

Counterfactual Selective Imitation for Long-Horizon Agents

Chunlin Yu¹ Ruiqi Xu¹ Huan Li² Chuliang Weng¹

¹School of Data Science and Engineering, East China Normal University

²College of Computer Science and Technology, Zhejiang University

{clyu,ruiqixu}@stu.ecnu.edu.cn, lihuan.cs@zju.edu.cn, clweng@dase.ecnu.edu.cn

Abstract

Long-horizon agents with structured actions execute tasks through sequences of dependent decisions, so a local decision mistake can change later observations and cause the task to fail. Stronger expert models offer a natural source of supervision, yet direct imitation or behavior cloning treats many differences between expert and base actions as equally useful targets. In this paper, we propose counterfactual selective imitation (CSI), a method that uses same-state counterfactual comparison to identify positive return differences. CSI assigns expert targets only when counterfactual rollouts show a high positive return difference, and routine states receive base-policy targets. The learned update is restricted to structured action spans, leaves the base model frozen, and does not require expert queries or replay at deployment. Experiments on AppWorld, ToolSandbox, and WebShop show that CSI raises average task success from **33.3%** to **49.7%** over the Qwen3-8B base and reduces average invalid/tool error from **9.3%** to **4.9%**. Compared with the strongest same-deployment adaptation baseline, CSI improves average task success by 1.3 points with only 5% additional deployed-inference latency.

1 Introduction

Long-horizon tool-use agents solve complex tasks through multi-turn sequences of structured decisions. This setting appears in ReAct-style reasoning and acting (Yao et al., 2023), Toolformer (Schick et al., 2023), Gorilla (Patil et al., 2024), and ToolLLM (Qin et al., 2024), where language-model outputs can call APIs, operate tools, or issue web actions. At each step, the agent may choose an action type, fill arguments, select an entity, or decide that the task should stop. These decisions determine what the agent observes next, so a local error can affect the rest of the trajectory. Stateful benchmarks such as AppWorld (Trivedi et al., 2024),

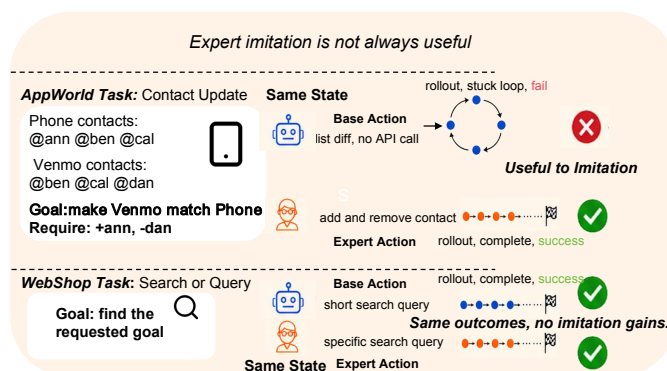


Figure 1: Imitate only when the expert helps.

ToolSandbox (Lu et al., 2025), and WebShop (Yao et al., 2022) expose this issue. Because later observations depend on earlier actions, an early invalid or off-target action can leave later correct-looking steps unable to solve the task.

A natural way to improve such agents is to use a stronger expert policy as supervision. Behavior cloning maximizes the likelihood of expert actions on recorded trajectories. Policy and sequence-level distillation match teacher outputs, token distributions, or teacher-generated sequences (Hinton et al., 2015; Rusu et al., 2016; Kim and Rush, 2016). Agent correction methods (Shinn et al., 2023; Fu et al., 2025; Lauffer et al., 2025) use failed trajectories or feedback to revise behavior. Critical-step methods (Chen et al., 2025; Li et al., 2026) focus supervision on selected decision points. These variants differ in data collection and update mechanism, but labeled decisions usually become expert targets. This route is attractive because expert actions provide concrete guidance for tool names, argument values, entity choices, and recovery behavior.

However, this line of work leaves three mismatches in long-horizon tool use. 1) A selected feedback state provides an expert action, but the label alone does not measure the **downstream effect**

of replacing the base action. 2) Surface disagreement is not the same as useful supervision. Different valid actions can lead to the same outcome, so treating every difference as an expert target can change behavior that does not need to change. 3) Broad updates can carry such changes into contexts where the base action was already valid. Deployment therefore calls for target selection that changes consequential action states while preserving routine actions.

Such selectivity requires a criterion that goes beyond action disagreement. Figure 1 gives the concrete example. It shows that the same pre-action state can be replayed with the base and expert actions, then judged by the resulting returns. The missing quantity is not whether the actions differ, but whether the expert replacement improves the downstream outcome enough to justify imitation. This same-state counterfactual comparison separates expert actions that should be copied from routine differences that should stay close to the base policy.

In this paper, we propose counterfactual selective imitation (CSI), a method that uses same-state counterfactual comparison to identify positive return differences and assigns expert targets only to the high-gain subset during training in long-horizon tool-use agents. First, CSI performs **counterfactual comparison**. It queries the expert and base policies on the same serialized history, replays their candidate next actions from the same execution point, and estimates whether expert imitation at that decision point improves downstream return and how large the gain is. Second, CSI constructs **selected expert targets**. Structured actions with high estimated gain receive expert targets, while matching actions, no-gain disagreements, and valid routine base actions keep base-policy targets. Third, CSI applies **span-restricted steering**. It freezes the base model and learns a low-dimensional action span steering module that generates a steering vector from the current context only for structured action tokens. At deployment, CSI runs only the steering-adapted policy once at each decision point and does not query the expert policy, replay the environment, or sample multiple candidate actions.

Our contributions are summarized as follows.

1. We introduce counterfactual comparison for long-horizon tool-use agents, estimating whether expert actions improve downstream return from the same pre-action state (Section 4.2).
2. We construct selected expert targets that imi-

tate the expert only for high-gain action states and use base-policy targets for routine states (Section 4.3).

3. We implement CSI with span-restricted steering and evaluate it on AppWorld, ToolSandbox, and WebShop (Sections 4.4 and 5). CSI improves average task success by **16.4** points over the Qwen3-8B base and by **1.3** points over the strongest same-deployment adaptation baseline, while lowering average invalid/tool error by **4.4** points with only **5%** additional deployed-inference latency over the Qwen3-8B base.

