WORLDTOKEN: TIME-FIRST SEQUENCE MODELING FOR ROBOTIC IMITATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action policies inherit both capabilities and input representations from pretrained vision-language models, showing great potential for robotic manipulation across diverse industrial settings. As these policies increasingly use interaction history, organizing the representation of historical observations determines how experiences enter temporal context and how relationships across time are modeled, which is a generally ignored challenge in previous works. In this work, we believe solving this challenge requires an architectural reconstruction and propose **WorldToken**. WorldToken encodes each timestep's observations into one world token, processes the resulting history with a causal Transformer, and generates action chunks with a diffusion action head. Unified token enables long horizon tasks while relieving the memory requirement of the temporal backbone, hence improving performance. This design also offers high interpretability and allows advances in language modeling, such as pre-training and scaling, to be transferred to robot interaction policies. In RoboCasa experiments, WorldToken successfully handles most tasks with 85M parameters and achieves 59.4% mean closed-loop success close to $\pi_{0.5}$ with 3.35B parameters. On the memory benchmark RM-Bench Blocks Ranking, WorldToken can reach the context of two minutes and achieves a success rate of 95%. In addition, WorldToken's holdout action RMSE is well described by power-law fits, and its closed-loop success rate improves consistently with increasing training data size in a study involving approximately 350,000 closed-loop evaluation episodes across 50 trained policies on RoboCasa, showing its scaling potential. We also conduct experiments to analyze information preservation and history use in WorldToken, providing empirical grounding for future work.

1 INTRODUCTION

Vision-language-action models (VLA) inherit both capabilities and input representations from pretrained vision-language models (VLM) (Brohan et al., 2023a; Kim et al., 2025; Black et al., 2025), showing great potential in applications like manipulation. Early VLMs such as LLaVA used an input image as context for understanding and generating text (Liu et al., 2023). Inherently, VLAs predict actions from the current observation and perform effectively on a range of manipulation tasks (Kim et al., 2025). However, as tasks become complex, policies have increasingly incorporated interaction history, and observations must also provide context for decisions across time (Koo et al., 2026; Torne et al., 2026). The memory mechanism, which summarizes historical information with memory modules, extends the context of VLAs and gains popularity (Koo et al., 2026). However, these memory modules generally retain the representational structure of VLMs, which can entangle within-timestep multimodal processing with cross-timestep history modeling. This makes it less straightforward to organize robot interaction histories around a unified sequence interface analogous to that used in language modeling. This raises a question: should the representational granularity inherited from vision-language models be carried over to history-conditioned robot policies?

To answer this question, we propose WorldToken, an architecture-level way to reconsider the organization of historical observations for VLAs. WorldToken formulates a time-first organization, separating multimodal encoding within each timestep from history modeling across timesteps. As shown in Figure 1, an encoder encodes each policy timestep's multimodal observations into one learned world token and processes the resulting history with a causal Transformer. An LM head

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

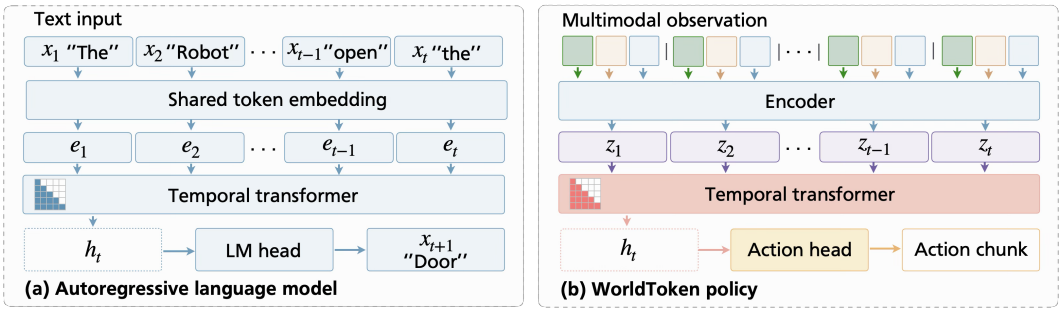


Figure 1: Structure of a language model and WorldToken. (a) An LM head maps the language model's contextual representation to a distribution over the next text token, which is appended to the sequence. (b) WorldToken encodes each policy timestep's multimodal observations into one world token and uses a diffusion action head to generate an action chunk from the history-conditioned state. Executed actions affect the environment, whose next observation supplies the next world token.

converts the language model's contextual representation into a distribution over text tokens. A DiT action head plays the corresponding role by generating an action chunk from the history-conditioned state. The encoder, temporal backbone, and action head have clear primary roles in perception, history modeling, and action generation. This design relieves the study of time relations for the temporal backbone, allowing VLA models to benefit from LLM-style scaling and pretraining without modification. Also, the design makes it easier to study component changes and interactions.

