LATENTCUA: A DECOUPLED WORLD MODEL FOR GUI PREDICTION AND PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Computer-use agents typically choose actions from the current screen without explicitly reasoning about their future effects. We introduce LatentCUA, a compact world model that enables an agent to compare candidate actions through latent prediction. LatentCUA learns a semantic representation of GUI screens, predicts how this representation evolves under an action, and scores imagined future states against the agent's goals. Its modules are trained in separate stages, allowing representation learning, dynamics prediction, and state evaluation to use supervision suited to each role. During planning, a vision-language agent proposes candidate action sequences and LatentCUA ranks their predicted outcomes before execution. Offline ablations validate these design choices, with the final configuration providing the strongest overall balance of representation quality, multi-step prediction, and candidate discrimination. On OSWorld, direct world-model selection improves lower-performing planners, while selective assistance improves all five tested planners. We further provide a decision-sufficiency framework that relates representation, prediction, and scoring errors to action-selection quality.

1 INTRODUCTION

Computer-use agents operate graphical interfaces through screenshots and mouse or keyboard actions (Xie et al., 2024; Lin et al., 2024; Qin et al., 2025; Wang et al., 2025). Most agents decide from the current screen and execute the selected action immediately. This reactive loop is effective for short interactions, but it does not let the agent compare how alternative actions may change the interface before committing to one. A world model provides this capability by predicting future states under candidate actions and supporting planning over their outcomes (Ha & Schmidhuber, 2018; Hafner et al., 2018; 2025).

GUI world modeling poses a distinct representation problem. Screens are high-resolution, text-rich, and dominated by content that remains unchanged, while a small semantic change can determine the next decision. GUI actions are also heterogeneous, combining discrete commands, coordinates, key presses, and typed text. Existing approaches simulate future computer states as text, structured elements, screenshots, or executable interface code (Gu et al., 2025; Chae et al., 2025; Mei et al., 2026; Cao et al., 2026; Luo et al., 2026; Rivard et al., 2026; Koh et al., 2026; Guan et al., 2026). These representations make future states explicit, but repeatedly producing them for many candidate branches is costly. A compact latent model can instead preserve the distinctions needed for decision making while predicting directly in a fixed-size state space.

We introduce **LatentCUA**, a decoupled latent world model for GUI prediction and planning. It contains three modules. A *Screen Encoder* compresses each screenshot into semantic latent slots. A *Predictor* rolls these slots forward under candidate GUI actions. A goal-conditioned *State Scorer* evaluates real or predicted future states against the task and the agent's short-horizon goals. The representation is learned from screenshots at their original resolution, while structured screen information is used only as training supervision. This gives the deployed Screen Encoder a standardized image-only interface and keeps rollout cost independent of screen resolution.

The three modules are trained sequentially. The Screen Encoder first learns which screen content the latent should preserve. It is then frozen while the Predictor learns action-conditioned dynamics, after which the State Scorer is trained on mixed real and predicted states. This separation prevents

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 prediction objective from replacing semantic state distinctions with a trivially predictable representation. It also provides dense, module-specific training signals and permits each component to be evaluated independently.

LatentCUA is integrated with a vision-language planner through receding-horizon control. At each step, the planner proposes a depth-three tree of feasible actions and their expected short-horizon effects. The Predictor imagines each branch, and the State Scorer evaluates the resulting states. The agent then uses these scores according to either a direct selection policy or a selective assistance policy. The planner remains responsible for proposing and grounding actions, while the world model compares their predicted consequences.

We evaluate the representation, dynamics, and scoring modules on held-out AgentNet and ScaleCUA trajectories (Wang et al., 2025; Liu et al., 2026), and evaluate the complete planning system with five planner backends on OSWorld (Xie et al., 2024). The Predictor remains action-sensitive over multi-step rollouts, and the State Scorer distinguishes closely related candidate outcomes. Online, direct world-model selection improves the three lower-performing planners, while selective assistance improves all five. We also develop a decision-sufficiency framework that decomposes action-value error into representation, dynamics, and scoring terms, and characterizes when the world model should directly select an action or selectively revise the planner's ordering.

Our contributions are:

- **A compact GUI world model** that combines semantic screen encoding, action-conditioned latent prediction, and goal-conditioned state scoring in a decoupled training pipeline.
- **A planning interface** that composes the frozen modules with a vision-language agent to compare depth-three candidate action branches before execution.
- **An empirical and theoretical account of latent planning** that connects module-level prediction quality, decision sufficiency, and the authority assigned to the world model across different planner regimes.

2 Related Work

World models for GUI planning. Model-based reinforcement learning learns compact dynamics for planning (Hafner et al., 2018; 2025; Hansen et al., 2024), while DINO-WM predicts in the feature space of a pretrained visual encoder (Zhou et al., 2025). For GUIs, Chae et al. (2025) learns text-based environment dynamics for web navigation, with related approaches using prompted or retrieval-augmented textual simulation (Gu et al., 2025; Mei et al., 2026). Other work predicts structured element sketches (Cao et al., 2026), generated screenshots (Luo et al., 2026; Rivard et al., 2026; Guan et al., 2026), or renderable interface code (Koh et al., 2026). Screen2Vec learns screen embeddings from interaction traces but does not model action-conditioned transitions (Li et al., 2021). LatentCUA combines compact latent rollouts with goal-conditioned outcome scoring for high-resolution GUI actions, and learns its representation, dynamics, and scorer sequentially from offline interaction logs.

Non-reconstructive representation learning. Joint-embedding methods learn visual representations without reconstructing pixels (Grill et al., 2020; Bardes et al., 2021; Assran et al., 2023). V-JEPA extends feature prediction to video (Bardes et al., 2024), and V-JEPA 2 combines large-scale video pretraining with an action-conditioned latent model for physical planning (Assran et al., 2025). LeWorldModel trains an action-conditioned JEPA end to end from pixels using a prediction loss and a distributional latent regularizer (Maes et al., 2026). LatentCUA instead fixes a semantically supervised GUI representation before learning its action-conditioned dynamics.

3 The LatentCUA Architecture

LatentCUA contains three modules. A *Screen Encoder* maps each screenshot to a compact GUI state, a *Predictor* rolls this state forward under candidate actions, and a goal-conditioned *State Scorer* evaluates the resulting futures. The modules are trained sequentially and then composed for planning, as summarized in Figures 1 and 2.

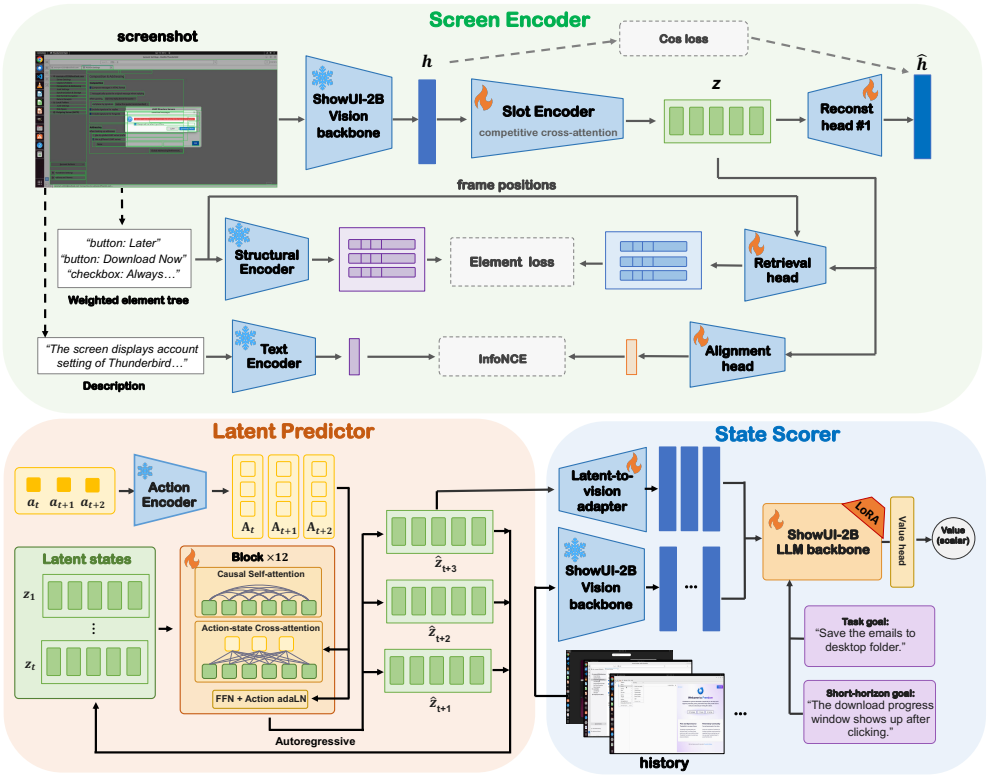

Figure 1: **LatentCUA overview.** The Screen Encoder forms semantic latent slots, the action-conditioned Predictor rolls them forward, and the State Scorer evaluates real or imagined states against the task and short-horizon goals.

3.1 Model and Training

Staged optimization. We first train the Screen Encoder and freeze it, then fit the Predictor on its cached latents, and finally train the State Scorer with both upstream modules fixed. This gives each stage a dense, module-specific learning signal and allows later stages to use larger effective batches without repeated high-resolution backbone passes. It also prevents prediction and scoring objectives from redefining the semantic state learned by the Encoder. Section 4 gives the formal motivation, and Appendix A.2 provides the complete loss definitions.

GUI-specific representation. GUI decisions depend on sparse semantic changes, such as a control state, label, or newly opened panel, rather than on photometric detail. We therefore compress the screen with semantically supervised queries instead of preserving a JEPA-style spatial patch grid (Assran et al., 2023; Bardes et al., 2024); the competitive query design is inspired by slot attention (Locatello et al., 2020). The visual backbone still receives the screenshot at its native resolution so that small controls and text remain visible. The deployed Encoder takes only the screenshot; accessibility elements and screen narrations are privileged supervision rather than inputs. Multiple latent slots avoid forcing the screen into one averaged global token, while a fixed slot budget keeps prediction compact. Our ablations in Section 5 examine this balance.

Screen Encoder. Given a screenshot o_t, the frozen ShowUI-2B visual backbone (Lin et al., 2024) produces GUI-aware tokens h_t at its native dynamic resolution. Two rounds of competitive cross-attention compress them into $z_t \in \mathbb{R}^{32\times256}$. Competition distributes content across the slots without assigning them to predetermined spatial cells, and the fixed latent size makes downstream cost independent of screenshot resolution.

Three objectives determine what the slots preserve. Feature reconstruction encourages broad visual coverage, narration alignment supplies screen-level semantics, and element matching recovers the visible GUI elements as an unordered token set. The Encoder objective is

$$\mathcal{L}_E = .5\mathcal{L}_{\text{rec}} + .5\mathcal{L}_{\text{aln}} + \mathcal{L}_{\text{elem}}.$$

Element targets jointly describe type, state, text, location, and numeric value, but neither accessibility records nor narrations enter the Encoder. These frame-specific readouts prevent a constant representation without an additional variance or covariance regularizer. After training, the auxiliary heads are discarded and the frozen slots are passed directly downstream.