2 Preliminaries

Long-horizon structured-action agents. Long-horizon structured-action agents interact with an environment $\mathcal{E}$ through observations, structured actions, and state updates. At decision i , context x_i serializes the instruction, prior observations, and previous actions, and policy $\pi(\cdot | x_i)$ produces the next action output. The benchmark wrapper may keep execution handle s_i to restore the environment before action i , but the policy does not observe s_i . The policy emits structured action a_i , whose execution can change later observations, database state, or final success. Thus, each decision is both a generated output and an intervention that can alter the future state distribution. At the end, the benchmark returns scalar R , such as binary task success or a benchmark-specific score.¹

Policies and action spans. We distinguish the base policy π_0 , the expert policy π_E , and the adapted policy π_θ . All three take the same serialized context x_i and produce a distribution over the next action output. Let $y_i = (y_{i,1}, \dots, y_{i,T_i})$ denote the tokenized output that contains the structured action for decision i . The action span $I_i \subseteq \{1, \dots, T_i\}$ is the set of token positions corresponding to the executable action, including the action name and serialized arguments when they are present. We refer to tokens in I_i as action tokens. The output format requires a single structured-action line, which the benchmark parser validates against the action schema. Thus, the action span is where generated text becomes an environment-changing decision.

¹We use the benchmark-defined scalar return under the official evaluation protocols of AppWorld, ToolSandbox, and WebShop (Trivedi et al., 2024; Lu et al., 2025; Yao et al., 2022). The comparison only requires a consistent scalar return within each benchmark.

Span-level adaptation. Span-level adaptation applies a target-matching objective only on the action span. For a target distribution q_i over action tokens, the loss on example i is

$$\mathcal{L}_{\text{span}}(i) = - \sum_{t \in I_i} \sum_v q_{i,t}(v) \log \pi_{\theta}(v \mid x_i, y_{i,<t}). \quad (1)$$

Here q_i may be derived from the expert policy, from the base policy, or from a hard action target when logits are not available. Equation 1 specifies where the training signal is applied, but not which action first becomes the target or how π_{θ} is parameterized.

Imitation and correction. Existing imitation and correction methods mainly differ in how supervised records are obtained. Behavior cloning fits expert actions, while policy and sequence-level distillation fit teacher distributions or teacher-generated sequences (Hinton et al., 2015; Rusu et al., 2016; Kim and Rush, 2016). DAgger-style methods ask the expert to label learner-visited states (Ross et al., 2011), and on-policy distillation samples histories from the student or current model and fits a teacher or reference distribution there (Agarwal et al., 2024; Gu et al., 2024).

Agent tuning collects interaction traces for supervision (Chen et al., 2023; Zeng et al., 2024; Chen et al., 2024). Agent correction uses feedback, critiques, or teacher fixes (Shinn et al., 2023; Gou et al., 2024; Fu et al., 2025; Lauffer et al., 2025). Critical-step methods choose important trajectory positions before applying expert or corrected labels (Chen et al., 2025; Li et al., 2026). In these pipelines, a selected record usually receives its attached expert label, teacher distribution, or correction as the training target.

For replayable long-horizon agents, the selected record can be checked one step further. The base and expert policies decode candidate next actions from the same x_i . The handle s_i restores the same pre-action state. Downstream returns under the same continuation rule indicate whether to imitate the expert action, keep a base target, or omit the record from expert imitation. Only after this action-level selection do we instantiate the span target distribution q_i used in Eq. 1.

Update mechanisms. The adaptation mechanism also matters for preserving routine behavior. Full fine tuning, low-rank adapters (Hu et al., 2022), and distillation can fit the same action-span labels, but their learned changes are shared across many prompts and positions. Representation-level interventions (Zou et al., 2023; Li et al., 2023) and activation steering (Turner et al., 2023; Rinsky et al.,

2024; Stolfo et al., 2025) leave the base weights fixed and modify hidden states during generation. Steering provides a local mechanism for executable tokens, but it still needs evidence for which expert actions are worth copying. The next section asks whether useful expert supervision is spread across many expert-base disagreements or concentrated in high-gain same-state replacements.

3 Motivation

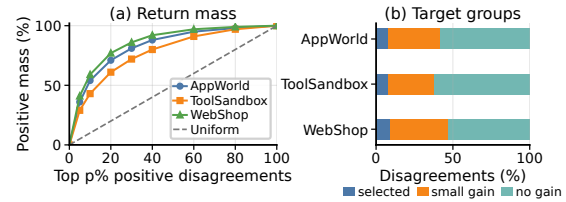


Figure 2: Useful expert supervision is concentrated. Panel (a) ranks positive-gain expert-base disagreements by $\hat{\Delta}$ and plots cumulative positive return-difference mass. Panel (b) partitions expert-base disagreements into selected expert targets under the main p setting, small positive-gain cases, and no-gain cases. Percentages in panel (b) are measured over expert-base disagreements.

Useful imitation concentrates. A first design question is how many disagreements between base and expert actions should become expert targets. A positive counterfactual gain is only a candidate signal. We need to know whether most of the positive gain is concentrated in a smaller high-gain subset. We rank disagreements by their estimated downstream effect rather than by trajectory source or surface difference. This asks whether useful expert targets are spread across many positive cases or concentrated in a smaller subset.

For an eligible decision record with serialized context x_i and execution handle s_i , let a_i^0 be the action decoded by the base policy and a_i^E the action decoded by the expert policy. The selection signal is the estimated counterfactual return difference,

$$\hat{\Delta}_i = \hat{R}(s_i, a_i^E) - \hat{R}(s_i, a_i^0), \quad (2)$$

where each term estimates the downstream return after executing one candidate next action from the same execution handle. Positive values identify candidate states where the expert action improves the estimated outcome. They do not by themselves define expert targets. A later target-construction step should therefore select only the highest-gain positive disagreements for expert imitation. When the actions match, replay shows no estimated gain,

Benchmark	Disagreement	Positive return	Selected targets	Top 10% mass	Top 20% mass
AppWorld	34	42	8	54	71
ToolSandbox	29	38	8	43	61
WebShop	41	47	9	59	77

Table 1: Disagreement statistics across AppWorld, ToolSandbox, and WebShop. Disagreement is measured over eligible records. Positive return is measured over expert-base disagreements. Selected targets report the main p -selection rate converted to a percentage of expert-base disagreements. Top-mass columns are measured over positive return-difference mass.

or the base action is already a valid routine action, expert imitation is not justified by this comparison alone. This quantity is not a policy-gradient advantage. It is an offline selection signal for deciding which action spans should imitate the expert.

Figure 2 asks whether positive expert gains are diffuse or concentrated. Panel (a) sorts positive-gain disagreements by $\hat{\Delta}$ and plots cumulative positive return-difference mass. The curves rise far above the uniform line. Across AppWorld, ToolSandbox, and WebShop, the top 10% of positive-gain disagreements account for 43–59% of the positive mass, and the top 20% account for 61–77%. Panel (b) shows that most disagreements are either no-gain cases or small positive-gain cases under the main selection setting. This supports selecting a high-gain subset rather than turning every expert-base disagreement into an expert target.

