FROM FUTURE TO PRESENT: UTILITY-ROUTED FUTURE DISTILLATION FOR MOBILE AGENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Mobile GUI agents rely on current screens and interaction history, yet anticipating the state a task should reach next offers additional guidance for action decisions. Although logged trajectories naturally provide future screens, these observations are unavailable at deployment and are not uniformly beneficial for action prediction. We propose **Utility-Routed Future Distillation (URFD)**, which transfers useful future representations into a current-only GUI policy without additional annotations or future-image generation. **Task- and Action-Conditioned Successor Initialization (TASI)** combines dual-view action prediction with future-selection tasks to establish a future-aware initialization. URFD distills future-conditioned representations into a compact latent slot within a current-only policy. A frozen dual-view teacher quantifies future utility through the gain in demonstrated-action log-likelihood, jointly governing the interpolation of hidden-state targets and the weighting of policy distillation across current- and future-conditioned views. This utility-guided formulation mitigates interference from unhelpful future observations during distillation. The experimental results show that URFD improves average action match by 2.26 percentage points across two offline benchmarks and online task success by 3.5 points on AndroidWorld. Comprehensive analyses examine the utility of future supervision, strategies for representation alignment, and their impact on action decisions. These results demonstrate that URFD enables models to approximate hidden representations of real future states and improve action prediction accuracy without observing real future states at deployment.

1 INTRODUCTION

> *"present of things past, memory; present of things present, sight; present of things future, expectation."* —— Augustine

Rapid advances in vision-language models (VLMs) have enabled GUI agents to help users automate actions on mobile phones. Their decisions involve three complementary temporal perspectives. The **past** provides task progress and information recalled from earlier interactions, motivating historical screenshot resampling, history compression, and structured memory management (Lu et al., 2025; Chen et al., 2025; Zhou et al., 2025b; Tian et al., 2025). The **present** requires fine-grained screen understanding and effective reasoning to identify actionable elements and choose an appropriate operation (Cheng et al., 2024; Lin et al., 2025; Xu et al., 2025; Qin et al., 2025). The **future** provides an expectation of what an action should accomplish and which interface state should result. While much of GUI-agent research focuses on understanding present interfaces and past history, learning to anticipate and leverage future states to guide current decisions is also equally important.

Existing approaches to incorporating future-state information into GUI agents follow two complementary directions: **(1) predicting the state after a specified action**, as in GUI world models with text/sketch (Li et al., 2025; Cao et al., 2026), image (Luo et al., 2025), or code-based outputs (Koh et al., 2026; Zheng et al., 2026; Xu et al., 2026b). Their objective is to model action-conditioned outcomes rather than task-advancing future states; text/sketch and code-based variants also require corresponding successor targets. **(2) Using future states to regularize action prediction**, as in MIRAGE (Yang et al., 2026), which aligns latent CoT with future features. However, it relies on explicit reasoning supervision. We therefore ask: *can an agent learn useful future expectations to improve action prediction without additional annotations beyond existing GUI trajectories?*

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.

The most direct way is to provide the agent with the extra future state, which empirically improves action prediction accuracy (Section 4.4.1). Such training examples can be constructed directly from existing GUI trajectories, which already record the screen reached after each action, without additional manual annotation. However, the real future state is unavailable when the agents decide which action to execute at deployment. We therefore replace the future screen with a placeholder and train its hidden representations to approximate those induced by the real future state, rather than generating future pixels. Realizing this transfer requires addressing two challenges: **(1) establishing an understanding of plausible future states** from the task and current interaction context, and **(2) learning the future signals that are useful for action decisions**, rather than transferring potentially distracting information.

To address these challenges, we introduce **Task- and Action-Conditioned Successor Initialization (TASI)** and **Utility-Routed Future Distillation (URFD)**. Our design turns a 32-token copy of the current screen into a learnable placeholder for the future, without synthesizing future pixels. TASI aims to provide a future-aware initialization for subsequent alignment and distillation. It combines dual-view action prediction with two future-selection tasks: task-conditioned selection teaches goal-directed state progression, while action-conditioned selection teaches the expected consequences of a specified action. All supervision is constructed from existing trajectories without additional annotations. URFD aims to transfer future representations according to their value for action decisions, rather than treating every future as equally useful. It initializes a frozen dual-view teacher and a trainable CopyState student from the TASI model, and measures future utility by the teacher's improvement in demonstrated-action likelihood when given the real future. This utility jointly controls layer-wise hidden alignment and policy distillation: higher-utility examples receive stronger FutureState supervision, while unhelpful futures retain the CopyState teacher as an anchor. Hidden alignment guides the student's future-slot representations, while policy distillation and direct action supervision guide their use in action prediction. **The student thus learns to approximate useful future representations from current observations, improving action prediction accuracy.**

URFD demonstrates consistent improvements across benchmarks, increasing overall action match by an average of 2.26 percentage points across the two offline benchmarks (AndroidControl and GUI-Odyssey) and online task success by 3.5 points on AndroidWorld. Improvements are most pronounced for actions that induce visible state transitions, such as Click, Scroll, and Press. Further analyses support utility-based selection of beneficial futures and shallow-emphasized alignment. Together, these results support selective future distillation with a compact latent representation for more accurate current-state decisions.

Our main contributions are summarized as follows:

- We introduce **TASI** to establish future-state awareness from existing GUI trajectories without additional annotations, providing an initialized model with stable, future-aware representations for subsequent alignment and distillation.
- We propose **URFD** to jointly route hidden-state alignment and policy distillation according to future decision utility, which enables a copy-state student to approximate useful future representations for more accurate action prediction.
- We demonstrate consistent improvements across offline and online benchmarks and systematically analyze the utility of future supervision, strategies for representation alignment, and their impact on action decisions. These findings support utility-guided distillation for more accurate actions.

2 Related Work

GUI agents have progressed from visual grounding (Cheng et al., 2024; Wu et al., 2025) to integrated perception–action policies (Lin et al., 2025; Xu et al., 2025; Qin et al., 2025), with trajectory training and reinforcement learning expanding their navigation capabilities (Ye et al., 2025; Xu et al., 2026a; Gu et al., 2025; Venus Team et al., 2026; Zhou et al., 2025a). Context compression, structured memory, and history-aware optimization further support long-horizon interaction (Chen et al., 2025; Tian et al., 2025; Zhou et al., 2025b; Huang et al., 2025). Beyond understanding current and past states, GUI world models anticipate action consequences through images (Luo et al., 2025), semantic descriptions or sketches (Li et al., 2025; Cao et al., 2026), and renderable code (Koh et al., 2026; Zheng et al., 2026). These predictions support planning, policy guidance, and simulated train-

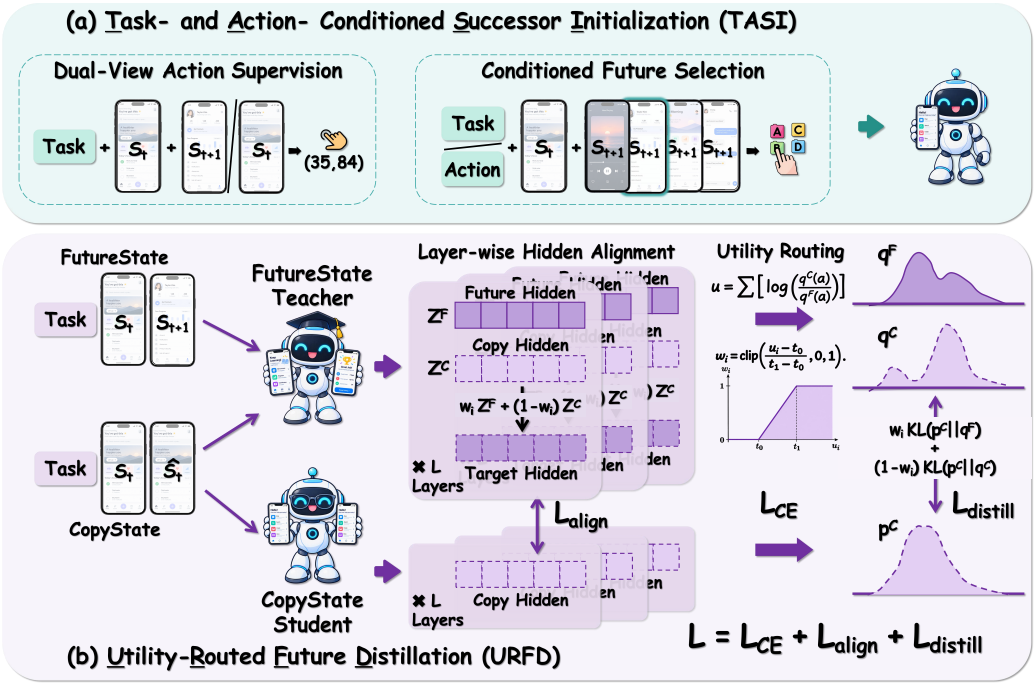


Figure 1: **Overview of TASI and URFD. (a)** TASI combines dual-view action supervision with task- and action-conditioned future selection to establish a future-aware initialization. **(b)** URFD trains a CopyState student with a frozen dual-view teacher. Future decision utility interpolates layer-wise hidden targets and weights policy distillation between CopyState and FutureState, alongside direct action supervision. Only the CopyState student is used at deployment.

ing (Chae et al., 2024; Gu et al., 2024; Guan et al., 2026; Xiao et al., 2026; Xu et al., 2026b), while latent approaches incorporate future-screen supervision into implicit reasoning (Yang et al., 2026). Our focus is utility-guided future distillation for current-only action prediction, without additional annotations or future-image generation.

