507.14843, 2025. doi: 10.48550/ARXIV.2507.14843. URL https://doi.org/10.48550/arXiv.2507.14843.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, 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 Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

Xiao Yu, Baolin Peng, Ruize Xu, Yelong Shen, Pengcheng He, Suman Nath, Nikhil Singh, Jiangfeng Gao, and Zhou Yu. Reinforcement world model learning for llm-based agents. *CoRR*, abs/2602.05842, 2026. doi: 10.48550/ARXIV.2602.05842. URL https://doi.org/10.48550/arXiv.2602.05842.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *CoRR*, abs/2504.13837, 2025. doi: 10.48550/ARXIV.2504.13837. URL https://doi.org/10.48550/arXiv.2504.13837.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of *Findings of ACL*, pp. 3053–3077. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.181. URL https://doi.org/10.18653/v1/2024.findings-acl.181.

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. *CoRR*, abs/2510.08558, 2025. doi: 10.48550/ARXIV.2510.08558. URL https://doi.org/10.48550/arXiv.2510.08558.

Yuxin Zuo, Zikai Xiao, Li Sheng, Fei Huang, Jianhong Tu, Yuxuan Liu, Tianyi Tang, Xiaomeng Hu, Yang Su, Qingfeng Lan, Yantao Liu, Qin Zhu, Yinger Zhang, Bowen Yu, Haiquan Zhao, Haiyang Xu, Jianxin Yang, Jiayang Cheng, Junyang Wang, Lianghao Deng, Mingfeng Xue, Tianyi Bai, Yang Fan, Yubo Ma, Yucheng Li, Zeyu Cui, Zhihai Wang, Zhihui Xie, Zhuorui Ye, An Yang, Dayiheng Liu, Jingren Zhou, and Ning Ding. Qwen-agentworld: Language world models for general agents. *CoRR*, abs/2606.24597, 2026. doi: 10.48550/ARXIV.2606.24597. URL https://doi.org/10.48550/arXiv.2606.24597.

A Evaluation protocol and statistical procedures

Reporting convention. We evaluate all 89 Terminal-Bench 2.0 tasks and all 225 aider-polyglot tasks. Here, n denotes the number of attempts per task. The headline Terminal-Bench 2.0 results pool four independently launched batches of four attempts per task, giving each evaluated SFT and RL checkpoint 16 attempts per task. Other evaluations use four or eight attempts per task as indicated. Model-caused failures, including formatting failures, receive a score of zero. A timeout can still count as a success if the verifier confirms that the task was completed. Infrastructure failures from the harness, verifier, or Docker are rerun once under the same configuration. We compute pass@k with the unbiased estimator of Chen et al. (2021).

Uncertainty estimates. Tables report the point estimate and one bootstrap standard error, computed from 20,000 resamples. Each bootstrap replicate holds the benchmark task set fixed, samples the observed binary attempts with replacement within each task, recomputes the pass@k estimator, and averages it across tasks. The resulting uncertainty captures finite-attempt sampling variation.

Evaluation budget. Using 16 attempts per task for every headline comparison reduces sampling variance in the pass@k estimates, enabling a more reliable comparison in Table 1. All headline checkpoints use the same task set, attempt count, and serving configuration, so each comparison has a matched evaluation budget. The same samples support estimates from pass@1 through pass@16 without changing the evaluation protocol across sampling budgets.

Serving configuration. Terminal-Bench 2.0 task limits are defined in wall-clock time, making serving latency part of the evaluation protocol. We pinned the serving setup after a nine-setting sweep over tensor parallelism, model colocation, and trial concurrency. For Terminal-Bench 2.0 evaluation, a single vLLM engine uses tensor parallelism across eight NVIDIA A100 GPUs on one node, colocated with the Docker trials. The setup allows 12 concurrent trials, a 40,960-token context, and vLLM GPU-memory utilization of 0.90. Decoding follows the checkpoint's generation configuration: $T = 0.6$, top-p 0.95, and top-k 20, except for the explicit temperature-matching analysis in Section 4.2.

B Complete results and ablations

Complete 8B matrix and control definitions. Table 2 contains the primary comparison together with the diagnostic variants. *act2act* repeats action-only SFT for a second epoch. *ActObs-100* applies joint action-observation supervision for the first 100 steps and then continues with action-only SFT. *Shuffled obs* pairs actions with observations sampled from other trajectories, whereas *wild logs* replaces observations with token- and format-matched production logs unrelated to the preceding action. The *salient* variants remove command echoes, repeated lines, and regular scaffold text before observation supervision.

The primary rows show how the 8B ordering changes with sampling budget. Before RL, ActionSFT has the highest point estimate at pass@1, while the three principal SFT methods remain within a narrow band over the full pass@k range. After the same GRPO procedure, ActObs matches the leading pass@4 score and becomes strongest at larger budgets, reaching 23.6 at pass@8 and 27.0 at pass@16. It also solves 24 of 89 tasks, compared with 21 for ActionSFT→GRPO. The sequential Obs→Act method receives both forms of supervision but does not reproduce this high-budget advantage, supporting joint rather than temporally separated supervision.

The remaining rows identify what produces that separation. A second epoch of action-only SFT closely tracks ActionSFT, so additional optimization alone does not recover the ActObs behavior. ActObs-100 reaches 19.1 pass@8 after GRPO, below the 23.6 obtained when observation supervision continues through SFT, indicating that the observation objective remains useful beyond the early updates. Observation-only SFT and action SFT followed by observation-only training solve no tasks, showing that observation prediction complements action learning rather than replacing it. Correct pairing also matters: shuffled observations reduce pass@1 from 7.9 to 3.7 and pass@4 from 16.1 to 9.0, whereas unrelated but clearly distinguishable wild logs retain 7.9 pass@1. Finally, supervision on the complete terminal response is stronger than selecting only salient lines, consistent with the payload-level effects in Figure 6(b,c).

Table 2: **Complete Qwen3-8B results on Terminal-Bench 2.0.** The results use the pinned serving configuration and the reporting protocol in Appendix A. Rows above the horizontal rule report SFT checkpoints, and rows below it report RL checkpoints. Each checkpoint in the primary comparison is evaluated with 16 attempts per task.