Concentration holds across benchmarks. Table 1 reports the same concentration pattern with the corresponding disagreement and target rates. Because the columns use different denominators, the table should be read as a distributional summary rather than a single global rate.

Routine states remain useful. Unselected contexts still need constraints, since ignoring all routine states would allow the adapted policy to drift on actions that were already acceptable. The resulting design should keep context source separate from target source and assign base targets only to routine states. This motivates CSI, which Section 4 formalizes through the corresponding expert and base target sets.

4 Method

As shown in Figure 3, CSI is an execution flow with an offline target-construction stage and a frozen-base policy-implementation stage. First, Counterfactual Comparison queries the base and expert policies on the same pre-action context and replays their candidate next actions from the same execution handle to estimate a return difference. Sec-

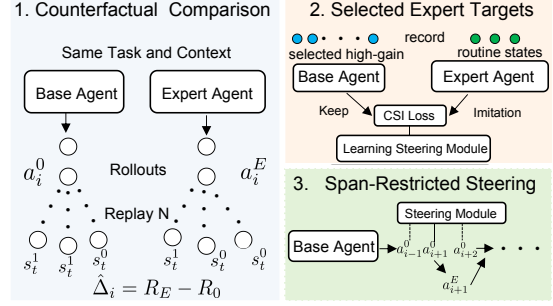


Figure 3: CSI consists of offline counterfactual comparison, selected target construction, and span-restricted steering. The offline stage compares base and expert actions from the same execution handle, builds expert and base target sets, and trains the action-span steering module. Deployment runs the adapted base policy once, without expert queries or replay.

ond, Selected Expert Targets uses this estimate to put only high-gain positive records in $\mathcal{D}_E$ while assigning routine records to $\mathcal{D}_0$ with base-policy targets. Third, Span-Restricted Steering trains an action-span module on these fixed target sets while keeping the base model frozen. The following subsections define the record buffer, comparison and target-set construction, steering module, span-level objective, and deployment.

4.1 Context Records

CSI starts from two sources of pre-action decision records. Expert rollouts provide expert-reached records $\mathcal{B}_E$, and base rollouts provide base-reached records $\mathcal{B}_0$. The combined record buffer is

$$\mathcal{B} = \mathcal{B}_E \cup \mathcal{B}_0. \quad (3)$$

Each element of $\mathcal{B}$ is a decision record,

$$e_i = (x_i, s_i, \text{src}_i), \quad \text{src}_i \in \{E, 0\}. \quad (4)$$

We treat $\mathcal{B}$ as a buffer rather than a duplicate-removing set. The serialized context x_i is the text input observed by the policies. The handle s_i is not observed by any policy. It is an execution handle used by the benchmark wrapper to resume environment execution immediately before action i . Depending on the benchmark, s_i may be implemented as a checkpoint id, a database handle, or a task id together with the logged action prefix. The source label src_i records whether the record was collected from an expert rollout or a base rollout. It does not determine the training target. This separation is important because an expert-reached state can still be routine for the base policy, and a base-reached state can contain a consequential decision where the expert action should be imitated.

Only eligible records are used for target assignment. A record is eligible for counterfactual comparison when the execution handle s_i can resume the environment, the context is nonterminal, the expert action can be parsed under the structured-action interface, and the corresponding action span can be identified. A record is eligible for a base-policy target only when the base action is also parseable, executable, and routine. A routine base action is one that is schema-valid, does not trigger a benchmark-level invalid/tool error, and has no positive expert-gain evidence requiring expert imitation. We write this condition as $\text{routine}_i^0 = 1$. If the base action is invalid, the record may still be used for expert comparison, but it is not used as a base target. When the benchmark defines an invalid-action transition, that transition is executed and its resulting return is used for the invalid branch. If no such transition is defined, the record is excluded from counterfactual comparison. This prevents the base target set from preserving actions that cannot be executed by the environment.

4.2 Counterfactual Comparison

For each eligible record e_i , the base and expert policies are queried on the same serialized context.

$$\begin{aligned} a_i^0 &= \text{Decode}(\pi_0(\cdot \mid x_i)), \\ a_i^E &= \text{Decode}(\pi_E(\cdot \mid x_i)). \end{aligned} \quad (5)$$

The comparison is therefore local to a single pre-action history, even if that history was originally reached by only one policy. After decoding, each action is parsed into a canonical structured action, a tokenized action, an action span, and a validity flag,

$$\text{Parse}(a_i^p) = (\bar{a}_i^p, y_i^p, I_i^p, \text{valid}_i^p), \quad p \in \{0, E\}. \quad (6)$$

Here $\bar{a}_i^p$ is used for canonical comparison, y_i^p and I_i^p are used later for span-level training, and valid_i^p indicates whether the action can be parsed and executed. The disagreement indicator is

$$d_i = 1[\bar{a}_i^E \neq \bar{a}_i^0]. \quad (7)$$

When the canonical actions match and the base action is executable, the record is not treated as an expert target. The base policy already proposes the same executable action as the expert after canonicalization. When canonical actions differ, CSI estimates whether replacing the base next action with the expert next action improves downstream return. The benchmark wrapper uses the same execution

handle s_i for both branches. One branch resumes execution from s_i and executes a_i^E . The other resumes from the same s_i and executes a_i^0 . After the first action, both branches are completed with the same continuation policy π_c and the same return definition. In all experiments, π_c is the frozen base policy π_0 with the same deterministic decoding used at deployment. This gives

$$J_i(a; \pi_c) = \mathbb{E}[R(\tau) \mid s_i, a_i = a, \pi_c], \quad (8)$$

where $R(\tau)$ is the scalar return of the completed trajectory. The counterfactual return difference is

$$\Delta_i = J_i(a_i^E; \pi_c) - J_i(a_i^0; \pi_c). \quad (9)$$

In practice, CSI uses

$$\begin{aligned} \hat{\Delta}_i &= \frac{1}{M} \sum_{m=1}^M [R_{i,m}^E - R_{i,m}^0], \\ R_{i,m}^p &= \text{Return}(s_i, a_i^p, \pi_c; m). \end{aligned} \quad (10)$$

Here $\text{Return}(s_i, a, \pi_c; m)$ is the return from replay m , resuming s_i , executing a , and continuing with π_c . This isolates the current action replacement.

4.3 Selected Expert Targets

The expert target set contains states with large positive estimated gains.

$$\mathcal{D}_E = \text{TopFrac}_p \left(\{i : d_i = 1, \hat{\Delta}_i > 0\}, \hat{\Delta}_i \right). \quad (11)$$

The operator TopFrac_p ranks positive-gain disagreement records by $\hat{\Delta}_i$ and keeps the top fraction p for expert imitation. Thus, p is defined within the positive-gain disagreement set, not over all records.² Small positive estimates that are not selected are ignored by default rather than forced toward either policy. This avoids copying small or noisy differences while also avoiding base targets on states where the expert branch appears better.