3 METHOD

3.1 FORMULATION AND PAIRED-VIEW OVERVIEW

Mobile agents can autonomously complete user-specified tasks by interacting with mobile interfaces through a sequence of actions. At each step t, the agent π_θ receives a task instruction I, an interaction history H_t, and the current screenshot S_t, and predicts the next executable action a_t:

$$a_t \sim \pi_\theta(\cdot \mid I, H_t, S_t). \tag{1}$$

Thus, when making a decision, the agent can only rely on the task, interaction history, and current state. Empirically, action prediction improves substantially when the agent is additionally shown the screen state it is expected to reach next. This information is naturally available during training: the screen state reached after executing action a_t is already recorded in the logged trajectory as S_{t+1}. We therefore define the future-visible input x_t^F and future-informed agent π_θ^F as:

$$x_t^F = (I, H_t, S_t, S_{t+1}), \qquad a_t \sim \pi_\theta^F(\cdot \mid x_t^F), \qquad \mathcal{D}_F^{\text{act}} = \{(x_t^F, a_t)\}_{t \in \mathcal{T}}. \tag{2}$$

The real successor cannot be observed at deployment, since it becomes available only after a_t has been executed. We therefore reserve the second visual slot for a placeholder whose latent representation will be trained to carry future-relevant information. We instantiate this placeholder with a copy of S_t, the natural starting point of the transition toward S_{t+1}. Rather than synthesizing future pixels, we train the hidden states of this *CopyState* slot, together with the resulting action policy,

to approximate those induced by the future-informed view. We define the deployable current-only input x_t^C and its corresponding policy π_θ^C as

$$x_t^C = (I, H_t, S_t, S_t), \qquad a_t \sim \pi_\theta^C(\cdot \mid x_t^C), \qquad \mathcal{D}_C^{\text{act}} = \left\{(x_t^C, a_t)\right\}_{t \in \mathcal{T}}. \tag{3}$$

Our framework follows a two-stage procedure. The first stage, *Task- and Action-Conditioned Successor Initialization* (TASI), learns a shared successor-aware initialization over both views. The second stage, *Utility-Routed Future Distillation* (URFD), freezes the initialized policy as a paired-view teacher and optimizes a deployable CopyState student.

3.2 Task- and Action-Conditioned Successor Initialization

The central challenge for the CopyState student is to infer, from the task, history, and current screen, a latent expectation of a successor that has not yet been observed. If the model has not yet developed a stable representation of plausible future states, directly aligning its copy-state hidden features with future-state targets would provide a poorly grounded and unstable training signal. We therefore introduce *Task- and Action-Conditioned Successor Initialization* (TASI) as a preparatory stage that equips the model with successor awareness before distillation. TASI jointly trains dual-view action prediction together with two complementary successor-selection tasks.

Dual-view action modeling. For every logged transition, we construct two action-generation examples with the same target a_t. One uses x_t^C, teaching the model to act through the deployable placeholder interface; the other uses x_t^F, teaching it to exploit the observed future state. Jointly training the two views gives us a single policy that can be evaluated with or without the real successor and avoids introducing a separate teacher–student representation space before distillation begins.

Task- and action-conditioned successor selection. To explicitly teach what future state should be expected, we add two conditioned successor-selection tasks. The task-conditioned input asks for the state expected after the next correct action, whereas the action-conditioned input asks for the immediate state resulting from the specified action. These two tasks model goal-directed task progress and local action effects, respectively. For each non-terminal transition, the four candidates comprise: (i) the logged immediate successor S_{t+1}; (ii) a temporal negative sampled from another state in the same trajectory; (iii) a task negative sampled from a trajectory with a similar instruction; and (iv) a visual negative selected among screens most similar to S_{t+1}. We uniformly permute the four candidates to form $\mathcal{C}_t$, which satisfies $c_t^\star \sim \text{Uniform}(\{\text{A}, \text{B}, \text{C}, \text{D}\})$. The candidate list, conditioned inputs, and corresponding datasets are jointly defined as

$$\mathcal{C}_t = \text{Shuffle}\left(S_{t+1}, \widetilde{S}_t^{\text{temp}}, \widetilde{S}_t^{\text{task}}, \widetilde{S}_t^{\text{vis}}\right). \tag{4a}$$

$$x_t^{\text{task}} = (I, H_t, S_t, \mathcal{C}_t), \qquad \mathcal{D}_{\text{task}}^{\text{succ}} = \left\{(x_t^{\text{task}}, c_t^\star)\right\}_{t \in \mathcal{T}}. \tag{4b}$$

$$x_t^{\text{action}} = (S_t, a_t, \mathcal{C}_t), \qquad \mathcal{D}_{\text{action}}^{\text{succ}} = \left\{(x_t^{\text{action}}, c_t^\star)\right\}_{t \in \mathcal{T}}. \tag{4c}$$

TASI trains the union of the paired action data and successor-selection data with the standard autoregressive cross-entropy objective. This produces an initialized policy π_{θ_0} that can predict actions through both the copy-state and future-state interfaces and identify plausible successors from task- and action-conditioned context:

$$\mathcal{D}_{\text{TASI}} = \mathcal{D}_C^{\text{act}} \cup \mathcal{D}_F^{\text{act}} \cup \mathcal{D}_{\text{task}}^{\text{succ}} \cup \mathcal{D}_{\text{action}}^{\text{succ}}, \tag{5}$$

$$\mathcal{L}_{\text{TASI}} = \mathbb{E}_{(x,y) \sim \mathcal{D}_{\text{TASI}}}\left[-\log \pi_{\theta_0}(y \mid x)\right]. \tag{6}$$

3.3 Utility-Routed Future Distillation

TASI provides a shared, successor-aware starting point, but not every logged successor is an equally useful distillation target. Some successors expose visual evidence that clarifies the demonstrated action, whereas others are nearly unchanged, redundant with the current screen, or even make that action less likely. Forcing every copy-state representation fully toward its future-state counterpart can therefore transfer unhelpful future information. Therefore, we propose *Utility-Routed Future Distillation* (URFD), which measures how much the observed successor improves the demonstrated decision and uses this utility to control both representation-level and policy-level distillation.

Paired-view teacher and deployable student. Starting from the TASI policy π_{θ_0}, we create a frozen teacher π_θ^F and a trainable student π_θ^C, with $\pi_\theta^F = \pi_\theta^C = \pi_{\theta_0}$ at initialization. For each example, we feed both x_i^C and x_i^F to the frozen teacher to obtain a copy-state reference and a future-state target, while the trainable student receives only the deployable input x_i^C. The three branches produce

$$\{Z_{i,l}^C\}_{l=1}^L, \quad q_i^C = \pi_\theta^F(x_i^C), \tag{7}$$

$$\{Z_{i,l}^F\}_{l=1}^L, \quad q_i^F = \pi_\theta^F(x_i^F), \tag{8}$$

$$\{\widehat{Z}_{i,l}^C\}_{l=1}^L, \quad p_i = \pi_\theta^C(x_i^C). \tag{9}$$

Here, $Z_{i,l}^C$ and $Z_{i,l}^F$ are the teacher hidden states at the matched positions of the second visual slot in decoder layer l, and $\widehat{Z}_{i,l}^C$ is the corresponding student state. The distributions q_i^C, q_i^F, and p_i are the causal distributions over action tokens. The teacher remains frozen throughout training, and all quantities derived from it are treated as stop-gradient targets.

Decision utility of the observed successor. To determine whether future-state is worth transferring, we compare the frozen teacher's likelihood of the same demonstrated action under the copy-state and future-state views. Let $\mathcal{A}_i$ denote the supervised action-token positions of action a_i. We define its utility as the reduction in action prediction likelihood provided by the future-state view:

$$u_i = \frac{1}{|\mathcal{A}_i|} \sum_{j \in \mathcal{A}_i} \left[\log q_{i,j}^C(a_{i,j}) - \log q_{i,j}^F(a_{i,j})\right]. \tag{10}$$

The score is averaged over tokens to prevent actions with longer arguments from dominating the utility. A positive u_i indicates that the observed successor provides a utility gain for action prediction, whereas a non-positive value indicates that it provides no utility gain.

