PREDICTIONS ABOUT UNCHOSEN ACTIONS IMPROVE LEARNING FROM AGENT TRAJECTORIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Interactive agents generate trajectories that are later reused for training, yet each decision records an outcome only for the action that was executed. We ask whether the same realized interaction becomes more useful when it also retains pre-action predictions about actions that were available but not chosen. We introduce *Candidate logging*, which stores the consequence predicted by the trajectory-generating model (the *Source*) for each candidate action without executing the alternatives. In a controlled latent-rule environment, where an unobserved rule maps mode–action pairs to stochastic consequences, *Candidate logging* improves held-out consequence prediction from 69.0% to 75.7% (+6.7 percentage points) compared with retaining a prediction only for the executed action. The improvement transfers to autonomous behavior, with no logged predictions or reasoning available at evaluation: relative to the same baseline, task success rises by 5.1 points on ALFWorld `valid_unseen` (to 52.5%) and by 4.7 points on WebShop (to 39.7%). On both tasks, it also outperforms a baseline that retains the Source's native pre-action reasoning. Control conditions that keep the logged fields but remove or misalign their action-specific content do not reproduce the controlled gain, which increases as predictions for more unchosen actions are retained and decreases as downstream model capacity and training data grow. These patterns are consistent with explicit predictions reducing the need to recover action–consequence relationships from limited realized experience.

1 INTRODUCTION

Interactive agents generate trajectories that are later reused for imitation learning, offline reinforcement learning, and sequence-based agent training (Ross et al., 2011; Fujimoto et al., 2019; Chen et al., 2021). Language agents similarly reuse interaction records for reflection, adaptation, and skill acquisition (Shinn et al., 2023; Wang et al., 2024), as well as for fine-tuning (Chen et al., 2023; Zeng et al., 2024). At each decision state, however, multiple actions may be available while only one is executed. The resulting trajectory records the selected action and its realized outcome, but provides no environment outcome for the alternatives. A downstream model (the *Learner*) must therefore infer how different actions relate to their consequences from records in which each visited state reveals only one realized action–outcome pair.

Knowledge distillation, learning using privileged information, and supervision from intermediate reasoning show that additional training-time signals can improve downstream learning (Hinton et al., 2015; Vapnik & Vashist, 2009; Lopez-Paz et al., 2016; Hsieh et al., 2023; Hu & Clune, 2023). In an interactive trajectory, the trajectory-generating model (the *Source*) can provide such information about more than the executed action: using only pre-action information, it can also predict the consequences of the other actions available at the same state. **We ask whether retaining these action-indexed predictions for unchosen actions can make the same realized interaction more useful for downstream learning.**

We study this question through *Candidate logging*. At each decision state, the Source predicts the consequence of each candidate action. *Candidate* stores these action-indexed predictions alongside the realized transition, while only the selected action is executed and the remaining actions receive no environment outcomes. We freeze the realized interaction and construct matched training views from the same record. *Public* retains the realized interaction, *Chosen* additionally retains the pre-

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.

diction for the executed action, and *Candidate* retains predictions for all candidate actions. When the Source uses native thinking (reasoning it emits before acting), we also compare against *Native Thinking*, which retains that reasoning. *Candidate* predictions are elicited through separate Source probes that do not alter the frozen interaction. Within each Source setting, the episodes, executed actions, and realized outcomes are matched across training views, and all added training-time fields are removed at evaluation. Figure 1 summarizes the setup.

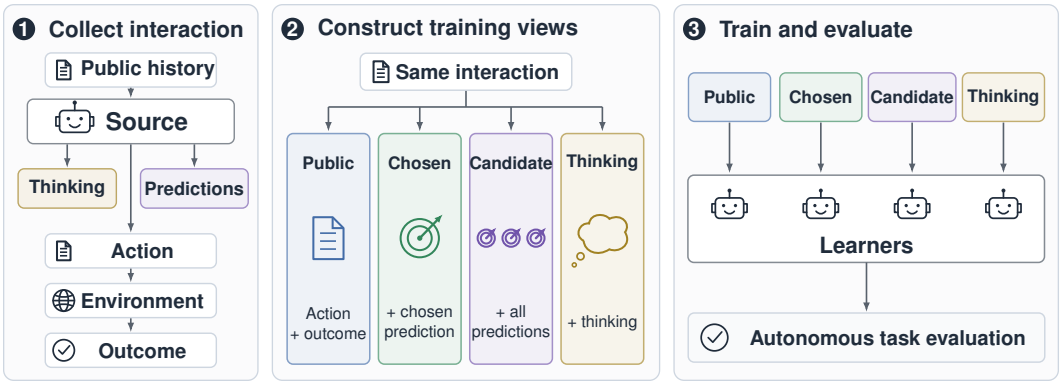


Figure 1: Source–Logger–Learner design. All training views share the same realized interaction: only the chosen action is executed, and the executed actions and realized outcomes are identical across views. *Public* retains this realized record, *Chosen* additionally retains the Source's predicted consequence for the executed action, *Candidate* retains predicted consequences for all candidate actions, and *Native Thinking* retains the Source's native pre-action reasoning. Each view trains a separate Learner, and all added training-time fields are removed at evaluation. Native Thinking is abbreviated as Thinking in the figure.

In a controlled latent-rule environment, where an unobserved rule maps mode–action pairs to stochastic consequences, ***Candidate* improves held-out consequence prediction over *Chosen* by 6.7 percentage points**. The benefit transfers to autonomous behavior: relative to *Chosen*, *Candidate* improves task success from 47.5% to 52.5% (**+5.1 points**) on ALFWorld `valid_unseen` (Shridhar et al., 2021) and from 35.0% to 39.7% (**+4.7 points**) on WebShop (Yao et al., 2022), while also outperforming *Native Thinking* in both environments. The controlled gain persists when the Source uses native thinking and across all nine pairings of three Source and three Learner model families. Controls that preserve the logged fields but replace predictions with token-matched neutral text, make the same prediction for every action, or reassign predictions to other actions do not reproduce the improvement. The gain increases as predictions for more unchosen actions are retained, and is larger for smaller Learners and with less downstream training data.

We make three contributions. First, we introduce *Candidate logging*, which augments a fixed realized trajectory with action-indexed pre-action consequence predictions without executing the alternative actions. Second, we show that predictions about unchosen actions improve both consequence learning and autonomous task performance beyond *Chosen* and *Native Thinking*. Third, we characterize when these predictions are most useful: the benefit depends on informative, action-aligned predictions, increases with coverage of unchosen actions, and is larger when the Learner has less capacity or less training data. When agents already evaluate alternatives during planning or search, *Candidate logging* can turn that computation into reusable training signal.

2 Problem Setup

Formally, the Source interacts with an environment, the Logger constructs training views, and a separate Learner is trained on each view. At decision step t, let H_t denote the public pre-action history. The Source produces action a_t, which is executed in the environment and yields outcome Y_t. The common realized record is

$$B_t = (H_t, a_t, Y_t),$$

and is held fixed across logger conditions.

For predictive logging, let $\mathcal{A}_t = \{a_1, \ldots, a_M\}$ denote the candidate set. In the controlled task, this is the fixed environment action set. In ALFWorld, candidates are drawn from the environment-provided admissible-action list at the saved pre-action state; in WebShop, candidates are proposed from saved public pre-action information. The executed action is included in each candidate set.

Using only pre-action information, separate probes query the Source for the consequence of each candidate action, $c_S(a, H_t)$ for $a \in \mathcal{A}_t$, giving the action-indexed predictive map

$$C_t = \{(a, c_S(a, H_t)) : a \in \mathcal{A}_t\}.$$

Each probe receives an immutable copy of H_t, with no realized outcome or future information. Probes do not write to the Source history, alter the environment, or influence action selection.

The Logger constructs four training views:

$$D_{\text{Public}} = B_t, \qquad D_{\text{Chosen}} = B_t \oplus c_S(a_t, H_t),$$

$$D_{\text{NT}} = B_t \oplus R_t, \qquad D_{\text{Candidate}} = B_t \oplus C_t,$$

where $\oplus$ denotes the inclusion of an additional training field and NT denotes *Native Thinking*. R_t is the native reasoning produced during the live generation that yields a_t. It precedes action execution and the realized outcome Y_t, and is excluded from the public history H_t and from all consequence-probe contexts. Native reasoning from previous decisions is not carried forward across turns.

The primary contrast is *Candidate* versus *Chosen*: both retain the executed-action prediction in the same representation, while only *Candidate* adds predictions for unchosen candidate actions. Source predictions and retained reasoning are unavailable to the Learner at evaluation.

Learner training interface. Each decision state yields a common task example and, when applicable, a logger-specific auxiliary example. Task and auxiliary examples are processed as separate causal sequences, so task predictions cannot attend to auxiliary targets. Common task examples, task-example exposure, and optimizer-update counts are identical across logger conditions, while conditions with auxiliary supervision process an additional auxiliary batch and incur additional computation. For autonomous action learning, the task objective supervises action selection and the predictive auxiliary objective supervises consequences; both update the same Learner parameters. Appendix D gives the training details. The ALFWorld example below illustrates the shared policy target and the additional consequence-prediction targets used by *Chosen* and *Candidate*.

Training Example — **ALFWorld**

Task: Put a clean apple in the fridge.

Current observation (abridged): You are at countertop 1, where you see apple 1. Sinkbasin 1 and fridge 1 are available.

All conditions share the policy target below; *Candidate* adds predictions for unchosen candidate actions to the *Chosen* supervision.

Training signal	Target / supervision
Common policy	`take apple 1 from countertop 1`
Chosen **auxiliary**	`take apple 1 from countertop 1` `→ (take, apple 1, held_by_agent, success)`

Candidate **auxiliary**	`take apple 1 from countertop 1`
	`→ (take, apple 1, held_by_agent, success)`
	`go to sinkbasin 1`
	`→ (navigate, sinkbasin 1, at:sinkbasin 1, success)`
	`go to fridge 1`
	`→ (navigate, fridge 1, at:fridge 1, success)`

3 CONTROLLED EVALUATION

3.1 CONTROLLED LATENT-RULE ENVIRONMENT

We first evaluate the logging intervention in a controlled latent-rule environment. Each six-turn episode is governed by one of six unobserved rules that map the current Mode and action to one of two consequences, Signal P or Signal Q. The environment has two Modes and three actions (A, B, and C); the rules form action-driven, mode-driven, and action–mode interaction families. Under an interaction rule, for example, action B favors P in Mode 1 and Q in Mode 2, while action C shows the opposite pattern. For each Mode–action pair, the rule-consistent signal occurs with probability 0.9, so individual transitions provide noisy evidence and a Learner must accumulate evidence across turns to infer the active rule. At every decision, one action is executed while the other two remain unexecuted, so the realized trajectory reveals only one branch of the local action–consequence structure. Appendix A.1 gives the complete rule mappings and stochastic outcome model.