The base target set contains routine states that should stay close to the original policy.

$$\mathcal{D}_0 = \mathcal{D}_{\text{match}} \cup \mathcal{D}_{\leq 0} \cup \mathcal{D}_{\text{valid}}, \quad (12)$$

where $\mathcal{D}_{\text{match}} = \{i : d_i = 0, \text{valid}_i^0 = 1, \text{routine}_i^0 = 1\}$, $\mathcal{D}_{\leq 0} = \{i : d_i = 1, \hat{\Delta}_i \leq 0, \text{valid}_i^0 = 1, \text{routine}_i^0 = 1\}$, and $\mathcal{D}_{\text{valid}}$ contains valid routine base-rollout records that do not belong to $\mathcal{D}_E$ and do not have positive expert-gain evidence. Equivalently, weak positive records that are not selected into $\mathcal{D}_E$ are ignored by default. States in $\mathcal{D}_E$ use the expert action as the target, while states in $\mathcal{D}_0$ use the base action as the target.

²Appendix Table 4 reports the corresponding selected-target rates under disagreement and eligible-record denominators.

4.4 Span-Restricted Steering

CSI implements the selected target supervision by freezing the base model and training only a steering module. Each record used for training has a target source, expert for records in $\mathcal{D}_E$ and base for records in $\mathcal{D}_0$. We write this source as $z_i \in \{E, 0\}$. This is only a bookkeeping variable that identifies which action provides the training target. It is not a learned gate. If $z_i = E$, the target action is y_i^E with span I_i^E . If $z_i = 0$, the target action is y_i^0 with span I_i^0 . We write these uniformly as $y_i^{z_i}$ and $I_i^{z_i}$. At layer ℓ , the frozen base model produces hidden state $h_{i,j}^{(\ell)}$ for token position j . The router maps a pooled representation u_i of the pre-action context to a K -dimensional steering code, and $P^{(\ell)}$ maps this code to an additive vector at layer ℓ .

$$\tilde{h}_{i,j}^{(\ell)} = h_{i,j}^{(\ell)} + \mathbf{1}[j \in I_i^{z_i}] P^{(\ell)} r_\phi(u_i). \quad (13)$$

Only the router and steering maps are trainable, while all base-model parameters remain frozen.

The intervention is local to executable action tokens and is trained with expert targets only on $\mathcal{D}_E$, limiting changes to the decisions supported by counterfactual evidence.

Algorithm 1 Counterfactual Selective Imitation

Require: Replayable environment $\mathcal{E}$, policies π_0, π_E
Ensure: Adapted policy π_θ

- 1: Set continuation policy $\pi_c \leftarrow \pi_0$
- 2: Collect expert-reached records $\mathcal{B}_E$ by rolling out π_E
- 3: Collect base-reached records $\mathcal{B}_0$ by rolling out π_0
- 4: Set $\mathcal{B} \leftarrow \mathcal{B}_E \cup \mathcal{B}_0$
- 5: Initialize all target sets to $\emptyset$
- 6: **for all** eligible records $e_i = (x_i, s_i, \text{src}_i) \in \mathcal{B}$ **do**
- 7: Decode actions a_i^0, a_i^E from x_i
- 8: Parse actions into $(\bar{a}_i^p, y_i^p, I_i^p, \text{valid}_i^p)$
- 9: Set $d_i \leftarrow \mathbf{1}[\bar{a}_i^E \neq \bar{a}_i^0]$
- 10: **if** $d_i = 0$ and $\text{valid}_i^0 = 1$ and $\text{routine}_i^0 = 1$ **then**
- 11: Add i to $\mathcal{D}_{\text{match}}$
- 12: **else**
- 13: Use execution handle s_i to evaluate expert and base branches
- 14: Estimate $\hat{\Delta}_i \leftarrow \text{expert return} - \text{base return}$
- 15: **if** $\hat{\Delta}_i \leq 0$ and $\text{valid}_i^0 = 1$ and $\text{routine}_i^0 = 1$ **then**
- 16: Add i to $\mathcal{D}_{\leq 0}$
- 17: **end if**
- 18: **end if**
- 19: **if** i is a valid routine base-rollout state with no positive expert-gain evidence **then**
- 20: Add i to $\mathcal{D}_{\text{valid}}$
- 21: **end if**
- 22: **end for**
- 23: $\mathcal{D}_E \leftarrow \text{TopFrac}_p(\{i : d_i = 1, \hat{\Delta}_i > 0\}, \hat{\Delta}_i)$
- 24: Build $\mathcal{D}_0$ from matching, non-positive, and valid routine base records
- 25: Freeze all parameters of π_0
- 26: Train only the span-restricted steering module using $\mathcal{L}_{\text{CSI}}$
- 27: **return** adapted policy π_θ

4.5 Span-Level Training Objective

Using the target source z_i defined above, let $q_{i,j}^{z_i}$ be the target distribution for token j in the selected action span. For states in $\mathcal{D}_E$, this target is derived from the expert action or expert distribution. For states in $\mathcal{D}_0$, it is derived from the frozen base policy. The training objective is

$$\mathcal{L}_{\text{CSI}} = \mathbb{E}_{i \sim \text{Mix}(\mathcal{D}_E, \mathcal{D}_0)} [\ell_i],$$

$$\ell_i = \sum_{j \in I_i^{z_i}} \text{KL}(q_{i,j}^{z_i} \parallel \pi_\theta(\cdot \mid x_i, y_{i,<j}^{z_i})). \quad (14)$$

The sampler $\text{Mix}(\mathcal{D}_E, \mathcal{D}_0)$ uses a fixed ratio between selected expert-target records and base-target records. When full target logits are not available, $q_{i,j}^{z_i}$ is the one-hot distribution for the target token, reducing the corresponding term to token-level negative log likelihood. The objective itself is standard span-level policy distillation. The method-specific part is the construction of $\mathcal{D}_E$ and $\mathcal{D}_0$, which decides which records imitate the expert and which records remain close to the base policy.

4.6 Putting It Together

Algorithm 1 summarizes the full procedure. It separates offline target construction from the frozen-base steering update.

At deployment, CSI receives only the current serialized context and generates one action with the adapted policy. It does not use execution handles s_i , replay the environment, query the expert, or sample multiple candidate actions. The steering module is active only during action generation. Under the single-line structured-action format, CSI identifies the inference-time action span online using the first serialized action token and the end-of-action delimiter as its boundaries. The lightweight steering module is the only source of additional test-time computation; deployment otherwise consists of a single deterministic rollout of the adapted policy.