Predictor. The Predictor uses the latest 13 latent states and a structured GUI action to predict up to three future states autoregressively. Its action tokenizer produces three tokens for the action verb, target element, and argument-with-location, together with a global conditioning vector. A block-causal Transformer lets every slot attend to its current and preceding frames, then conditions each block on the action through cross-attention and adaptive layer normalization (Peebles & Xie, 2022). The history supplies interaction context while preserving all 32 semantic slots at every step.

The Predictor outputs the complete next state rather than only a delta. The absolute target has the same size and preserves the stable layout and screen context shared across transitions, giving the model a stronger prediction signal than sparse changes alone. For rollout steps beyond the first, precise coordinates and element bindings are unavailable from the planner, so future action inputs emphasize the verb, target semantics, and textual arguments; coordinate grounding remains the planner's responsibility. Training combines change-normalized regression with same-screen sibling transitions:

$$\mathcal{L}_P = \mathcal{L}_{\text{norm-mse}} + \mathcal{L}_{\text{sib}}.$$

The normalization prevents large page changes from dominating small but decision-critical changes. Sibling transitions pair the same screen with different real actions and successors, directly enforcing action-sensitive predictions.

State Scorer. The State Scorer is a goal-conditioned ShowUI-2B judge. It receives the task instruction, short-horizon goals, the most recent eight real states, and one candidate rollout containing up to three future states. It receives no candidate action, so an action affects its score only through the state imagined by the Predictor. Real screenshots use the frozen ShowUI vision tower; real or predicted latents pass through a trainable adapter that maps the raw 32×256 slots directly into the same visual-token space. The adapter is therefore a token-space decoder, not a pixel decoder.

The ShowUI vision tower and base language model remain frozen, while LoRA, the latent adapter, and a scalar value head are trained to rank candidate futures. The objective combines next-state identification, short-goal matching, task-goal matching, trajectory progress, and transition quality:

$$\mathcal{L}_V = \mathcal{L}_{\text{frm}} + \mathcal{L}_{\text{short}} + \mathcal{L}_{\text{task}} + \mathcal{L}_{\text{prog}} + \mathcal{L}_{\text{qual}}.$$

The matching terms bind states to the correct local and long-horizon goals; the ordering terms reward progress and rank wrong or redundant transitions below useful ones. Training pools mix encoded real states with frozen-Predictor rollouts and include near-copy, same-trajectory, and semantically related alternatives. These deployment-like comparisons train the Scorer to attend to the state differences that separate plausible candidate actions.

3.2 Planning with the World Model

Agent interface. Our online agent retains the planning architecture of Agent S3 but standardizes its outputs to match the offline transition data. Each tree node contains one primitive GUI action and a short-horizon goal describing the observable effect expected after execution. We disable compound actions, WAIT, terminal operations, coding-agent mode, and unnecessary hotkeys. DONE and FAIL are controller signals that terminate the episode rather than actions passed to the world model.

WM-guided selection. At each step, the VLM planner searches feasible actions and proposes a depth-three tree. The Screen Encoder maps the current screenshot to z_t, the Predictor rolls each root-to-leaf branch to $\hat{z}_{t+1}, \ldots, \hat{z}_{t+3}$, and the State Scorer evaluates the imagined states against the task and corresponding short-horizon goals. The integration policy then selects among the proposed

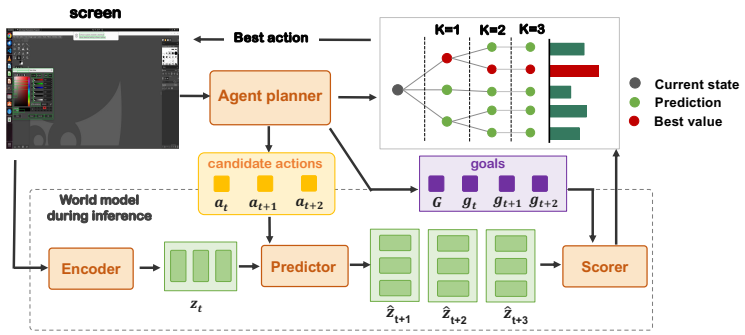

Figure 2: **Planning loop.** A VLM planner proposes a depth-three action tree. The frozen Predictor rolls out each branch, and the State Scorer evaluates its imagined states. The integration policy uses these scores to select the next root action.

root actions, the agent observes the next real screen, and planning repeats. The frozen world model compares the consequences of planner-proposed actions; action proposal and precise grounding remain with the VLM planner.

4 Theory: Decision-Sufficient Latent Planning

We characterize what a latent must preserve, how module errors affect action selection, and when the WM should replace or revise the planner's ordering. Proofs appear in Appendix B.

Decision model. Let s_t, a_t, and o_t denote the computer state, action, and screenshot, with transition kernel $P(s_{t+1} \mid s_t, a_t)$ and observation kernel $O(o_t \mid s_t)$. The agent acts from its observable history $h_t = (o_0, a_0, \ldots, a_{t-1}, o_t)$. For goal set G_g and budget T, let $J_T(\pi, g)$ be the probability that policy π reaches G_g by T, and let $Q_t^\pi(h, a)$ be this probability after taking a. At a decision, the planner proposes $(a_1, \ldots, a_n)$, with baseline a_1. Under continuation π_0, define

$$q_i := Q_t^{\pi_0}(h, a_i), \qquad q_* := \max_i q_i, \qquad I^* := \arg\max_i q_i. \tag{1}$$

LatentCUA scores branch $\mathbf{a} = (a_t, \ldots, a_{t+k-1})$ using bounded short-horizon utility U_g:

$$Q_g^k(h, \mathbf{a}) := \mathbb{E}[U_g(h_{t+k}) \mid h_t = h, \mathrm{do}(\mathbf{a})]. \tag{2}$$

Claim 1: decision sufficiency preserves candidate values.

Definition 1 (Decision-sufficient representation)**.** For a family of goals, horizons $k \le H$, and action-sequence queries, a representation Φ is decision-sufficient if

$$\Phi(h) = \Phi(h') \Longrightarrow Q_g^k(h, \mathbf{a}) = Q_g^k(h', \mathbf{a}) \quad \text{for every query } (g, k, \mathbf{a}). \tag{3}$$

The coarsest sufficient representation groups histories with identical candidate values, as formalized in Proposition B.1. Thus minimality concerns decision-relevant information rather than pixel fidelity or a fixed number of tokens. This connects LatentCUA to state abstraction and latent-MDP theory (Abel et al., 2016; Gelada et al., 2019).

Claim 2: representation, dynamics, and scoring errors control decisions. Let P_Φ^k and $\widehat{P}_\Phi^k$ be the true and predicted distributions of the encoded future trajectory. Let $V_{g,k}^\star$ be an ideal latent scorer that is L_V-Lipschitz under trajectory metric d_k, and let $\widehat{V}_{g,k}$ be the learned scorer. Assume uniformly over deployment queries that

$$\left| Q_g^k - \mathbb{E}_{P_\Phi^k} V_{g,k}^\star \right| \le \epsilon_{\mathrm{rep}}, \tag{4}$$

$$W_1(P_\Phi^k, \widehat{P}_\Phi^k) \le \epsilon_{\mathrm{dyn}}, \tag{5}$$

$$\mathbb{E}_{\widehat{P}_\Phi^k} |\widehat{V}_{g,k} - V_{g,k}^\star| \le \epsilon_{\mathrm{val}}. \tag{6}$$

These terms measure representation insufficiency, rollout-distribution error, and scoring error on predicted trajectories, respectively. Define $\widehat{Q}_g^k := \mathbb{E}_{\widehat{P}_\Phi^k} \widehat{V}_{g,k}$.

Theorem 4.1 (Decision error decomposition)**.** *Under* (4)–(6)*, let* $\mathbf{a}^\star$ *and* $\widehat{\mathbf{a}}$ *maximize* Q *and* $\widehat{Q}$ *over the same candidates. Then*

$$|\widehat{Q}_g^k - Q_g^k| \le B_k, \qquad B_k := \epsilon_{\text{rep}} + L_V \epsilon_{\text{dyn}} + \epsilon_{\text{val}}, \tag{7}$$

$$Q_g^k(h, \mathbf{a}^\star) - Q_g^k(h, \widehat{\mathbf{a}}) \le 2B_k. \tag{8}$$

For a unique optimum with margin $\Delta_g^k(h) := Q_g^k(h, \mathbf{a}^\star) - \max_{\mathbf{a} \neq \mathbf{a}^\star} Q_g^k(h, \mathbf{a})$*,* $2B_k < \Delta_g^k(h)$ *guarantees exact selection.*

The factor two appears because error can lower the best candidate by B_k and raise a competitor by another B_k.

To connect short branches to task completion, assume each branch rooted at a_i satisfies

$$|Q_g^k(h, \mathbf{a}_i) - q_i| \le \epsilon_{\text{cont}}, \tag{9}$$

$$|\widehat{Q}_i - q_i| \le C_k, \qquad C_k := B_k + \epsilon_{\text{cont}}. \tag{10}$$

The continuation term isolates the alignment between short-horizon progress and eventual success from the three learned-module errors.

Claim 3: WM authority depends on relative decision quality. Let $J = \arg\max_i \widehat{Q}_i$. Direct selection executes a_J; assisted selection retains a_1 unless its gate accepts the override. Define expected candidate regret on a shared distribution of decision contexts. Direct selection improves expected task value exactly when

$$\underbrace{\mathbb{E}[q_* - q_J]}_{R_{\text{WM}}} < \underbrace{\mathbb{E}[q_* - q_1]}_{R_{\text{planner}}}. \tag{11}$$

The task-value bound gives two operational guarantees:

$$2\mathbb{E}[C_k] < R_{\text{planner}} \Longrightarrow R_{\text{WM}} < R_{\text{planner}}, \qquad \widehat{Q}_J - \widehat{Q}_1 > 2C_k \Longrightarrow q_J > q_1. \tag{12}$$

The first supports direct selection when WM error is below planner regret; the second certifies an assisted override. Thus strong planners favor selective assistance, weaker ordering permits more WM authority, and both remain bounded by candidate quality q_*. Proofs appear in Appendices B.4 and B.7.

Implications for LatentCUA. The Screen Encoder preserves candidate-value distinctions, the Predictor keeps action outcomes distinguishable, and the State Scorer learns on predicted trajectories. Sequential training fixes decision-relevant semantics before dynamics learning; complexity and multi-step results appear in Appendix B.5.

5 Offline Evaluation

We evaluate the three modules separately before placing them in a decision loop. Our experiments ask whether the latent retains decision-relevant screen content, whether the predictor models action-conditioned state changes, and whether the value model can rank predicted outcomes under a goal. We report both state-prediction metrics and candidate-ranking metrics because low prediction error alone does not imply useful counterfactual separation.

5.1 Experimental Setup

Dataset. We pretrain on the Windows and macOS portions of ScaleCUA and AgentNet (Liu et al., 2026; Wang et al., 2025). AgentNet provides approximately 18K trajectories and 320K transitions. ScaleCUA provides 907 trajectories and 7,027 screenshots: 525/4,918 from Windows and 382/2,109 from macOS. We then fine-tune on Ubuntu from both datasets: AgentNet has 5,000 trajectories and 82,448 steps, while ScaleCUA has 1,201 exploration trajectories and 16,514 frames,

Table 1: Encoder ablations. Rec. is per-token feature cosine, and Narration is 32-way retrieval accuracy. Element matching reports type, 12-attribute state exact accuracy, text accuracy, and normalized numeric-value MAE. Adj. cos. averages corresponding-token cosine across consecutive frames, where lower is more temporally discriminative. Effective rank is normalized by token width. Dashes denote unavailable native heads; no replacement probes are fitted.