Continuous Copy-to-Future routing. We map the detached utility score to a routing coefficient

$$w_i = \mathrm{clip}\left(\frac{u_i - \tau_{\mathrm{start}}}{\tau_{\mathrm{full}} - \tau_{\mathrm{start}}}, 0, 1\right). \tag{11}$$

An example with $u_i \le \tau_{\mathrm{start}}$ remains anchored to the copy-state teacher, while one with $u_i \ge \tau_{\mathrm{full}}$ receives the full future-state target. Intermediate examples are routed continuously between the two. At decoder layer l, this gives the target

$$\widetilde{Z}_{i,l} = (1 - w_i) Z_{i,l}^C + w_i Z_{i,l}^F. \tag{12}$$

The routed target lies directly on the teacher's sample-specific Copy-to-Future trajectory. Thus, w_i determines how far the student should move toward future-state rather than merely deciding whether to retain or drop the example.

Layer-wise future alignment. The two views have identical token layouts, allowing corresponding tokens of the second visual slot to be matched directly. For P_i slot tokens with hidden dimension d, the layer-l distance is

$$D_{i,l} = \frac{1}{P_i d} \sum_{k=1}^{P_i} \left\| \widehat{Z}_{i,l,k}^C - \mathrm{sg}\left(\widetilde{Z}_{i,l,k}\right) \right\|_2^2. \tag{13}$$

We aggregate these distances across the batch and decoder layers using normalized reverse-exponential weights, which place greater emphasis on earlier layers and give transferred successor information more opportunity to propagate into the subsequent action computation:

$$\mathcal{L}_{\mathrm{align}} = \frac{1}{B} \sum_{i}^{B} \sum_{l=1}^{L} \omega_l D_{i,l}, \qquad \omega_l = \frac{\mathrm{e}^{(L-l)/(L-1)}}{\sum_{r=1}^{L} \mathrm{e}^{(L-r)/(L-1)}}. \tag{14}$$

Decision-level policy distillation. Future alignment alone does not guarantee that the action decoder will use the transferred information. We therefore route the teacher's action distributions with the same utility coefficient w_i. Let $M = \sum_i |\mathcal{A}_i|$ be the total number of supervised action tokens in the batch. The policy distillation loss is

$$\mathcal{L}_{\mathrm{distill}} = \frac{1}{M} \sum_{i} \sum_{j \in \mathcal{A}_i} \left[(1 - w_i) D_{\mathrm{KL}}\left(p_{i,j} \| \,\mathrm{sg}(q_{i,j}^C)\right) + w_i D_{\mathrm{KL}}\left(p_{i,j} \| \,\mathrm{sg}(q_{i,j}^F)\right)\right]. \tag{15}$$

When $w_i = 0$, the student preserves the copy-state teacher; when $w_i = 1$, it imitates the future-state teacher. Intermediate values interpolate the two KL objectives. Sharing w_i across latent and policy distillation moves both targets by the same sample-specific amount toward future-state targets.

Training objective and deployment. Direct supervision on the demonstrated action keeps the deployable copy-state policy grounded in the original action task:

$$\mathcal{L}_{\text{action}} = -\frac{1}{M} \sum_{i} \sum_{j \in \mathcal{A}_i} \log p_{i,j}(a_{i,j}). \tag{16}$$

The complete URFD objective is

$$\mathcal{L}_{\text{URFD}} = \mathcal{L}_{\text{action}} + \lambda_{align} \mathcal{L}_{\text{align}} + \lambda_{distill} \mathcal{L}_{\text{distill}}, \tag{17}$$

At deployment, we retain only the student policy π_θ^C, which receives $x_t^C = (I, H_t, S_t, S_t)$. From this input, the student produces hidden representations in the copy-state slot that approximate those induced by the unavailable future-state slot, thereby enabling future-informed action prediction without observing S_{t+1}.

4 Experiments

4.1 Experimental setup

Datasets and evaluation. We train on the training splits of AndroidControl (Li et al., 2024) and GUI-Odyssey (Lu et al., 2025), together with open-source AndroidWorld trajectories from MobileForge (Liu et al., 2026) and OpenMobile (Cheng et al., 2026). We evaluate on two offline benchmarks, AndroidControl-High and GUI-Odyssey, using GUIEvalKit (Qu et al., 2026), and one online benchmark, AndroidWorld (Rawles et al., 2024). For offline evaluation, we report step-level action-type accuracy (S.type) and action match (S.match), as well as trajectory-level success rate (T.succ) and prefix progress (T.prog). For AndroidWorld, we report cumulative task success on Easy, Medium, and Hard tasks. We report Pass@3 success rate over different complexity split.

Baselines and implementation. We compare URFD with ten GUI agents from seven model families—UI-TARS (Qin et al., 2025), GUI-Owl (Ye et al., 2025), UI-Venus (Gu et al., 2025), MAI-UI (Zhou et al., 2025a), GUI-Owl-1.5 (Xu et al., 2026a), UI-Venus-1.5 (Venus Team et al., 2026) and Mirage (Yang et al., 2026)—covering 2B and 7–8B parameter scales. All baselines receive a single current screenshot encoded into fewer than 1,500 visual tokens, together with a text-only action history without past screenshots. URFD additionally uses a copy-state slot with 32 visual tokens during both training and deployment. We train TASI for one epoch and URFD for three epochs, both with a global batch size of 128 and a learning rate of $1e^{-5}$ with cosine learning rate schedule. For URFD, we set the utility-routing interval to $[0, 0.01]$ and use loss coefficients $\lambda_{\text{align}} = \lambda_{\text{distill}} = 1$.

4.2 Main results

4.2.1 Offline Benchmark Results

AndroidControl. As shown in Table 1, URFD achieves 77.96% action match, exceeding the strongest released baseline, UI-TARS (7B), by 4.25 percentage points. Relative to current-only SFT, URFD improves action type and match by 0.62 and 1.20 points, respectively, while prefix progress increases from 47.27% to 50.49%. Offline trajectory success also increases from 29.58% to 31.31%. The larger gain in action match ($+1.20$ percentage points) than in action-type accuracy ($+0.62$ points) suggests that URFD's latent approximation of future states supports not only action-category selection but also more precise action-argument prediction.

GUI-Odyssey. URFD reaches 83.38% action match, surpassing GUI-Owl-1.5 (8B) by 2.43 points and current-only SFT by 3.32 points. Compared with SFT, URFD also improves action-type accuracy by 1.70 points, trajectory success by 5.65 points, and prefix progress by 4.87 points. The larger improvement in trajectory success, from 10.68% to 16.33%, is particularly relevant: this metric requires every evaluated step to match, so the benefit is not limited to isolated correct actions. These results show that URFD's gains extend beyond individual decisions to more fully correct offline trajectories on GUI-Odyssey, supporting its effectiveness for sequential action prediction.

Table 1: **Offline benchmark results on AndroidControl-High and GUI-Odyssey.** We report step-level action-type accuracy (S.type) and action match (S.match), as well as trajectory-level success rate (T.succ) and prefix progress (T.prog). *Best results are highlighted in bold.*

		AndroidControl				GUI-Odyssey			
Method	**Params**	**S.type**	**S.match**	**T.succ**	**T.prog**	**S.type**	**S.match**	**T.succ**	**T.prog**
UI-TARS	2B	77.12	69.53	17.92	26.87	85.76	63.53	1.08	17.96
GUI-Owl-1.5	2B	77.04	66.95	10.00	36.38	89.38	76.34	9.24	35.30
UI-Venus-1.5	2B	73.40	62.31	13.45	29.91	78.97	58.27	1.56	21.08
Mirage	4B	73.43	64.27	16.28	41.95	-	-	-	-
UI-TARS	7B	80.47	73.71	22.02	32.46	87.57	70.39	4.02	26.36
GUI-Owl	7B	79.23	70.09	18.75	32.52	77.49	60.36	1.56	20.16
UI-Venus	7B	74.75	64.80	9.88	24.68	85.98	69.36	3.66	26.51
MAI-UI	8B	72.54	64.11	13.21	23.95	85.27	69.24	3.48	21.58
GUI-Owl-1.5	8B	78.04	67.94	16.96	39.59	92.46	80.95	14.95	41.59
UI-Venus-1.5	8B	75.93	67.25	14.52	35.86	81.44	62.28	1.62	20.95
Qwen3-VL-Instruct	8B	75.19	66.33	14.17	24.55	83.97	64.34	2.28	20.55
Qwen3-VL-SFT	8B	84.62	76.76	29.58	47.27	91.71	80.06	10.68	39.08
URFD (ours)	**8B**	**85.24**	**77.96**	**31.31**	**50.49**	**93.41**	**83.38**	**16.33**	**43.95**

4.2.2 Online Benchmark Results

AndroidWorld online evaluation. Table 2 shows that URFD achieves 64.7% pass@3 (75/116 tasks), exceeding SFT, MAI-UI, and GUI-Owl-1.5 by 3.5, 5.2, and 6.9 percentage points, respectively. Its advantage over released GUI agents is concentrated on Easy and Medium tasks: URFD surpasses GUI-Owl-1.5 by 9.9 points on Easy tasks and both MAI-UI and GUI-Owl-1.5 by 5.5 points on Medium tasks. Relative to SFT, the gain comes from Medium tasks (58.3% versus 47.2%), while Easy and Hard success rates remain unchanged. Unlike offline action matching, these results measure task completion along the agent's own interaction sequence, extending the evaluation beyond individual decisions on recorded screens. Anticipating the next screen may help local decisions such as triggering functions or reaching target pages, whereas harder tasks may require long-horizon state management beyond one-step future supervision.