We evaluate WorldToken through extensive experiments. Across over 400,000 closed-loop evaluation episodes with more than 60 trained policies on RoboCasa (Nasiriany et al., 2024), WorldToken supports effective control with an 85M model size. Experiments across data and model sizes show consistent gains from additional demonstrations, showing the potential for scaling. We also vary the number of tokens per timestep and the available history to examine their effects on control performance and the computation required for temporal modeling to analyze if the information compression sacrifices performance. On RMBench Blocks Ranking (Chen et al., 2026), WorldToken reaches a context of two minutes, showing long horizon context ability. These results demonstrate that WorldToken can efficiently tokenize historical observations and also support policy timesteps as a useful basis for robot sequence modeling.

2 WorldToken Policy Architecture

We study how to organize multimodal observation histories for sequence modeling. We define a policy timestep as one cycle in which the robot observes, generates an action chunk, and executes part of it before observing again. We use this cycle as the basic sequence unit, separating multimodal encoding within each timestep from history modeling across timesteps. This design leads to three modules (Figure 2): a within-timestep encoder E_θ, a temporal backbone T_ϕ, and a DiT action head $\mathcal{D}_\psi$ (Peebles & Xie, 2023). We factorize the policy as

$$z_t = E_\theta(o_t), \qquad h_t = [T_\phi(z_{\le t})]_{\text{last}}, \qquad \hat{A}_t \sim \mathcal{D}_\psi(\cdot \mid h_t). \tag{1}$$

At each policy timestep t, WorldToken takes the current observation o_t as input and outputs an action chunk $\hat{A}_t$. Previously encoded world tokens can be reused without re-encoding past observations.

Within-timestep multimodal encoding. The within-timestep encoder E_θ takes the multimodal observation o_t as input and produces one world token z_t. We define

$$o_t = \{I_t^{1:K_{\text{view}}}, p_t, c\}, \tag{2}$$

where $I_t^{1:K_{\text{view}}}$ denotes the images from K_{view} cameras, p_t is proprioception, and c is the task condition. A CNN visual stem maps each camera image I_t^i to visual tokens V_t^i. Proprioception and the task condition are projected into tokens u_t^{prop} and u_t^{task}. The *observation tokens* are:

$$X_t = \left[V_t^1; \ldots; V_t^{K_{\text{view}}}; u_t^{\text{prop}}; u_t^{\text{task}}\right]. \tag{3}$$

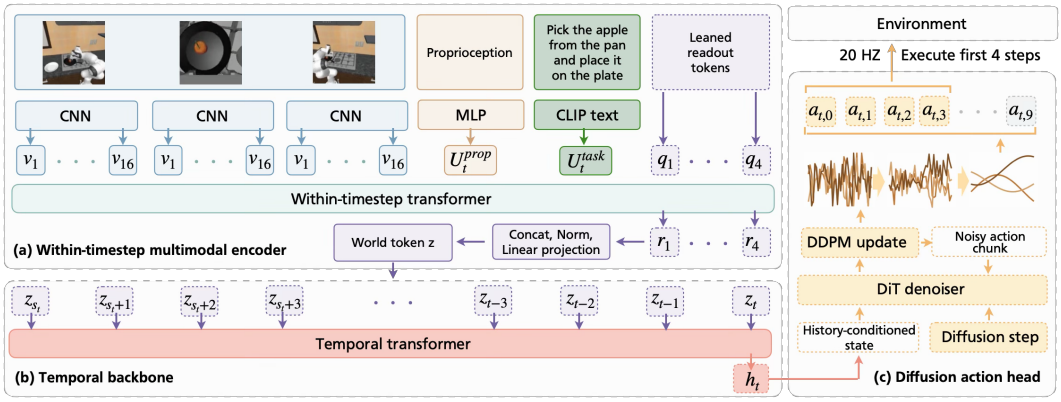


Figure 2: WorldToken architecture. The encoder maps each observation to one world token.

We append four learned readout tokens to X_t and process the combined sequence with a Transformer. Observation tokens attend only to observation tokens. Readout tokens attend to both observation and readout tokens. The output representations of the four readout tokens are concatenated, normalized, and linearly projected into z_t, which is passed to the temporal backbone.