Model	n	pass@1	pass@2	pass@4	pass@8	pass@16	solved/89
Base (no SFT)	16	2.7$_{\pm 0.4}$	4.6$_{\pm 0.5}$	7.0$_{\pm 0.6}$	9.3$_{\pm 0.8}$	11.2$_{\pm 1.0}$	10
ActionSFT	16	9.2$_{\pm 0.5}$	13.6$_{\pm 0.7}$	**18.2**$_{\pm 0.8}$	**21.9**$_{\pm 1.0}$	**24.7**$_{\pm 1.2}$	**22**
ActObs	16	7.9$_{\pm 0.5}$	11.8$_{\pm 0.7}$	16.1$_{\pm 0.8}$	19.7$_{\pm 0.9}$	22.5$_{\pm 1.2}$	20
Obs→Act	16	8.3$_{\pm 0.5}$	12.0$_{\pm 0.6}$	15.9$_{\pm 0.8}$	19.8$_{\pm 1.1}$	**24.7**$_{\pm 1.6}$	**22**
act2act (2 epochs action)	16	**9.3**$_{\pm 0.5}$	13.2$_{\pm 0.6}$	17.0$_{\pm 0.8}$	20.7$_{\pm 1.1}$	**24.7**$_{\pm 1.4}$	**22**
ActObs-100 (first 100 steps)	16	8.4$_{\pm 0.5}$	12.6$_{\pm 0.7}$	17.0$_{\pm 0.8}$	20.9$_{\pm 0.9}$	23.6$_{\pm 1.1}$	21
$\lambda = 0.05$	4	9.0$_{\pm 0.5}$	13.3$_{\pm 0.7}$	18.0$_{\pm 0.8}$	–	–	–
$\lambda = 0.25$	4	**9.3**$_{\pm 0.6}$	**13.9**$_{\pm 0.7}$	18.0$_{\pm 0.8}$	–	–	–
$\lambda = 0.5$	4	7.9$_{\pm 0.7}$	12.0$_{\pm 0.8}$	16.9$_{\pm 0.9}$	–	–	–
$\lambda = 2.0$	4	8.1$_{\pm 0.5}$	11.4$_{\pm 0.6}$	14.6$_{\pm 0.8}$	–	–	–
Obs only	4	0.0	0.0	0.0	–	–	–
Shuffled obs	4	3.7$_{\pm 0.7}$	6.0$_{\pm 0.9}$	9.0$_{\pm 1.2}$	–	–	–
Wild logs	4	7.9$_{\pm 0.9}$	12.2$_{\pm 1.2}$	16.9$_{\pm 1.5}$	–	–	–
Salient-filtered obs	4	7.2$_{\pm 0.8}$	10.3$_{\pm 1.0}$	13.5$_{\pm 1.3}$	–	–	–
Salient Obs→Act	4	8.1$_{\pm 0.9}$	12.4$_{\pm 1.1}$	15.7$_{\pm 1.2}$	–	–	–
Action→obs-only	4	0.0	0.0	0.0	–	–	–
ActionSFT→GRPO	16	**12.3**$_{\pm 0.5}$	**15.9**$_{\pm 0.5}$	18.7$_{\pm 0.6}$	21.0$_{\pm 0.8}$	23.6$_{\pm 1.2}$	21
ActObs→GRPO	16	11.0$_{\pm 0.5}$	15.0$_{\pm 0.7}$	19.3$_{\pm 0.9}$	**23.6**$_{\pm 1.1}$	**27.0**$_{\pm 1.3}$	**24**
Obs→Act→GRPO	16	11.9$_{\pm 0.5}$	**15.9**$_{\pm 0.6}$	19.3$_{\pm 0.7}$	21.7$_{\pm 0.7}$	23.6$_{\pm 1.0}$	21
ActionSFT→ECHO	16	9.6$_{\pm 0.5}$	13.0$_{\pm 0.6}$	16.5$_{\pm 0.9}$	20.6$_{\pm 1.2}$	25.8$_{\pm 1.6}$	23
ActObs→ECHO	16	11.4$_{\pm 0.5}$	15.1$_{\pm 0.6}$	18.6$_{\pm 0.8}$	22.0$_{\pm 0.9}$	24.7$_{\pm 1.2}$	22
$\lambda = 0.05$ →GRPO	8	11.4$_{\pm 0.7}$	15.2$_{\pm 0.8}$	18.8$_{\pm 1.0}$	21.3$_{\pm 1.1}$	–	–
$\lambda = 0.25$ →GRPO	8	11.0$_{\pm 0.8}$	**15.9**$_{\pm 0.9}$	**20.1**$_{\pm 1.0}$	22.5$_{\pm 1.0}$	–	–
ActObs-100→GRPO	8	11.0$_{\pm 0.6}$	14.0$_{\pm 0.7}$	16.5$_{\pm 0.9}$	19.1$_{\pm 1.1}$	–	–

Complete 4B matrix. Table 3 repeats the full primary comparison at the smaller model scale. It provides a stricter test of whether the downstream effect is specific to one model size or to the 8B crossover pattern.

The 4B SFT checkpoints are almost indistinguishable at pass@1, with point estimates between 4.4 and 4.5. Observation-supervised SFT nevertheless reaches 18.0 at pass@16 and solves one additional task. The difference becomes much clearer after optimization: among the three plain-GRPO policies, ActObs→GRPO is strongest at every reported sampling budget. It improves on ActionSFT→GRPO by 1.5 points at pass@1 and retains an advantage through pass@16. Thus, the 4B result expresses the same initialization effect without requiring a crossover in the final pass@k ordering.

The ECHO rows provide a complementary RL objective. With the ActObs initialization, ECHO reaches 20.2 pass@16 and solves 18 tasks, the largest high-budget point estimate and task count in the 4B matrix. Across both GRPO and ECHO, the observation-aware initialization therefore supports stronger pass@k performance as the sampling budget grows.

Direct endpoint differences. Table 4 subtracts the ActionSFT score from the ActObs score at matched model scale, training stage, and sampling budget. This presentation removes the common performance level and exposes how the gap changes during RL.

At 4B, a near-zero SFT difference becomes a positive post-GRPO difference at every k, with gains of 1.1–1.5 points. At 8B, ActObs begins below ActionSFT at the SFT checkpoint, but GRPO shifts the comparison steadily toward ActObs as the sampling budget grows: the difference crosses zero between pass@2 and pass@4 and reaches +3.4 points at pass@16. The direction of change at both scales shows that the central effect is produced by how RL uses the initialization, rather than by a uniformly stronger SFT checkpoint.

Table 3: **Complete Qwen3-4B results on Terminal-Bench 2.0.** Rows above the horizontal rule report SFT checkpoints, and rows below it report RL checkpoints. Each checkpoint is evaluated with 16 attempts per task.