The primary corpus contains 3,000 episodes, split into 2,400 training, 300 development, and 300 test episodes. The corpus is frozen before logger construction, so the realized interactions and train/dev/test assignments are identical across conditions.

The Source is `Qwen/Qwen3-8B` and the Learner is `Qwen/Qwen3-1.7B-Base`, adapted with LoRA. We train ten paired Learner seeds per logger condition. At evaluation, the Learner receives a held-out public history and a proposed action and predicts whether its consequence will be Signal P or Signal Q. Held-out episodes draw from the same six latent rules, so evaluation measures inference of the active rule within new episodes. Model and optimization settings are given in Appendix D.

The primary Source has native thinking disabled (Thinking-OFF); a separate corpus from the same Source with native thinking enabled (Thinking-ON) tests robustness (Section 3.3), with logger comparisons made within each corpus.

3.2 MAIN LOGGER COMPARISON: THINKING-OFF

Table 1 compares the three training views constructed from the primary Thinking-OFF corpus. *Public* and *Chosen* differ by less than one point (68.2% and 69.0%), whereas *Candidate* reaches 75.7%, exceeding *Chosen* by 6.7 percentage points (95% CI $[5.5, 7.8]$).

Table 1: Primary controlled experiment (Source native thinking disabled): held-out consequence-prediction accuracy (%). The right panel reports the paired contrast with its 95% confidence interval.

Training view	Accuracy (%)
Public	68.2
Chosen	69.0
Candidate	75.7

Paired contrast	Difference (pp)	95% CI (pp)
Candidate − *Chosen*	+6.7	$[5.5, 7.8]$

3.3 LOGGER COMPARISON: THINKING-ON

Within the Thinking-ON corpus, the *Candidate* advantage persists (Table 2). *Candidate* reaches 76.8%, compared with 70.8% for *Chosen* and 73.5% for *Native Thinking*, corresponding to gains of 6.0 and 3.3 percentage points, respectively (95% CIs $[4.7, 7.2]$ and $[2.2, 4.5]$). The result shows that predictions about unchosen actions remain useful when the Source itself performs native reasoning.

3.4 CROSS-FAMILY ROBUSTNESS

We cross three Source families with three Learner families in the controlled environment (Table 3; checkpoints in Appendix D). *Candidate* yields positive mean gains over *Chosen* in all nine Source–Learner pairings, ranging from 4.8 to 6.7 percentage points.

Table 2: Thinking-ON extension: held-out consequence-prediction accuracy (%); all four views come from one frozen corpus. The last column reports *Candidate* minus the row view, with 95% confidence intervals in brackets.

Training view	Accuracy (%)	*Cand.* − view (pp)
Public	70.0	–
Chosen	70.8	+6.0 [4.7, 7.2]
Native Thinking	73.5	+3.3 [2.2, 4.5]
Candidate	76.8	–

Table 3: *Candidate–Chosen* gain across Source and Learner model families (percentage points).

Source \ Learner	Qwen	Llama	Gemma
Qwen	+6.7	+5.8	+5.1
Llama	+5.9	+5.3	+4.8
Gemma	+6.2	+5.5	+5.0

4 What Makes Predictions About Unchosen Actions Useful?

The *Candidate* advantage depends on both what predictive structure is retained and how readily the Learner can recover that structure from realized experience. We first test whether the benefit depends on informative, action-aligned predictions, then vary how many unchosen-action predictions are retained, and finally examine how the gain changes with Learner capacity and training data. The following analyses use a Thinking-OFF Source.

4.1 Informative and Action-Aligned Predictions

Token-matched neutral fields preserve serialized length and prediction-field positions while replacing predictive content with episode-independent tokens; they reach 68.4%. *Uniform* candidate predictions preserve the action-indexed field structure but assign the same prediction to every action and reach 68.8%. Both remain near *Chosen* at 69.0%, whereas aligned *Candidate* reaches 75.7% (Figure 2A). Neither additional length nor prediction-field structure reproduces the *Candidate* gain.

An alignment control preserves the executed-action prediction and swaps the predictions assigned to the two unchosen actions at each decision state, preserving the prediction values. Action-Misaligned *Candidate* reaches 69.6%, compared with 75.7% for aligned *Candidate*, a difference of 6.1 percentage points (95% CI $[5.7, 6.6]$). This result indicates that the gain depends on action–prediction alignment.

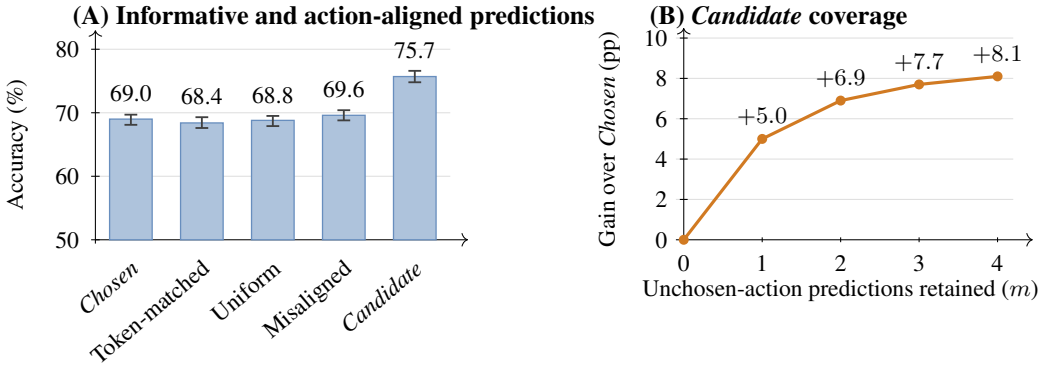


Figure 2: What makes predictions about unchosen actions useful? **(A)** Added length, non-informative prediction fields, and action-misaligned predictions do not reproduce the aligned *Candidate* gain. **(B)** *Candidate* gain increases with the number of retained unchosen-action predictions, with diminishing returns; retaining two of four alternatives recovers about 85% of the full gain. Error bars in **(A)** show pointwise 95% bootstrap confidence intervals across Learner training seeds; **(B)** reports mean paired gains across seeds.

4.2 *Candidate* COVERAGE

We next ask how much coverage of unchosen actions is needed to obtain the *Candidate* benefit. Because the primary three-action environment provides only two unchosen alternatives per decision, we use a five-action extension for this analysis (Appendix A.1). Within this extension, *Chosen* retains no unchosen branches ($m = 0$), whereas full *Candidate* retains all four ($m = 4$). Intermediate conditions retain predictions for $m = 1, 2, 3$ unchosen actions, sampled uniformly without replacement at each decision state. All coverage conditions are constructed from the same frozen five-action trajectories, and every retained prediction remains aligned with its action.

Performance improves with candidate coverage, with diminishing returns (Figure 2B). Relative to *Chosen* ($m = 0$), the gain is $+5.0$ percentage points with one retained unchosen-action prediction, $+6.9$ with two, $+7.7$ with three, and $+8.1$ with all four. Retaining only two of the four unchosen alternatives therefore recovers 85% of the full *Candidate* gain.

4.3 WHEN IS *Candidate* MOST USEFUL?

Returning to the primary three-action setting, we examine how the *Candidate–Chosen* gain varies with Learner capacity and training data. The gain decreases from $+9.1$ to $+3.9$ percentage points as Learner size increases from 0.6B to 8B parameters (Figure 3A). It also decreases from $+12.3$ to $+2.5$ points as the training corpus grows from 250 to 10,000 episodes (Figure 3B; complete values in Appendix Table 9).

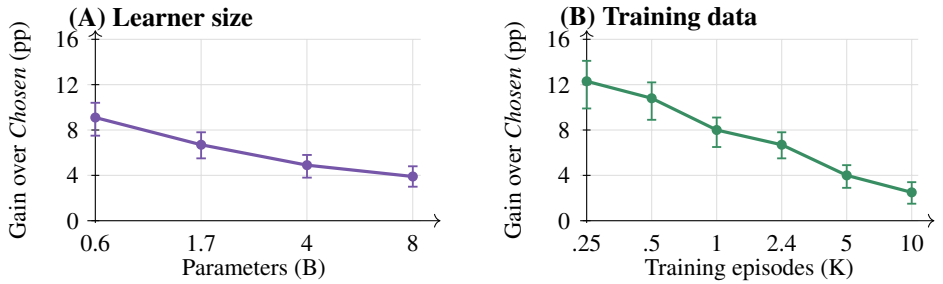


Figure 3: *Candidate* gains are larger with smaller Learners and less training data. Its gain over *Chosen* decreases with **(A)** larger Learners and **(B)** more downstream training episodes. Error bars denote pointwise 95% bootstrap confidence intervals across paired Learner-seed differences.

Given the known rule family, prior, and transition probabilities, the posterior predictive can be computed exactly from public history (Appendix A.6). The larger gains for smaller Learners and less training data are consistent with explicit predictive supervision easing learning from realized trajectories.

5 AUTONOMOUS MULTI-STEP TASK EVALUATION

We next test whether the logging advantage extends to autonomous multi-step behavior in two interactive domains: ALFWorld (Shridhar et al., 2021), which evaluates household-task completion in seen and unseen environments, and WebShop (Yao et al., 2022), which requires agents to search, navigate product pages, select requested attributes, and complete purchases from natural-language shopping instructions. For both benchmarks, the Source is `Qwen/Qwen3-32B` and the downstream Learner is `Qwen/Qwen3-4B-Base`. The Source generates the frozen live trajectories with native thinking enabled and a fixed one-shot demonstration, while consequence-prediction probes use the same checkpoint with native thinking disabled. All four views are constructed from the same frozen interaction corpus within each benchmark, and the Learner acts autonomously at evaluation without the Source, constructed candidate set, retained reasoning, or prediction fields. Confidence intervals reflect variability across paired Learner training seeds, with the Source corpus and evaluation tasks held fixed (Appendix E).

5.1 ALFWorld Autonomous Task Success

The ALFWorld corpus contains 3,553 Source trajectories: 3,200 for Learner training and 353 for development. Evaluation uses all 140 `valid_seen` and 134 `valid_unseen` games. Candidate sets are constructed offline from the environment-provided admissible-action list at each saved pre-action state.

For every candidate action, the Source predicts a normalized immediate consequence offline from the saved pre-action context. Consequences are represented as (*effect*, *target*, *relation*, *status*) tuples. All four Learner conditions receive the same environment-provided admissible-action list during policy training and autonomous evaluation, with no in-context demonstration.