Table 2: **Pass@3 results on AndroidWorld.**

Method	**Easy**	**Medium**	**Hard**	**Overall**
General-agent references				
GPT 5.5	91.8	66.7	31.6	74.1
Seed 2.0	86.9	61.1	36.8	70.7
Open-source GUI-agent references				
MAI-UI	70.5	52.8	36.8	59.5
GUI-Owl-1.5	72.1	52.8	21.1	57.8
UI-Venus-1.5	65.6	33.3	15.8	47.4
GUI-Owl	60.7	41.7	10.5	46.6
UI-Venus	62.3	30.6	15.8	44.8
Qwen3-VL controls				
Qwen3VL-Inst	59.0	30.6	15.8	43.1
Qwen3VL-SFT	82.0	47.2	21.1	61.2
URFD (ours)	82.0	58.3	21.1	64.7

4.3 Ablation analysis

Removing any component reduces performance (Table 3; full results in Appendix C). Without utility routing or successor initialization, average match drops by 0.60 or 0.55 points, supporting their roles in filtering distracting future information and establishing a prior over plausible successors. Removing hidden alignment or policy KL costs 0.51 or 0.68 points: the former guides future representations, while the latter guides their use in action prediction. The drop without copy-state

Table 3: **Overall ablation results of URFD.**

Method	**Type**	**Match**
URFD	89.32	80.67
- w/o utility routing	88.82	80.07
- w/o successor initialization	88.84	80.12
- w/o hidden alignment	88.90	80.16
- w/o policy KL	88.81	79.99
- w/o copy-state CE	88.83	79.91

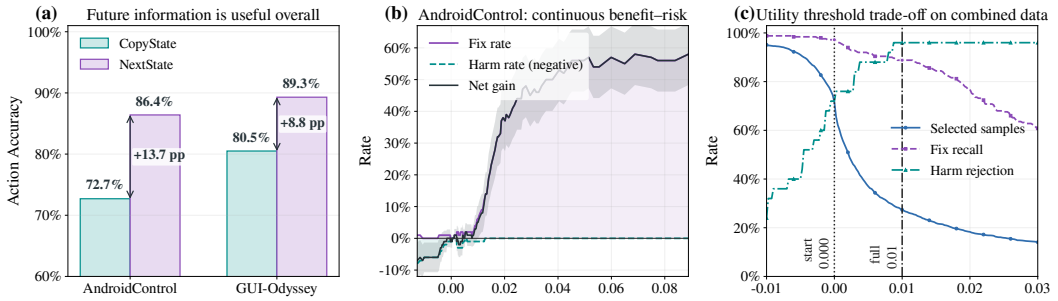


Figure 2: **Utility analysis of TASI teacher. (a)** CopyState versus FutureState action match. **(b)** Fix and harm rates by utility. **(c)** Selection rate, fix retention, and harm rejection across utility thresholds.

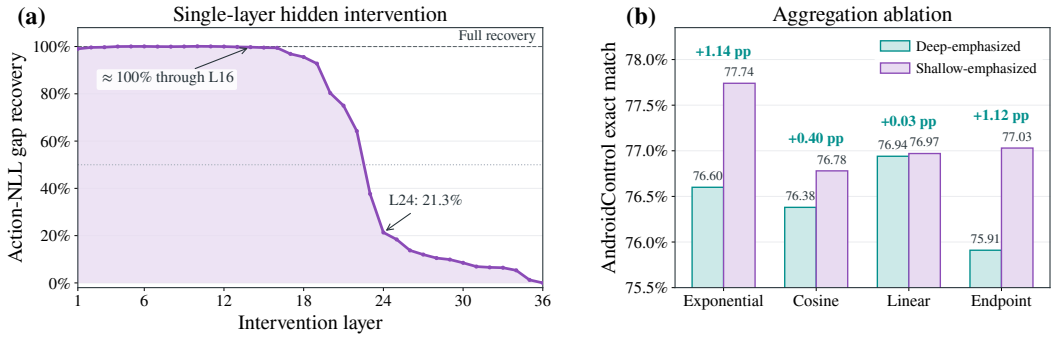


Figure 3: **Layer-wise future alignment. (a)** Action-NLL gap recovery from single-layer future-slot patching within the same model. **(b)** Shallow- versus deep-emphasized aggregation under AndroidControl Align+CE training; labels show action-match differences.

CE further supports retaining direct action supervision. **Together, URFD selects informative future representations and learns to approximate and exploit them for more accurate actions.**

4.4 Analysis of URFD design

4.4.1 Future utility and selection

Q1: Are future observations useful, and how can we select useful ones? On 1,000 non-terminal examples per dataset, the frozen TASI teacher improves action match by 13.7/8.8 points on AndroidControl/GUI-Odyssey when given the real successor (Figure 2(a)). Yet future observations can both correct errors (*fixes*) and invalidate correct decisions (*harms*). Low-utility groups contain as many or more harms than fixes, while fixes concentrate at higher utility (Figure 2(b)), supporting selective rather than uniform future supervision. To identify useful futures, a zero utility threshold selects 71.65% of the pooled examples, retains 97.2% of fixes, and rejects 72.0% of harms. Raising it to 0.01 selects 27.35%, retains 88.8% of fixes, and rejects 96.0% of harms (Figure 2(c)). **These results show that future observations are useful but not uniformly beneficial, and that action utility helps retain useful corrections while filtering harmful future supervision.**

4.4.2 Layer-wise future alignment

Q2: Where should future information be aligned? We patch the CopyState second-slot hidden states with FutureState states at one layer at a time, using the same NextState-trained model. Early interventions through layer 16 recover nearly the full action-NLL advantage, whereas recovery falls to 64.2% at layer 22 and 21.3% at layer 24 (Figure 3(a)). Earlier injection leaves more downstream computation to incorporate future information into action prediction. We further compare different layer-aggregation strategies (Figure 3(b)), and the results show that **assigning larger weights to shallower layers generally yields better performance, while the best all-layer strategy outperforms supervision at either endpoint alone**.

4.4.3 ALIGNMENT DISTANCE METRIC

Q3: Which align metric is most effective? We compare four alignment distances under uniform Future distillation: MSE, cosine, normalized MSE, and relative MSE. As shown in Table 4, MSE achieves the highest action match (83.52%), exceeding cosine distance by 0.38 percentage points and both normalized and relative MSE by 0.65 points. Unlike cosine distance, MSE constrains both feature direction and magnitude; normalization or relative scaling instead changes the emphasis placed on feature discrepancies. The results suggest that **preserving the original feature scale can benefit future alignment and downstream action prediction**.

Table 4: **Analysis of align metrics.**

Distance	Type	Match
MSE	**93.35**	**83.52**
+ Normalized	93.08	82.87
+ Relative	92.89	82.87
Cosine	93.09	83.14

4.4.4 FUTURE-SLOT VISUAL BUDGET

Q4: Does a larger future slot improve deployment? We compare 8–256 visual patches using DualView references under FutureState and CopyState inputs, alongside Align+CE students deployed with CopyState. Larger budgets improve the FutureState reference, but student accuracy peaks at 32 patches (Figure 4). FutureState match rises from 80.91% to 88.96%, while CopyState remains at 75.65–76.55% and students at 77.58–78.01%. Thus, encoding more observed future detail does not automatically translate into better current-only prediction, potentially because aligning a larger number of hidden-state tokens makes the distillation task more challenging. **These trends suggest a trade-off between alignment complexity and useful future information**.