Model	pass@1	pass@2	pass@4	pass@8	pass@16	solved/89
Base (no SFT)	$1.2_{\pm 0.2}$	$2.1_{\pm 0.4}$	$3.3_{\pm 0.5}$	$4.7_{\pm 0.6}$	$5.6_{\pm 0.7}$	5
ActionSFT	$\mathbf{4.5}_{\pm 0.4}$	$\mathbf{7.0}_{\pm 0.6}$	$\mathbf{10.4}_{\pm 0.8}$	$14.0_{\pm 1.0}$	$16.9_{\pm 1.2}$	15
ActObs	$4.4_{\pm 0.4}$	$6.8_{\pm 0.6}$	$9.9_{\pm 0.9}$	$14.0_{\pm 1.1}$	$\mathbf{18.0}_{\pm 1.4}$	**16**
Obs→Act	$4.4_{\pm 0.4}$	$6.8_{\pm 0.6}$	$10.1_{\pm 0.9}$	$\mathbf{14.2}_{\pm 1.1}$	$\mathbf{18.0}_{\pm 1.4}$	**16**
ActionSFT→GRPO	$5.6_{\pm 0.4}$	$8.3_{\pm 0.5}$	$11.1_{\pm 0.7}$	$14.0_{\pm 1.0}$	$18.0_{\pm 1.4}$	16
ActObs→GRPO	$\mathbf{7.2}_{\pm 0.4}$	$\mathbf{9.7}_{\pm 0.5}$	$12.5_{\pm 0.7}$	$15.5_{\pm 1.0}$	$19.1_{\pm 1.4}$	17
Obs→Act→GRPO	$5.3_{\pm 0.4}$	$8.1_{\pm 0.6}$	$11.4_{\pm 0.8}$	$14.7_{\pm 0.9}$	$16.9_{\pm 1.0}$	15
ActionSFT→ECHO	$5.9_{\pm 0.4}$	$8.6_{\pm 0.6}$	$11.6_{\pm 0.8}$	$14.7_{\pm 1.0}$	$18.0_{\pm 1.3}$	16
ActObs→ECHO	$5.9_{\pm 0.4}$	$9.0_{\pm 0.6}$	$\mathbf{12.7}_{\pm 0.8}$	$\mathbf{16.5}_{\pm 1.1}$	$\mathbf{20.2}_{\pm 1.4}$	**18**

Table 4: **Direct ActObs versus ActionSFT comparisons.** Each cell reports the ActObs minus ActionSFT difference on Terminal-Bench 2.0 in pass@k points. The ECHO rows compare the corresponding ECHO checkpoints.

Comparison	Δ pass@1	Δ pass@2	Δ pass@4	Δ pass@8	Δ pass@16
4B after SFT	-0.1	-0.2	-0.4	0.0	+1.1
4B after GRPO	+1.5	+1.5	+1.4	+1.4	+1.1
4B after ECHO	0.0	+0.4	+1.1	+1.8	+2.2
8B after SFT	-1.3	-1.8	-2.0	-2.2	-2.2
8B after GRPO	-1.3	-0.8	+0.7	+2.7	+3.4
8B after ECHO	+1.8	+2.2	+2.2	+1.4	-1.1

The ECHO comparison is also informative. At 4B, its ActObs advantage grows from parity at pass@1 to +2.2 points at pass@16. At 8B, the advantage is concentrated at smaller and intermediate budgets. This distinction reinforces that observation supervision changes the distribution of successful behavior, while the precise conversion of that distribution depends on the downstream RL objective.

Improvement during RL. Final-policy differences combine two quantities: where each method starts and how much it learns during RL. Table 5 isolates the second by subtracting each method's own SFT score from its corresponding RL score.

The 8B contrast is sharpest in this view. GRPO from ActionSFT gains 3.1 points at pass@1, but the gain contracts at larger k and is negative at pass@8 and pass@16. GRPO from ActObs gains at least 3.1 points at every budget, with the improvement increasing to 4.5 points at pass@16. Sequential Obs→Act supervision produces strong gains at small k but does not retain them at pass@16. Joint supervision is therefore the only 8B initialization whose GRPO improvement remains positive and substantial throughout the reported sampling range.

At 4B, ActObs likewise produces the largest plain-GRPO gain from pass@1 through pass@8. Its +2.7-point pass@1 gain is more than twice the +1.1 points from ActionSFT, while its pass@8 gain is +1.5 rather than +0.1. The ActObs ECHO row also remains positive across the full range and is strongest at intermediate and large budgets. Together, these rows show that observation supervision improves the amount and breadth of performance that downstream optimization can extract from an SFT checkpoint.

Observation-loss weight. The binary comparison between ActionSFT and ActObs corresponds to the endpoints $\lambda = 0$ and $\lambda = 1$ in Eq. 1. Table 6 adds intermediate weights to test whether the final behavior changes gradually with the amount of observation supervision.

The SFT block shows that small observation weights preserve the immediate benchmark level: $\lambda = 0.05$ and $\lambda = 0.25$ remain close to ActionSFT through pass@4. At the same time, the

Table 5: **Performance gains during RL.** Each cell reports the change in pass@k relative to the corresponding SFT initialization, in points. Every checkpoint is evaluated with 16 attempts per task.

	Init	Δ pass@1	Δ pass@2	Δ pass@4	Δ pass@8	Δ pass@16
8B	ActionSFT	+3.1	+2.2	+0.5	-1.0	-1.1
	ActObs	+3.1	+3.2	+3.2	**+3.9**	**+4.5**
	Obs→Act	**+3.6**	**+3.8**	**+3.4**	+1.9	-1.1
	ActionSFT (ECHO)	+0.4	-0.7	-1.7	-1.4	+1.1
	ActObs (ECHO)	**+3.6**	+3.3	+2.5	+2.2	+2.2
4B	ActionSFT	+1.1	+1.3	+0.8	+0.1	+1.1
	ActObs	**+2.7**	**+3.0**	+2.6	+1.5	+1.1
	Obs→Act	+0.8	+1.3	+1.3	+0.5	-1.1
	ActionSFT (ECHO)	+1.4	+1.6	+1.2	+0.7	+1.1
	ActObs (ECHO)	+1.5	+2.2	**+2.7**	**+2.5**	**+2.2**

Table 6: **Observation-loss weight sweep at 8B.** The SFT block reports checkpoints trained with different values of λ, together with the change in the command-token top1-to-top2 margin relative to ActionSFT on a fixed trace set. Intermediate values use four attempts per task; the $\lambda = 0$ and $\lambda = 1$ endpoints use the ActionSFT and ActObs evaluations with 16 attempts per task. The GRPO block reports the resulting policies after the common RL recipe, using eight attempts per task for $\lambda \in \{0.05, 0.25\}$ and 16 attempts per task for the endpoints.

After SFT λ	pass@1	pass@2	pass@4	Δ cmd margin
0 (ActionSFT)	9.2$_{\pm 0.5}$	13.6$_{\pm 0.7}$	**18.2**$_{\pm 0.8}$	–
0.05	9.0$_{\pm 0.5}$	13.3$_{\pm 0.7}$	18.0$_{\pm 0.8}$	-0.29
0.25	**9.3**$_{\pm 0.6}$	**13.9**$_{\pm 0.7}$	18.0$_{\pm 0.8}$	-0.35
0.5	7.9$_{\pm 0.7}$	12.0$_{\pm 0.8}$	16.9$_{\pm 0.9}$	-0.36
1 (ActObs)	7.9$_{\pm 0.5}$	11.8$_{\pm 0.7}$	16.1$_{\pm 0.8}$	-0.48
2.0	8.1$_{\pm 0.5}$	11.4$_{\pm 0.6}$	14.6$_{\pm 0.8}$	-0.52
After GRPO λ	pass@1	pass@2	pass@4	pass@8
0 (ActionSFT)	**12.3**$_{\pm 0.5}$	**15.9**$_{\pm 0.5}$	18.7$_{\pm 0.6}$	21.0$_{\pm 0.8}$
0.05	11.4$_{\pm 0.7}$	15.2$_{\pm 0.8}$	18.8$_{\pm 1.0}$	21.3$_{\pm 1.1}$
0.25	11.0$_{\pm 0.8}$	**15.9**$_{\pm 0.9}$	**20.1**$_{\pm 1.0}$	22.5$_{\pm 1.0}$
1 (ActObs)	11.0$_{\pm 0.5}$	15.0$_{\pm 0.7}$	19.3$_{\pm 0.9}$	**23.6**$_{\pm 1.1}$