Temporal backbone. The temporal backbone T_ϕ processes world tokens (without KV caching). We use a maximum context length of C policy timesteps. At timestep t, the input is $z_{s_t:t}$, with

$$h_t = [T_\phi(z_{s_t:t})]_{\text{last}}, \qquad s_t = \max(1, t - C + 1). \tag{4}$$

We use $K = 1$ world token per policy timestep, so the input contains at most C tokens. A causal Transformer processes these tokens in time order, allowing each token to attend to itself and earlier tokens. Its last output h_t integrates the observation history $o_{s_t:t}$ and is passed to the DiT action head.

DiT action head. The DiT action head $\mathcal{D}_\psi$ takes h_t from the temporal backbone as input and generates an H-step action chunk $\hat{A}_t \in \mathbb{R}^{H \times d_a}$, where d_a is the action dimension. At inference, action generation starts from Gaussian noise. At each diffusion step, the DiT denoiser ϵ_ψ receives the noisy action chunk, the diffusion step, and h_t, and predicts the noise. Stochastic DDPM sampling (Ho et al., 2020) iteratively denoises the action chunk to produce $\hat{A}_t$. The generated actions are unnormalized before execution. The robot executes the first H_{exec} actions before observing again at timestep $t + 1$.

During training, we normalize an expert action chunk A_t using statistics from the training demonstrations. At diffusion step k, we add Gaussian noise ϵ to obtain $A_t^{(k)}$. The DiT denoiser ϵ_ψ takes $A_t^{(k)}$, k, and h_t as inputs and predicts the injected noise. The default training objective is

$$\mathcal{L}_{\text{act}} = \mathbb{E}_{t,k,\epsilon}\left[\left\|\epsilon - \epsilon_\psi\left(A_t^{(k)}, k, h_t\right)\right\|_2^2\right]. \tag{5}$$

All three modules are trained jointly with the action-generation objective, without auxiliary losses.

3 Evaluation Framework

Evaluation measures. For RoboCasa, we use two complementary measures. Holdout action root-mean-square error (RMSE) assesses how well the policy imitates expert behavior. Given observations and histories from held-out demonstrations, we compare sampled action chunks with expert action chunks. We compute RMSE in the original, unnormalized action space and average the taskwise values across the 23 tasks. Closed-loop success rate (SR) assesses task completion when the policy's own actions determine subsequent observations. RMSE provides a continuous measure for fitting scaling trends and shows less variability than SR across training seeds in our experiments. The two measures broadly agree across our configuration grid, but meaningful gaps remain between action imitation and closed-loop task success. We therefore interpret them jointly (Appendix G).

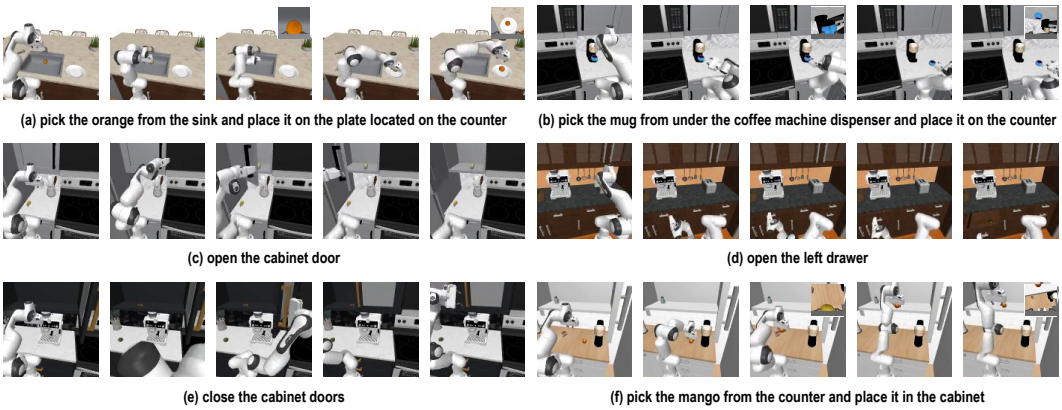


Figure 3: Examples of manipulation process.

Data and model configurations. Performance at a single data scale cannot fully characterize a model's capabilities. The benefit of an architectural choice may also depend on the amount of training data. We therefore evaluate WorldToken at five data scales and five model sizes to study performance trends and compare design choices across data scales. For RoboCasa, we use $D \in \{50, 100, 300, 1000, 2900\}$ training demonstrations per task, denoted D50, D100, D300, D1000, and D2900, respectively. The five model configurations (Appendix B.2) are N1 (44.3M parameters), N2 (85.3M), N3 (218.8M), N4 (648.9M), and N5 (1.49B). Parameter counts include the visual stems and policy modules, excluding the frozen CLIP text encoder.