		Rec.	Narration	Element matching				Latent	
Group	Variant	cos. ↑	@32 ↑	Type ↑	State exact ↑	Text ↑	Value MAE ↓	Adj. cos. ↓	Eff. rank (%)
Default	**LatentCUA** (32×256)	.588	.611	.959	.906	.880	.068	.721	28.3
Loss	w/o $\mathcal{L}_{\text{elem}}$	.451	.510	–	–	–	–	.780	19.9
	w/o $\mathcal{L}_{\text{aln}}$	.475	–	.932	.908	.884	.069	.765	27.3
	w/o $\mathcal{L}_{\text{rec}}$	–	.547	.855	.903	.872	.061	.938	4.7
	+ VICReg	.575	.612	.965	.902	.871	.069	.723	41.4
Latent	single global token (1×8192)	.280	.564	.830	.832	.708	.090	.872	–
	64×256 latent	.536	.602	.944	.899	.845	.064	.725	23.8
	32×512 latent	.608	.609	.961	.907	.886	.059	.743	16.4
	spatial grid structure	.709	–	–	–	–	–	.976	–
Input	half-resolution input	.611	.506	.945	.900	.209	.162	.987	23.4
Training	joint Enc./Pred. training	.224	.323	.408	.389	.322	.275	.964	9.4

including 15.3K consecutive-frame and 6.3K same-screen sibling pairs. We exclude its 2,958 human demonstrations. Trajectory-level held-out sets contain 600 ScaleCUA pairs and 1,500 AgentNet frames for the Encoder, 400 windows per dataset for the Predictor, and 300 AgentNet anchors for the State Scorer.

5.2 What Does the Latent Preserve?

Table 1 evaluates the Encoder with metrics tied to its three training objectives, temporal separation, and latent collapse. All variants share the visual backbone, held-out split, and evaluation protocol; we report only their native training heads, without replacement probes.

The three objectives provide complementary signals. Without reconstruction, adjacent-frame cosine rises from .721 to .938 and effective rank falls from 28.3% to 4.7%. Half-resolution input retains reconstruction quality but loses text recovery (.880 to .209) and temporal separation (.721 to .987), supporting native-resolution perception before compression. VICReg raises effective rank to 41.4% without consistently improving semantic readout, so rank measures latent spread rather than the amount of GUI knowledge.

Latent structure matters independently of scalar capacity. The 1×8192 baseline matches the default scalar budget but pools local elements and changes into one global update; semantic slots instead let distinct changes evolve separately. Increasing the representation to 64×256 or 32×512 does not improve the overall balance. The spatial baseline retains a grid of 24-dimensional per-cell latents with 3×3 context and per-cell decoding. It reconstructs well but performs little semantic compression, leaving consecutive states highly similar. Joint Encoder and Predictor training also reduces the effective batch and mixes objectives that favor semantic detail and predictability, respectively. We therefore select 32×256 for its balance of semantic readout, temporal separation, compactness, and downstream utility.

5.3 Action-Conditioned Prediction and Discrimination

We evaluate multi-step fidelity, dependence on the supplied action, and counterfactual separation among actions executed from the same screen.

LatentCUA improves over copying through three steps and remains action-sensitive: it attains .735 same-screen oracle accuracy, 3.5 times chance, while preserving .831 of real-outcome separation. Unweighted MSE instead lets large screen changes dominate small decision-relevant ones, and removing the sibling loss sharply reduces counterfactual separation. Normalizing by copy error and training on sibling outcomes address these two failures.

The remaining ablations explain the architecture. Distinct action tokens, adaLN, and cross-attention are all needed to propagate action identity through the dynamics. Coordinate dropout shifts reliance

Table 2: Predictor ablations. Fidelity reports relative MSE $r_k = \mathrm{MSE}(\hat{z}_{t+k}, z_{t+k})/\mathrm{MSE}(z_t, z_{t+k})$ and direction cosine; copy has $r_k = 1$ and cosine .790/.748/.704 for $k = 1/2/3$. Benefit is $(e_{\mathrm{null}} - e_{\mathrm{act}})/e_{\mathrm{copy}}$; Adv is $e_{\mathrm{wrong}} - e_{\mathrm{act}}$; Sens is cosine between forecasts under two actions. Same-screen sets contain a mean of 5.6 executed actions (chance .210). Oracle matches each forecast to its real outcome, and Sep is predicted pairwise distance divided by real pairwise distance.

		Fidelity (MSE ↓ / Dir ↑)			Action dependence			Same-screen discrimination	
Group	Variant	k=1	k=2	k=3	Benefit ↑	Adv ↑	Sens ↓	Oracle ↑	Sep ↑
Default	**LatentCUA**	.478/.856	.522/.827	.565/.803	.295	.204	.574	.735	.831
Loss	unweighted MSE	.410/.860	.505/.830	.560/.805	.236	.199	.703	.412	.495
	w/o $\mathcal{L}_{\mathrm{sib}}$	.483/.847	.492/.826	.538/.791	.208	.171	.825	.384	.460
Action	w/o adaLN	.679/.813	.693/.805	.725/.796	.223	.105	.987	.289	.482
	w/o action cross-attention	.577/.820	.615/.813	.641/.794	.226	.130	.981	.368	.423
	single action token	.892/.810	.903/.799	.911/.773	.014	.051	.994	.257	.356
Design	w/o coord. dropout	.458/.856	.589/.831	.622/.810	.260	.192	.708	.589	.540
	Δ prediction	.460/.865	.542/.823	.586/.806	.234	.153	.693	.402	.471
	history: 1 frame	.821/.853	.880/.818	.889/.792	.101	.165	.825	.320	.519

Table 3: State Scorer ablations. M1 ranks the executed rollout in 8/16-way pools; M2 retrieves the real next state from nearby-task and hard same-trajectory pools. M3/M4 retrieve the matching short/task goal. M5 orders states by trajectory progress, and M6 prefers the state before a labelled wrong step. M1–M6 use 300 AgentNet anchors. Sibling selects among same-screen ScaleCUA actions using real/predicted successors, zero-shot.

Group	Variant	Action sel. M1 @8/@16 ↑	State identity M2 real/hard ↑	Goal grounding M3/M4 @16 ↑	Progress M5/M6 ↑	Same-screen real/pred. ↑
Default	**LatentCUA**	.697/.580	.867/.632	.833/.703	.967/.958	.847/.771
Structure	w/o LoRA	.351/.267	.653/.542	.565/.535	.796/.855	.552/.484
	history: 1 frame	.185/.095	.220/.159	.350/.201	.340/.426	.596/.548
Data	w/o near-copy negatives	.250/.175	.501/.398	.683/.580	.940/.893	.532/.601
	w/o mixed real / predicted pools	.192/.108	.472/.240	.660/.491	.930/.928	.637/.529

toward semantics because future actions cannot be grounded to unobserved screens. Predicting the next state retains the stable layout and screen content as an anchor; a delta has the same output size but isolates a sparse, noisy change. Finally, reducing history from 13 frames to one raises one-step relative error from .478 to .821, confirming that interaction history supplies essential context.

5.4 Goal-Conditioned State Scoring

We test whether the State Scorer ranks fine-grained outcomes, goals, and trajectory progress without observing candidate actions directly.

Training must expose the Scorer to deployment-like distinctions. Removing near-copy negatives or mixed real/predicted pools lowers M1@16 from .580 to .175 and .108, although coarse wrong-step accuracy remains high. These hard comparisons prevent task or trajectory shortcuts and teach the state details needed to rank similar actions. LoRA consistently improves all axes, while reducing history from eight frames to one collapses M1@16 to .095: the current screen alone does not reveal whether an action advances, repeats, or reverses progress. With adaptation, full history, and mixed hard pools, the Scorer reaches .847/.771 on zero-shot real/predicted sibling outcomes.

6 Online Evaluation: When Does World-Model Guidance Help?

We use OSWorld (Xie et al., 2024) to study when a world model improves an agent's decisions. We compare direct and assisted guidance across five planners, then isolate representation and prediction gaps from real future states.

6.1 Setup and Integration Policies

All conditions run the same 361 tasks with the Agent S3-based architecture (Section 3.2), changing only the planner backend and integration policy. Each planner returns a depth-three tree of primitive

Table 4: OSWorld score (%) over 361 tasks and three seeds. Subscripts give seed std.; Assist./override reports assisted score/mean override rate; arrows give change from Agent.

Planner	Agent	Direct	Assist./override
Gemini 3 Flash	$55.1_{\pm 1.1}$	$49.1_{\pm 0.2}$ ↓ 6.0	$63.2_{\pm 0.7}$ ↑ 8.1/25.5
Kimi-K2.5	$33.4_{\pm 1.2}$	$38.1_{\pm 0.7}$ ↑ 4.7	$38.7_{\pm 0.6}$ ↑ 5.3/45.6
Qwen3-VL-30B	$20.8_{\pm 0.9}$	$25.3_{\pm 2.3}$ ↑ 4.5	$24.2_{\pm 1.4}$ ↑ 3.4/38.3
Qwen3-VL-Flash	$26.7_{\pm 0.8}$	$29.4_{\pm 0.3}$ ↑ 2.7	$29.7_{\pm 0.4}$ ↑ 3.0/30.1
GPT-5.4-mini	$63.5_{\pm 1.0}$	$59.2_{\pm 1.5}$ ↓ 4.3	$67.0_{\pm 0.9}$ ↑ 3.5/21.2

Table 5: Outcome discrimination from three future states. Arrows show the change from GT image; green denotes a decrease. The GT image-to-latent drop measures encoding and adapter loss, while the further drop to Pred. latent measures rollout loss.

Test	GT image	GT latent	Pred. latent
Overall AUROC	.814	.775 ↓ 3.9	.759 ↓ 5.5
Within-task 3-way	.781	.720 ↓ 6.1	.686 ↓ 9.5
Same-page pair	.752	.719 ↓ 3.3	.703 ↓ 4.9

Table 6: Within-task three-way accuracy by OSWorld domain, pooling planners. Each test ranks one successful state against two failed states from the same task. LO denotes LibreOffice; Multi denotes multi-application tasks.

Input	OS	Chrome	T-bird	VLC	GIMP	VS Code	LO Calc	LO Writer	LO Impress	Multi
GT image	71.0	83.8	80.3	81.2	79.3	70.0	76.4	79.1	76.4	77.1
GT latent	68.1	79.2	75.1	74.6	67.4	64.1	71.7	75.5	73.3	73.1
Pred. latent	64.3	75.2	69.1	68.6	61.4	60.1	70.7	66.5	67.3	67.7

actions and expected screens; episodes end at completion or $T = 30$. Each task-policy condition uses three independent seeds, and Table 4 reports mean terminal score $r_m \in [0, 1]$ over 1,083 episodes.