5 Experiments

Benchmarks and models. We evaluate on AppWorld (Trivedi et al., 2024), ToolSandbox (Lu et al., 2025), and WebShop (Yao et al., 2022), which cover stateful API tasks, simulated tool conversations, and web shopping with structured actions. All methods use the same benchmark action interface, parser, and evaluation split. The student policy is Qwen3-8B-Instruct, and the offline expert policy is Qwen3-32B (Qwen Team, 2025). We use one model family to keep the tokenizer, action interface, and expert source controlled, so that differences mainly reflect target construction and

Method	AppWorld		ToolSandbox		WebShop		Relative latency ↓
	Task success ↑	Invalid/tool error ↓	Task success ↑	Invalid/tool error ↓	Task success ↑	Invalid/tool error ↓	
Qwen3-8B base	24.0	10.8	31.0	9.8	45.0	7.4	1.00×
Qwen3-32B expert	42.0	4.1	50.0	4.0	66.0	2.4	1.55×
Base@4	32.0	9.2	40.2	8.4	57.8	6.5	3.80×
Dense expert distillation	34.5	13.1	39.5	11.2	54.5	9.8	1.00×
Dense LoRA SFT	32.8	12.4	37.8	10.4	53.2	8.9	1.00×
Agent correction	38.9	7.2	44.6	6.4	61.7	5.2	1.00×
Critical imitation	37.2	9.4	42.4	7.8	60.0	6.4	1.00×
Dense steering	36.4	5.5	42.2	5.6	58.6	4.6	1.04×
CSI	39.6	5.8	46.3	5.1	63.1	3.8	1.05×

Table 2: Main results on AppWorld, ToolSandbox, and WebShop. Rows above the separator are reference policies that change model scale or test-time sampling budget. Rows below the separator are adapted Qwen3-8B baselines trained and deployed under the same structured-action protocol. Relative latency is deployed inference latency normalized by the Qwen3-8B base policy.

adaptation rather than model-family effects. All trainable baselines start from Qwen3-8B-Instruct. When a baseline requires expert labels or corrections, it uses the same Qwen3-32B expert as CSI, so differences come from data construction and adaptation rather than expert-policy strength.

Metrics. Task success is the benchmark-defined final success score, reported as a percentage. Invalid/tool error is the benchmark-defined rate of invalid structured actions or tool execution errors. For AppWorld, collateral database changes are counted in this family only when they correspond to an executed action that violates the task state. For ToolSandbox, milestone progress is used for analysis, while the main table reports final task success for consistency. Relative latency measures deployed inference time normalized by the Qwen3-8B base under the same decoding protocol.

Baselines. We compare reference policies and adapted baselines under the common protocol above. The references include the Qwen3-8B base, the Qwen3-32B expert, and Base@4. The expert changes model scale, and Base@4 changes the test-time sampling budget by drawing four base-policy rollouts. The adapted baselines re-implement prior algorithms with the same student, expert, action interface, and train/test split. *Dense expert distillation* tests dense teacher imitation through distillation or supervised action-span fitting (Hinton et al., 2015; Rusu et al., 2016; Kim and Rush, 2016). *Dense LoRA SFT* tests the same dense teacher signal with a LoRA update (Hu et al., 2022). *Agent correction* tests supervised adaptation from learner-state expert corrections (Lauffer et al., 2025). *Critical imitation* tests ATLAS-style critical-step imitation (Chen et al., 2025). *Dense steering* tests frozen-base action-span steering with dense expert targets, following activation-steering methods

(Turner et al., 2023; Stolfo et al., 2025). Appendix Table 8 adds record sources, target rules, update mechanisms, and offline construction costs.

Implementation details. All compared systems use deterministic single-action decoding, except Base@4, whose four samples define the test-time sampling baseline. The main CSI setting uses offline replay count $M = 3$ for each expert and base branch, $p = 20\%$, steering basis size $K = 16$, and one expert-target record for every two base-target records. These replays are used only for target assignment and are excluded from relative latency. Appendix Table 7 gives the remaining layer, router, optimizer, and decoding settings.

Main results. Table 2 reports per-benchmark scores. CSI improves task success over the Qwen3-8B base on AppWorld, ToolSandbox, and WebShop while keeping invalid/tool error below Dense expert distillation and Dense LoRA SFT in each case. Compared with Base@4, CSI reaches higher success on ToolSandbox and WebShop and comparable success on AppWorld without extra test-time samples. Against Dense steering, CSI improves success most clearly on ToolSandbox and WebShop, with comparable or lower invalid/tool error.

Success-validity trade-off. Figure 4 summarizes the main comparison as displacement from the base policy. Among single-rollout adapted baselines, CSI gives the largest average success gain while also reducing invalid/tool error relative to the base. Dense steering already improves validity, but it stays left of CSI. The arrow from Dense steering to CSI therefore isolates the part of the gain that comes from target construction under the same frozen action-span steering interface.

Steering capacity. Figure 5 varies the steering basis size K under the same target construction.

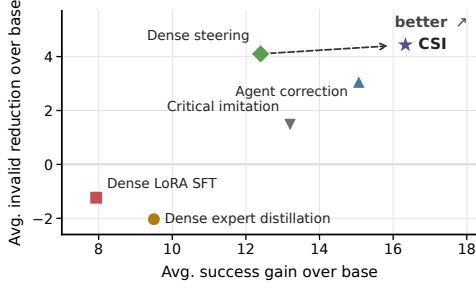


Figure 4: Average success-validity displacement from Qwen3-8B base. Points are single-rollout adapted base-lines averaged over the main benchmark suite. Higher and farther right is better.

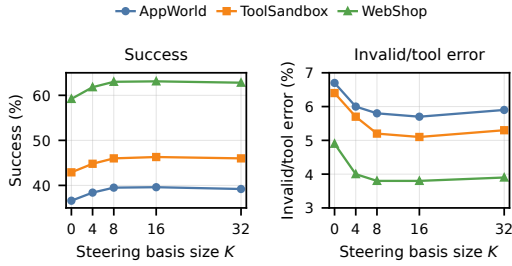


Figure 5: Effect of steering basis size K under fixed CSI target sets. Panels report per-benchmark success and invalid/tool-error rates.

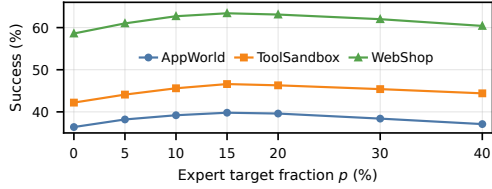


Figure 6: Effect of selected expert target fraction p within positive-gain disagreements. The sweep reports per-benchmark success.