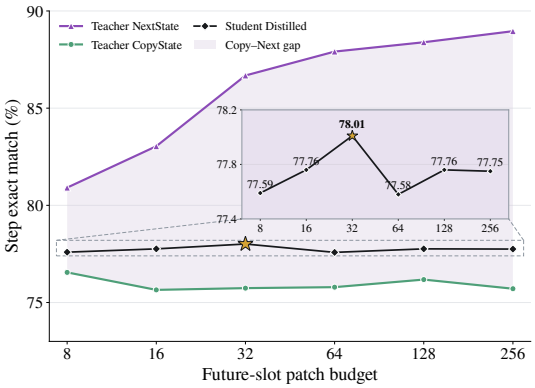


Figure 4: **Analysis of future-slot budget.**

4.4.5 ACTION-WISE PERFORMANCE

Q5: Which actions benefit from distillation? URFD improves full-action match over the TASI CopyState initialization on five of six action categories (Figure 5). Scroll and Press gain the most (7.67/7.83 points), whereas Stop declines by 3.48 points. One plausible explanation is that successor screens expose scroll direction and displacement, or the destination of a system-button press, reducing ambiguity about the demonstrated action. The large teacher Future–Copy gaps for Scroll and Press (24.26/17.56 points) are consistent with this interpretation, although they do not establish its causal mechanism. **Transfer benefits are action-dependent, with stopping behavior remaining a limitation.** Appendix D provides the numerical breakdown and further analysis.

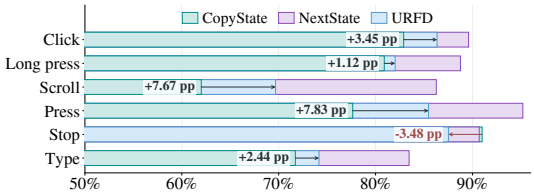


Figure 5: **Action-wise analysis of URFD**

5 CONCLUSION

We present a two-stage training framework that leverages logged future screenshots while preserving current-only deployment. TASI provides a successor-aware initialization; URFD then performs utility-routed hidden-state and policy distillation. The full framework improves step- and trajectory-level performance over the evaluated baselines on both offline benchmarks. Our analyses show that future information has uneven decision value and that transfer gains vary across actions. Recognizing an observed successor does not, by itself, demonstrate current-only anticipation; future work should improve predictability, causal use of the latent slot, and stopping behavior.

6 AI USE STATEMENT

Generative AI tools were used as research assistants in this work to support code development, experimental result aggregation, exploratory data analysis. Generative AI also assisted with manuscript drafting, language editing, and formatting in the official ICLR LaTeX template. The authors have reviewed and verified all AI-assisted work and take responsibility for the final text, claims, code, data, and artifacts.

7 ETHICS STATEMENT

This work studies GUI agents using existing trajectory datasets and benchmark environments. Such agents may expose sensitive information or execute unintended actions when deployed. Real-world applications should therefore require appropriate user authorization, privacy safeguards, and human confirmation for sensitive or irreversible operations.

8 REPRODUCIBILITY STATEMENT

Section 3 describes the paired-view formulation, TASI initialization, and URFD objectives. Appendix A summarizes dataset sizes and distributions, while Appendix B details the implementation, training hyperparameters, evaluation metrics, and comparison protocols. Appendices C–F provide additional ablation results and analyses of action-wise performance, future selection, and layer-wise alignment. Appendix G reproduces the action-generation prompt and an example output.

REFERENCES

Yilin Cao, Yufeng Zhong, Zhixiong Zeng, Siran Dai, Liming Zheng, Jing Huang, Haibo Qiu, Peng Shi, and Wenji Mao. MobileDreamer: Generative sketch world model for GUI agent. *arXiv preprint arXiv:2601.04035*, 2026.

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. *arXiv preprint arXiv:2410.13232*, 2024.

Gongwei Chen, Xurui Zhou, Rui Shao, Yibo Lyu, Kaiwen Zhou, Shuai Wang, Wentao Li, Yinchuan Li, Zhongang Qi, and Liqiang Nie. Less is more: Empowering GUI agent with context-aware simplification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5901–5911, 2025.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9313–9332, 2024.

Kanzhi Cheng, Zehao Li, Zheng Ma, Nuo Chen, Jialin Cao, Qiushi Sun, Zichen Ding, Fangzhi Xu, Hang Yan, Jiajun Chen, Anh Tuan Luu, Jianbing Zhang, Lewei Lu, and Dahua Lin. OpenMobile: Building open mobile agents with task and trajectory synthesis. *arXiv preprint arXiv:2604.15093*, 2026.

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. *arXiv preprint arXiv:2411.06559*, 2024.

Zhangxuan Gu, Zhengwen Zeng, Zhenyu Xu, Xingran Zhou, Shuheng Shen, Yunfei Liu, Beitong Zhou, Changhua Meng, Tianyu Xia, Weizhi Chen, et al. UI-Venus technical report: Building high-performance UI agents with RFT. *arXiv preprint arXiv:2508.10833*, 2025.

Yiming Guan, Rui Yu, John Zhang, Lu Wang, Chaoyun Zhang, Liqun Li, Bo Qiao, Si Qin, He Huang, Fangkai Yang, Pu Zhao, Lukas Wutschitz, Samuel Kessler, Huseyin A Inan, Robert Sim, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. Computer-using world model. *arXiv preprint arXiv:2602.17365*, 2026.

Kung-Hsiang Huang, Haoyi Qiu, Yutong Dai, Caiming Xiong, and Chien-Sheng Wu. GUI-KV: Efficient GUI agents via KV cache with spatio-temporal awareness. *arXiv preprint arXiv:2510.00536*, 2025.

Woosung Koh, Sungjun Han, Segyu Lee, Se-Young Yun, and Jamin Shin. Generative visual code mobile world models. *arXiv preprint arXiv:2602.01576*, 2026.

Shufan Li, Konstantinos Kallidromitis, Akash Gokul, Yusuke Kato, Kazuki Kozuka, and Aditya Grover. MobileWorldBench: Towards semantic world modeling for mobile agents. *arXiv preprint arXiv:2512.14014*, 2025.

Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on ui control agents. *Advances in Neural Information Processing Systems*, 37:92130–92154, 2024.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for gui visual agent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19498–19508, June 2025.

Guangyi Liu, Pengxiang Zhao, Gao Wu, Yiwen Yin, Mading Li, Liang Liu, Congxiao Liu, Zhang Qi, Mengyan Wang, Liang Guo, Jiangning Zhang, and Yong Liu. MobileForge: Annotation-free adaptation for mobile GUI agents with hierarchical feedback-guided policy optimization. *arXiv preprint arXiv:2606.19930*, 2026.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Lingxiao Du, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, and Ping Luo. GUIOdyssey: A comprehensive dataset for cross-app GUI navigation on mobile devices. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22404–22414, October 2025.

Dezhao Luo, Bohan Tang, Kang Li, Georgios Papoudakis, Jifei Song, Shaogang Gong, Jianye Hao, Jun Wang, and Kun Shao. ViMo: A generative visual GUI world model for app agents. *arXiv preprint arXiv:2504.13936*, 2025.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

Heng Qu, Yike Liu, Renren Jin, Wenzong Zhang, Pengzhi Gao, Wei Liu, and Jian Luan. Scaling, benchmarking, and reasoning of vision-language agents for mobile GUI navigation. *arXiv preprint arXiv:2605.27134*, 2026.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. AndroidWorld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*, 2024.

Shizuo Tian, Hao Wen, Yuxuan Chen, Jiacheng Liu, Shanhui Zhao, Guohong Liu, Ju Ren, Yunxin Liu, and Yuanchun Li. AgentProg: Empowering long-horizon GUI agents with program-guided context management. *arXiv preprint arXiv:2512.10371*, 2025.

Venus Team, Changlong Gao, Zhangxuan Gu, Yulin Liu, Xinyu Qiu, Shuheng Shen, Yue Wen, Tianyu Xia, Zhenyu Xu, Zhengwen Zeng, Beitong Zhou, Xingran Zhou, Weizhi Chen, Sunhao Dai, Jingya Dou, Yichen Gong, Yuan Guo, Zhenlin Guo, Feng Li, Qian Li, Jinzhen Lin, Yuqi Zhou, Linchao Zhu, Liang Chen, Zhenyu Guo, Changhua Meng, and Weiqiang Wang. UI-Venus-1.5 technical report. *arXiv preprint arXiv:2602.09082*, 2026.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. Os-atlas: Foundation action model for generalist gui agents. In *International Conference on Learning Representations*, volume 2025, pp. 5090–5108, 2025.

Zikai Xiao, Jianhong Tu, Chuhang Zou, Yuxin Zuo, Zhi Li, Peng Wang, Bowen Yu, Fei Huang, Junyang Lin, and Zuozhu Liu. WebWorld: A large-scale world model for web agent training. *arXiv preprint arXiv:2602.14721*, 2026.