Let v_i and ℓ_i be the post- and pre-softmax WM scores, respectively, and $J = \arg\max_i v_i$. Agent executes candidate 1, Direct executes J, and Assisted overrides only when $J \neq 1$ and $v_J > 2v_1$, equivalently $\ell_J - \ell_1 > \log 2$. This planner-agnostic gate is fixed across planners, tasks, and seeds.

6.2 Results

Table 4 exhibits the two regimes in Claim 3. For the higher-capability Gemini and GPT agents, direct selection lowers performance by 6.0 and 4.3 points. Their assisted policies have the lowest override rates, 25.5% and 21.2%, and instead exceed the agent-only point estimates by 8.1 and 3.5 points. For the three lower-capability agents, direct selection improves performance by 2.7–4.7 points, and assistance overrides more often, on 30.1%–45.6% of decisions. Their ceiling remains candidate quality because reranking cannot recover a missing action. Thus strong agents benefit from preserving their prior ordering, whereas weaker agents can grant the WM more authority. Override frequency alone is insufficient: Claim 3 attributes the gain to the conditional advantage $\mathbb{E}[q_J - q_1 \mid S = 1]$ of the selected overrides.

Future-state diagnostic. Table 5 scores completed trajectories from their next three states, supplied as real screenshots, encoded real latents, or Predictor rollouts. Encoding costs 3.3–6.1 points relative to images, and prediction adds 1.6–3.4, leaving predicted latents 4.9–9.5 points below the image reference. All ten domains preserve GT image > GT latent > predicted latent; GIMP has the largest representation gap (11.9 points), while LO Calc has the smallest end-to-end gap (5.7 points). Table 5 uses domain-frequency weighting, so its within-task result differs from the unweighted category mean. Protocol details are in Appendix C.

7 Conclusion

LatentCUA is a three-stage latent world model combining semantic screen representation, action-conditioned multi-step dynamics, and goal-conditioned state scoring. Offline evaluations validate its representation, action-sensitive rollouts, and fine-grained candidate discrimination. On OSWorld, direct selection improves three weaker planners by 2.7–4.7 points, while selective assistance improves all five by 3.0–8.1 points. Our theory relates these regimes to representation, rollout, and scoring error relative to the candidate margin, establishing compact latent prediction as an effective basis for adaptive GUI planning.

AI USE STATEMENT

We used generative AI tools to aid and polish writing, retrieve and discover relevant literature, and support research ideation and execution through experiments, and mathematical formulations and proofs. The authors also selected the final research questions and methods, ran the experiments, verified every reported result and citation, and reviewed all AI-assisted work. No AI-generated datasets or experimental measurements are reported. The authors take responsibility for the final content of this work.

REFERENCES

David Abel, David Ellis Hershkowitz, and Michael L. Littman. Near optimal behavior via approximate state abstraction. In *International Conference on Machine Learning*, pp. 2915–2923, 2016.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael G. Rabbat, Yann LeCun, and Nicolas Ballas. Self-Supervised learning from images with a Joint-Embedding predictive architecture. In *Computer Vision and Pattern Recognition*, pp. 15619–15629. IEEE, 2023. doi: 10.1109/CVPR52729.2023.01499.

Mahmoud Assran, Adrien Bardes, David Fan, Q. Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv.org*, 2025. doi: 10.48550/arXiv.2506.09985.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-Invariance-Covariance regularization for Self-Supervised learning. In *International Conference on Learning Representations*, 2021.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *Trans. Mach. Learn. Res.*, 2024. doi: 10.48550/arXiv.2404.08471.

Yilin Cao, Yufeng Zhong, Zhixiong Zeng, Liming Zheng, Jing Huang, Haibo Qiu, Peng Shi, Wenji Mao, and Guanglu Wan. MobileDreamer: Generative sketch world model for GUI agent. *arXiv.org*, 2026. doi: 10.48550/arXiv.2601.04035.

Hyungjoo Chae, Namyoung Kim, Kai Tzu-iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. Web agents with world models: Learning and leveraging environment dynamics in web navigation. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2410.13232.

Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous Markov decision processes. *SIAM journal on computing (Print)*, 40(6):1662–1714, 2011. doi: 10.1137/10080484X.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://papers.neurips.cc/paper_files/paper/2017/hash/4a8423d5e91fda00bb7e46540e2b0cf1-Abstract.html.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. DeepMDP: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pp. 2170–2179, 2019.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to Self-Supervised learning. In *arXiv (Cornell University)*. Cornell University, 2020. doi: 10.48550/arxiv.2006.07733.

Yu Gu, Boyuan Zheng, Boyu Gou, Kai Zhang, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. Is your LLM secretly a world model of the internet? model-based planning for web agents. *Trans. Mach. Learn. Res.*, 2025. doi: 10.48550/arXiv.2411.06559.

Yiming Guan, Rui Yu, John Zhang, Lu Wang, Chaoyun Zhang, Liqun Li, Bo Qiao, Si Qin, He Huang, Fangkai Yang, et al. Computer-using world model. *arXiv.org*, 2026. doi: 10.48550/arXiv.2602.17365.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Neural Information Processing Systems*, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, 2018.

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

Nick Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024. doi: 10.48550/arXiv.2310.16828.

Biwei Huang, Chaochao Lu, Liu Leqi, José Miguel Hernández-Lobato, C. Glymour, B. Schölkopf, and Kun Zhang. Action-sufficient state representation learning for control with structural constraints. In *International Conference on Machine Learning*, pp. 9260–9279, 2021.

Woosung Koh, Sungjun Han, Segyu Lee, Se-Young Yun, and Jamin Shin. Generative visual code mobile world models. In *arXiv.org*, 2026. doi: 10.48550/arXiv.2602.01576.

Toby Jia-Jun Li, Lindsay Popowski, Tom M. Mitchell, and Brad A. Myers. Screen2Vec: Semantic embedding of GUI screens and GUI components. In *International Conference on Human Factors in Computing Systems*, pp. 1–15. ACM, 2021. doi: 10.1145/3411764.3445049.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Ze-Chen Bai, Weixian Lei, Lijuan Wang, and M. Shou. ShowUI: One Vision-Language-Action model for GUI visual agent. In *Computer Vision and Pattern Recognition*, pp. 19498–19508. IEEE, 2024. doi: 10.1109/CVPR52734.2025.01816.

Zhaoyang Liu, Jingjing Xie, Zichen Ding, Zehao Li, Bowen Yang, Zhenyu Wu, Xuehui Wang, Qiushi Sun, Shi Liu, Weiyun Wang, et al. ScaleCUA: Scaling open-source computer use agents with cross-platform data. In *arXiv.org*, 2026. doi: 10.48550/arXiv.2509.15221.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Ajay Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Neural Information Processing Systems*, 2020.

Dezhao Luo, Bo-Han 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. In *arXiv.org*, 2026. doi: 10.48550/arXiv.2504.13936.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv.org*, 2026. doi: 10.48550/arXiv.2603.19312.

Kai Mei, Jiang Guo, Shuaichen Chang, Mingwen Dong, Dongkyu Lee, Xing Niu, and Jiarong Jiang. R-WoM: Retrieval-augmented world model for computer-use agents. In *arXiv.org*, 2026. doi: 10.48550/arXiv.2510.11892.

Li Meng, Morten Goodwin, Anis Yazidi, and Paal Engelstad. State representation learning using an unbalanced atlas. In *International Conference on Learning Representations*, 2024. doi: 10.48550/arXiv.2305.10267.

William Peebles and Saining Xie. Scalable diffusion models with Transformers. In *IEEE International Conference on Computer Vision*, pp. 4172–4182. IEEE, 2022. doi: 10.1109/ICCV51070.2023.00387.

Yujia Qin, Yining Ye, Junjie Fang, Hao-Ming Wang, Shihao Liang, Shizuo Tian, Jun-Da Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. UI-TARS: Pioneering automated GUI interaction with native agents. *arXiv.org*, 2025. doi: 10.48550/arXiv.2501.12326.

Luke Rivard, Sun Sun, Hongyu Guo, Wenhu Chen, and Yuntian Deng. NeuralOS: Towards simulating operating systems via neural generative models. In *arXiv.org*, 2026. doi: 10.48550/arXiv.2507.08800.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Charles J. Stone. Optimal global rates of convergence for nonparametric regression. *The Annals of Statistics*, 10(4):1040–1053, 1982. doi: 10.1214/AOS/1176345969.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, et al. OpenCUA: Open foundations for computer-use agents. *Advances in Neural Information Processing Systems 38*, pp. 155016–155066, 2025. doi: 10.52202/085713-4669.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiao-Chuan Li, Siheng Zhao, Ruisheng Cao, T. Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems 37*, pp. 52040–52094. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024. doi: 10.52202/079017-1650.

Yao Zhao, Connor Stephens, Csaba Szepesvari, and Kwang-Sung Jun. Revisiting simple regret: Fast rates for returning a good arm. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 42110–42158, 2023. URL https://proceedings.mlr.press/v202/zhao23g.html.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *International Conference on Machine Learning*, 2025. doi: 10.48550/arXiv.2411.04983.

A Implementation Details

Modules. Within the **screen encoder**, the slot encoder runs two rounds of competitive cross-attention from 32 learnable slots (256-d) to the backbone tokens h of the frozen ShowUI-2B vision Transformer, with attention normalized across slots. The **reconstruction head** (training only) maps the 32 slots back to the backbone token sequence and is trained with a per-token cosine loss. The training-only **element matching module** uses a frozen structural encoder to form target tokens for selected GUI elements and an element retrieval head to map latent slots into the same token space. Each target jointly represents element type, state, text, location, and numeric value. The **narration head** A maps z to one 384-d vector. The **action tokenizer** maps the structured action and its description to a conditioning vector c and 3 action tokens. The **latent dynamics predictor** has 12 blocks at width 384, each with block-causal self-attention, cross-attention to the 3 action tokens, and adaLN modulation by c. Predictions are fed back with a "predicted" role embedding for rollout depths 2 and 3. The **goal-conditioned state scorer**, or value model, uses ShowUI-2B with LoRA on the language-model attention projections. It conditions on the task goal, short-horizon goals, eight real history frames, and up to three candidate future states, but receives no candidate action. A trainable adapter maps each raw 32×256 latent directly to ShowUI's visual-token space under token-level supervision. A learned scalar head reads the final language-model state to produce the value score $V \in \mathbb{R}$ used by the ranking losses in Section 3.1.

Table 7: Detailed architecture of the LatentCUA Screen Encoder. Parameter counts are trainable parameters and are rounded to two decimals.

Component	Configuration	Output	Params	Use
ShowUI visual backbone	Frozen ShowUI-2B vision tower; bf16; produces a 38×68 token grid at feature width 1536	$h_t \in \mathbb{R}^{2584\times1536}$	0	train/test
Slot encoder	$\mathrm{LN}(1536)$, linear $1536\to256$, fixed 2-D sinusoidal positions, 32 learned slots, and two competitive-attention rounds. Each update uses 256-d queries, keys, and values followed by an MLP $256\to512\to256$ and output layer normalization	$z_t \in \mathbb{R}^{32\times256}$	0.87M	train/test
Reconstruction decoder	2584 learned queries cross-attend to the 32 slots with 4 heads, followed by layer normalization, an MLP $256\to512\to256$, and a linear $256\to1536$ projection	$\hat{h}_t \in \mathbb{R}^{2584\times1536}$	1.58M	train only
Narration head A	Flattened latent $8192\to512\to384$ with GELU; matched to frozen narration embeddings with InfoNCE	one 384-d token	4.39M	train only
Element matching module H_{elem}	Frozen structural target encoder and a two-layer, $d = 256$, 8-head element retrieval Transformer that maps the latent slots into the target element-token space	up to 32 matched element tokens	2.40M	train only
Screen encoder total	Slot encoder, reconstruction decoder, narration head, and element matching module	latent and training readouts	9.24M	train

Table 8: Training configuration of the LatentCUA Screen Encoder.