Across all three benchmarks, increasing K gives most of its gains at small capacities and then plateaus around $K = 8$ – 16 , while invalid/tool error remains stable. This suggests that selected target supervision does not require a large update to the base model. A small action-span steering module is enough in the evaluated setting. The plateau is useful for deployment because it separates the target-selection benefit from simply adding more trainable capacity.

Sparse target rate. Figure 6 varies the fraction of positive-gain disagreements that receive expert targets. Performance improves when p moves away from the very sparse setting, which means that keeping only a few top-ranked records discards useful expert information. The gains flatten around

Variant	AppWorld		ToolSandbox		WebShop	
	Task success $\uparrow$	Invalid/tool error $\downarrow$	Task success $\uparrow$	Invalid/tool error $\downarrow$	Task success $\uparrow$	Invalid/tool error $\downarrow$
CSI	39.6	5.8	46.3	5.1	63.1	3.8
w/o counterfactual comparison	36.2	9.4	42.8	8.3	59.4	7.0
w/o selected expert targets	38.5	7.2	44.8	6.2	61.9	5.1
w/o span-restricted steering	38.2	9.6	44.9	7.9	62.0	6.3

Table 3: Core ablations on Qwen3-8B-Instruct with Qwen3-32B offline supervision. Rows marked ‘w/o’ remove the corresponding CSI component under the same benchmark suite and metrics. The steering ablation replaces the module with a LoRA update on the same targets.

$p = 15$ – 20 and then fall when p is pushed to 30 or 40. This pattern supports the target-construction rule in Section 4.3. Broad positive-gain imitation is still too wide because lower margin records begin to dilute the routine behavior preservation supplied by base targets. Appendix A.2 reports the paired p and K sweeps, the replay-count check, and the target-ratio check.

Core ablations. Table 3 reports the three component checks under the same benchmark and metric scope. Removing counterfactual comparison reduces success and raises invalid/tool error, showing that using every disagreement between expert and base actions is too broad a target. Removing selected expert targets also weakens the trade-off, which suggests that CSI should imitate the expert only at selected states with clear estimated gain rather than every positive return disagreement. Replacing span-restricted steering with a low-rank adapter update under the same targets keeps success comparable but raises invalid/tool error, suggesting that the frozen-base action-span implementation helps preserve routine behavior.

6 Conclusion

In this paper, we propose CSI, a return-aware target-selection framework for long-horizon agents. CSI uses same-state counterfactual rollouts to identify decisions where expert actions yield high return gains, assigns expert targets to those decisions, preserves routine behavior with base targets, and confines steering to executable action spans. Experiments on three representative benchmarks for long-horizon structured-action agents show higher success and fewer invalid/tool errors with only slight deployment-time overhead. Overall, CSI imitates the expert only for high-gain decisions while protecting routine base behavior.

Limitations

CSI relies on replayable environments, hidden-state interventions, and identifiable action spans, so its current form assumes access and environmental control that may not be available in some deployments. Our evaluation uses a single Qwen3 student-expert pair on structured-action benchmarks, leaving model diversity open. Different model families, tokenizers, decoding behaviors, and expert strengths may change which expert-base disagreements become high-gain targets. CSI also assumes stable same-state replay. In dynamic environments, external state may change between counterfactual branches, making the estimated return difference less reliable. The frozen base continuation estimates base-policy-realizable one-action gain, so it can miss expert actions that require expert-style follow-up, and the estimate may shift after adaptation. Future work should test CSI across more student-expert pairs, study broader continuation policies, and develop approximations for settings without stable replay handles or consistent environment snapshots.

Ethical Considerations

CSI improves agents that can modify external state through APIs or tools, but this may also amplify risks from misconfiguration, unauthorized use, or malicious actors. Real-world deployments should therefore enforce least-privilege permissions and require confirmation for irreversible or high-impact actions.

Acknowledgments

We sincerely thank the anonymous reviewers for their valuable feedback and guidance. This work was supported by the National Natural Science Foundation of China under Grant No. 62272171. Chuliang Weng is the corresponding author.

References

- Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. 2024. [On-policy distillation of language models: Learning from self-generated mistakes](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. [Fireact: Toward language agent fine-tuning](#). *CoRR*, abs/2310.05915.
- Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. 2024. [Agent-flan: Designing data and methods of effective agent tuning for large language models](#). In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of *Findings of ACL*, pages 9354–9366. Association for Computational Linguistics.
- Zhixun Chen, Ming Li, Yuxuan Huang, Yali Du, Meng Fang, and Tianyi Zhou. 2025. [ATLAS: agent tuning via learning critical steps](#). In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, volume ACL 2025 of *Findings of ACL*, pages 25334–25349. Association for Computational Linguistics.
- Dayuan Fu, Keqing He, Yejie Wang, Wentao Hong, Zhuoma Gongque, Weihao Zeng, Wei Wang, Jingang Wang, Xunliang Cai, and Weiran Xu. 2025. [Agentrefine: Enhancing agent generalization through refinement tuning](#). In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.
- Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujia Yang, Nan Duan, and Weizhu Chen. 2024. [CRITIC: large language models can self-correct with tool-interactive critiquing](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. [Minillm: Knowledge distillation of large language models](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. [Distilling the knowledge in a neural network](#). *CoRR*, abs/1503.02531.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [Lora: Low-rank adaptation of large language models](#). In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.
- Yoon Kim and Alexander M. Rush. 2016. [Sequence-level knowledge distillation](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1317–1327. The Association for Computational Linguistics.
- Niklas Lauffer, Xiang Deng, Srivatsa Kundurthy, Brad Kenstler, and Jeff Da. 2025. [Imitation learning for multi-turn LM agents via on-policy expert corrections](#). *CoRR*, abs/2512.14895.
- Kenneth Li, Oam Patel, Fernanda B. Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. [Inference-time](#)

- intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Mukai Li, Qingcheng Zeng, Tianqing Fang, Zhenwen Liang, Linfeng Song, Qi Liu, Haitao Mi, and Dong Yu. 2026. [Verified critical step optimization for LLM agents](#). In *Findings of the Association for Computational Linguistics, ACL 2026, San Diego, California, United States, July 2-7, 2026*, pages 39627–39639. Association for Computational Linguistics.
- Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. 2025. [Toolsandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities](#). In *Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, volume NAACL 2025 of *Findings of ACL*, pages 1160–1183. Association for Computational Linguistics.
- Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. [Gorilla: Large language model connected with massive apis](#). In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. [Toolllm: Facilitating large language models to master 16000+ real-world apis](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Qwen Team. 2025. [Qwen3 technical report](#). *CoRR*, abs/2505.09388.
- Nina Rimskey, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. 2024. [Steering llama 2 via contrastive activation addition](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 15504–15522. Association for Computational Linguistics.
- Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. [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, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pages 627–635. JMLR.org.
- Andrei A. Rusu, Sergio Gomez Colmenarejo, Çağlar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. 2016. [Policy distillation](#). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. [Toolformer: Language models can teach themselves to use tools](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. [Reflexion: language agents with verbal reinforcement learning](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Alessandro Stolfo, Vidhisha Balachandran, Safoora Yousefi, Eric Horvitz, and Besmira Nushi. 2025. [Improving instruction-following in language models through activation steering](#). In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.
- Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. 2024. [Appworld: A controllable world of apps and people for benchmarking interactive coding agents](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 16022–16076. Association for Computational Linguistics.
- Alexander Matt Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. 2023. [Activation addition: Steering language models without optimization](#). *CoRR*, abs/2308.10248.
- Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. [Webshop: Towards scalable real-world web interaction with grounded language agents](#). In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. [React: Synergizing reasoning and acting in language models](#). In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2024. [Agenttuning: Enabling generalized agent abilities for llms](#). In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of *Findings of ACL*, pages 3053–3077. Association for Computational Linguistics.

Andy Zou, Long Phan, Sarah Li Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, and 2 others. 2023. [Representation engineering: A top-down approach to AI transparency](#). *CoRR*, abs/2310.01405.

A Appendix Details

A.1 Target and ablation details

Table 4 makes the denominator conventions explicit for Table 1 and Figure 2. The main setting uses $p = 20\%$ within positive-gain expert-base disagreements. Under this setting, selecting the top p percent of positive-gain disagreements corresponds to 8–9% of expert-base disagreements, or 2.2–3.9% of eligible records.

These percentages use two denominators. Disagreement is measured over eligible records. Positive, selected, small positive, and no-gain groups are measured over expert-base disagreements. The selected column applies the main $p = 20$ setting to the positive-gain subset, then reports the result over all disagreements.

Benchmark	Disagreement over eligible	Positive over disag.	Selected over disag.	Small positive over disag.	No gain over disag.	Selected over eligible	Weak positive over eligible	Base-target candidates over eligible
AppWorld	34	42	8	34	58	2.9	11.4	85.7
ToolSandbox	29	38	8	30	62	2.2	8.8	89.0
WebShop	41	47	9	38	53	3.9	15.4	80.7

Table 4: Target construction statistics under $p = 20\%$. All values are percentages. Disag. abbreviates expert-base disagreements.

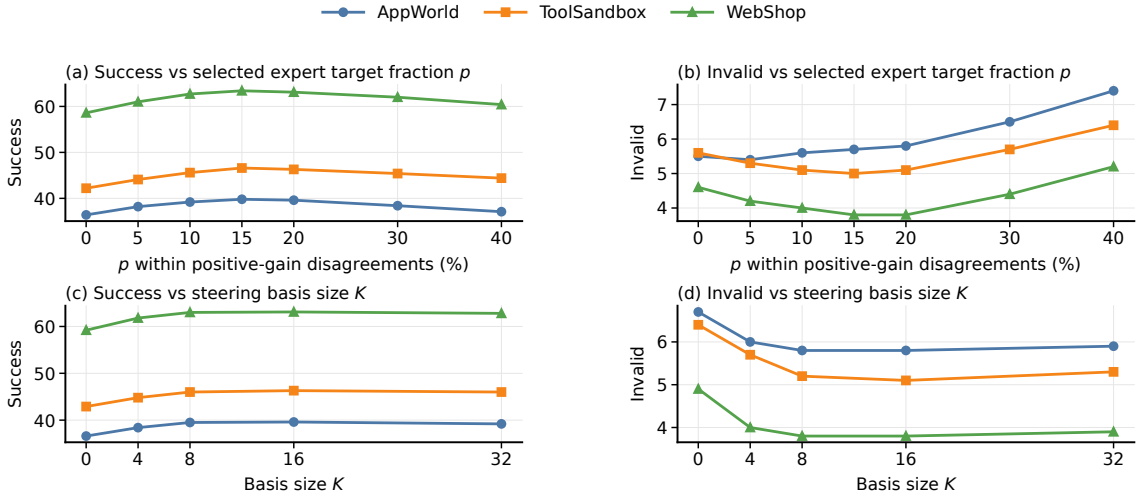


Figure 7: Hyperparameter sweeps for CSI. The plots vary expert target fraction p and steering basis size K around the main setting, $p = 20\%$ and $K = 16$.

M	AppWorld Succ.	AppWorld Invalid	ToolSandbox Succ.	ToolSandbox Invalid	WebShop Succ.	WebShop Invalid	Avg. Succ.	Avg. Invalid	Selected-set overlap with $M = 3$
1	38.8	6.3	45.4	5.7	62.2	4.4	48.8	5.5	78%
2	39.3	6.0	46.0	5.3	62.8	4.0	49.4	5.1	89%
3	39.6	5.8	46.3	5.1	63.1	3.8	49.7	4.9	100%
5	39.7	5.7	46.4	5.0	63.2	3.8	49.8	4.8	95%

Table 5: Effect of replay count M , with $M = 3$ as the main setting.

A.2 Sensitivity analysis

The sensitivity analysis varies the settings most directly tied to target construction and steering, the expert target fraction p , steering basis size K , replay count M , and the expert-to-base target ratio. Moderate p values keep success high without increasing invalid/tool error, while overly broad expert targeting starts to disturb routine actions. The K sweep plateaus around $K = 8$ to 16, which supports using a small steering basis.

Table 5 varies the replay count used to estimate the same-state comparison signal.

Table 6 varies the expert-to-base target sampling ratio used during span training.

Target ratio $\mathcal{D}_E : \mathcal{D}_0$	1:0	1:1	1:2	1:4	1:8
Avg. success	49.4	49.9	49.7	49.2	48.3
Avg. invalid	6.1	5.3	4.9	4.8	4.7

Table 6: Effect of the D_E to D_0 target ratio, with 1 to 2 as the main setting.

A.3 Implementation parameters

Setting	Value
Student policy	Qwen3-8B-Instruct
Expert policy	Qwen3-32B
Replay trials M	3 per branch, offline only
Continuation policy	frozen Qwen3-8B base policy
Expert target fraction p	20% of positive-gain disagreements
Steering basis size K	16
Injection layers	three fixed middle-late transformer layers
Router r_ϕ	two-layer MLP
Target sampling ratio	1 expert-target record to 2 base-target records
$D_E : D_0$	
Optimizer	AdamW
Learning-rate search range	$1e-4$ to $5e-4$
Main decoding	deterministic single-action decoding
Deployment	one adapted Qwen3-8B rollout, no expert query, no replay
Offline hardware	four NVIDIA L20 GPUs for decoding and replay, one NVIDIA A40 GPU for steering training

Table 7: Default CSI values and searched ranges for the main results.