Haiyang Xu, Xi Zhang, Haowei Liu, Junyang Wang, Zhaozai Zhu, Shengjie Zhou, Xuhao Hu, Feiyu Gao, Junjie Cao, Zihua Wang, Zhiyuan Chen, Jitong Liao, Qi Zheng, Jiahui Zeng, Ze Xu, Shuai Bai, Junyang Lin, Jingren Zhou, and Ming Yan. Mobile-Agent-v3.5: Multi-platform fundamental GUI agents. *arXiv preprint arXiv:2602.16855*, 2026a.

Weikai Xu, Kun Huang, Yunren Feng, Jiaxing Li, Yuhan Chen, Yuxuan Liu, Zhizheng Jiang, Heng Qu, Pengzhi Gao, Wei Liu, Jian Luan, Xiaolin Hu, and Bo An. How mobile world model guides GUI agents? *arXiv preprint arXiv:2605.10347*, 2026b.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous GUI interaction. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 69772–69805. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/xu25ae.html.

Zhichao Yang, Yuanze Hu, Haojie Hao, Longkun Hao, Dongshuo Huang, Hongyu Lin, Gen Li, Lanqing Hong, Yihang Lou, and Yan Bai. MIRAGE: Mobile agents with implicit reasoning and generative world models. *arXiv preprint arXiv:2606.04627*, 2026.

Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, Jitong Liao, Qi Zheng, Fei Huang, Jingren Zhou, and Ming Yan. Mobile-Agent-v3: Fundamental agents for GUI automation. *arXiv preprint arXiv:2508.15144*, 2025.

Yuhao Zheng, Li'an Zhong, Yi Wang, Rui Dai, Kaikui Liu, Xiangxiang Chu, Linyuan Lv, Philip Torr, and Kevin Qinghong Lin. Code2World: A GUI world model via renderable code generation. *arXiv preprint arXiv:2602.09856*, 2026.

Hanzhang Zhou, Xu Zhang, Panrong Tong, Jianan Zhang, Liangyu Chen, Quyu Kong, Chenglin Cai, Chen Liu, Yue Wang, Jingren Zhou, and Steven Hoi. MAI-UI technical report: Real-world centric foundation GUI agents. *arXiv preprint arXiv:2512.22047*, 2025a.

Xurui Zhou, Gongwei Chen, Yuquan Xie, Zaijing Li, Kaiwen Zhou, Shuai Wang, Shuo Yang, Zhuotao Tian, and Rui Shao. HiconAgent: History context-aware policy optimization for GUI agents. *arXiv preprint arXiv:2512.01763*, 2025b.

A Additional Experimental Details

A.1 Dataset Statistics

Table 5 summarizes the training and offline evaluation data. The four training sources contain 22,035 source episodes and 208,766 action steps in total: 88,317 from AndroidControl, 102,086 from GUI-Odyssey, 8,772 from OpenMobile, and 9,591 from MobileForge. The dataset rows count source episodes and action records after processing; the TASI total additionally accounts for expansion into its four training tasks. Here, "steps" refers to data examples, not optimizer updates.

Table 6 complements the dataset sizes with episode-length and action distributions. GUI-Odyssey contains the longest training trajectories (mean 15.31 steps, P90 24), more than twice the average length of AndroidControl (6.49 steps). This explains why GUI-Odyssey contributes $102{,}086$ training steps from only $6{,}668$ episodes and also makes trajectory-level success particularly demanding. The two AndroidWorld-derived sources have intermediate lengths, with means of 10.98 and 9.94 steps. The offline test splits have closely matched length distributions: AndroidControl-High has mean 6.44 versus 6.49 in training, and GUI-Odyssey has mean 15.49 versus 15.31.

The action mixture is highly imbalanced. Click accounts for 59.59% of the 208,766 source transitions, followed by swipe (10.76%) and terminate (10.57%); long press contributes only 0.38%.

Table 5: **Processed dataset sizes.** Dataset rows count source episodes and action steps. The TASI total counts task-specific episode instances and training examples after expansion into two action-modeling and two successor-selection tasks; terminate actions are excluded from both selection tasks. The URFD total counts source transitions once.

Dataset / source	Usage	Episodes	Steps
AndroidControl	TASI and URFD training	13,603	88,317
GUI-Odyssey	TASI and URFD training	6,668	102,086
OpenMobile (retained subset)	TASI and URFD training	799	8,772
MobileForge (retained subset)	TASI and URFD training	965	9,591
TASI expanded total		**88,140**	**790,994**
URFD total		**22,035**	**208,766**
AndroidControl-High	Offline evaluation	1,680	10,814
GUI-Odyssey	Offline evaluation	1,666	25,807
Offline evaluation total		**3,346**	**36,621**

Table 6: **Episode-length and action distributions.** Left: episode length is the number of annotated action steps in one source episode; P75 and P90 are episode-length percentiles. Right: distribution of raw source action labels over the 208,766 URFD training transitions.

Dataset	Mean	Med.	P75	P90	Max
AndroidControl	6.49	6	8	11	92
GUI-Odyssey	15.31	14	18	24	59
OpenMobile	10.98	10	13	17	30
MobileForge	9.94	9	13	17	44
AndroidControl-High	6.44	6	8	10	25
GUI-Odyssey	15.49	14	19	24	51

Action	Count	Share
Click	124,410	59.59
Swipe	22,455	10.76
Terminate	22,060	10.57
Type	17,449	8.36
System button	11,295	5.41
Wait	5,211	2.50
Open app	5,099	2.44
Long press	787	0.38

TASI constructs four task-specific versions of the source trajectories: CopyState action modeling, FutureState action modeling, task-conditioned successor selection, and action-conditioned successor selection. Its 88,140 episode instances therefore represent four task-specific copies of the 22,035 source episodes, not 88,140 independently collected trajectories. The two action-modeling tasks retain all 208,766 action steps each, including terminate actions. In contrast, both successor-selection tasks exclude terminate actions, since ending a task does not provide a subsequent interaction state to select. Under the counts in Table 5, excluding one terminate step per source episode leaves 186,731 examples for each selection task. The difference of 44,070 examples is exactly the two omitted selection examples for each of the 22,035 terminate steps. URFD uses each of the 208,766 source transitions once as a training example; its paired teacher forwards do not constitute additional dataset steps.

AndroidWorld online evaluation comprises 116 task instances (61 Easy, 36 Medium, and 19 Hard), rather than a fixed collection of annotated steps. Episode attempts and executed step counts depend on the evaluated policy and attempt budget, and are therefore not included as fixed dataset episode/step counts in Table 5.

B Implementation Details

Backbone and input representation. We initialize from Qwen3-VL-8B-Instruct and fine-tune the language model while freezing the vision tower and multimodal projector in both stages. We use full-parameter language-model fine-tuning rather than low-rank adapters. The current screenshot has a maximum pixel budget of 1,536,000, corresponding to no more than 1500 merged visual tokens; the second visual slot uses a budget of 32,768 pixels, corresponding to at most 32 merged visual

Table 7: **Full component ablation results.** Overall is the unweighted average of AndroidControl and GUI-Odyssey. All scores are percentages.

	AndroidControl		GUI-Odyssey		Overall	
Method	Type	Match	Type	Match	Type	Match
URFD	85.24	77.96	93.41	83.38	89.32	80.67
w/o utility routing	84.76	77.48	92.88	82.66	88.82	80.07
w/o successor initialization	84.85	77.69	92.82	82.54	88.84	80.12
w/o hidden alignment	84.95	77.50	92.84	82.81	88.90	80.16
w/o policy KL	84.81	77.52	92.81	82.46	88.81	79.99
w/o copy-state CE	85.06	77.83	92.60	81.99	88.83	79.91

tokens in the main configuration. CopyState fills this slot with a low-resolution copy of the current screen, whereas FutureState uses the logged successor. Inputs also contain the task instruction and text-only action history; past screenshots are not included.

Two-stage optimization. TASI jointly trains CopyState/FutureState action modeling and the two successor-selection tasks for one epoch. We train TASI for one epoch with a global batch size of 128. URFD initializes both the student and the frozen teacher from this checkpoint, then trains the student for three epochs with a global batch size of 128, finishing at step 4,893. Both stages use AdamW with a learning rate of 10^{-5}, weight decay of 0.01, momentum coefficients of 0.9 and 0.999, and numerical epsilon of 10^{-8}. We use a 10% linear warm-up followed by cosine decay to zero, and lip the gradient norm at 1.0.