Setting	Value
Screenshot resolution	1064×1904
Backbone representation	$38 \times 68 = 2584$ tokens, width 1536
Latent representation	32 slots, width 256; 8192 scalars per frame
Slot-attention rounds	2
Loss weights	$0.5\mathcal{L}_{\mathrm{rec}} + 0.5\mathcal{L}_{\mathrm{aln}} + \mathcal{L}_{\mathrm{elem}}$
Optimizer	AdamW, weight decay 10^{-4}, gradient clipping at 1.0
Learning rate	3×10^{-4} with cosine decay to 3×10^{-5}
Batch composition	32 ScaleCUA frames and 12 AgentNet frames per step
Initialization	Slot encoder, reconstruction decoder, and narration head warm-started; element matching module initialized from scratch
Training schedule	Planned 5000 steps; early stopped at 4750; validation every 250 steps with patience 3
Selected checkpoint	Step 4000, selected by text accuracy + type accuracy + state macro AUC
Encoder training cost	Approximately 14 seconds per step and 17.5 GB memory
Element records	16,514 ScaleCUA frames with 53,426 distinct strings

A.1 Screen Encoder Details

Table 7 decomposes the screen encoder. The frozen ShowUI tower is excluded from the trainable parameter count. The reported 9.24M parameters comprise the slot encoder, reconstruction decoder, narration head, and element matching module.

The element matching module receives up to 32 visible accessibility elements per frame. The frozen target encoder jointly embeds each element's type, 12 state attributes, text, normalized box, and numeric value. The retrieval head predicts an unordered set of tokens from z_t, and $\mathcal{L}_{\mathrm{elem}}$ matches this set to the frozen targets. Thus the accessibility record supervises the representation without becoming an input to the screenshot encoder. The reconstruction decoder predicts the residual ShowUI feature after subtracting a fixed positional mean μ_{pos}.

Action encoding. The LatentCUA Action Tokenizer separately embeds the action verb and optional second verb, the natural-language action description, typed text, normalized coordinates with Fourier features, the scroll magnitude, and a 64-key multi-hot vector. When an element binding is available, the target token also receives the element box and its owner slot, defined by the share of each slot's attention mass inside that box. The resulting components form a conditioning vector and three tokens for the verb, target, and argument-plus-location. Coordinates, boxes, and slot bindings are removed from future actions during both training and evaluation.

Table 9: Hyperparameters of the LatentCUA modules. The screen-encoder batch contains 32 ScaleCUA frames and 12 AgentNet frames. Predictor batch size is shown before gradient accumulation. Teacher forcing is annealed from 1 to 0 over the first 10% of predictor training; evaluation is autoregressive.

	Screen encoder	Latent predictor	Value model
trainable params	9.24M	37.48M	4.36M LoRA + adapter
latent / model width	256	384	256→1536 adapter
layers / blocks	2 slot rounds + 2 decoder	12	28 LM layers
attention heads	8	8	12 query / 2 KV
lr (AdamW)	3e-4 cosine	1e-4 constant	3e-4 constant
weight decay	1e-4	1e-4	1e-2
batch	32+12 frames	64×2 accum.	8 anchors
optimization length	4750 steps	8000 steps	5000 steps
checkpoint selection	step 4000	step 7750	step 4800

A.2 Training Objective Details

Write $\cos(u, v)$ for cosine similarity, $\mathrm{sg}[\cdot]$ for stop-gradient, and $\|\cdot\|_F$ for the Frobenius norm between latent arrays.

Screen Encoder. Let $\tilde{h}$ be the ShowUI features after subtracting the fixed positional mean used by the reconstruction decoder, and let n_t be the frozen embedding of the step narration. The reconstruction and alignment terms are

$$\mathcal{L}_{\mathrm{rec}} = 1 - \tfrac{1}{N}\textstyle\sum_i \cos\big(\mathrm{Dec}(z)_i, \mathrm{sg}[\tilde{h}_i]\big),$$
$$\mathcal{L}_{\mathrm{aln}} = \mathrm{CE}\big(\{\cos(A(z_t), n)/\tau\}_{n\in\mathcal{N}_t}, n_t\big),$$

where $\mathcal{N}_t$ contains the positive and contrastive narrations. If $T_{\mathrm{elem}}(\mathcal{E}_t)$ is the frozen encoding of the selected element set and $H_{\mathrm{elem}}(z_t)$ is the retrieved set, then

$$\mathcal{L}_{\mathrm{elem}} = \mathrm{Match}(H_{\mathrm{elem}}(z_t), \mathrm{sg}[T_{\mathrm{elem}}(\mathcal{E}_t)]), \qquad \mathcal{L}_E = .5\mathcal{L}_{\mathrm{rec}} + .5\mathcal{L}_{\mathrm{aln}} + \mathcal{L}_{\mathrm{elem}}.$$

Predictor. For prediction depth k, define the copy error as $b_{i,1} = \|z_{i+1} - z_i\|_F^2$ for dense one-step targets and $b_{t,k} = \|z_{t+k} - z_t\|_F^2$ for anchored rollouts. We use

$$u_{i,k} = \frac{1}{\max(b_{i,k}, \tau_c^2\overline{\|z\|_F^2})}, \qquad w_{i,k} = \frac{u_{i,k}}{\mathrm{mean}_{j,l}\, u_{j,l}},$$
$$\mathcal{L}_{\text{norm-mse}} = \mathrm{mean}_{i,k}\, w_{i,k}\|\hat{z}_{i+k} - z_{i+k}\|_F^2, \qquad k \in \{1,2,3\}.$$

For a sibling action a_i' from the same screen with observed successor z_{i+1}',

$$\mathcal{L}_{\mathrm{sib}} = \mathrm{mean}_{(z_i, a_i', z_{i+1}')}\, \|P(z_{\le i}, a_i') - z_{i+1}'\|_F^2, \qquad \mathcal{L}_P = \mathcal{L}_{\text{norm-mse}} + \mathcal{L}_{\mathrm{sib}}.$$

State Scorer. Write $V(S; s, g)$ for the score of state sequence S under short-horizon goal s and task goal g. The three matching losses are 16-way cross-entropies that identify the real next state, its short goal, and its task instruction:

$$\mathcal{L}_{\mathrm{frm}} = \mathrm{CE}_{16}(\text{next state}), \qquad \mathcal{L}_{\mathrm{short}} = \mathrm{CE}_{16}(\text{short goal}), \qquad \mathcal{L}_{\mathrm{task}} = \mathrm{CE}_{16}(\text{task goal}).$$

For correctly progressing frames $j > i$, and a transition at index q labeled wrong or redundant, the ordering losses are

$$\mathcal{L}_{\mathrm{prog}} = \max\big(0, .5 - (V_j - V_i)\big),$$
$$\mathcal{L}_{\mathrm{qual}} = \mathbf{1}_{\mathrm{wrong}} \max\big(0, .5 - (V_{q-1} - V_q)\big) + \mathbf{1}_{\mathrm{redundant}} \max\big(0, V_q - V_{q-1}\big).$$

The complete objective is $\mathcal{L}_V = \mathcal{L}_{\mathrm{frm}} + \mathcal{L}_{\mathrm{short}} + \mathcal{L}_{\mathrm{task}} + \mathcal{L}_{\mathrm{prog}} + \mathcal{L}_{\mathrm{qual}}$. The latent adapter also receives token-space supervision from the frozen ShowUI representation of the corresponding screenshot.

B Proofs and Additional Remarks for Section 4

B.1 Environment dynamics and policy-induced trajectories

Fix a task, initial-state distribution, observation kernel, and action budget. The environment kernel P specifies what follows each action, including incorrect and unobserved actions. It is not inferred by declaring a human trajectory to be the only valid outcome. For a fully observed state and a Markov policy, the state distribution evolves as

$$d^{\pi}_{t+1}(s' \mid g) = \sum_{s,a} d^{\pi}_{t}(s \mid g)\pi_t(a \mid s, g)P(s' \mid s, a). \tag{13}$$

With history-dependent policies, let $P_H(h' \mid h, a)$ be the environment-induced history kernel. The corresponding expression is

$$d^{\pi}_{t+1}(h' \mid g) = \sum_{h,a} d^{\pi}_{t}(h \mid g)\pi_t(a \mid h, g)P_H(h' \mid h, a). \tag{14}$$

Sums stand for integrals in continuous spaces. The full history, or an exact belief state, supplies a Markov information state. A finite screenshot window need not do so. Policy randomness used to generate proposals can be included in this information state; the proposal procedure is held fixed when comparing selection rules.

Equations (13) and (14) follow by conditioning on the preceding state and action. They show that the environment has one transition law, while different policies induce different screen distributions. A fixed Markov policy induces a chain, generally time-inhomogeneous for a finite budget. Because success is absorbing, $J_T(\pi, g) = \Pr_{\pi}(s_T \in G_g)$. This is the relevant task-level quantity; distance to a particular demonstration distribution is not equivalent to it.

Human expertise and the decision oracle. Let π^* maximize task success over policies with the same observation and action interface. A demonstrator π_H is an expert reference within that class, not an assumed optimizer. If one additionally assumes $J_T(\pi^*, g) - J_T(\pi_H, g) \le \epsilon_H$, then any agent π_A satisfies

$$J_T(\pi^*, g) - J_T(\pi_A, g) \le \epsilon_H + J_T(\pi_H, g) - J_T(\pi_A, g). \tag{15}$$

This follows by adding and subtracting $J_T(\pi_H, g)$. The expert-optimality assumption must be justified separately. The agent may use a different successful path or even outperform the demonstrator. Moreover, demonstrations alone do not identify counterfactual outcomes or recovery behavior away from their support; policy-induced distribution shift is a central issue in imitation learning (Ross et al., 2011).

What an action value means. At $t = T$, set $V^{\pi}_{T}(h) = \Pr(s_T \in G_g \mid h_T = h)$; this quantity is independent of the continuation policy. For $t < T$, conditioning on the first action gives

$$Q^{\pi}_{t}(h, a) = \mathbb{E}[V^{\pi}_{t+1}(h_{t+1}) \mid h_t = h, a_t = a], \quad V^{\pi}_{t}(h) = \mathbb{E}_{a\sim\pi_t} Q^{\pi}_{t}(h, a). \tag{16}$$

Thus $Q^{\pi}_{t}(h, a)$ already includes all possible subsequent successes and failures. A product of per-click accuracies would require a different model, such as a fixed sequence of necessary substeps with failure terminating the task. It is not a general GUI-task success formula.