Candidate reaches 52.5% task success on `valid_unseen`, exceeding *Chosen* by 5.1 percentage points (95% CI [4.0, 6.1]) and *Native Thinking* by 3.3 points (95% CI [1.2, 4.9]). The same ordering holds for the means on `valid_seen` (Table 4). The Source uses a larger model and the one-shot action-generation setup above, so its success rates are reported only as contextual references.

Appendix B gives the dataset construction, prediction schema, prediction diagnostics, and seed-level results.

Table 4: ALFWorld autonomous task success (%). Learner means are over five paired training seeds on the complete standard evaluation splits; the Source row is one evaluation reported as a contextual reference. The lower panel reports paired contrasts on `valid_unseen` with pointwise 95% confidence intervals.

Training view	`valid_seen`	`valid_unseen`
Source (`Qwen3-32B`)	54.3	64.9
Public	49.1	45.8
Chosen	50.6	47.5
Native Thinking	51.7	49.3
Candidate	**54.1**	**52.5**
Unseen contrast	Difference (pp)	95% CI (pp)
Candidate − *Chosen*	+5.1	[4.0, 6.1]
Candidate − *Native Thinking*	+3.3	[1.2, 4.9]

5.2 WebShop Autonomous Task Success

We next evaluate *Candidate logging* in multi-step web shopping using WebShop (Yao et al., 2022). The frozen Source corpus contains 2,000 training trajectories and 300 disjoint development trajectories collected from the official training split; final evaluation uses the standard 500-task test set. Candidate actions are constructed offline from saved pre-action states without access to the frozen live action, which is inserted afterward if absent; candidate sets are capped at six. Each candidate action receives a normalized (effect, target, status) prediction of its immediate consequence.

At autonomous evaluation, the Learner receives only the shopping instruction, current observation, currently available actions, and its own realized interaction history.

Candidate reaches 39.7% task success, compared with 35.0% for *Chosen*, 36.0% for *Native Thinking*, and 33.0% for *Public* (Table 5). *Candidate* exceeds *Chosen* by 4.7 percentage points (95% CI [3.6, 5.8]) and *Native Thinking* by 3.6 points (95% CI [2.6, 4.6]). The frozen `Qwen3-32B` Source reaches 45.6%.

Appendix C gives the dataset construction, candidate-construction procedure, consequence schema, and seed-level results; complete prompts are provided in Appendix F.

Table 5: WebShop autonomous task success (%). Learner means are over five paired training seeds on the standard 500-task test set. The Source row is one evaluation reported as a contextual reference. The lower panel reports paired contrasts with pointwise 95% confidence intervals.

Training view	Task success	
Source (`Qwen3-32B`)	45.6	
Public	33.0	
Chosen	35.0	
Native Thinking	36.0	
Candidate	39.7	
Contrast	Difference (pp)	95% CI (pp)
Candidate − *Chosen*	+4.7	[3.6, 5.8]
Candidate − *Native Thinking*	+3.6	[2.6, 4.6]

6 Related Work

Learning from alternative and counterfactual actions. Early Experience executes alternatives at expert-trajectory states and uses the resulting states for next-state prediction or self-reflection supervision (Zhang et al., 2026). SAND rolls out candidate actions to construct deliberation supervision from observations and rewards (Xia et al., 2025). Agent Q converts search-derived action values into preference pairs for policy optimization (Putta et al., 2024), while ACT trains action-quality judgments on expert-versus-sampled action pairs (Liu et al., 2026). Counterfactually-Guided Policy Search uses structural causal models to infer alternative outcomes of logged episodes for policy evaluation and search (Buesing et al., 2019); counterfactual data augmentation constructs additional transitions under structural assumptions about environment dynamics (Pitis et al., 2020). *Candidate* instead elicits Source predictions using only saved pre-action information and retains them as auxiliary targets for a separate Learner. It neither performs causal counterfactual inference conditioned on realized outcomes nor uses the predictions to evaluate alternative policies.

World models for language agents. WebDreamer simulates candidate web-action outcomes and evaluates them before selecting an action (Gu et al., 2025); WMA similarly uses predicted state transitions to guide web-agent action selection (Chae et al., 2025). Reinforcement World Model Learning trains action-conditioned world models using reinforcement-learning rewards based on semantic agreement between simulated and realized next states (Yu et al., 2026). *Candidate* freezes the realized interaction, probes candidate actions from saved pre-action information, and uses Source predictions as auxiliary targets for a separate Learner without observing unchosen outcomes.

Distillation, privileged information, and reasoning supervision. Distillation and privileged-information methods use teacher-provided or training-only signals to supervise a student (Hinton et al., 2015; Vapnik & Vashist, 2009; Lopez-Paz et al., 2016). Policy distillation transfers teacher policy or action-value information to a student (Rusu et al., 2016; Czarnecki et al., 2019); intermediate or teacher-generated reasoning can likewise serve as supervision (Hsieh et al., 2023; Hu & Clune, 2023). *Candidate* distills predicted environment consequences indexed by candidate actions, whereas policy distillation typically transfers teacher action preferences or values. Free-form native reasoning may discuss alternatives without systematically covering the full candidate set or explicitly linking each action to a predicted consequence. Our *Native Thinking* baseline directly compares retaining this reasoning with explicit candidate–consequence supervision.

Model-based learning and auxiliary prediction. Model-based methods use predictions of future states to support planning and policy learning (Sutton, 1991; Chua et al., 2018; Hafner et al., 2020; Schrittwieser et al., 2020). Learned dynamics also supply imagined experience for value estimation or policy optimization, as in MVE, MBPO, and MOPO (Feinberg et al., 2018; Janner et al., 2019; Yu et al., 2020). Predictive auxiliary objectives also exploit interaction: UNREAL adds auxiliary control and reward-prediction tasks (Jaderberg et al., 2017), while SPR predicts future latent representations conditioned on actions (Schwarzer et al., 2021).

Agent trajectory construction and synthesis. Agent trajectories support imitation, sequence-based learning, and agent adaptation (Ross et al., 2011; Chen et al., 2021; 2023; Zeng et al., 2024). Agent-FLAN redesigns agent-tuning corpora and their training signals (Chen et al., 2024), while Synatra synthesizes web demonstrations from indirect procedural knowledge (Ou et al., 2024). AgentBank scales fine-tuning over large collections of interaction trajectories (Song et al., 2024), and AgentGym trains and evolves agents through interaction across diverse environments (Xi et al., 2024). The distinction is that *Candidate* does not expand or replace the trajectory corpus: it preserves the visited states, executed actions, and realized outcomes, adding only Source predictions about unexecuted candidate actions.

7 Discussion

A realized trajectory reveals the consequence of only one executed action at each visited state. *Candidate* increases predictive coverage at that same state by retaining the Source's predictions for other candidate actions, without collecting additional environment transitions. The controlled analyses indicate that this coverage, rather than additional serialized content alone, is important. Token-matched neutral fields and uniform predictions do not reproduce the gain, while reassigning predictions to the wrong actions substantially reduces it. The benefit also grows as predictions for more unchosen actions are retained. Together, these results point to informative, action-aligned predictive coverage as the useful training signal.

One possible explanation is that *Candidate* reduces the amount of local action–consequence structure that a Learner must reconstruct from realized experience alone. Consistent with this view, the *Candidate–Chosen* gain decreases as Learner capacity and downstream training data increase. In the controlled environment, the exact posterior predictive provides a useful reference: given the rule family and transition model, the relevant action–consequence relationships are recoverable from public history, whereas *Candidate* exposes a Source-estimated version of that local structure during training. Taken together, these patterns are consistent with a reconstruction account: *Candidate* is most useful when the Learner has limited capacity or limited realized experience from which to recover action–consequence structure.

Importantly, *Candidate* does not rely on oracle counterfactual labels. The Source's predictions are fallible beliefs about what alternative actions would produce, and unchosen actions are never executed to obtain their counterfactual outcomes. In ALFWorld, for example, executed-action consequence predictions reach 77.8% exact match rather than perfect accuracy, yet *Candidate* still improves downstream autonomous performance. Free-form native reasoning may contain comparisons among alternatives, but *Candidate* makes the association between each action and its predicted consequence explicit. Under our training interface, its advantage over *Native Thinking* indicates that explicit action-indexed predictions can be more useful than retaining the Source's free-form native reasoning alone. A useful next step is to disentangle how representation structure, predictive content, coverage, and supervision length each contribute to this advantage.

More broadly, *Candidate* suggests that decision-time predictive computation can remain useful after the decision itself has been made. Model-based methods use predicted transitions or imagined trajectories to support planning, action selection, and policy learning (Sutton, 1991; Chua et al., 2018; Hafner et al., 2020; Schrittwieser et al., 2020). *Candidate* preserves action-indexed predictions as auxiliary supervision for later Learners. Future work can examine longer-horizon predictions, larger candidate spaces, and tighter integration with native planning and search.

8 Conclusion

Candidate logging augments a fixed realized trajectory with Source predictions about unchosen actions, improving consequence prediction and downstream autonomous performance without executing those alternatives. The controlled experiments show that the gain depends on informative, action-aligned predictions, increases with candidate coverage, and decreases as Learner capacity and training data grow. These findings suggest that predictions about what could happen under unchosen actions can increase the training value of what actually happened.

Ethics Statement

The experiments use a synthetic latent-rule environment, simulated ALFWorld household tasks through a text interface, and the simulated WebShop shopping environment, and introduce no new human-subject data.

Reproducibility Statement

The appendix reports the checkpoints, dataset construction, logger interfaces, candidate-generation procedures, evaluation settings, diagnostic checks, and statistical aggregation procedures for the three experimental settings, including the complete WebShop Source, candidate-proposal, consequence-prediction, and Learner policy prompts and seed-level success counts for all four training views in ALFWorld and WebShop. The experiment code, configurations, and evaluation scripts will be released on GitHub to support reproduction of the reported results.

AI Use Statement

An LLM-based writing and coding assistant was used for language editing, code review, figure generation support, and preparation of experiment scripts. The authors reviewed all generated material and remain responsible for the analyses, claims, and final text.

References

Lars Buesing, Theophane Weber, Yori Zwols, Sebastien Racaniere, Arthur Guez, Jean-Baptiste Lespiau, and Nicolas Heess. Woulda, coulda, shoulda: Counterfactually-guided policy search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJG0voC9YQ.

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 *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.13232.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. FireAct: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023. URL https://arxiv.org/abs/2310.05915.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

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 *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 9354–9366, 2024. doi: 10.18653/v1/2024.findings-acl.557. URL https://aclanthology.org/2024.findings-acl.557/.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Wojciech M. Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. Distilling policy distillation. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 1331–1340, 2019. URL https://proceedings.mlr.press/v89/czarnecki19a.html.

Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I. Jordan, Joseph E. Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018. URL https://arxiv.org/abs/1803.00101.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2052–2062, 2019. URL https://proceedings.mlr.press/v97/fujimoto19a.html.

Gemma Team. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024. URL https://arxiv.org/abs/2408.00118.

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. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=c6l7yA0HSq.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8003–8017, 2023. doi: 10.18653/v1/2023.findings-acl.507. URL https://aclanthology.org/2023.findings-acl.507/.

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

Shengran Hu and Jeff Clune. Thought cloning: Learning to think while acting by imitating human thinking. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJ6yPD5xg.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://arxiv.org/abs/1906.08253.

Weize Liu, Minghui Liu, Sy-Tuyen Ho, Souradip Chakraborty, Xiyao Wang, and Furong Huang. Agentic critical training. *arXiv preprint arXiv:2603.08706*, 2026. URL https://arxiv.org/abs/2603.08706.

David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1511.03643.

Meta AI. Introducing Meta Llama 3.1: Our most capable models to date. https://ai.meta.com/blog/meta-llama-3-1/, 2024a.

Meta AI. Llama 3.2 model card. https://github.com/meta-llama/llama-models/blob/main/models/llama3_2/MODEL_CARD.md, 2024b.

Tianyue Ou, Frank F. Xu, Aman Madaan, Jiarui Liu, Robert Lo, Abishek Sridhar, Sudipta Sengupta, Dan Roth, Graham Neubig, and Shuyan Zhou. Synatra: Turning indirect knowledge into direct demonstrations for digital agents at scale. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/a6a6891cf1dfc64d664f086cf5976e93-Abstract-Conference.html.

Silviu Pitis, Elliot Creager, and Animesh Garg. Counterfactual data augmentation using locally factored dynamics. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/file/294e09f267683c7ddc6cc5134a7e68a8-Paper.pdf.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent Q: Advanced reasoning and learning for autonomous AI agents. *arXiv preprint arXiv:2408.07199*, 2024.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, 2011.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1511.06295.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588: 604–609, 2020. doi: 10.1038/s41586-020-03051-4.

Max Schwarzer, Ankesh Anand, Rishabh Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=uCQfPZwRaUu.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R. Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://papers.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.03768.

Yifan Song, Weimin Xiong, Xiutian Zhao, Dawei Zhu, Wenhao Wu, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. AgentBank: Towards generalized LLM agents via fine-tuning on 50000+ interaction trajectories. *arXiv preprint arXiv:2410.07706*, 2024. URL https://arxiv.org/abs/2410.07706.

Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163, 1991. doi: 10.1145/122344.122377.

Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5–6):544–557, 2009. doi: 10.1016/j.neunet.2009.06.042.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. AgentGym: Evolving large language model-based agents across diverse environments. *arXiv preprint arXiv:2406.04151*, 2024. URL https://arxiv.org/abs/2406.04151.

Yu Xia, Yiran Jenny Shen, Junda Wu, Tong Yu, Sungchul Kim, Ryan A. Rossi, Lina Yao, and Julian McAuley. SAND: Boosting LLM agents with self-taught action deliberation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3062–3077. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.152.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757, 2022.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y. Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://arxiv.org/abs/2005.13239.

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. *arXiv preprint arXiv:2602.05842*, 2026. URL https://arxiv.org/abs/2602.05842.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling generalized agent abilities for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3053–3077, 2024. doi: 10.18653/v1/2024.findings-acl.181. URL https://aclanthology.org/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. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=N3dXUHY5dD.

A CONTROLLED EXPERIMENT

This section specifies the controlled three-action environment and reports the supporting analyses. Unless stated otherwise, the controls, sweeps, audits, posterior-predictive construction, and cost accounting in this appendix belong to the primary controlled experiment: the `Qwen/Qwen3-8B` Source with native thinking disabled and its *Public*, *Chosen*, and *Candidate* views. The Thinking-ON extension is specified in Appendix A.2. The candidate-coverage analysis of Section 4.2 uses the five-action extension specified below and is reported in Appendix A.4.

A.1 ENVIRONMENT, LATENT RULES, AND DATASET CONSTRUCTION

Main three-action environment. The main controlled comparison uses three candidate actions A, B, and C, two public Modes, and binary consequences Signal P and Signal Q. Each episode is governed by one of six latent rules, sampled uniformly and held fixed throughout the episode. The six rules form three paired structural families: action-driven rules depend on the selected action but not Mode; mode-driven rules depend on Mode but not the selected action; and interaction rules assign Mode-dependent consequence profiles to the actions. Within each family, the paired rule reverses Signal P and Signal Q. Table 6 gives the canonical consequence for every Mode–action combination.

At each turn, the public Mode M_t is sampled independently and uniformly from $\{M_1, M_2\}$. The Source observes the current public history and Mode and is instructed to choose one action from the fixed action set $\{A, B, C\}$.

Table 6: Canonical consequences in the three-action controlled environment.

Type	Rule	M1 + A	M1 + B	M1 + C	M2 + A	M2 + B	M2 + C
Action-driven	R1	P	Q	Q	P	Q	Q
Action-driven	R2	Q	P	P	Q	P	P
Mode-driven	R3	P	P	P	Q	Q	Q
Mode-driven	R4	Q	Q	Q	P	P	P
Interaction	R5	P	P	Q	P	Q	P
Interaction	R6	Q	Q	P	Q	P	Q

Stochastic outcome model. For every Mode–action pair, the canonical consequence occurs with probability 0.9, while the other signal occurs with probability 0.1. At each decision, exactly one of the three candidate actions is executed and the other two remain unexecuted. Let $f_Z(M_t, a_t)$ denote the canonical consequence under rule Z. The realized outcome is

$$Y_t = \begin{cases} f_Z(M_t, a_t), & \text{with probability } 0.9, \\ \text{the alternative signal}, & \text{with probability } 0.1. \end{cases} \tag{1}$$

The Learner never observes Z directly.

Five-action extension. The five-action environment uses actions $\{A, B, C, D, E\}$ with binary action feature

$$x(A) = 0, \quad x(B) = 1, \quad x(C) = 0, \quad x(D) = 1, \quad x(E) = 0, \tag{2}$$

and mode feature $m(M1) = 0$ and $m(M2) = 1$. With binary output 0 mapped to Signal P and output 1 to Signal Q, the six rules are

$$R1 : y = x(a), \qquad R2 : y = 1 - x(a), \tag{3}$$

$$R3 : y = m, \qquad R4 : y = 1 - m, \tag{4}$$

$$R5 : y = x(a) \text{ XOR } m, \qquad R6 : y = 1 - [x(a) \text{ XOR } m]. \tag{5}$$

Table 7 expands these definitions.

Table 7: Canonical consequences in the five-action controlled extension.

Rule	M1+A	M1+B	M1+C	M1+D	M1+E	M2+A	M2+B	M2+C	M2+D	M2+E
R1	P	Q	P	Q	P	P	Q	P	Q	P
R2	Q	P	Q	P	Q	Q	P	Q	P	Q
R3	P	P	P	P	P	Q	Q	Q	Q	Q
R4	Q	Q	Q	Q	Q	P	P	P	P	P
R5	P	Q	P	Q	P	Q	P	Q	P	Q
R6	Q	P	Q	P	Q	P	Q	P	Q	P

The same 0.9 canonical-outcome reliability applies. The repeated action-feature pattern increases candidate-set size while preserving the six latent rule families and their underlying inference structure. At every decision one action is executed and four remain unchosen. The extension is used specifically for the candidate-coverage analysis.

Pre-action consequence prediction. *Candidate* predictions are generated offline after the realized Source corpus has been frozen. For each saved decision state, each probe receives an immutable copy of the public context H_t available immediately before the frozen action, together with the candidate action being queried. No realized outcome of the current action, future information, or Source-private reasoning is provided. Thus, "pre-action" refers to the probe's information set rather than the wall-clock time at which the query is executed.

In the controlled environment, we instantiate the generic consequence prediction $c_S(a, H_t)$ as the normalized distribution $q_S(Y \mid a, H_t)$ over Signal P and Signal Q. The three-action setting records $q_S(Y \mid a, H_t)$ for each $a \in \{A, B, C\}$, and the five-action extension analogously records predictions for all five actions. Only the frozen live action has a realized environment outcome; unexecuted actions receive Source predictions only.

Dataset generation and freezing. The primary controlled corpus is generated by the `Qwen/Qwen3-8B` Source with native thinking disabled. It contains 3,000 six-turn Source episodes, with the six hidden rules balanced across the dataset. Episodes are split into 2,400 training, 300 development, and 300 test episodes before logger construction. The complete primary corpus is frozen before logger construction. Consequently, all logger conditions have identical episode identities, Modes and public contexts, executed actions, realized outcomes, ordering, and split membership; only the training-time augmentation fields differ under the specified design.

The Thinking-ON extension corpus is generated separately with the same task and Source checkpoint and native thinking enabled, and is frozen before logger construction in the same way. Equality of the realized record across logger views holds within each Source configuration; views from the two corpora are never compared with each other.

For the training-data sweep, the 2,400-episode condition reuses the primary controlled corpus; auxiliary matched corpora, generated with the same latent-rule environment, Source procedure, rule balance, and logging protocol, provide the remaining training-set sizes up to 10,000 episodes. These auxiliary corpora do not alter the latent task or logger definitions. Training-set sizes are nested: each smaller condition is a subset of the next larger corpus, with the evaluation set held fixed across the sweep.

A.2 Logger and Serialization Details

Logger conditions. Table 8 specifies the content views for the two controlled Source configurations. These are field-level semantics. Thinking on or off refers to the Source generation configuration, not to whether reasoning tokens are serialized into the training record.

In the controlled environment, the candidate action set is the fixed alphabet $\{A, B, C\}$, shared across all conditions, and is therefore treated as environment metadata rather than as an additional logger-specific field.

Table 8: Controlled-environment logger conditions by Source configuration.

Logger	Retained content
Primary experiment: Source native thinking disabled	
Public	H_t, a_t, Y_t
Chosen	*Public* plus $q_S(Y \mid a_t, H_t)$
Candidate	*Public* plus $q_S(Y \mid a, H_t)$ for every candidate action a
Thinking-ON extension: Source native thinking enabled	
Public	H_t, a_t, Y_t; emitted reasoning omitted
Chosen	*Public* plus $q_S(Y \mid a_t, H_t)$; emitted reasoning omitted
Native Thinking	*Public* plus native pre-action reasoning R_t
Candidate	*Public* plus $q_S(Y \mid a, H_t)$ for every candidate action a; emitted reasoning omitted

***Native Thinking* implementation.** *Native Thinking* is defined only in the Thinking-ON extension. R_t is the native reasoning produced during the live generation that yields a_t. It precedes action execution and the realized outcome Y_t, and is excluded from the public history H_t and from all consequence-probe contexts. Native reasoning from previous decisions is not carried forward across turns. Native reasoning is generated with a maximum budget of 96 tokens per decision, with earlier termination permitted at the model's natural stopping point, and is not regenerated or elicited post hoc. The *Native Thinking* logger retains R_t for downstream training, whereas *Public*, *Chosen*, and *Candidate* omit that reasoning field from their logged training records. For *Native Thinking*, the retained reasoning trace forms the supervised target of its auxiliary sequence and uses the same target-token loss weighting as the other auxiliary conditions. Common task examples, task-example exposure, and optimizer-update counts are identical across conditions.