command-token top-1 to top-2 margin decreases progressively as λ grows, showing that the action distribution changes even when SFT benchmark scores remain similar. After GRPO, pass@8 increases monotonically from 21.0 at $\lambda = 0$ to 21.3, 22.5, and 23.6 as the observation weight rises. Pass@1 moves from 12.3 to 11.0 over the same sweep. Observation weight therefore controls a tradeoff between single-attempt reliability and multi-attempt task reach: lower weight favors one attempt, while higher weight allocates more probability across behaviors that become useful with multiple samples.

The intermediate checkpoints are important because they rule out an all-or-nothing endpoint effect. In particular, $\lambda = 0.25$ reaches 20.1 pass@4 and 22.5 pass@8 after GRPO while starting from an SFT checkpoint whose pass@1 and pass@4 closely match ActionSFT. The downstream change is therefore already visible before the full ActObs weight is applied.

Absolute performance across sampling budgets. Figure 7 complements the improvement plot in Figure 2 by showing the absolute scores of the four primary SFT and GRPO policies. It makes the scale-dependent pattern easy to compare: the 4B ActObs GRPO policy is stronger at every evaluated k, whereas the 8B curves cross between $k = 2$ and $k = 4$ before the ActObs advantage expands at larger budgets.

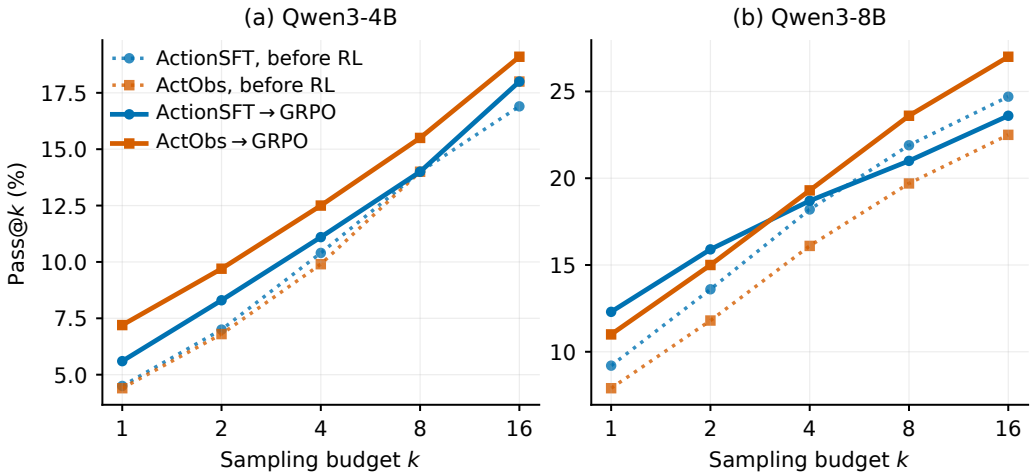

Figure 7: **Absolute pass@k curves.** The primary comparison at both model scales complements Figure 2. At 4B, the point estimate for ActObs→GRPO exceeds that for ActionSFT→GRPO at every k. At 8B, the point-estimate curves cross between $k = 2$ and $k = 4$.

Taken together, the control ladder isolates three ingredients. Action competence must be maintained, mismatched observations are harmful, and the two objectives are most effective when optimized jointly throughout SFT. The weight sweep then shows that this effect is graded rather than binary. These findings connect the complete result matrices to the mechanism in Sections 4.1–5.2: observation supervision changes the checkpoint in ways that are only partially reflected by its immediate SFT score but become consequential during RL and repeated sampling.

C RL TRAINING DYNAMICS

Figures 8 and 9 extend the training-dynamics analysis in Section 4.1. Both figures report *training entropy*, the token-level policy entropy on the newly sampled on-policy batch at each RL update. This differs from the endpoint self-entropy in Figure 4(a) and the fixed-state entropy in Appendix D.

Figure 8 isolates the default GRPO comparison at 8B. The reward curves in Figure 8(a) overlap and reach the same endpoint, but the entropy curves in Figure 8(b) separate near step 20 and remain apart. Figure 8(c) rules out rollout length as a simple explanation: ActObs is longer through the middle of training, yet the two methods finish at similar lengths while their entropy gap remains. Together with the shorter final ActObs rollouts at 4B, this shows that its higher late-stage entropy is not a mechanical consequence of longer generations.

Figure 9 broadens the comparison to all five RL methods at both model scales. At 4B, Figure 9(a) shows ActionSFT losing entropy late in standard GRPO while ActObs remains higher. At 8B, Figure 9(b) shows the ActObs separation emerging earlier and growing through training. The sequential Obs→Act method finishes below both at each scale, so observation exposure alone does not reproduce joint supervision.

The ECHO trajectories are less monotonic and finish with high entropy, but the benchmark ordering is not determined by entropy magnitude alone. At 4B, ActionSFT→ECHO and ActionSFT→GRPO attain the same pass@16 and solve the same number of tasks. At 8B, ActObs→ECHO has the second-highest training-entropy endpoint but solves 22 tasks, compared with 24 for ActObs→GRPO (Table 2). Meanwhile, the 8B methods finish with closely matched reward, and pass@16 on the RL training tasks saturates near 0.9. The persistent entropy separation therefore reflects different optimization trajectories that aggregate reward does not reveal.

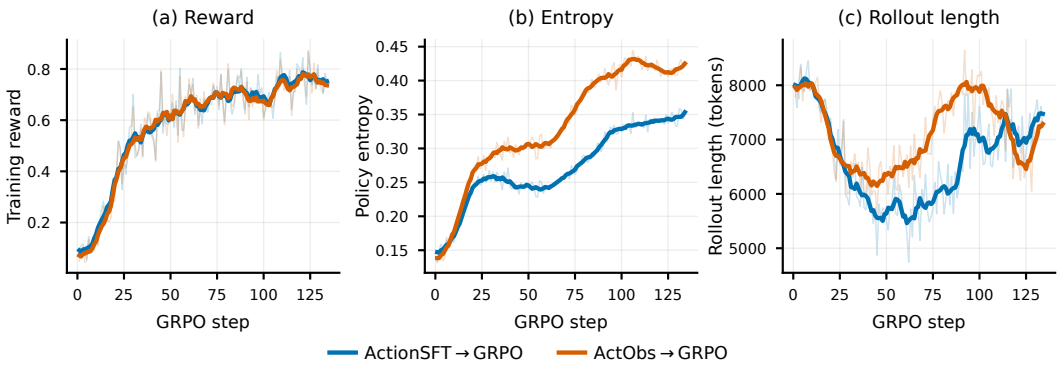

Figure 8: **GRPO training dynamics at 8B.** This is the counterpart of Figure 3: (a) training reward, (b) on-policy training entropy, and (c) rollout length. The two methods reach the same reward while following clearly separated entropy trajectories. Both produce shorter rollouts early in training; ActObs remains longer through the middle of training, and the two methods end at similar lengths.