B.2 Minimum decision-sufficient representation

Proposition B.1 (The coarsest sufficient representation)**.** *Let $h \sim_{\mathcal{G},H} h'$ when their scores in* (2) *agree for every goal, horizon, and query under consideration. Then $\Phi_{\min}(h) = [h]$ is decision-sufficient. Every representation Φ satisfying Definition 1 refines this quotient: there is a map r such that $\Phi_{\min} = r \circ \Phi$.*

Proof of Proposition B.1. Equality of all the candidate scores is reflexive, symmetric, and transitive, so $\sim_{\mathcal{G},H}$ is an equivalence relation. If $\Phi_{\min}(h) = \Phi_{\min}(h')$, then h and h' belong to the same equivalence class. Their candidate scores are therefore equal by construction, which proves decision sufficiency.

Now let Φ be any decision-sufficient representation. If $\Phi(h) = \Phi(h')$, Definition 1 implies $h \sim_{\mathcal{G},H} h'$, so $[h] = [h']$. Hence $r(\Phi(h)) := [h]$ is well-defined on the image of Φ and satisfies $\Phi_{\min} = r \circ \Phi$. Thus every sufficient representation distinguishes at least the equivalence classes distinguished by $\Phi_{\min}$. □

The proposition concerns minimal information, not a unique coordinate system or a minimum Euclidean dimension. Any invertible reparameterization of $\Phi_{\min}$ is equally sufficient. With a restricted neural architecture, the smallest realizable representation can also be larger than the abstract quotient. Nor is this quotient necessarily the coarsest representation preserving only the maximizing action: histories can have different candidate values and still have the same maximizer. Our definition deliberately retains the values needed to compare margins and decide whether to override.

B.3 Decision error and exact selection

Proof of Theorem 4.1. Fix $(h, g, k, \mathbf{a})$ and abbreviate the true and predicted rollout distributions by P and $\widehat{P}$. Add and subtract the two expectations of $V^{\star}_{g,k}$ and apply the triangle inequality:

$$|\widehat{Q} - Q| \le \left|Q - \mathbb{E}_P V^{\star}_{g,k}\right| + \left|\mathbb{E}_P V^{\star}_{g,k} - \mathbb{E}_{\widehat{P}} V^{\star}_{g,k}\right| + \left|\mathbb{E}_{\widehat{P}} V^{\star}_{g,k} - \mathbb{E}_{\widehat{P}} \widehat{V}_{g,k}\right|.$$

The first term is at most ϵ_{rep}. By the Kantorovich–Rubinstein inequality and the L_V-Lipschitz property, the second is at most $L_V W_1(P, \widehat{P}) \le L_V \epsilon_{\mathrm{dyn}}$. Jensen's inequality and (6) bound the last term by ϵ_{val}. This proves (7).

For the candidate-value loss, optimality under $\widehat{Q}$ and the uniform bound give

$$Q(\mathbf{a}^{\star}) - Q(\widehat{\mathbf{a}}) \le [\widehat{Q}(\mathbf{a}^{\star}) + B_k] - [\widehat{Q}(\widehat{\mathbf{a}}) - B_k] \le 2B_k.$$

For exact selection, every $\mathbf{a} \neq \mathbf{a}^{\star}$ satisfies

$$\begin{aligned}\widehat{Q}(\mathbf{a}^{\star}) - \widehat{Q}(\mathbf{a}) &\ge Q(\mathbf{a}^{\star}) - Q(\mathbf{a}) - 2B_k \\ &\ge \Delta^k_g(h) - 2B_k > 0.\end{aligned}$$

Therefore $\mathbf{a}^{\star}$ is the unique maximizer of $\widehat{Q}$. □

The strict inequality is necessary if tie breaking is unspecified. When the errors vary by candidate, the sharper sufficient condition is $B(\mathbf{a}^{\star}) + \max_{\mathbf{a} \neq \mathbf{a}^{\star}} B(\mathbf{a}) < \Delta$. The factor two appears only after replacing both candidate-specific terms by the same uniform bound.

From branch scores to root-action task values. The triangle inequality applied to (7) and (9) proves (10). To see when continuation error vanishes, execute the root action a and follow π_0 for another $k - 1$ steps, using $V^{\pi_0}_{t+k}$ as the terminal utility. Iterated conditional expectation then gives

$$Q^{\pi_0}_t(h, a) = \mathbb{E}_{P,\pi_0}[V^{\pi_0}_{t+k}(h_{t+k}) \mid h_t = h, a_t = a]. \tag{17}$$

This statement uses $k \le T - t$ and averages over the baseline's adaptive continuations. A fixed proposed action suffix does not in general have that distribution. Neither a short-horizon progress label nor a terminal scorer trained only to rank logged frames establishes (17). The continuation-error assumption makes this otherwise missing condition explicit.

B.4 Safe overrides and task-level improvement

The following statements formalize the distinction between beneficial individual overrides and improvement over a complete task.

Proposition B.2 (A sufficient condition for a beneficial override)**.** *Suppose $|\widehat{Q}_i - q_i| \le C_i$ simultaneously for all candidates. For a required improvement $\eta \ge 0$, use*

$$S = \mathbf{1}\{J \neq 1,\ \widehat{Q}_J - \widehat{Q}_1 > C_J + C_1 + \eta\}. \tag{18}$$

Every accepted override improves $Q^{\pi_0}_t$ by more than η; otherwise the baseline action is preserved.

The uniform threshold is $2C_k + \eta$. If the target is only short-horizon utility, $2B_k + \eta$ suffices for that target, but not for task completion. A raw Value Model score gap is not a calibrated success-probability gap. The deployed assisted policy instead applies a fixed ratio to scores normalized over the current candidates. If ℓ_i is the raw score and $v_i = \mathrm{softmax}(\ell)_i$, its gate is

$$S_2 = \mathbf{1}\{J \neq 1,\ v_J > 2v_1\} = \mathbf{1}\{J \neq 1,\ \ell_J - \ell_1 > \log 2\}. \tag{19}$$

This yields a planner-agnostic relative-evidence rule with a fixed log-score margin. We use the same rule across all planners, tasks, and random seeds, separately from the additive certificate above.

Proof of Proposition B.2. On an accepted override, the simultaneous error bounds give

$$\begin{aligned} q_J - q_1 &\geq (\widehat{Q}_J - C_J) - (\widehat{Q}_1 + C_1) \\ &= \widehat{Q}_J - \widehat{Q}_1 - C_J - C_1 > \eta. \end{aligned}$$

Otherwise the selected index equals 1 and the gain is zero. Thus the decision never lowers utility on the error-bound event. If the event holds with probability at least $1 - \delta$ over a deployment decision, bounded utility implies an unconditional expected gain of at least $-\delta$. The bound must hold jointly for all candidates because J is selected after observing their scores. Separate marginal confidence intervals without correction for this selection do not establish the proposition. □

Theorem B.3 (From local improvement to task success)**.** *Use the same initial-state distribution and budget T for π and π_0. If the assisted policy satisfies Proposition B.2 at every decision it can visit, then*

$$J_T(\pi, g) \geq J_T(\pi_0, g). \tag{20}$$

More generally, this holds whenever $\mathbb{E}_{a\sim\pi_t(\cdot|h,g)} Q_t^{\pi_0}(h, a) \geq V_t^{\pi_0}(h)$ at each visited history.

For $A_t^{\pi_0}(h, a) := Q_t^{\pi_0}(h, a) - V_t^{\pi_0}(h)$, the finite-horizon performance-difference identity is

$$J_T(\pi, g) - J_T(\pi_0, g) = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} A_t^{\pi_0}(h_t, a_t) \right]. \tag{21}$$

This identity sums advantages along trajectories of the new policy; it does not multiply per-step success probabilities.

Proof of Theorem B.3. Write $Y = \mathbf{1}\{s_T \in G_g\}$ and $A_t^{\pi_0}(h, a) = Q_t^{\pi_0}(h, a) - V_t^{\pi_0}(h)$. Under trajectories generated by π, the environment's next-history law is unchanged, so

$$\mathbb{E}_\pi A_t^{\pi_0}(h_t, a_t) = \mathbb{E}_\pi[V_{t+1}^{\pi_0}(h_{t+1}) - V_t^{\pi_0}(h_t)].$$

Summing over $t = 0, \ldots, T-1$ cancels all intermediate terms. The final term is $\mathbb{E}_\pi V_T^{\pi_0}(h_T) = \mathbb{E}_\pi Y = J_T(\pi, g)$, and the initial term is $J_T(\pi_0, g)$ because the initial distribution is shared. This proves (21). Nonnegative expected baseline advantage at every visited history therefore implies (20).

For the assisted policy, condition on each generated proposal set, included in the decision context. The baseline takes a_1, so $V_t^{\pi_0}(h) = Q_t^{\pi_0}(h, a_1) = q_1$. On an accepted override, Proposition B.2 gives $q_J - q_1 > \eta \geq 0$; otherwise the advantage is zero. The preceding identity completes the proof. □

The same argument applies to a fractional terminal evaluator score $R_T \in [0, 1]$. Set $V_T^\pi(h) = \mathbb{E}_\pi[R_T \mid h_T = h]$ and replace the success indicator Y by R_T throughout the proof. This is the version used for the OSWorld evaluation in Section 6.

The identity also gives an approximate version. If the expected baseline advantage is at least $-\xi_t$ at every history visited at step t, then

$$J_T(\pi, g) - J_T(\pi_0, g) \geq -\sum_{t=0}^{T-1} \xi_t. \tag{22}$$

For example, if the simultaneous candidate bounds can fail with probability at most δ_t at step t under the new policy's visitation distribution, bounded task values give $\xi_t = \delta_t$ in the integrated argument.

The resulting bound is $-\sum_t \delta_t$, not a claim that a single decision-level confidence statement guarantees an entire task.

This proof compares all actions using the same baseline continuation, even though π may subsequently override again. There is no circular requirement to know Q^π before defining π. Direct selection also admits task-level improvement if its expected baseline advantage is nonnegative at each visited history. Merely showing lower value loss on a fixed collection of baseline snapshots does not establish that condition.

B.5 Design implications and their assumptions

B.5.1 Multi-step accumulation

For future latents conditional on a fixed history, use

$$d_k(y, y') = \sum_{j=1}^{k} w_j d_{\mathcal{Z}}(z_j, z'_j), \qquad w_j > 0. \tag{23}$$

The encoded past context is fixed in these comparisons. In a deterministic latent Markov system, suppose the true transition F_a is L_F-Lipschitz and $\sup_{z,a} d_{\mathcal{Z}}(\widehat{F}_a(z), F_a(z)) \le \epsilon_1$. Starting from the same encoded state, the depth-j error satisfies

$$e_j \le \epsilon_1 \sum_{i=0}^{j-1} L_F^i, \qquad \epsilon_{\mathrm{dyn}} \le \sum_{j=1}^{k} w_j e_j. \tag{24}$$

If the transition requires memory, its sufficient latent context must be included in the state for this argument. The stochastic analogue requires uniform one-step error and Lipschitz continuity of the true kernel in W_1.