Downstream training and evaluation. The Learner checkpoint is `Qwen/Qwen3-1.7B-Base`, adapted with LoRA, with ten paired training seeds per logger condition. Logger conditions train on matched frozen episodes. Source predictions are available only as training-time trajectory content and are absent at evaluation. Held-out evaluation asks the Learner to predict the consequence of a proposed action from public history. Held-out episodes use known rule families, so this measures inference of the active rule within a new episode rather than extrapolation to a novel rule family. The primary corpus uses six-turn episodes. The reconstructability analyses in Section 4.3 use this same three-action environment.

Task and auxiliary examples. In the controlled environment, write $H_t = (h_t, M_t)$, where h_t contains the realized public interactions preceding the current decision and M_t is the current public Mode. The controlled task uses only executed transitions as common training targets. For each decision state, the common task example is

$$(h_t, M_t, a_t) \longrightarrow Y_t,$$

where h_t is the realized public interaction history before the current decision, M_t is the current public Mode, a_t is the action executed in the frozen trajectory, and Y_t is its realized outcome. Unexecuted actions never receive realized-outcome labels; they contribute only Source-predicted auxiliary targets.

The logger-specific auxiliary examples are

$$\begin{aligned}
\textit{Public}: &\quad \emptyset, \\
\textit{Chosen}: &\quad (h_t, M_t, a_t) \longrightarrow q_S(Y \mid a_t, M_t, h_t), \\
\textit{Candidate}: &\quad (h_t, M_t, \mathcal{A}_t) \longrightarrow \{(a, q_S(Y \mid a, M_t, h_t)) : a \in \mathcal{A}_t\}, \\
\textit{Native Thinking}: &\quad (h_t, M_t) \longrightarrow R_t.
\end{aligned}$$

The predictive target for each action is the Source's normalized distribution over Signal P and Signal Q, obtained from forced-choice complete-sequence log probabilities. Predictive targets are se-

rialized in a fixed candidate-action order using normalized probabilities rounded to three decimal places. *Chosen* and *Candidate* use the same formatter; *Chosen* retains only the executed-action entry, whereas *Candidate* retains one entry for every candidate action. The checkpoint's standard chat template and EOS termination are used throughout. Prompt tokens are masked from the loss, all supervised target tokens receive unit weight, and sequences are not packed across example boundaries. When truncation is required, the oldest public-history tokens are removed first; the current decision state and supervised target are retained in full.

Native Thinking uses the public pre-action state (h_t, M_t) as its auxiliary input and the current live reasoning trace R_t as its supervised target. Prompt tokens are masked from the loss and all supervised target tokens receive unit loss weight.

Controlled evaluation interface. Held-out evaluation uses executed transitions from the frozen test corpus. Each query supplies (h_t, M_t, a_t) from one held-out decision state, and the realized outcome Y_t of that executed transition provides the reference label. The Learner predicts Signal P or Signal Q. No Source prediction, candidate-prediction map, retained reasoning, or Source model is available at evaluation, and no additional fine-tuning is performed after logger training.

A.3 Control Construction Details

The controls in Section 4.1 are constructed from the primary Thinking-OFF corpus while preserving the common task examples and training procedure. *Token-matched neutral* preserves the positions and serialized length of the *Candidate* prediction fields but replaces their predictive content with tokenizer-matched neutral tokens sampled independently of the episode. *Uniform* preserves the action-indexed prediction-field structure but assigns the same prediction to every candidate action. *Action-Misaligned Candidate* keeps the executed-action prediction unchanged and swaps the prediction assignments between the two unchosen actions at each decision state in the primary three-action environment. This preserves the candidate actions and prediction values while changing only the action assignments of the unchosen predictions. With two unchosen actions, the swap is the unique derangement; if their predictions are identical, the swap leaves their predictive content unchanged. Results are reported in Section 4.1 and Figure 2A.

A.4 *Candidate* Coverage

The candidate-coverage analysis uses the five-action extension of Appendix A.1. Realized episodes, candidate sets, the logging interface, and the Learner configuration are fixed; only the number of retained unchosen-action predictions changes. *Chosen* retains no unchosen branch ($m = 0$), whereas full *Candidate* retains all four ($m = 4$). For $m \in \{1, 2, 3\}$, retained alternatives are sampled uniformly without replacement from the unchosen candidates at each decision state. The executed-action prediction is always retained, and every retained prediction remains aligned with its action. Results are reported in Section 4.2 and Figure 2B.

A.5 Resource-Bounded Reconstructability

Section 4.3 and Figure 3 present the capacity and training-data sweeps. Both sweeps use the primary three-action environment and compare *Candidate* with *Chosen*. The Learner-size sweep uses `Qwen/Qwen3-0.6B-Base`, `Qwen/Qwen3-1.7B-Base`, `Qwen/Qwen3-4B-Base`, and `Qwen/Qwen3-8B-Base`. The 1.7B point in the Learner-size sweep and the 2,400-episode point in the training-data sweep reuse the primary controlled configuration, whose *Candidate–Chosen* gain is $+6.7$ points. The remaining sweep points use matched runs constructed with the same environment, logging protocol, and optimization procedure; logger views remain matched within each corpus. Exact values are reported in Table 9. These sweeps assess whether the benefit decreases as reconstructing predictive structure becomes easier.

Table 9: Absolute held-out consequence-prediction accuracies and paired gains underlying Figure 3. Accuracy is reported in percent and gain in percentage points. Bold rows correspond to the primary controlled configuration.

(A) Learner-size sweep			
Learner size	*Chosen* (%)	*Candidate* (%)	Gain (pp)
0.6B	63.8	72.9	+9.1
1.7B	**69.0**	**75.7**	**+6.7**
4B	73.6	78.5	+4.9
8B	76.9	80.8	+3.9
(B) Training-data sweep			
Training episodes	*Chosen* (%)	*Candidate* (%)	Gain (pp)
250	55.9	68.2	+12.3
500	59.8	70.6	+10.8
1,000	64.8	72.8	+8.0
2,400	**69.0**	**75.7**	**+6.7**
5,000	74.0	78.0	+4.0
10,000	77.2	79.7	+2.5

A.6 *Public*-History Posterior Predictive

Given the known rule prior and transition model, an exact posterior predictive can be computed from realized public history:

$$p(Y \mid a, H_t) = \sum_z p(Y \mid a, M_t, z)\, p(z \mid H_t). \tag{6}$$

Here, $p(z \mid H_t)$ is obtained by updating the uniform rule prior using the likelihood of the realized transitions. This construction uses no Source predictions and assumes knowledge of the rule family and transition model, which are not supplied to the Learner. It does not imply that a finite history uniquely identifies the active rule.

A.7 Controlled Cost Accounting

Candidate logging increases both stored trajectory size and Source-side collection cost. In the primary controlled experiment, the mean serialized record sizes are 104.2 tokens for *Public*, 137.5 for *Chosen*, and 238.7 for *Candidate*, corresponding to $1.00\times$, $1.32\times$, and $2.29\times$ the *Public* record size, respectively. These values describe stored logger records, not total Learner training tokens or FLOPs under the separate task and auxiliary training streams defined in Appendix D.

The experimental corpus is collected once with predictions for every candidate action, after which *Chosen* and *Candidate* are constructed from the same frozen prediction buffer. For a standalone logger, *Chosen* requires one consequence prediction for the executed action at each decision state, whereas *Candidate* requires one prediction for each retained candidate action. In ALFWorld, candidates are drawn from the environment-provided admissible-action list, without a Source call for candidate generation. In WebShop, candidate proposal introduces additional Source calls beyond the consequence-prediction queries. We therefore treat the reported comparison as a downstream-utility comparison rather than a compute-matched one.

A.8 Seed-Level Controlled Results

Table 10 reports the seed-level results for the primary Thinking-OFF experiment. All ten paired Learner seeds use the same frozen test corpus. The paired confidence interval is computed as described in Appendix E.

Table 10: Primary controlled experiment: correct predictions / 1,800 held-out decisions (300 six-turn episodes), with accuracy in parentheses. Every seed is evaluated on the same test decisions. Means and paired differences are computed from the unrounded counts and rounded only for display.

Seed	*Public*	*Chosen*	*Candidate*	*Candidate* − *Chosen*
1	1214/1800 (67.44%)	1232/1800 (68.44%)	1294/1800 (71.89%)	+3.44 pp
2	1228/1800 (68.22%)	1248/1800 (69.33%)	1370/1800 (76.11%)	+6.78 pp
3	1231/1800 (68.39%)	1239/1800 (68.83%)	1352/1800 (75.11%)	+6.28 pp
4	1240/1800 (68.89%)	1255/1800 (69.72%)	1381/1800 (76.72%)	+7.00 pp
5	1204/1800 (66.89%)	1225/1800 (68.06%)	1382/1800 (76.78%)	+8.72 pp
6	1232/1800 (68.44%)	1244/1800 (69.11%)	1403/1800 (77.94%)	+8.83 pp
7	1220/1800 (67.78%)	1236/1800 (68.67%)	1387/1800 (77.06%)	+8.39 pp
8	1240/1800 (68.89%)	1250/1800 (69.44%)	1403/1800 (77.94%)	+8.50 pp
9	1232/1800 (68.44%)	1246/1800 (69.22%)	1329/1800 (73.83%)	+4.61 pp
10	1235/1800 (68.61%)	1245/1800 (69.17%)	1325/1800 (73.61%)	+4.44 pp
Mean	**68.20%**	**69.00%**	**75.70%**	**+6.70** pp

B ALFWorld Experiment

B.1 Environment and Dataset Construction

We use the text-based ALFWorld household-task environment. The frozen Source corpus contains all 3,553 ALFWorld training games. We use 3,200 trajectories for Learner training and reserve 353 for development. Both successful and unsuccessful Source trajectories are retained. Evaluation uses the complete standard `valid_seen` and `valid_unseen` splits, containing 140 and 134 games, respectively.

The live Source is Qwen/Qwen3-32B with native thinking enabled. During collection, the Source receives the task goal, the current textual observation, the realized interaction history, and the complete environment-provided admissible-action list. It selects one action, that action alone is executed in ALFWorld, and the resulting environment response is appended to the realized trajectory.