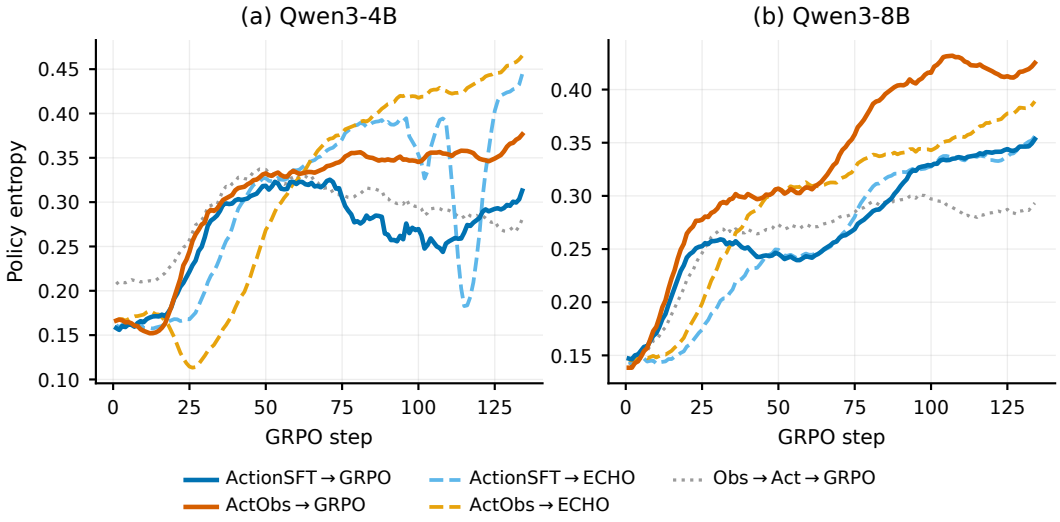

Figure 9: **Training entropy across RL methods and model scales.** Seven-step moving averages are shown for (a) Qwen3-4B and (b) Qwen3-8B. Solid lines denote standard GRPO from ActionSFT or ActObs, dashed lines denote ECHO, and the dotted line denotes standard GRPO from the sequential Obs→Act initialization.

D Entropy and Command-Margin Probes

Probe construction. Table 7 reports student-forcing measurements for the 8B checkpoints. Self-entropy is computed on 200 Terminal-Bench 2.0 rollouts generated by each checkpoint, covering 81 to 87 tasks. Fixed-state entropy instead scores every checkpoint on the same 200 ActionSFT traces. Command entropy and margin restrict the calculation to emitted commands; a smaller margin places the second-most likely token closer to the top choice.

Before RL. ActObs is consistently flatter than ActionSFT. Its self-entropy rises from 0.683 to 0.706, and the same ordering holds on fixed states, showing that the difference is not only due to state visitation. Its command margin also falls from 7.00 to 6.56. The sequential Obs→Act checkpoint is similarly flat before RL but does not retain this structure after GRPO. The controls sharpen this interpretation: a second action-only epoch makes the policy sharper; shuffled observations reproduce the ActObs entropy and margin signature but perform much worse; and wild logs reproduce neither effect. Flatter command distributions are therefore a consequence of observation supervision, while strong performance also requires pairing actions with their actual feedback.

Table 7: **Entropy and command margins at 8B.** Each checkpoint scores 200 of its own Terminal-Bench 2.0 evaluation rollouts (self) and a shared set of 200 traces (fixed). Command entropy and margin are computed at emitted command tokens; a lower top-1 to top-2 margin indicates a flatter distribution. Within each stage, boldface marks the highest entropy and lowest margin.

Model	self-entropy	fixed-state entropy	cmd entropy	cmd margin
ActionSFT	0.683	0.683	0.308	7.00
ActObs	**0.706**	0.708	0.321	**6.56**
Obs→Act	0.704	0.711	**0.328**	6.65
act2act	0.678	0.678	0.296	7.05
Shuffled obs	0.702	**0.728**	0.317	6.59
Wild logs	0.654	0.693	0.285	7.01
ActionSFT→GRPO	0.716	0.719	0.312	7.55
ActObs→GRPO	0.779	0.755	0.320	7.24
Obs→Act→GRPO	0.684	0.717	0.259	7.78
ActionSFT→ECHO	0.794	**0.788**	0.355	7.21
ActObs→ECHO	**0.804**	0.784	**0.363**	**6.44**

After RL. The post-GRPO rows reproduce the training-entropy ordering from Appendix C. ActObs retains substantially more self-entropy than ActionSFT, 0.779 versus 0.716. The advantage remains on the shared trace set, 0.755 versus 0.719, and its command margin remains flatter, 7.24 versus 7.55. The sequential policy moves in the opposite direction, ending with the lowest self-entropy and command entropy among the standard-GRPO policies. Thus, the SFT similarity between ActObs and Obs→Act breaks decisively during GRPO. ECHO raises entropy further: ActObs→ECHO has the highest self-entropy and command entropy in the table and the lowest margin, yet it trails ActObs→GRPO at pass@16 and solves fewer tasks (Table 2). High entropy and a flat margin can support exploration, but their magnitude alone does not determine success.

E Extended-budget evaluation

We test whether observation-supervised initializations make productive use of longer interaction horizons within Terminal-Bench 2.0 by reevaluating six 8B checkpoints after doubling the agent wall-clock limit while leaving the verifier timeout and serving configuration unchanged. Table 8 compares these four-attempt evaluations with the headline results under the standard budget.

All four observation-supervised checkpoints have higher pass@1 point estimates under the doubled budget. The SFT gains are +2.2 points for ActObs and +1.5 for Obs→Act, and the post-GRPO gains are +1.7 and +0.2 points. The standard ActionSFT checkpoint is the only SFT policy that does not benefit from the additional interaction time. This consistent directional pattern shows that policies initialized with observation supervision continue to convert a longer interaction budget into successful trajectories.

F Behavioral studies

Within-task strategy diversity. We conducted three blinded LLM-judge analyses with two judges per unit and the method identity concealed. The judges counted distinct solution strategies among the successful attempts for each task. Exact inter-judge agreement ranges from 91% to 94% across the three analyses. In the largest analysis, the methods average 1.5–1.9 strategies per solved task, and their pairwise command-set Jaccard similarities are also closely matched. These probes localize the ActObs advantage to *outcome diversity*: its successes are spread across more tasks, rather than repeating more distinct strategies on tasks that are already solved.