Proof of Equation 24. Let z_j and $\widehat{z}_j$ be the true and predicted latent at depth j, with $e_j = d_{\mathcal{Z}}(z_j, \widehat{z}_j)$ and $e_0 = 0$. The triangle inequality, the one-step model-error bound, and Lipschitz continuity give

$$\begin{aligned} e_{j+1} &= d_{\mathcal{Z}}(F_{a_j}(z_j), \widehat{F}_{a_j}(\widehat{z}_j)) \\ &\le d_{\mathcal{Z}}(F_{a_j}(z_j), F_{a_j}(\widehat{z}_j)) + d_{\mathcal{Z}}(F_{a_j}(\widehat{z}_j), \widehat{F}_{a_j}(\widehat{z}_j)) \\ &\le L_F e_j + \epsilon_1. \end{aligned}$$

Unrolling the recursion yields $e_j \le \epsilon_1 \sum_{i=0}^{j-1} L_F^i$. Summing with the trajectory weights gives the second part of (24). For Markov kernels, replace point distances with W_1 and assume $W_1(P_a(\cdot \mid z), P_a(\cdot \mid z')) \le L_F d_{\mathcal{Z}}(z, z')$; the same coupling argument applies. □

LatentCUA executes only the root action and re-encodes the next real screen. This removes the accumulated *open-loop prediction* error before the next candidate rollout. It does not undo a poor real action, restore lost history information, or return the agent to the previous visitation distribution.

B.5.2 From latent error to screen error

Latent error and pixel error require a common metric before they can be compared. Let $D : \mathcal{Z} \to \mathcal{X}$ decode latents into a screen-state space. Assume

$$\begin{aligned} d_{\mathcal{X}}(D(\phi(x)), x) &\le \delta_{\mathrm{dec}}, \\ d_{\mathcal{X}}(D(z), D(z')) &\le L_D d_{\mathcal{Z}}(z, z'). \end{aligned} \tag{25}$$

Then, for a predicted latent $\widehat{z}$ and actual screen x',

$$d_{\mathcal{X}}(D(\widehat{z}), x') \le \delta_{\mathrm{dec}} + L_D d_{\mathcal{Z}}(\widehat{z}, \phi(x')). \tag{26}$$

If U_g is L_X-Lipschitz in the corresponding screen-trajectory metric, the score error from decoding and rollout is at most $L_X(\delta_{\mathrm{dec}} \sum_j w_j + L_D \epsilon_{\mathrm{dyn}})$. If the learned score differs from U_g applied to the decoded trajectory by at most $\epsilon_{\mathrm{score},X}$ in expectation under the predicted rollout, then

$$|\widehat{Q} - Q| \le L_X \left(\delta_{\mathrm{dec}} \sum_j w_j + L_D \epsilon_{\mathrm{dyn}} \right) + \epsilon_{\mathrm{score},X}. \tag{27}$$

Exact action selection follows whenever this bound is below the local decision margin divided by two. This is an alternative route to bounding local score error, not an unconditional property of raw pixels. A latent rollout has a no-larger screen-error bound than a pixel model with trajectory-error bound ϵ_{pix} when

$$\delta_{\text{dec}} \sum_j w_j + L_D \epsilon_{\text{dyn}} \le \epsilon_{\text{pix}}. \tag{28}$$

Proof of Equation 26. Insert $D(\phi(x'))$ and apply the triangle inequality and (25):

$$\begin{aligned} d_{\mathcal{X}}(D(\widehat{z}), x') &\le d_{\mathcal{X}}(D(\widehat{z}), D(\phi(x'))) + d_{\mathcal{X}}(D(\phi(x')), x') \\ &\le L_D d_{\mathcal{Z}}(\widehat{z}, \phi(x')) + \delta_{\text{dec}}. \end{aligned}$$

Weighting and summing this inequality over the rollout yields $\delta_{\text{dec}} \sum_j w_j + L_D \epsilon_{\text{dyn}}$. If the screen-space utility is L_X-Lipschitz, its score difference is at most L_X times this quantity. Theorem 4.1 then gives the stated margin condition. □

This result is only as meaningful as $d_{\mathcal{X}}$. Small RGB error need not preserve a button label, and a large RGB change caused by animation need not matter for the task. A GUI metric based on element identity, text, layout, and interaction state is better aligned with the Lipschitz assumption.

B.5.3 Compactness: information, estimation, and computation

The quotient in Proposition B.1 motivates removing distinctions that do not affect values. This is a conditional benefit: excessive compression can merge histories that require different decisions. Under nested representation classes of dimension m, increasing m can reduce representation bias while making dynamics estimation harder. For a ν-smooth regression function on an m-dimensional support, standard nonparametric theory gives an illustrative root-mean-square rate

$$r_n(m) \asymp n^{-\nu/(2\nu+m)}. \tag{29}$$

This rate requires regularity, sampling, and noise assumptions (Stone, 1982). It describes an estimation tradeoff, not a uniform deployment bound for our neural Predictor. In particular, deriving ϵ_{dyn} in Theorem 4.1 from this rate would need additional coverage and concentration assumptions. Neither the nominal coordinate count Kd_z nor effective rank alone identifies the statistical dimension m.

The computational benefit is more direct. Storing a length-k rollout of K tokens of width d_z takes $O(kKd_z)$ entries. For fixed layer depth, dense self-attention over K tokens has an attention term $O(K^2 d_z)$ and projection terms $O(Kd_z^2)$ per frame. Reducing K therefore reduces these components of rollout cost. Cross-attention, the frozen visual backbone, and planner inference incur additional costs, so this accounting is not an end-to-end speedup claim. The relevant empirical test is whether the smaller representation retains decision quality at lower measured cost.

B.5.4 Collapse and decoupled training

Proposition B.4 (Prediction-only collapse)**.** *For the joint consistency objective*

$$\mathcal{L}_{\text{pred}}(\phi, F) = \mathbb{E}\|\phi(o_{t+1}) - F(\phi(o_t), a_t)\|^2, \tag{30}$$

the constant encoder $\phi(o) \equiv c$ and predictor $F(c, a) = c$ attain the global minimum zero.

Proof of Proposition B.4. For $\phi(o) = c$ and $F(c, a) = c$, every integrand is $\|c - c\|^2 = 0$. The objective is nonnegative, so zero is a global minimum. □

This is a failure of predictive consistency alone, not a claim that all jointly trained world models collapse. To see how decoding can oppose it, consider a nonconstant target Y decoded only from the latent with squared loss. For any constant representation,

$$\inf_D \mathbb{E}\|Y - D(c)\|^2 = \mathbb{E}\|Y - \mathbb{E}Y\|^2 > 0. \tag{31}$$

This follows from the bias-variance decomposition. If a nonconstant representation achieves a smaller loss, the constant solution cannot minimize that supervised objective. The argument requires that the decoder cannot bypass the latent to read the target. It also does not exclude partial collapse, certify semantic sufficiency, or settle the optimum of an arbitrarily weighted joint objective.

LatentCUA first learns its semantically supervised Encoder, freezes it while fitting the Predictor, and then trains the Value Model while both world-model modules remain frozen. Freezing prevents downstream objectives from changing upstream representations or dynamics. The larger batches and denser training signals enabled by separate stages are implementation benefits to measure, not guarantees supplied by the collapse proposition.

B.5.5 Deployment distribution and coverage

Applying a value model across real and predicted trajectories creates a separate distribution shift. Let P_{real} and P_{pred} be real and predicted latent-trajectory distributions. For any L_ℓ-Lipschitz evaluation loss ℓ_V, the Kantorovich–Rubinstein inequality gives

$$\left|\mathbb{E}_{P_{\text{pred}}}\ell_V - \mathbb{E}_{P_{\text{real}}}\ell_V\right| \le L_\ell W_1(P_{\text{pred}}, P_{\text{real}}). \tag{32}$$

The State Scorer is trained on mixed pools of encoded real states and frozen Predictor rollouts, including near-copy and same-trajectory alternatives. This aligns its scoring interface across the two state sources. The matched real-image, real-latent, and predicted-latent evaluation in Section 6 separately measures the representation and prediction components of the remaining decision gap.

B.6 Relation to latent-state theory

The framework follows the value-preservation view of state abstraction (Abel et al., 2016), while its transition-to-value step is closest in spirit to latent MDP analyses such as DeepMDP (Gelada et al., 2019). Our setting differs by using goals, finite candidate action sequences, partial observability, and a separately learned state scorer applied to predicted trajectories. Bisimulation metrics offer a stronger, policy-independent route to value preservation (Ferns et al., 2011), while action-sufficient representations characterize task-relevant latent variables under structural assumptions (Huang et al., 2021). The unbalanced Atlas method (Meng et al., 2024) concerns multi-chart geometry for self-supervised representations. It may motivate a charted latent when GUI states lie on several local manifolds, but it does not itself supply a decision-sufficiency or planning guarantee and is not required by our theorem.

B.7 Direct and assisted selection

A model of ordering, not of action success. The utilities q_i are the eventual success probabilities in (1), evaluated with the common baseline continuation. They need not sum to one. By contrast, $p_i = \Pr(I^* = i)$ is a probability distribution over the position of the best available candidate, with $\beta = p_1$ denoting the planner's top-choice accuracy. One possible sensitivity model is

$$p_i(\lambda) = \frac{\exp[-\lambda(i-1)]}{\sum_{j=1}^{n}\exp[-\lambda(j-1)]}, \qquad \lambda \ge 0. \tag{33}$$

Increasing λ makes the first rank more reliable. This truncated geometric family is an illustrative model of ordering, not an assumed law of GUI success. It should be checked against observed ranks and stratified by candidate count. Changing p at fixed candidates does not change $\max_i q_i$; it therefore isolates ordering from proposal quality.

Selection accuracy and the real-future reference. Let a_{img} be the Value Model's selection accuracy given three real future screenshots for every candidate, a_z its accuracy when those same screenshots are encoded and mapped back to visual tokens, and $b = \Pr(J = I^*)$ its accuracy given predicted latents. The differences $a_{\text{img}} - a_z$ and $a_z - b$ measure the representation-adapter and dynamics gaps on matched candidates. For the accuracy-ratio example below, set $\rho = b/a_z$ when $a_z > 0$. The real-latent condition removes prediction error, but not representation, scoring, or continuation error. Neither real-future condition is the decision oracle π^* or a guaranteed upper bound on b.

There is no universal sufficient fraction ρ. Under the illustrative assumption of a unique best candidate and the same constant utility loss $\Delta_0 > 0$ for every incorrect choice, define $G_{\text{direct}} := \mathbb{E}[q_J - q_1]$. Then

$$G_{\text{direct}} = \Delta_0(b - \beta), \qquad G_{\text{direct}} > 0 \iff \rho > \beta/a_z. \tag{34}$$

Assistance instead depends on reliability within the accepted subset, as derived below. This dependence on both planner quality and intervention coverage is related to selective prediction (Geifman & El-Yaniv, 2017).