The Source corpus, split membership, realized actions and observations, terminal outcomes, and trajectory ordering are frozen before *Candidate* construction or consequence-prediction logging. *Candidate* construction therefore does not alter the realized interaction record.

The Source action-generation template is reproduced in Appendix F.1; the complete fixed one-shot demonstration is included in the released experiment configuration.

The full admissible-action list returned by ALFWorld is supplied to the live Source without truncation. The same demonstration and instruction template are used throughout collection; the task, observation, realized history, and admissible actions vary with the current environment state.

Native reasoning. Qwen3 native thinking is enabled during live Source action generation. The model's native reasoning is generated before the final action response and is not sent to the environment. We define R_t for the *Native Thinking* logger as the native reasoning associated with the live Source generation at decision t. *Public*, *Chosen*, and *Candidate* omit this reasoning field.

B.2 *Candidate* Construction and Predictive Fields

***Candidate* construction.** *Candidate* sets are constructed offline from the environment-provided admissible-action list at each saved pre-action state after the realized Source trajectory has been frozen. The frozen live action is included in the final candidate set. *Candidate* construction uses no post-action observation, realized outcome, reward, completion signal, future state, candidate-consequence prediction, or hidden simulator state.

Consequence prediction. For every $a \in A_t$, Qwen/Qwen3-32B is independently queried to predict the immediate consequence

$$c_t(a) = (\text{effect}, \text{target}, \text{relation}, \text{status}).$$

These consequence probes are also performed offline after the realized trajectory has been frozen. "Pre-action" therefore refers to the information available to each probe, rather than to the wall-clock time at which the query is executed.

Each probe receives an immutable copy of the saved public context available immediately before the corresponding frozen action: the task goal, current public textual observation, realized public interaction history preceding that action, and the candidate action a whose consequence is being queried. It does not receive the realized post-action observation, environment outcome, reward, completion signal, future state, or hidden simulator state. For candidate actions other than the frozen live action, no environment transition is ever generated.

The realized outcome of the frozen live action is not supplied to the consequence-prediction query. Consequently, the prediction for the executed branch is also a Source prediction from the restricted pre-action information set rather than a label derived from the already observed transition.

Live Source action generation uses native thinking enabled. Consequence prediction uses the same Qwen/Qwen3-32B checkpoint with native thinking disabled and does not receive the live Source's private reasoning R_t.

Unexecuted alternatives therefore carry Source-predicted consequences only; they never receive environment-supplied counterfactual outcomes.

Table 11: ALFWorld consequence representation.

Field	Representation
Effect	navigate, open, close, take, put, clean, heat, cool, toggle, inspect, no_change, error
Target	Normalized affected object, receptacle, or location
Relation	held_by_agent, in:<receptacle>, on:<receptacle>, at:<location>, open, closed, heated, cooled, cleaned, none
Status	success, no_change, error

B.3 Logger Views and Autonomous Evaluation

All four logger views are constructed only after the Source corpus has been frozen. They share the same task identities, split membership, public observations, executed actions, realized environment feedback, final outcomes, and trajectory ordering.

Public contains the common realized record without Source-generated auxiliary fields. *Chosen* additionally retains the Source consequence prediction for the executed action. *Candidate* retains the aligned Source consequence predictions for the complete constructed candidate set. *Native Thinking* instead retains the native pre-action reasoning R_t produced during the live Source generation that selected the executed action.

The Source collection prompt is reproduced in Appendix F.1. Downstream Learners receive no in-context demonstration.

Separate Qwen/Qwen3-4B-Base Learners are adapted from each logger view using five paired training seeds and the optimization configuration in Appendix D.

Policy training interface. All logger conditions contain identical autonomous-policy examples. For each decision state, the policy example contains the task goal, current textual observation, realized public interaction history, and current environment-provided admissible-action list. The supervised target is the executed environment command.

The policy example contains no Source candidate set, Source consequence prediction, retained Source reasoning, or one-shot demonstration. In particular, the policy target cannot attend to any logger-specific auxiliary field.

Chosen, *Candidate*, and *Native Thinking* additionally receive separate logger-specific auxiliary examples constructed from the same frozen decision states. *Chosen* predicts the Source consequence for the executed action; *Candidate* predicts the action-indexed Source consequences for the complete candidate set; and *Native Thinking* predicts R_t. Task and auxiliary examples are processed as separate causal sequences.

The complete Learner policy prompt is reproduced in Appendix F.1.

Autonomous evaluation. The logger-adapted checkpoint is evaluated directly, with no additional policy fine-tuning. Evaluation uses all 140 `valid_seen` and 134 `valid_unseen` games, held fixed across logger conditions and training seeds.

At each evaluation step, the Learner receives the task goal, current environment observation, its own realized action/observation history, and the current environment-provided admissible-action list. It emits one command, that command is executed in ALFWorld, and the resulting environment response is appended to that Learner's own history before the next decision.

The Source model, Source one-shot demonstration, constructed candidate set, Source consequence predictions, and retained Source reasoning are unavailable throughout autonomous evaluation. Task success is scored using ALFWorld's native task-completion signal. All four logger conditions use the same decoding configuration, action interface, interaction limit, and failure handling (Appendix D).

B.4 Prediction Diagnostics

The Source's ALFWorld consequence predictions reach 77.8% exact-match accuracy on executed actions, where exact match requires agreement on all four normalized consequence fields. These values are fallible Source predictions, not oracle counterfactual labels.

Candidate construction and consequence prediction use only frozen pre-action information; no unchosen action is executed, and the frozen realized trajectory is never modified.

Table 12: ALFWorld prediction and candidate-set diagnostics.

Quantity	Result
Executed-action consequence exact match	77.8%
Parse success	99.6%
Mean final candidate-set size	7.6
States with at least two distinct predicted tuples	71.2%

B.5 Seed-Level Seen and Unseen Results

Table 13: Successful tasks / 140 on ALFWorld `valid_seen`, by paired Learner training seed. Means are computed from success counts before rounding.

Seed	*Public*	*Chosen*	*Native Thinking*	*Candidate*
17	67/140	69/140	71/140	77/140
43	72/140	72/140	75/140	74/140
101	65/140	70/140	69/140	73/140
211	70/140	73/140	74/140	79/140
503	70/140	70/140	73/140	76/140
Mean (%)	49.1	50.6	51.7	54.1

Candidate exceeds *Chosen* in every seed on both `valid_seen` and `valid_unseen`, while comparisons with *Native Thinking* vary in isolated seeds; across-seed means follow *Public* $<$ *Chosen* $<$ *Native Thinking* $<$ *Candidate* on both splits. Appendix E specifies the paired-bootstrap procedure.

Table 14: Successful tasks / 134 on ALFWorld `valid_unseen`, by paired Learner training seed. Means are computed from success counts before rounding.

Seed	*Public*	*Chosen*	*Native Thinking*	*Candidate*
17	60/134	63/134	65/134	72/134
43	64/134	62/134	68/134	67/134
101	58/134	64/134	62/134	69/134
211	63/134	66/134	69/134	73/134
503	62/134	63/134	66/134	71/134
Mean (%)	45.8	47.5	49.3	52.5

C WebShop Experiment

C.1 Environment and Dataset Construction

We use WebShop (Yao et al., 2022) to evaluate trajectory augmentation in multi-step web shopping. We sample 2,000 instructions from the official training split for Source trajectory collection and reserve 300 disjoint training-side instructions for development. Final evaluation uses the standard 500 test instructions. Split membership is fixed before Source collection and logger construction. Both successful and unsuccessful Source trajectories are retained.

The live Source is `Qwen/Qwen3-32B` with native thinking enabled. At each step, it receives the shopping instruction, current WebShop observation, realized interaction history, and currently available actions, and emits one valid WebShop action. Episodes terminate on successful purchase, environment termination, or the standard interaction limit. The Source corpus, split membership, realized actions and observations, rewards, terminal outcomes, and trajectory ordering are frozen before *Candidate* construction or consequence-prediction logging.

The complete Source rollout prompt, including the fixed one-shot demonstration, is reproduced in Appendix F.

The demonstration is collection-time scaffolding only. It is not serialized into the frozen trajectory and is not exposed to any downstream Learner.

Native reasoning. Qwen3 native thinking is enabled during live Source action generation. *Native Thinking* retains the native reasoning associated with the live generation that produces the executed action. This reasoning precedes environment execution and is not sent to WebShop. *Public*, *Chosen*, and *Candidate* omit this reasoning field. The visible demonstration text in the rollout prompt is not part of the retained native reasoning.

C.2 *Candidate* Construction and Consequence Representation

Candidate sets and consequence predictions are constructed offline from the frozen pre-action state after the realized Source corpus has been frozen. The procedure is:

1. Query `Qwen/Qwen3-32B` from the frozen pre-action state—the shopping instruction, current WebShop observation, currently available actions, and realized interaction history—to propose up to five distinct plausible next actions. *Candidate* proposal uses the same `Qwen/Qwen3-32B` checkpoint with native thinking disabled and greedy decoding.
2. Do not reveal the frozen live action to the proposer.
3. Canonicalize the proposed candidates, remove duplicates, and filter invalid actions.
4. After proposal is complete, insert the frozen live action when it is absent.
5. Deduplicate again.
6. Cap the final candidate set at six actions.
7. Independently predict the immediate consequence of every retained candidate action.
8. Never execute unchosen alternatives.

No post-action observation, reward, future state, task-completion signal, or consequence prediction enters candidate construction.

The complete candidate-proposal and consequence-prediction prompts are reproduced in Appendix F.2.

Consequence schema. Immediate consequences are represented as $(\textit{effect}, \textit{target}, \textit{status})$ tuples (Table 15). The schema describes the immediate transition, not eventual task success.

Table 15: WebShop immediate-consequence representation.

Field	Representation
Effect	show_search_results, open_product, select_option, navigate, reveal_information, purchase, no_change, error
Target	The normalized product, query, option, page, section, or other object directly affected by the action; "none" if no specific target applies
Status	success, no_change, error

Consequence probes use `Qwen/Qwen3-32B` with native thinking disabled and receive no live Source reasoning. For unexecuted actions, the retained tuple is a Source prediction only; no environment transition is generated.

Candidate proposal and consequence prediction use only frozen pre-action information; no unchosen action is executed, and the frozen realized trajectory is never modified.

C.3 Logger Views and Autonomous Evaluation

All four logger views share the same frozen shopping instructions, observations, executed actions, realized environment feedback, final rewards, and trajectory ordering. *Public* contains the common realized record. *Chosen* additionally retains the Source consequence prediction for the executed action. *Candidate* retains the aligned Source predictions for the complete constructed candidate set. *Native Thinking* instead retains the native pre-action reasoning produced during live Source generation.