Composition of the task-coverage difference. At 8B, ActObs→GRPO and ActionSFT→GRPO solve 19 tasks in common. ActObs solves five additional tasks: *code-from-image*, *fix-git*, *kv-store-grpc*, *mcmc-sampling-stan*, and *qemu-startup*. ActionSFT solves two different tasks, giving ActObs a net advantage of three tasks, 24 versus 21. Three of the five ActObs-only tasks, *code-from-image*,

Table 8: **Pass@1 after doubling the Terminal-Bench 2.0 agent wall-clock budget.** The extended-budget evaluation uses four attempts per task; the standard-budget column reports the 16-attempt headline results. Every observation-supervised checkpoint improves under the longer interaction budget.

Model	pass@1 at 1×	pass@1 at 2×	Δ
ActionSFT	9.2	7.9	-1.3
ActObs	7.9	10.1	+2.2
Obs→Act	8.3	9.8	+1.5
ActionSFT→GRPO	12.3	14.0	+1.8
ActObs→GRPO	11.0	12.6	+1.7
Obs→Act→GRPO	11.9	12.1	+0.2

Table 9: **Success distribution across tasks.** Results are computed on Terminal-Bench 2.0 using 16 attempts per task for each checkpoint. Win entropy is the Shannon entropy of the checkpoint's success distribution over its solved tasks; higher values indicate more evenly spread successes. Boldface marks the largest value within each stage and column.

Model	Successes	Tasks solved	Successes/task	Win entropy (bits)
ActionSFT	**131**	**22**	**5.95**	**4.11**
ActObs	112	20	5.60	3.97
Obs→Act	118	**22**	5.36	3.94
ActionSFT→GRPO	**175**	21	**8.33**	4.05
ActObs→GRPO	156	**24**	6.50	**4.16**
Obs→Act→GRPO	169	21	8.05	4.10
ActionSFT→ECHO	137	23	5.96	3.93
ActObs→ECHO	163	22	7.41	4.06

kv-store-grpc, and *qemu-startup*, receive no success across the 64 combined SFT attempts per task or the 16 ActionSFT→GRPO attempts. Blinded review identifies two recurring patterns: less correlated execution of a shared procedure across attempts and broader sampling of parameters within a trajectory.

The per-task ledger supports the same picture. Relative to its SFT initialization, ActObs→GRPO adds six tasks while losing only two; ActionSFT→GRPO adds three and loses four. Across ActionSFT→GRPO, ActObs→GRPO, Obs→Act→GRPO, and ActionSFT→ECHO, 31 of 89 tasks are solved by at least one policy. Table 9 shows how successes are distributed over those tasks. At the SFT stage, all three checkpoints average 5.4–6.0 successes per solved task. After RL, ActionSFT→GRPO concentrates 8.3 successes per solved task over 21 tasks, whereas ActObs→GRPO distributes 6.5 successes per task over 24 tasks and achieves the highest win entropy. This broader success portfolio is the counterpart of its stronger high-k performance.

Case studies. We examine *qemu-startup*, *fix-git*, and *kv-store-grpc*, three of the five tasks solved by ActObs→GRPO but not by ActionSFT→GRPO. Appendix G reproduces trajectory excerpts for *qemu-startup* and *fix-git*. On *qemu-startup*, all methods use the same general procedure: launch QEMU in the background with a serial TCP server, then terminate and retry unsuccessful launches. The successful ActObs→GRPO trajectory systematically changes one launch parameter at a time. In particular, it varies guest memory from `-m 128M` to `512M` and then `1024M`; QEMU's 128 MiB default is insufficient to boot the image. None of the lower-entropy attempts samples this sequence.

On *fix-git*, every method locates the dangling commit with `git reflog`, but the successful ActObs trajectories continue trying new integration tactics. The trajectory in Appendix G works through failed cherry-pick and patch attempts, reapplies the recovered content, and commits it; another successful trajectory reaches the recovered state with `git reset --hard`. Two ActionSFT trajectories reach the integration step but subsequently lose progress because of pager-handling failures and an unsupported file rewrite. On *kv-store-grpc*, the methods follow the same high-level procedure, but the ActObs attempts fail at less correlated points and are more likely to complete the full sequence.

Table 10: **Behavioral patterns in post-RL trajectories.** Statistics are computed over all available Terminal-Bench 2.0 trajectories for each displayed policy. Each value is paired with its policy; the counts in the final row are totals within the two groups.

Measure	Higher self-entropy	Lower self-entropy
Self-entropy	ActObs→GRPO: 0.779; ActionSFT→ECHO: 0.794	ActionSFT→GRPO: 0.716; Obs→Act→GRPO: 0.684
Trajectory shape	ActObs→GRPO uses 10.8 fewer commands per attempt; ActionSFT→ECHO uses 29.8 fewer. Both use more turns.	More command-heavy repetition
Opening diversity	ActObs→GRPO: 0.797; ActionSFT→ECHO: 0.772	ActionSFT→GRPO: 0.760; Obs→Act→GRPO: 0.723
Response when stalled	42 outward escalations across both policies, including new tools and orchestration scripts	16 outward escalations across both policies, with more terminal resets and repeated rewrites

Behavioral patterns in post-RL trajectories. We analyze the available Terminal-Bench 2.0 trajectories from four post-RL policies. Based on the self-entropy measurements in Table 7, we compare the higher-entropy ActObs→GRPO and ActionSFT→ECHO policies with the lower-entropy ActionSFT→GRPO and Obs→Act→GRPO policies. The opening-diversity score measures variation among the first three commands across attempts; a larger value indicates less repetitive openings. Command-count differences are measured relative to ActionSFT→GRPO.

Table 10 shows a consistent behavioral separation. The higher-entropy policies begin with more varied command sequences, issue fewer commands across more turns, and more often change tactics when progress stalls. The lower-entropy policies rely more heavily on dense command sequences, terminal resets, and repeated rewrites. Opening diversity is a correlate of entropy rather than the main source of success. In a blinded review of 12 successful trajectories from tasks where the methods differ, eight successes arise from less correlated executions of the same high-level procedure, two from trying different parameter values, and two from sustained persistence.

Trajectory-level statistics. Table 11 separates overall trajectory length from the length of successful trajectories. The two quantities capture different behaviors: the first reflects how long a policy continues interacting across both solved and unsolved attempts, while the second measures how directly its successful attempts reach a solution. Before RL, ActObs has a longer median trajectory than ActionSFT, 59 versus 51 steps, but its successful trajectories are much shorter, 13 versus 21 steps. The sequential Obs→Act policy has the shortest successful trajectories at 11 median steps. Observation-supervised SFT can therefore combine greater persistence overall with more efficient execution on the attempts that succeed.