Comparison with the planner. The expected value losses R_{WM} and R_{planner} correspond to expected simple regret restricted to the proposed candidate set (Zhao et al., 2023). They measure the gap to the best available candidate, not cumulative regret over a task. All expectations in this subsection use the same distribution of histories, ordered candidate sets, and evaluation outcomes. Candidate utilities use a common continuation protocol, so differences compare decisions rather than different evaluators. Subtracting the two expected value losses gives

$$R_{\text{planner}} - R_{\text{WM}} = \mathbb{E}[q_J - q_1], \tag{35}$$

which proves (11). If every nonoptimal candidate is exactly Δ_0 below the unique optimum, then $q_i = \max_j q_j - \Delta_0 \mathbf{1}\{i \neq I^*\}$. Therefore

$$R_{\text{planner}} = \Delta_0(1 - \beta), \qquad R_{\text{WM}} = \Delta_0(1 - b),$$
$$\mathbb{E}[q_J - q_1] = \Delta_0(b - \beta).$$

Substituting $b = \rho a_z$ gives (34). If the gap varies between decisions, the correct expression is $\mathbb{E}[\Delta(h)(\mathbf{1}\{J = I^*\} - \mathbf{1}\{1 = I^*\})]$; one cannot replace it by mean gap times mean accuracy difference without an additional independence assumption. With arbitrary gaps, the value-loss comparison remains exact. Equations (10) and (8) imply $R_{\text{WM}} \leq 2\mathbb{E}[C_k]$. Consequently $2\mathbb{E}[C_k] < R_{\text{planner}}$ is sufficient for improvement on the shared decision distribution. Dropping continuation error and using B_k instead would certify only short-branch utility.

An illustrative accuracy threshold. Suppose $n = 3$, real-latent accuracy is $a_z = .90$, and predicted-latent accuracy is $b = .75$, giving $\rho \approx .833$. Consider a stronger ordering with $p = (.80, .15, .05)$ and a weaker ordering with $p = (.45, .35, .20)$. In the equal-gap model, direct selection changes utility by $-.05\Delta_0$ and $+.30\Delta_0$, respectively. The corresponding break-even ratios are $.80/.90 \approx .889$ and $.45/.90 = .50$. These values are a constructed example, not experiment results. They illustrate why retaining the same fraction of real-future accuracy can be insufficient for one planner and sufficient for another.

Accuracy gains need not imply utility gains. For example, suppose the planner is correct on half of the decisions, with utility gap 1, and incorrect on the other half, with gap .01. A WM that corrects every decision in the second half but breaks 40% of all decisions in the first half has accuracy .60 versus the planner's .50, yet changes utility by $.50 \cdot .01 - .40 \cdot 1 = -.395$. The distribution of mistake costs is essential.

What real futures establish. The three-input diagnostic supplies actual future screenshots for all n candidates at depths $1, 2, 3$. Direct screenshot input retains scoring and finite-horizon errors; encoding those screenshots and applying the visual-token adapter additionally exposes the representation bottleneck; replacing the encoded latents with Predictor rollouts adds dynamics error. Consequently neither $a_z \leq a_{\text{img}}$ nor $b \leq a_z$ is guaranteed. If J_z is the real-latent selected index and $d = \Pr(J \neq J_z)$, the elementary bound

$$|b - a_z| \leq d \tag{36}$$

follows because the correctness indicators can differ only when the selected indices differ. This comparison must use the same candidate sets and outcome labels. Replacing only the logged action's prediction with a real state does not measure a_z. A single realized trajectory also does not reveal an expected success probability in a stochastic environment; repeated executions or an explicit deterministic evaluation assumption are required.

The reliability needed for assistance. Let $S \in \{0, 1\}$ indicate whether the WM replaces the planner's first choice. Its exact expected gain is

$$G_{\text{assist}} = \mathbb{E}[S(q_J - q_1)] = \Pr(S = 1)\mathbb{E}[q_J - q_1 \mid S = 1]. \tag{37}$$

This follows by subtracting q_1 from the selected utility $(1 - S)q_1 + Sq_J$ and taking expectations. With zero override probability the gain is zero. In the equal-gap model it becomes

$$G_{\text{assist}} = \Delta_0 \Pr(S = 1)(b_S - \beta_S). \tag{38}$$

Here $b_S = \Pr(J = I^* \mid S = 1)$ and $\beta_S = \Pr(I^* = 1 \mid S = 1)$. If $a_{z,S} > 0$ is real-latent accuracy on these same accepted decisions, positive coverage improves utility exactly when $b_S > \beta_S$, equivalently $b_S/a_{z,S} > \beta_S/a_{z,S}$. The global ratio b/a_z does not determine these conditional quantities. For instance, a WM with lower overall accuracy can still improve the strong planner in the preceding example if it intervenes on 10% of decisions where $b_S = .85$ and $\beta_S = .40$. Its expected gain is then $.045\Delta_0$. This example assumes an informative gate; a confidence threshold alone does not establish those conditional accuracies.

For arbitrary utility gaps, set $c = \Pr(S = 1)$ and $r = \Pr(q_J > q_1 \mid S = 1)$. Let $g = \mathbb{E}[q_J - q_1 \mid S = 1, q_J > q_1]$ and $\ell = \mathbb{E}[q_1 - q_J \mid S = 1, q_J \le q_1]$. With zero contributions for empty events,

$$G_{\text{assist}} = c\{rg - (1 - r)\ell\}. \tag{39}$$

For $c > 0$ and $g + \ell > 0$, improvement requires $r > \ell/(g + \ell)$. The required override precision is therefore higher when incorrect replacements are costly. This condition involves neither a universal oracle fraction nor an assumption that WM and planner mistakes are independent.

Combining planner order with WM evidence. Let C contain the available planner information and let E denote WM evidence. Among retaining the first candidate and following J, the optimal decision for expected utility is to override exactly when

$$\mathbb{E}[q_J - q_1 \mid C, E] > 0. \tag{40}$$

If a switch has utility cost c_{switch}, replace zero by that cost. In the equal-gap model, a useful explicit form follows from $p_i(C) = \Pr(I^* = i \mid C)$ and the likelihood $f_i(E \mid C) = p(E \mid I^* = i, C)$:

$$\Pr(I^* = i \mid C, E) = \frac{p_i(C)f_i(E \mid C)}{\sum_j p_j(C)f_j(E \mid C)}. \tag{41}$$

For positive priors and likelihoods, candidate j outranks candidate 1 when

$$\log \frac{f_j(E \mid C)}{f_1(E \mid C)} > \log \frac{p_1(C)}{p_j(C)}. \tag{42}$$

With (33), the right side is $\lambda(j - 1)$. Stronger prior ordering thus requires more WM evidence to overturn the first choice, and lower-ranked alternatives require more evidence. The likelihood is conditional on planner information, allowing correlated errors; multiplying two uncalibrated confidence scores would not implement this rule. For varying mistake costs, use (40) rather than posterior top-choice accuracy. Under a correct conditional utility model, this decision rule is at least as good in expectation as either always following J or always retaining 1. Estimation error can invalidate that comparison in practice.

Explicit thresholds under symmetric WM errors. For an explicit example, assume the equal-gap model, $n \ge 2$, and

$$\Pr(J = j \mid I^* = i) = \begin{cases} b, & j = i, \\ (1 - b)/(n - 1), & j \ne i. \end{cases} \tag{43}$$

The conditional error rate is constant across the best candidate's rank; ordinary aggregate accuracy b does not establish this assumption. Given $J = j \ne 1$, the unnormalized posterior masses for the WM recommendation and the planner's first choice are $p_j b$ and $p_1(1 - b)/(n - 1)$. Their utility difference in the equal-gap model is

$$\mathbb{E}[q_j - q_1 \mid J = j] = \Delta_0 \frac{p_j b - p_1(1 - b)/(n - 1)}{p_j b + (1 - p_j)(1 - b)/(n - 1)}. \tag{44}$$

Whenever this event has positive probability, its sign is positive exactly when $b > p_1/\{p_1 + (n-1)p_j\}$. Substitution of $b = \rho a$ proves the conditional assistance threshold

$$\mathbb{E}[q_j - q_1 \mid J = j] > 0 \quad \Longleftrightarrow \quad \rho > \frac{p_1}{a\{p_1 + (n-1)p_j\}}. \tag{45}$$

For the geometric rank prior this threshold is $b > \{1 + (n-1)\exp[-\lambda(j-1)]\}^{-1}$, making both rank and planner strength explicit.

In the stronger-planner example, $b = .75$ is below $p_1 = .80$, so direct selection hurts. Yet accepting only $J = 2$ improves utility by

$$\Delta_0\{p_2 b - p_1(1-b)/2\} = \Delta_0(.1125 - .1000) = .0125\Delta_0. \tag{46}$$

The assistance threshold for candidate 2 is $b > .727$, or $\rho > .808$; candidate 3 would require $b > .889$, or $\rho > .988$. For the weaker ordering, the direct threshold is $b > .45$, while the two override thresholds are approximately .391 and .529. At $b = .75$, both WM recommendations are accepted and assistance coincides with direct selection. This illustrates the different integration policies appropriate for the two planner orderings.

This advantage is not automatic: if the prior is uniform, both thresholds are $b > 1/n$. With $n = 2$, a WM supplying only its top choice under (43) has the same assistance and direct threshold, $b > p_1$. Additional confidence information can still enable selection of a more reliable subset, but aggregate accuracy alone cannot provide it.

Candidate supply and task-level success. Every selector restricted to the proposal set satisfies $\mathbb{E}[q_{\text{selected}}] \leq \mathbb{E}[\max_i q_i]$. Flattening the rank prior while holding the candidates fixed tests ordering quality. Lowering all candidate utilities tests proposal quality, a different limitation. If all candidates fail, no reranking policy can succeed.

Theorem B.3 states the additional condition required for these local comparisons to improve complete agent runs. The relevant values are $Q_t^{\pi_0}$, and advantages must remain nonnegative under the new policy's visitation distribution. A three-step progress scorer does not automatically estimate these values, so online terminal evaluator score must still be measured separately. The real-future diagnostic is not a replacement for that evaluation.

C Online Outcome Diagnostic Details

For all three interfaces in Table 5, the State Scorer receives the same task goal, short-horizon goals, and eight-frame real history; only the three candidate future states change. The overall test reports held-out AUROC. The within-task three-way test ranks one anchor from a successful trajectory against two failed anchors of the same task. The same-page test pairs successful and failed tree-search trajectories at their first anchor. The screenshot judge is a measured reference rather than an oracle, and all uncertainty is resampled by task rather than by correlated anchors.

For Table 6, runs from all planners are pooled before domain stratification, and the within-task three-way pools are kept identical across the three future-state interfaces.

D Evaluation Protocol and Metric Suite

Encoder metrics use 600 held-out ScaleCUA pairs and 1,500 held-out AgentNet frames. Predictor metrics use 400 held-out windows from each dataset and 400 ScaleCUA same-screen sibling anchors. State Scorer metrics use 300 held-out AgentNet anchors, with negatives drawn from the 16 nearest-neighbor tasks by mean task-goal embedding. Latents are unit-normalized per token; cosines are computed per token and then averaged. Table 10 defines the scorer suite.

Table 10: State Scorer metric suite. M1 is the headline action-selection task; M2–M4 test state and goal identity, M5 measures progress monotonicity, M6 tests wrong-step ordering, and Sibling measures same-screen discrimination.

metric	question	chance
M1 ground-action@N	pick the true action among $N-1$ similar-task distractors, judging predictor rollouts vs. (step, task)	$1/N$
M2 real-state@N	pick the true next state	$1/N$
M3 ground-short@16	pick the true step intent	0.0625
M4 goal-retrieval@16	pick the true task goal	0.0625
M5 rank acc. / ρ	order states by task progress	0.5 / 0
M6 wrong-step acc.	score a wrong or redundant successor below its predecessor	0.5
Sibling real/pred.	choose the correct same-screen successor	empirical $\approx .46$