Separate `Qwen/Qwen3-4B-Base` Learners are adapted from each logger view using five paired training seeds. All logger conditions contain identical policy-training examples. *Chosen*, *Candidate*, and *Native Thinking* additionally receive separate logger-specific auxiliary examples. Task and auxiliary examples are processed as separate causal sequences.

Policy and auxiliary examples. For each decision state, the policy example contains the shopping instruction, current observation, currently available actions, and realized interaction history, and the supervised target is the executed WebShop action. *Public* receives only this common policy training. The *Chosen* auxiliary target contains the Source prediction for the executed action; the *Candidate* auxiliary target contains the action-indexed Source predictions for the complete candidate set; and the *Native Thinking* auxiliary target contains the live Source native reasoning. The policy target cannot attend to the auxiliary target. Prompt tokens have zero loss weight and supervised target tokens have unit loss weight. Task-example exposure and optimizer-update counts are matched across logger conditions; augmented conditions process an additional auxiliary batch (Appendix D).

The complete Learner policy prompt is reproduced in Appendix F.2.

Autonomous evaluation. The logger-adapted checkpoint is evaluated directly on all 500 standard test instructions, with no additional fine-tuning. At each step, the Learner receives the shopping instruction, current observation, currently available actions, and its own realized interaction history, and emits one WebShop action. The Source model, Source demonstration, constructed candidate set, Source consequence predictions, and retained Source reasoning are unavailable throughout evaluation. All four logger conditions use the same policy prompt, decoding configuration, interaction limit, parsing rules, and failure handling (Appendix D). The primary endpoint is autonomous task success. Following the standard WebShop metric, a test instruction is counted as successful if the environment reward is $r = 1$; all episodes with $r < 1$ are counted as failures.

C.4 Source and Dataset Diagnostics

The frozen `Qwen/Qwen3-32B` Source achieves 45.6% task success on the standard 500-task test set. This value is reported only as the contextual Source reference in Table 5; Source test rollouts never enter Learner training, development, candidate construction, or model selection.

C.5 Storage and Compute Accounting

The token and compute accounting conventions in Appendix A.7 also apply here. *Public* uses only the common policy stream, whereas *Chosen*, *Candidate*, and *Native Thinking* each process one matched auxiliary stream in addition to it. On the Source side, *Candidate* collection cost includes candidate proposal plus one consequence query per retained candidate action. *Chosen* requires only the executed-action consequence prediction. *Native Thinking* retains the native live-generation reasoning rather than generating *Candidate*-style consequence probes. Policy and auxiliary examples are separate sequences, so stored-record size and Source-side collection cost are not equivalent to Learner FLOPs, and Source-side candidate-proposal and consequence-query costs are distinct from Learner training cost. The reported comparison is therefore one of downstream utility rather than compute matching.

C.6 Seed-Level Behavioral Results

Table 16 reports seed-level WebShop task success for all four logger views. *Candidate* achieves the highest task success within every seed, exceeding *Chosen* by 15–31 tasks and *Native Thinking* by 9–26 tasks out of 500, and *Public* is lowest within every seed. *Native Thinking* exceeds *Chosen* in four of five seeds and falls below it by one task in seed 503. The paired contrasts are summarized in Table 5 and Appendix E.

D Model and Training Configuration

We use Qwen3 (Yang et al., 2025), Llama 3.1/3.2 (Meta AI, 2024a;b), and Gemma 2 (Gemma Team, 2024); Table 17 lists the exact checkpoints. Learners are adapted with LoRA (Hu et al., 2022).

Batching and loss aggregation. The reported effective batch sizes refer to the common task stream. At every optimizer update, all logger conditions process the same batch of B common task examples. *Chosen*, *Candidate*, and *Native Thinking* additionally process a matched batch of B logger-specific auxiliary examples. *Public* processes only the common task batch. Task-example exposure and optimizer-update counts are therefore identical across logger conditions.

For each sequence, cross-entropy is averaged over supervised target tokens; prompt tokens are masked from the loss. For augmented logger conditions, the task loss and auxiliary loss are each averaged over their respective batch and combined with equal weight,

$$\mathcal{L}_v = \mathcal{L}_{\text{task}} + \mathcal{L}_{\text{aux},v},$$

whereas

$$\mathcal{L}_{\text{Public}} = \mathcal{L}_{\text{task}}.$$

Gradients from the two losses are accumulated before one optimizer update. Epoch counts are defined with respect to the common task stream; auxiliary examples are added on top of that stream and do not reduce task-example exposure. Examples are not packed across sequence boundaries.

The effective batch size of 64 in the controlled experiment and 32 in ALFWorld and WebShop therefore refers to the common task batch; augmented conditions process an additional auxiliary batch of the same size at each optimizer update.

Controlled task. The primary controlled experiment uses the `Qwen/Qwen3-8B` Source with native thinking disabled and the `Qwen/Qwen3-1.7B-Base` Learner. The Thinking-ON extension uses the same Source checkpoint with native thinking enabled; its four logger views come from that single frozen corpus. The cross-family experiment (Section 3.4) crosses the Qwen, Llama, and Gemma Source checkpoints with the Qwen, Llama, and Gemma Learner checkpoints in Table 17 and reports the *Candidate–Chosen* gain for each pairing. Each Source–Learner pairing uses ten paired

Table 16: Seed-level WebShop autonomous task success. Each seed is evaluated on the same 500 standard test instructions. Parentheses give task success in percent; paired differences are in percentage points.

Seed	*Public*	*Chosen*	*Native Thinking*	*Candidate*	*Candidate–Chosen*	*Candidate–Native Thinking*
17	159/500 (31.8)	177/500 (35.4)	183/500 (36.6)	192/500 (38.4)	+3.0	+1.8
43	168/500 (33.6)	173/500 (34.6)	182/500 (36.4)	204/500 (40.8)	+6.2	+4.4
101	164/500 (32.8)	174/500 (34.8)	176/500 (35.2)	192/500 (38.4)	+3.6	+3.2
211	171/500 (34.2)	177/500 (35.4)	188/500 (37.6)	206/500 (41.2)	+5.8	+3.6
503	163/500 (32.6)	173/500 (34.6)	172/500 (34.4)	198/500 (39.6)	+5.0	+5.2
Mean	33.0	35.0	36.0	39.7	+4.7	+3.6

Table 17: Source and Learner checkpoints used in the experiments. "Post-trained hybrid" denotes the Qwen3 post-trained checkpoints whose native thinking can be enabled or disabled. †Live ALFWorld and WebShop Source action generation uses thinking enabled; the offline consequence-prediction side probes run with thinking disabled.

Role/family	Checkpoint	Type	Size
Source/Qwen (primary controlled)	`Qwen/Qwen3-8B`	Post-trained hybrid; thinking disabled	8B
Source/Qwen (Thinking-ON extension)	`Qwen/Qwen3-8B`	Post-trained hybrid; thinking enabled	8B
Source/Llama	`meta-llama/Llama-3.1-8B-Instruct`	Instruct	8B
Source/Gemma	`google/gemma-2-9b-it`	Instruct	9B
Learner/Qwen (primary Learner)	`Qwen/Qwen3-1.7B-Base`	Base	1.7B
Learner/Llama	`meta-llama/Llama-3.2-3B`	Base	3B
Learner/Gemma	`google/gemma-2-2b`	Base	2B
Learner/size sweep	`Qwen/Qwen3-0.6B-Base`	Base	0.6B
Learner/size sweep	`Qwen/Qwen3-1.7B-Base` (primary)	Base	1.7B
Learner/size sweep	`Qwen/Qwen3-4B-Base`	Base	4B
Learner/size sweep	`Qwen/Qwen3-8B-Base`	Base	8B
Source/ALFWorld	`Qwen/Qwen3-32B`	Post-trained hybrid; thinking enabled†	32B
Learner/ALFWorld	`Qwen/Qwen3-4B-Base`	Base	4B
Source/WebShop	`Qwen/Qwen3-32B`	Post-trained hybrid; thinking enabled†	32B
Learner/WebShop	`Qwen/Qwen3-4B-Base`	Base	4B

Learner training seeds on a frozen corpus generated by the corresponding Source; reported values are mean *Candidate–Chosen* accuracy differences across paired seeds. The Learner-size sweep (Section 4.3) uses the base checkpoints `Qwen/Qwen3-0.6B-Base`, `Qwen/Qwen3-1.7B-Base`, `Qwen/Qwen3-4B-Base`, and `Qwen/Qwen3-8B-Base`; the 1.7B point is the primary Learner. Qwen3-8B and Qwen3-32B Sources are the post-trained hybrid-thinking checkpoints, not separate instruct variants. In ALFWorld and WebShop, the Qwen3-32B Source generates live actions with native thinking enabled, and its offline consequence-prediction side probes run with thinking disabled; the Llama and Gemma Sources are instruct checkpoints without a native thinking mode. The controlled Source configuration uses greedy action decoding, temperature 1.0 for normalized outcome logits, no sampling, and a fixed candidate order. The controlled Learner configuration uses LoRA rank 16, $\alpha = 32$, dropout 0.05, AdamW, learning rate 2×10^{-4}, weight decay 0.01, effective batch size 64, three epochs, 3% warmup, gradient clipping at 1.0, and context length 2,048. Ten Learner seeds are

nested within each frozen Source dataset. Each run uses one 80GB accelerator, bfloat16 parameters, and deterministic data ordering.

ALFWorld and WebShop. Both autonomous-task Learners use `Qwen/Qwen3-4B-Base` with LoRA rank 16, $\alpha = 32$, and dropout 0.05. Training uses AdamW with learning rate 1×10^{-4}, weight decay 0.01, effective batch size 32, three epochs, 3% warmup, gradient clipping at 1.0, context length 8,192, and bfloat16 precision. Five paired Learner seeds are used for every logger condition in both benchmarks.

Autonomous evaluation configuration. At evaluation, actions are decoded greedily with a maximum of 64 generated tokens per decision in both ALFWorld and WebShop. Each episode is limited to 50 action attempts. Invalid or unparsable actions count toward this limit but do not themselves terminate the episode; interaction continues until the step limit is reached or the environment terminates.

E Statistical Inference

The reported confidence intervals are pointwise percentile intervals and are not multiplicity adjusted.

Controlled-task inference. For each reported controlled-task paired contrast, we first compute the accuracy difference within each of the ten paired Learner training seeds. Each bootstrap replicate samples ten seed differences with replacement and computes their mean. The 2.5th and 97.5th percentiles of 10,000 bootstrap replicates form the reported pointwise 95% confidence interval. The Source corpus and evaluation set are held fixed, so this interval quantifies variation across Learner training seeds rather than uncertainty from collecting a new Source corpus or sampling a new evaluation set. The underlying seed-level counts are reported in Table 10. Intervals are reported for the primary Thinking-OFF contrasts and the Thinking-ON extension (Table 2); the cross-family matrix (Table 3) reports point contrasts.