We use hard action-token targets when cached expert-policy logits are unavailable. Otherwise, we cache action-span logits only.

A.4 Baseline construction and offline cost

For CSI, $M = 3$, so the offline replay budget is $2M$ branch evaluations per expert-base disagreement. With the reported disagreement rates, this corresponds to 2040, 1740, and 2460 branch replays per 1000 eligible records for AppWorld, ToolSandbox, and WebShop. These replays are used only for target construction.

A.5 Selection diagnostics

CSI makes a target decision by comparing the expert and actual base branches from the same state under the same frozen-base continuation. This shared continuation isolates the effect of the candidate action rather than differences in downstream policies. The resulting graded return difference determines whether a record receives an expert target, a base target, or no target. Unlike on-policy correction, which supplies expert labels at learner-visited states, CSI also tests whether the expert action should replace the action actually taken by the base policy.

Table 9 illustrates three distinct failure modes: action serialization in AppWorld, an unmet tool precondition in ToolSandbox, and incorrect action ordering in WebShop. In each case, the selected expert action changes the downstream state so that the base-policy continuation can complete the task. Table 10 complements these examples with disagreements in both directions: CSI-only cases contain outcome-changing replacements, whereas Critical-imitation-only cases retain outcome-equivalent base actions.

B Use of AI Assistants

We used large language models to improve the language quality of the manuscript, including grammar correction, wording refinement, and clarity enhancement, and for limited coding assistance, such as debugging and implementation suggestions.

Method	Records and targets	Update	Offline extra	Deployment
Dense expert distillation	eligible records, dense Qwen3-32B expert targets	distillation or supervised action-span fitting	expert labels	one Qwen3-8B rollout, $1.00\times$
Dense LoRA SFT	same dense expert targets	LoRA	expert labels	one Qwen3-8B rollout, $1.00\times$
Agent correction	learner-visited states, Qwen3-32B correction or continuation	supervised adaptation	student rollouts and expert corrections	one Qwen3-8B rollout, $1.00\times$
Critical imitation	Qwen3-32B expert trajectories, ATLAS-selected critical steps	supervised adaptation	expert trajectories and critical-step selection	one Qwen3-8B rollout, $1.00\times$
Dense steering	eligible records, dense Qwen3-32B expert targets	frozen-base action-span steering	expert labels	one steered Qwen3-8B rollout, $1.04\times$
CSI	base-reached and expert-reached records, top- $\hat{\Delta}$ expert targets plus routine base targets	frozen-base action-span steering	expert and base decodes, $2M$ branch replays per expert-base disagreement	one steered Qwen3-8B rollout, $1.05\times$

Table 8: Baseline construction and offline cost. All adapted baselines use Qwen3-8B-Instruct as the student, the same structured-action interface, the same train/test split, and deterministic single-rollout evaluation. Methods requiring expert labels or corrections use the same Qwen3-32B expert. Offline cost is used only for training data construction and is not included in relative latency.

Benchmark case	Base action	Expert action	$\hat{\Delta}$	Why the expert action is selected
AppWorld, Venmo friend sync	for add_friend(...); for remove_friend(...)	...: to_add phone_friend_emails - ...: venmo_friend_emails	= +1.00	The compressed loops cause a syntax error. The expert first creates an executable intermediate assignment, after which the base continuation completes the update.
ToolSandbox, message with cellular off	send_message_with_phone_number(...)	set_cellular_service_status(enabled=True)	+1.00	The base action violates the cellular precondition. Enabling service lets the base continuation send the message.
WebShop, required option	click[Buy Now]	click[khaki]	+0.67	The base attempts to buy before selecting a required option. The expert first changes the product state so the continuation can complete a valid purchase.

Table 9: Representative CSI-selected states. Return difference is the expert-minus-base mean over three rollouts per branch under the same frozen-base continuation.

Benchmark	Overlap	CSI only	Critical imitation only
AppWorld	85.3%	Spotify queue cleanup, where the expert grounds the queue position.	A read-only listing where both actions reach the same outcome.
ToolSandbox	73.5%	The cellular precondition case in Table 9, where enabling service allows the message to be sent.	Contact canonicalization where both actions reach the same outcome.
WebShop	80.9%	The required-option case in Table 9, where choosing the option enables a valid purchase.	Alternative search wording that reaches the same product.

Table 10: Overlap and representative disagreements between CSI and Critical imitation selectors under the common Qwen3-8B student, Qwen3-32B expert, action interface, and data split.

C Prompts

Expert Prompt for Expert Policy π_E

You are the **expert policy** for a long-horizon structured-action environment. Your job is to complete the task with high success while avoiding invalid action calls, wrong arguments, and unintended state changes.

Input. At each step you receive the following fields.

- TASK, the user goal or benchmark instruction.
- OBSERVATION, the current environment state, visible text, tool outputs, or API state.
- HISTORY, previous actions and observations.
- ACTION_SCHEMA, the allowed structured actions for this benchmark.

Output format. Output exactly one line.

ACTION: <action_string>

Do not output reasoning, explanations, or extra text.

Action rules.

1. Choose only actions allowed by ACTION_SCHEMA.
2. Copy action names, argument names, entity identifiers, and option IDs exactly when they are provided by the observation.
3. Prefer actions that advance the task state and reduce uncertainty.
4. Avoid changing database, cart, booking, or profile state unless the task requires that change.
5. If no valid progress action is available, choose the safest allowed recovery action.

Now produce the next action.

Base Prompt for Base Policy π_0

You are the **base policy** for a long-horizon structured-action environment. Your goal is to complete the task by issuing one valid structured action at each step.

Input. At each step you receive the following fields.

- TASK, the user goal or benchmark instruction.
- OBSERVATION, the current environment state, visible text, tool outputs, or API state.
- HISTORY, previous actions and observations.
- ACTION_SCHEMA, the allowed structured actions for this benchmark.

Output format. Output exactly one line.

ACTION: <action_string>

Do not output reasoning, explanations, or extra text.

Validity rules.

1. Use only actions that are valid under the current observation.
2. For identifiers, copy the value exactly as shown.
3. Keep action strings minimal and deterministic.
4. Do not invent tools, fields, entity IDs, or hidden state.
5. Prefer fewer steps, but do not skip required confirmation or verification actions.

Now produce the next action.