Distillation configuration. The teacher remains frozen throughout URFD. For each example, it evaluates the CopyState and FutureState views under the same demonstrated action prefix. Utility is the reduction in token-averaged negative log-likelihood over the supervised action response, including its arguments, and is detached before routing. The main routing thresholds are 0 and 0.01: non-positive utility retains CopyState supervision, utility at least 0.01 uses full FutureState supervision, and intermediate values interpolate continuously. We align corresponding second-slot tokens across all 36 decoder layers using MSE without spatial pooling, with normalized reverse-exponential weights emphasizing shallow layers. Policy distillation uses reverse KL from the student to each teacher distribution at supervised response positions, with the same routing coefficient as hidden alignment. The CopyState action cross-entropy, hidden alignment, and routed policy-distillation terms each have coefficient 1.

Offline inference. The deployed student receives only the current screenshot and its low-resolution copy; neither the real successor nor the teacher is required. The GUIEvalKit adapter generates one action response per step, with a maximum of 2,048 new tokens, temperature 0.01, top-k of 1, top-p of 0.01, and repetition penalty 1.0.

B.1 Metrics and comparison protocols

S.type measures action-category accuracy; **S.match** additionally checks action arguments using the evaluator's action-specific matching rules. **T.succ** is the fraction of offline trajectories whose evaluated steps all match, not online emulator success. **T.prog** averages the fraction of each trajectory covered by the consecutive correctly matched prefix before the first error. AndroidWorld cumulative pass@k measures the fraction of tasks with at least one strict full-credit success within k attempts. The evaluation contains 61 Easy, 36 Medium, and 19 Hard tasks. External results with fewer than three attempts are reported only as lower bounds on pass@3. Released GUI agents use model-specific GUIEvalKit adapters on the same converted offline splits.

C Full component ablation results

Table 7 provides the dataset-level results corresponding to the Overall summary in Table 3.

D Detailed action-wise analysis on GUI-Odyssey

Table 8 reports the numerical values underlying Figure 5. CopyState and NextState evaluate the same TASI checkpoint with a copied current screenshot and the real successor, respectively. URFD evaluates the main distillation run under CopyState input. All three use the same 25,807 GUI-Odyssey steps. The scores measure complete action match under the evaluator's action-specific rules, including action arguments, rather than action-type accuracy alone.

Table 8: **Full-action match by action category on GUI-Odyssey.** Scores are percentages, reported to the precision of the saved evaluation summaries. C, F, and U denote CopyState, NextState, and URFD. Differences are percentage points; NextState has privileged future input and is not an input-matched deployment baseline.

Action	Count	CopyState	NextState	URFD	$U - C$	$F - U$
Click	16,658	82.90	89.60	86.35	+3.45	3.25
Long press	89	80.90	88.76	82.02	+1.12	6.74
Scroll	2,622	62.01	86.27	69.68	+7.67	16.59
Press	2,106	77.64	95.20	85.47	+7.83	9.73
Stop	1,666	91.00	90.70	87.52	−3.48	3.18
Type	2,666	71.72	83.46	74.16	+2.44	9.30
Overall	**25,807**	**79.71**	**89.15**	**83.38**	**+3.67**	**5.77**

Large per-category gains for Scroll and Press. URFD improves five of the six categories over the CopyState initialization, with the largest gains on Press ($+7.83$ points) and Scroll ($+7.67$). These categories also have large teacher-view gaps: observing the real successor improves the same teacher by 17.56 points on Press and 24.26 points on Scroll. A plausible explanation is that the successor reveals the destination of a system-button press or the direction and extent of a scroll, helping disambiguate the demonstrated operation. Nevertheless, the student remains 9.73 and 16.59 points below NextState on these categories. The results therefore indicate partial improvement, not complete recovery of the future-visible behavior.

Click dominates the aggregate improvement. Per-category gains and contributions to the overall score are different. Click accounts for 64.55% of the evaluated steps, so its 3.45-point gain contributes approximately 2.23 points to the overall improvement of 3.67 points. Scroll and Press contribute approximately 0.78 and 0.64 points, respectively.

E Task- and Action-Conditioned Future Selection

We evaluate four-way successor selection on 1,000 sampled test cases from each of AndroidControl and GUI-Odyssey. Task-conditioned selection asks which candidate follows the next correct action, without supplying that action; action-conditioned selection supplies the demonstrated action and asks for its resulting screen. Accuracy is computed by restricting the first response-token logits to A/B/C/D and checking whether the highest-scoring option is the logged successor.

Figure 6 shows that TASI reaches 93.8% and 96.0% task-conditioned accuracy on AndroidControl and GUI-Odyssey, respectively, improving over DualView by 31.9 and 41.8 percentage points. Under action conditioning, TASI reaches 93.4% and 95.8%, with gains of 31.1 and 32.2 points over DualView. Dual-view action modeling alone does not improve selection accuracy over the pretrained baseline in these evaluations, whereas adding the selection tasks substantially improves recognition of the appropriate successor. This supports the role of TASI in establishing successor awareness. However, choosing among visible candidate screens is distinct from inferring a latent future from current-only inputs; these results do not by themselves establish current-only predictability or downstream action improvements.

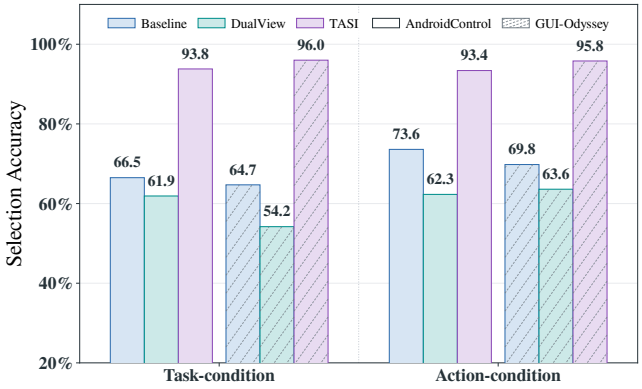


Figure 6: **Task- and action-conditioned future-selection accuracy.** Baseline denotes Qwen3-VL-8B-Instruct; DualView denotes initialization with CopyState and FutureState action modeling; Solid and diagonally patterned bars indicate AndroidControl and GUI-Odyssey, respectively. Each condition uses 1,000 cases per dataset.

F Layer-wise Alignment Dynamics

We examine the 36-layer alignment losses of URFD run with utility-routing interval $[0, 0.01]$, trained for 3 epochs. Each panel groups four consecutive layers; Layer 1 corresponds to `layer_00` in the log. Values are shown as logged, without additional layer weighting or normalization. All numerical comparisons below use unsmoothed values.

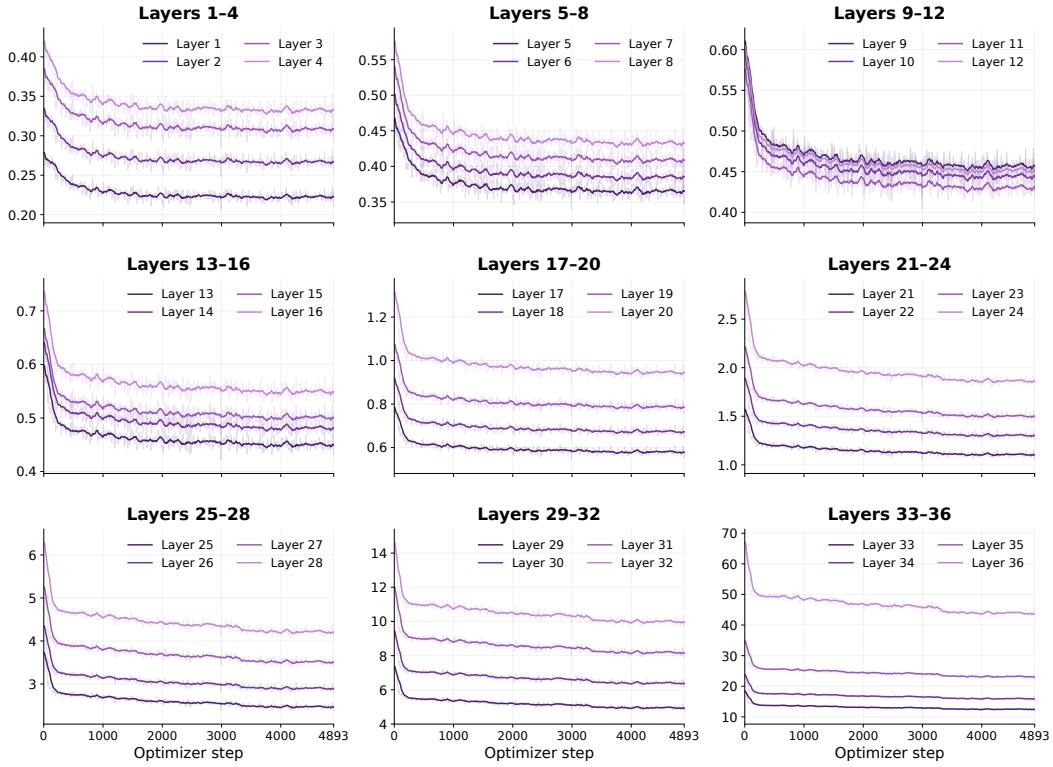


Figure 7: **Training alignment loss across all 36 layers.** Each panel shows four layers with independent, data-dependent linear y-axis limits.