The clearest shared effect of GRPO is a collapse in redundant command repetition. Across the three main SFT-to-GRPO comparisons, exact repetitions fall from 34.9–48.1 per trajectory to 1.0–1.2, and identical retries after an error fall from 48.0–55.5% to 0.7–0.9%. Errors per trajectory also decline for every main method. Overall trajectory length, however, does not uniformly shrink: the ActObs and Obs→Act endpoints remain longer than the ActionSFT endpoint. Their behavior is therefore better characterized by less repetitive interaction than by shorter interaction. The shuffled-observation control provides the opposite pattern. It has the highest error count and repeats the preceding command after most errors, showing that observation tokens help only when they are paired with the actions that produced them.

G Representative transcripts

This appendix reproduces excerpts from the ActObs→GRPO trajectories used in the *qemu-startup* and *fix-git* case studies in Appendix F. All text is quoted verbatim from the recorded trajectories; `[...]` marks elided text, and trailing blank screen lines are trimmed. Blue boxes show agent turns: the `analysis` and `plan` fields of the JSON response, followed by the submitted keystrokes. Or-

Table 11: **Trajectory-level behavioral statistics.** Statistics are computed over all available Terminal-Bench 2.0 trajectories for each policy. "Median steps" is the median number of interaction steps across all trajectories, whereas "Solve steps" is the median restricted to successful trajectories. "Repeats/traj." and "Errors/traj." are the mean numbers of exact command repetitions and error-bearing observations per trajectory, respectively. "Retry-same" is the fraction of error-bearing observations followed by an exact repetition of the preceding command.

Model	Median steps	Solve steps	Repeats/traj.	Errors/traj.	Retry-same
Base	13	5	19.0	10.2	40.5%
ActionSFT	51	21	45.7	23.7	55.5%
ActObs	59	13	48.1	26.8	51.9%
Obs→Act	49	11	34.9	19.7	48.0%
act2act	41	14	45.8	22.2	53.4%
Wild logs	37	14	33.3	15.0	42.6%
Shuffled obs	55	17	52.9	33.4	75.9%
ActionSFT→GRPO	41	26	1.0	12.4	0.7%
ActObs→GRPO	48	35	1.2	10.9	0.9%
Obs→Act→GRPO	52	33	1.0	12.0	0.9%
ActionSFT→ECHO	34	29	1.0	6.8	0.8%
ActObs→ECHO	29	21	0.6	10.4	0.9%

ange boxes show the observation returned to the model after execution. Long episodes also contain scaffold-managed context summarization turns, which we elide.

We first show the episode prompt that defines the environment. Every episode begins with a single user message containing the scaffold instructions, the task description, and the initial terminal screen; each subsequent user message contains only the new terminal output.

terminus-2 episode prompt (first user message, excerpt)

```
You are an AI assistant tasked with solving command-line tasks in a Linux
environment. You will be given a task description and the output from
previously executed commands. Your goal is to solve the task by providing
batches of shell commands.

Format your response as JSON with the following structure:

{
  "analysis": "Analyze the current state based on the terminal output
    provided. What do you see? What has been accomplished? What still
    needs to be done?",
  "plan": "Describe your plan for the next steps. [...]",
  "commands": [
    {
      "keystrokes": "ls -la\n",
      "duration": 0.1
    }
  ],
  "task_complete": true
}
[...]
IMPORTANT: The text inside "keystrokes" will be used completely verbatim
as keystrokes. Write commands exactly as you want them sent to the
terminal:
- You must end every command with a newline (\n) or it will not execute.
- For special key sequences, use tmux-style escape sequences:
  - C-c for Ctrl+C
[...]
Task Description:
<task text, see below>

Current terminal state:
Current Terminal Screen:
root@b6641dc52806:/app#
```

G.1 CASE STUDY: QEMU-STARTUP

Among the evaluated policies, ActObs→GRPO uniquely solves this task. The decisive difference is guest memory: the other attempts use the default 128 MiB, too little for the image to produce a login prompt on the serial port.

Task description

```
Start the /app/alpine.iso image in qemu in such a way that I can connect
to it via `telnet 127.0.0.1 6665`. When I run telnet I will expect to see
the login prompt; I'll log in. You should just get me to this point.
Start the image in the background and leave it running. Block until it's
ready.
```

The winning attempt begins with the same canonical launch every method uses:

Agent (early)

```
Analysis: File command not installed, but we can still start qemu. Need
to start the ISO image with serial console TCP on port 6665. We'll start
qemu in the background, then wait for it to be ready. [...]

Commands:
qemu-system-x86_64 -m 128M -cdrom /app/alpine.iso
  -serial tcp:127.0.0.1:6665,wait -nographic > /tmp/qemu.log 2>&1 &
```

Terminal (observation)

```
New Terminal Output:

root@b6641dc52806:/app# qemu-system-x86_64 -m 128M -cdrom /app/alpine.iso
-serial tcp:127.0.0.1:6665,wait -nographic > /tmp/qemu.log 2>&1 &
[1] 131
root@b6641dc52806:/app#
```

The VM never becomes reachable, and the log stays empty. Where the low-entropy policies reissue near-identical launches, this trajectory begins a search that changes one launch parameter at a time, killing the previous process between attempts. The session recording contains twelve distinct QEMU invocations before the final one, varying the serial backend, boot flags, drive interface, and, critically, guest memory:

QEMU launch variants attempted (from the session recording, condensed; each launch is followed by a check and a kill)

```
-m 128M ... -serial tcp:127.0.0.1:6665,wait -nographic
-m 128M ... -serial tcp:127.0.0.1:6665,server,nowait -nographic
-m 128M ... -serial tcp:...,server,nowait -nographic -nodefaults
-m 128M ... -boot c -serial tcp:...,server,nowait -nographic
-m 128M -drive file=/app/alpine.iso,if=ide,format=raw -serial tcp:...
-m 512M ... -serial tcp:...,server,nowait -nographic -nodefaults
-m 1024M ... -serial tcp:...,server,nowait -nographic -nodefaults
-m 128M ... -serial stdio -nographic
-m 256M ... -boot d -serial tcp:...,server,nowait -nographic -nodefaults
-m 512M ... -boot d -serial tcp:...,server,nowait -nographic -nodefaults
-m 256M ... -serial stdio -nographic -nodefaults
-m 1024M ... -serial file:/tmp/serial_fifo -nographic -nodefaults
```

Agent (mid-episode, during the memory sweep)

```
Analysis: We have killed the latest qemu process. Need to start the ISO
with a serial console TCP server. The ISO is valid. The previous attempts
were killed due to high CPU, indicating the image might be stuck booting.
[...]

Commands:
qemu-system-x86_64 -m 512M -cdrom /app/alpine.iso
  -serial tcp:127.0.0.1:6665,server,nowait -nographic -nodefaults
  > /tmp/qemu9.log 2>&1 &
```

Between launches the trajectory also runs a forensic detour, inspecting the image with `od`, `file`, `xorriso`, and `7z`, and attempting a FIFO-plus-`socat` serial relay. None of these succeeds, and the agent returns to the direct launch, now with the large memory setting:

Agent (late)

```
Analysis: [...] We should now start QEMU with the correct command. We'll
also need to block until it's ready, meaning we need to wait for the
system to boot. [...]

Commands:
qemu-system-x86_64 -m 1024M -cdrom /app/alpine.iso
  -serial tcp:127.0.0.1:6665,server,nowait -nographic -nodefaults
  > /tmp/qemu.log 2>&1 &
```

Terminal (observation)

```
New Terminal Output:

root@b6641dc52806:/app# qemu-system-x86_64 -m 1024M -cdrom /app/alpine.iso
-serial tcp:127.0.0.1:6665,server,nowait -nographic -nodefaults >
/tmp/qemu.log 2>&1 &
[1] 2694
root@b6641dc52806:/app#
```

Agent (verification)

```
Analysis: The QEMU process is running (PID 2694). The log file is empty,
indicating that QEMU hasn't produced any output yet. The system may still
be booting. We need to wait for the system to become ready (i.e., the
login prompt appears). [...]

Commands:
nc -z 127.0.0.1 6665 && echo 'Port 6665 is listening'
  || echo 'Port 6665 is not listening'
```

Terminal (observation)

```
New Terminal Output:

root@b6641dc52806:/app# nc -z 127.0.0.1 6665 && echo 'Port 6665 is
listening' || echo 'Port 6665 is not listening'
Port 6665 is listening
root@b6641dc52806:/app#
```

Terminal (observation, after `telnet 127.0.0.1 6665`)

```
New Terminal Output:

root@b6641dc52806:/app# telnet 127.0.0.1 6665 2>&1 | head -5
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
```

The trajectory then blocks, waiting for the guest to finish booting, and is still waiting when the 900-second task wall-clock expires. The verifier nevertheless finds the correctly configured VM running with its login prompt reachable and scores the trial 1.0. The attempt therefore uses its full interaction budget productively and leaves the environment in the required state.

G.2 Case study: fix-git

Task description

```
I just made some changes to my personal site and checked out master, but
now I can't find those changes. Please help me find them and merge them
into master.

Current terminal state:
```

```
Current Terminal Screen:
root@03738d59a47f:/app/personal-site#
```

Every method reaches the key discovery within a few turns; only ActObs→GRPO carries the integration through. The discovery:

Agent (early)

```
Analysis: Only master branch exists. Need to find where the changes are.
Could be that the changes are in a commit that is not yet on master
(maybe we have a local branch that we haven't merged). [...] Let's
examine git reflog [...]

Commands:
git reflog
```

Terminal (observation)

```
New Terminal Output:

root@03738d59a47f:/app/personal-site# git reflog
d7d3e4b (HEAD -> master) HEAD@{0}: checkout: moving from
650dba427e0a9dcd118f41a4c5e35c8017550a5a to master
650dba4 HEAD@{1}: commit: Move to Stanford
c4e38a1 HEAD@{2}: checkout: moving from master to HEAD~1
d7d3e4b (HEAD -> master) HEAD@{3}: reset: moving to d7d3e4b
b0d59cd HEAD@{4}: clone: from
https://github.com/TheMikeMerrill/personal-site.git
root@03738d59a47f:/app/personal-site#
```

The dangling commit `650dba4` ("Move to Stanford") is the lost work. The integration is where the methods separate. A first `git cherry-pick 650dba4` traps the session in a pager:

Terminal (observation, after `git cherry-pick 650dba4`)

```
Current Terminal Screen:
[...]
  -x [N[,...]]  --tabs=[N[,...]]
                  Set tab stops.
  -X  ........  --no-init
                  Don't use termcap init/deinit strings.
  -y [N]  ....  --max-forw-scroll=[N]
                  Forward scroll limit.
```

The trajectory escapes the pager and works through a sequence of failing alternatives, reading each error and switching tactics rather than re-issuing the same command:

Terminal (observation, after `git cherry-pick -m 1 650dba4`)

```
New Terminal Output:

root@03738d59a47f:/app/personal-site# git cherry-pick -m 1 650dba4
error: your local changes would be overwritten by cherry-pick.
hint: commit your changes or stash them to proceed.
fatal: cherry-pick failed
root@03738d59a47f:/app/personal-site#
```

Terminal (observation, after `git apply -p1 <(git show 650dba4)`)

```
New Terminal Output:

root@03738d59a47f:/app/personal-site# git apply -p1 <(git show 650dba4)
/dev/fd/63:17: trailing whitespace.
[...]
error: patch failed: _includes/about.md:1
error: _includes/about.md: patch does not apply
error: patch failed: _layouts/default.html:27
error: _layouts/default.html: patch does not apply
root@03738d59a47f:/app/personal-site#
```

Having already re-applied the commit's content to the working tree by hand, the agent stages it and commits:

Agent (final)

```
Analysis: The cherry-pick failed due to missing editor. We need to commit
the changes that are already staged. We can use git commit -m "Merge
changes from cherry-pick 650dba4". That should work because the changes
are already in the index. Let's try that.

Commands:
git commit -m "Merge changes from cherry-pick 650dba4"
```

Terminal (observation)

```
New Terminal Output:

root@03738d59a47f:/app/personal-site# git commit -m "Merge changes from
cherry-pick 650dba4"
[master ef1fb37] Merge changes from cherry-pick 650dba4
 Date: Sat Sep 13 17:54:33 2025 +0000
 2 files changed, 19 insertions(+), 17 deletions(-)
root@03738d59a47f:/app/personal-site#
```

The verifier scores the trial 1.0. For contrast, the ActionSFT→GRPO attempts on this task twice reach the same integration step (`git show 650dba4` followed by a merge attempt) and then lose the session to pager handling and an incorrect full-file rewrite (Appendix F). The difference in these transcripts is not the initial plan, which is shared, but the ability to keep sampling new tactics in response to observed errors until one succeeds.

H LIMITATIONS

Domain coverage. Our experiments are confined to terminal agents. Aider-polyglot adds a cross-domain evaluation on multilingual code editing. The results do not yet establish whether observation supervision provides the same benefit for agents operating in coding, web, GUI, or embodied environments. These settings also expose richer observations than terminal text, making them an important direction for testing the generality of the method.

Model and data scope. We study Qwen3 at 4B and 8B, which provides a cross-scale comparison but not a cross-family one because the models share a tokenizer and chat template. All SFT trajectories are generated by a single teacher, DeepSeek-V3.2, and contain only the consequence realized after each teacher action. Experiments with additional model families, teachers, and data collection pipelines would separate these factors.

Scaffold parity. The SFT, RL, and evaluation pipelines are not fully identical. They differ in observation formatting and context construction, and RL uses a fresh, non-interactive shell whereas Terminal-Bench 2.0 uses a persistent `tmux` session. Command handling, requested waits, generation templates, and termination and scoring behavior also vary across pipelines. Because every method within a stage uses the same scaffold, the central ActObs versus ActionSFT comparisons remain controlled, although these mismatches may affect absolute scores and comparisons between SFT and RL endpoints.