ALFWorld inference. For each `valid_unseen` contrast, we compute the task-success difference within each of the five paired Learner training seeds. We enumerate all 5^5 ordered resamples of the five seed-wise differences with replacement and average each resample. The 2.5th and 97.5th percentiles of this exact bootstrap distribution form the reported pointwise 95% confidence interval. The evaluation task set is held fixed, so these intervals quantify Learner-training variability rather than uncertainty from sampling a new ALFWorld evaluation set.

WebShop inference. For each reported WebShop task-success contrast, we compute the success-rate difference within each of the five paired Learner training seeds. We enumerate all 5^5 ordered resamples of the five seed-wise differences with replacement and average each resample. The 2.5th and 97.5th percentiles of this exact bootstrap distribution form the reported pointwise 95% confidence interval. The standard 500-task evaluation set is held fixed, so these intervals quantify Learner-training variability rather than uncertainty from sampling a new WebShop test set.

The resulting pointwise 95% intervals are reported in Table 5.

F PROMPTS AND DEMONSTRATIONS

F.1 ALFWORLD PROMPTS

Source rollout prompt. Source action generation uses a fixed one-shot demonstration in addition to the instruction template below. The demonstration is collection-time scaffolding only and is not serialized into the frozen trajectory or exposed to the downstream Learner. The complete demonstration is included in the released experiment configuration.

```
[SYSTEM]

You are an agent acting in the ALFWorld text-based household
environment.

Your goal is to complete the task described by the user through
a sequence of valid environment actions.

At every step, you will receive:
1. the task goal;
2. the current environment observation;
3. the interaction history so far; and
4. the list of currently admissible actions.

Use the current task information to decide what to do next.
Choose exactly one action from the admissible-action list.

Return exactly one line:

Action: <one admissible action exactly as written>

Do not output multiple actions.
Do not invent actions that are not in the admissible-action list.
Do not add text after the Action line.

The environment, not you, determines whether the task has been
completed.


[USER]

Now solve a new ALFWorld task.

The demonstration above specifies only the interaction format.
Do not assume that its objects, locations, or actions are
available in the new task. Use only the current task
information and the current admissible-action list.

Task:
{TASK_GOAL}

Current observation:
{CURRENT_OBSERVATION}

Interaction history:
{REALIZED_ACTION_OBSERVATION_HISTORY}

Admissible actions:
{FULL_ADMISSIBLE_ACTION_LIST}
```

```
Choose the next action.
```

Learner policy prompt. The same policy interface is used for *Public*, *Chosen*, *Native Thinking*, and *Candidate*. Unlike the Source rollout prompt, the Learner policy prompt contains no in-context demonstration and requests only an environment command.

```
[SYSTEM]

You are an agent acting in the ALFWorld text-based household
environment.

Your goal is to complete the task described by the user through
a sequence of valid environment actions.

At every step, you will receive the task goal, the current
environment observation, the interaction history so far, and
the list of currently admissible actions.

Choose exactly one action from the admissible-action list that
best advances the task.

Return exactly one line:

Action: <one admissible action exactly as written>

Do not provide reasoning, explanation, or additional text.


[USER]

Task:
{TASK_GOAL}

Current observation:
{CURRENT_OBSERVATION}

Interaction history:
{LEARNER_REALIZED_ACTION_OBSERVATION_HISTORY}

Admissible actions:
{FULL_ADMISSIBLE_ACTION_LIST}

Next action:
```

F.2 WebShop Prompts

Source rollout prompt. All WebShop Source rollouts use the following fixed prompt, which contains one fixed one-shot demonstration. The same demonstration and instruction template are used throughout collection; the shopping instruction, observation, available actions, and realized history vary with the current environment state.

```
[SYSTEM]

You are an agent acting in the WebShop environment.

Your goal is to satisfy the user's shopping instruction by
searching for products, examining relevant product information,
selecting the requested options, and purchasing the best
matching item.
```

```
At each step, you will receive:
1. the shopping instruction;
2. the current WebShop observation;
3. the realized interaction history; and
4. the actions currently available in the environment.

Valid actions have one of the following forms:

search[]
click[]

Use search[...] only when search is currently available.
For click[...], the argument must exactly match one of the
currently available clickable items.

Use the shopping instruction and complete interaction history
to choose the next action.
Consider all stated product requirements, including product
type, attributes, options, and price constraints.

Return exactly one line:

Action: <one valid WebShop action>

Do not include reasoning, explanation, or additional text in
the final response.
Do not output multiple actions.
Do not invent clickable items that are not currently available.
Do not purchase an item until the relevant product requirements
and options have been checked.


[USER]

The following is a WebShop demonstration.

Shopping instruction:
Find a women's waterproof hiking jacket in black, size medium,
for less than $80.

Current observation:
WebShop
Search

Available actions:
search

Interaction history:
None.

Choose the next action.


[ASSISTANT]

Action: search[womens waterproof hiking jacket black medium under 80]
```

```
[USER]

Shopping instruction:
Find a women's waterproof hiking jacket in black, size medium,
for less than $80.

Current observation:
Search results:
Women's Waterproof Hiking Jacket - $64.99
Women's Lightweight Rain Coat - $52.99
Women's Insulated Hiking Jacket - $92.00

Available actions:
click[Women's Waterproof Hiking Jacket]
click[Women's Lightweight Rain Coat]
click[Women's Insulated Hiking Jacket]
click[Back to Search]

Interaction history:
search[womens waterproof hiking jacket black medium under 80]

Choose the next action.


[ASSISTANT]

Action: click[Women's Waterproof Hiking Jacket]


[USER]

Shopping instruction:
Find a women's waterproof hiking jacket in black, size medium,
for less than $80.

Current observation:
Women's Waterproof Hiking Jacket
Price: $64.99
Color options: [black] [blue] [red]
Size options: [small] [medium] [large]
[Description] [Features] [Reviews] [Buy Now]

Available actions:
click[black]
click[blue]
click[red]
click[small]
click[medium]
click[large]
click[Description]
click[Features]
click[Reviews]
click[Buy Now]
click[< Prev]
click[Back to Search]

Interaction history:
search[womens waterproof hiking jacket black medium under 80]
click[Women's Waterproof Hiking Jacket]
```

```
Choose the next action.


[ASSISTANT]

Action: click[black]


[USER]

Shopping instruction:
Find a women's waterproof hiking jacket in black, size medium,
for less than $80.

Current observation:
Women's Waterproof Hiking Jacket
Price: $64.99
Selected color: black
Size options: [small] [medium] [large]
[Buy Now]

Available actions:
click[small]
click[medium]
click[large]
click[Buy Now]
click[< Prev]
click[Back to Search]

Interaction history:
search[womens waterproof hiking jacket black medium under 80]
click[Women's Waterproof Hiking Jacket]
click[black]

Choose the next action.


[ASSISTANT]

Action: click[medium]


[USER]

Shopping instruction:
Find a women's waterproof hiking jacket in black, size medium,
for less than $80.

Current observation:
Women's Waterproof Hiking Jacket
Price: $64.99
Selected color: black
Selected size: medium
[Buy Now]

Available actions:
click[Buy Now]
click[< Prev]
```

```
click[Back to Search]

Interaction history:
search[womens waterproof hiking jacket black medium under 80]
click[Women's Waterproof Hiking Jacket]
click[black]
click[medium]

Choose the next action.


[ASSISTANT]

Action: click[Buy Now]


[USER]

Now solve a new WebShop task.

The demonstration above specifies only the interaction format
and general procedure.
Do not assume that its product, options, price, or clickable
text appears in the new task.

Shopping instruction:
{TASK_GOAL}

Current observation:
{CURRENT_OBSERVATION}

Available actions:
{AVAILABLE_ACTIONS}

Interaction history:
{REALIZED_ACTION_OBSERVATION_HISTORY}

Choose the next action.
```

***Candidate*-proposal prompt.**

```
[SYSTEM]

You are given the state of a WebShop interaction immediately
before an action is taken.

Propose plausible next actions that could reasonably advance
the shopping instruction.
Use only information available in the current pre-action state.

Valid actions have one of the following forms:

search[]
click[]

A click action must exactly match one of the currently
available clickable items.
Use search[...] only when search is currently available.
```

```
Return up to five distinct candidate actions.
Candidates should represent meaningfully different plausible
choices rather than minor textual variations of the same
action.

Do not predict consequences.
Do not rank the candidates.
Do not explain the choices.

Return exactly:

Candidates:
1.
2.
...


[USER]

Shopping instruction:
{TASK_GOAL}

Current observation:
{CURRENT_OBSERVATION}

Available actions:
{AVAILABLE_ACTIONS}

Interaction history:
{REALIZED_HISTORY}

Propose the candidate actions.
```

Consequence-prediction prompt. For every retained candidate action, `Qwen/Qwen3-32B` is queried independently from the frozen pre-action state with the following prompt.

```
[SYSTEM]

Predict the immediate consequence of a proposed WebShop action
using only information available before the action is executed.

Do not use post-action information.
Do not predict later actions.
Do not predict eventual task success.

Return exactly one normalized tuple:

(effect, target, status)

Allowed effect values:
show_search_results
open_product
select_option
navigate
reveal_information
purchase
no_change
error
```

```
Allowed status values:
success
no_change
error

The target identifies the product, query, option, page,
section, or object most directly affected by the immediate
action.
Use "none" when no specific target applies.

Return only the tuple.


[USER]

Shopping instruction:
{TASK_GOAL}

Current observation:
{CURRENT_OBSERVATION}

Interaction history:
{REALIZED_HISTORY}

Candidate action:
{CANDIDATE_ACTION}

Predict its immediate consequence.
```

Learner policy prompt. The same policy prompt is used for *Public*, *Chosen*, *Native Thinking*, and *Candidate* during policy training and autonomous evaluation.

```
[SYSTEM]

You are an agent acting in the WebShop environment.

Your goal is to satisfy the shopping instruction by searching
for products, examining relevant information, selecting the
requested options, and purchasing the best matching item.

Valid actions have one of the following forms:

search[]
click[]

Use search[...] only when search is currently available.
For click[...], the argument must exactly match one of the
currently available clickable items.

Return exactly one line:

Action: <one valid WebShop action>

Do not provide reasoning or additional text.


[USER]

Shopping instruction:
```

```
{TASK_GOAL}

Current observation:
{CURRENT_OBSERVATION}

Available actions:
{AVAILABLE_ACTIONS}

Interaction history:
{LEARNER_REALIZED_HISTORY}

Choose the next action.
```