Rapid initial fitting followed by gradual refinement. Figure 7 shows an early decline followed by slower improvement with minibatch fluctuations. Every layer ends below its first logged value,

with reductions of 17.9–33.6%. The decrease across the decoder indicates that training adjusts representations at multiple depths rather than reducing the objective through a single layer alone. The later flattening suggests diminishing returns in this alignment objective, not necessarily saturation of downstream action performance.

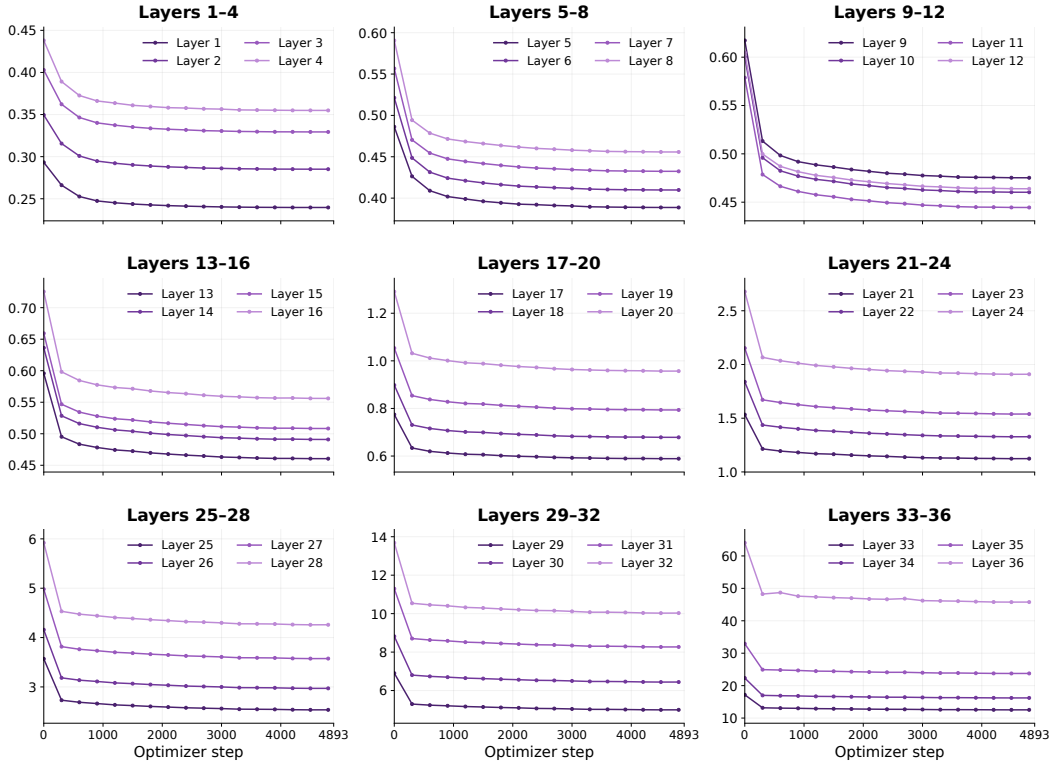


Figure 8: **Validation alignment loss during training.** Lines with markers show raw losses from 17 evaluations at 300-step intervals, including step 0, without smoothing. Panels use independent linear y-axis limits. The last evaluation is at step 4,800, before the final training step 4,893.

Validation follows the training trend. All 36 validation losses also decrease between the first and last evaluations, by 18.2–28.9% (Figure 8). By step 300, the median layer has achieved 76.5% of its eventual reduction at step 4,800. Thus, the training-loss reduction is accompanied by improvement on validation examples, without a broad late-stage reversal in alignment.

Depth-dependent discrepancies and shallow-layer supervision. At the last validation evaluation, the alignment loss is 0.240 at Layer 1 but 45.77 at Layer 36. One plausible explanation is that small representation discrepancies introduced in shallow layers propagate through successive attention and nonlinear transformations and can be amplified at greater depth. This interpretation motivates our reverse-exponential weighting, which assigns stronger alignment weights to shallow layers to constrain discrepancies early, before they propagate to deeper representations and the action decoder.

G Action-generation Prompt and Output Example

The system message specifies the `mobile_use` tool and constrains the response to a single tool call. The user message supplies the task, text action history, and two image slots. The blocks below retain the prompt wording; long lines wrap only for presentation. The placeholder `<image>` denotes an image input, not additional text describing that image.

System message.

```
You are a helpful assistant.
```

```
## Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML tags:
<tools>
{"type": "function", "function": {"name": "mobile_use", "description": "Use a touchscreen to
     interact with a mobile device, and take screenshots.
* This is an interface to a mobile device with touchscreen. You can perform actions like
     clicking, typing, swiping, etc.
* Some applications may take time to start or process actions, so you may need to wait and
     take successive screenshots to see the results of your actions.
* The screen's resolution is 999x999.
* Make sure to click any buttons, links, icons, etc with the cursor tip in the center of the
     element. Don't click boxes on their edges unless asked.", "parameters": {"properties": {"
     action": {"description": "The action to perform. The available actions are:
* `click`: Click the point on the screen with coordinate (x, y).
* `long_press`: Press the point on the screen with coordinate (x, y) for specified seconds.
* `swipe`: Swipe from the starting point with coordinate (x, y) to the end point with
     coordinates2 (x2, y2).
* `type`: Input the specified text into the activated input box.
* `answer`: Output the answer.
* `system_button`: Press the system button.
* `wait`: Wait specified seconds for the change to happen.
* `terminate`: Terminate the current task and report its completion status.", "enum": ["click
     ", "long_press", "swipe", "type", "answer", "system_button", "wait", "terminate"], "type
     ": "string"}, "coordinate": {"description": "(x, y): The x (pixels from the left edge)
     and y (pixels from the top edge) coordinates to move the mouse to. Required only by `
     action=click`, `action=long_press`, and `action=swipe`.", "type": "array"}, "coordinate2
     ": {"description": "(x, y): The x (pixels from the left edge) and y (pixels from the top
     edge) coordinates to move the mouse to. Required only by `action=swipe`.", "type": "array
     "}, "text": {"description": "Required only by `action=type` and `action=answer`.", "type
     ": "string"}, "time": {"description": "The seconds to wait. Required only by `action=
     long_press` and `action=wait`.", "type": "number"}, "button": {"description": "Back means
      returning to the previous interface, Home means returning to the desktop, Menu means
     opening the application background menu, and Enter means pressing the enter. Required
     only by `action=system_button`", "enum": ["Back", "Home", "Menu", "Enter"], "type": "
     string"}, "status": {"description": "The status of the task. Required only by `action=
     terminate`.", "type": "string", "enum": ["success", "failure"]}}, "required": ["action"],
      "type": "object"}}}
</tools>

For each function call, return a json object with function name and arguments within <
     tool_call></tool_call> XML tags:
<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>

# Response format

Response format for every step:
1) A single <tool_call>...</tool_call> block containing only the JSON: {"name": <function-name
     >, "arguments": <args-json-object>}.

Rules:
- Output exactly in the order: <tool_call>.
- Do not output anything else outside <tool_call> parts.
- If finishing, use action=terminate in the tool call.
```

User message template.

```
The user query:  {user_request}
Task progress (You have done the following operation on the current device): {history_actions
     }.
Current Screen:
<image> (no more than 1500 merged visual tokens)
Target Screen:
<image> (no more than 32 merged visual tokens)
```

{user_request} is replaced by the task instruction and {history_actions} by the formatted prior action history. The first image is the current screenshot. In CopyState, the Target Screen slot contains a low-resolution copy of that same screenshot; in FutureState, it contains the logged successor. The real successor is never supplied to the deployed CopyState student. This block

documents the action-generation interface, not the separate four-choice successor-selection prompts used in TASI.

Example action output. The following is an illustrative valid click response, not a newly generated evaluation prediction. Coordinates follow the prompt's 999-by-999 coordinate convention; the selected coordinate depends on the actual screenshot.

```
<tool_call>
{"name": "mobile_use", "arguments": {"action": "click", "coordinate": [500, 300]}}
</tool_call>
```

The output contains the complete action type and arguments without an explanation or explicit chain of thought outside the tool-call block.