Training and evaluation setup. We use causal attention and compute the action prediction loss at every valid timestep, following the sequence training structure of autoregressive language models. At position i in a training window, the model predicts an action chunk from the first i observations. Thus, C_{train} denotes the maximum history length, and $C_{\text{train}} = 10$ includes training with histories of length 1 through 10. For RoboCasa, all evaluations use the final checkpoint from each training run, without selecting checkpoints based on evaluation performance. At these checkpoints, RMSE is near its observed plateau across the scaling sweep (Appendix C.2). C_{test} denotes the visible history length at evaluation. Unless otherwise stated, we evaluate with $C_{\text{test}} = C_{\text{train}} = 10$ and report mean SR over three complete evaluations of the same checkpoint under identical test settings.

We evaluate WorldToken through four questions: whether it supports effective multitask control and how its performance scales with data and model capacity (Section 4); how the number of tokens per timestep affects performance and temporal computation (Section 5); whether trained policies rely on recent history (Section 6); and whether WorldToken can use longer histories (Section 7).

4 Can WorldToken achieve effective multitask control and scale systematically?

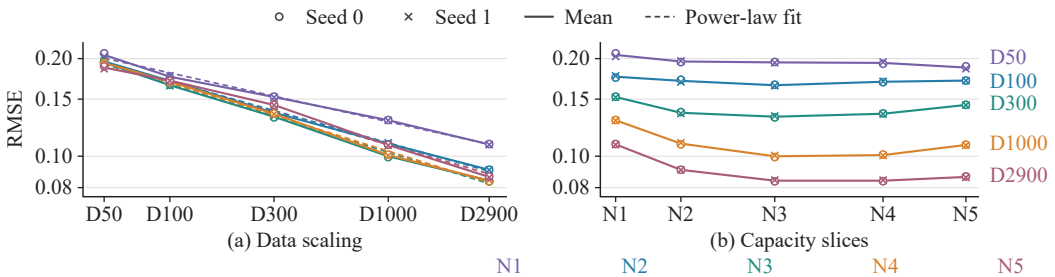
We first examine whether WorldToken supports effective multitask control and how performance scales with training data and model capacity. To this end, we evaluate WorldToken across the 5×5 grid defined in Section 3, training two policies with different random seeds for each configuration.

4.1 Effective multitask control using RoboCasa data

WorldToken achieves effective multitask control with an 85.3M-parameter policy. All trainable policy modules are initialized from scratch, while the pretrained CLIP text encoder remains frozen. At D300, WorldToken reaches a mean SR of 46.83% across two training seeds, compared with 31.28% for our reproduction of BC-Transformer using its official code and the same task set, data scale, and evaluation episodes (Appendix C.4). Increasing the training data to D2900 raises WorldToken's mean SR to 59.45% (Table 1). For context, the 3.35B-parameter $\pi_{0.5}$ model is reported to achieve 62.1% SR on RoboCasa Kitchen (Kim et al., 2026) . Manipulation examples are shown in Figure 3.

Table 1: Performance of WorldToken across dataset sizes and model capacities on RoboCasa.

		Training seed 0					Training seed 1				
		D50	D100	D300	D1000	D2900	D50	D100	D300	D1000	D2900
SR (%) ↑	N1	14.1	30.0	41.0	49.1	54.4	18.1	27.4	41.4	49.9	54.7
	N2	21.4	31.2	47.5	55.7	59.1	18.9	32.4	46.2	52.2	59.8
	N3	22.0	35.8	48.1	**58.5**	59.3	22.7	33.2	48.4	**57.8**	**61.2**
	N4	**27.1**	**36.5**	**51.2**	57.7	58.8	**24.0**	**36.3**	**51.4**	56.7	57.8
	N5	23.1	33.7	49.8	56.2	**59.7**	21.7	34.6	49.9	56.4	59.3
RMSE ↓	N1	0.208	0.175	0.153	0.129	0.109	0.203	0.177	0.152	0.129	0.109
	N2	0.195	0.172	0.137	0.109	0.091	0.197	0.169	0.135	0.110	0.091
	N3	0.195	**0.166**	**0.132**	**0.099**	**0.084**	0.194	**0.165**	**0.133**	0.101	**0.084**
	N4	0.193	0.169	0.135	0.101	**0.084**	0.195	0.170	0.135	**0.100**	**0.084**
	N5	**0.189**	0.171	0.144	0.109	0.087	**0.186**	0.171	0.144	0.108	0.086



(a) Data scaling (b) Capacity slices

	N1	N2	N3	N4	N5
Scaling exponent α	0.151	0.190	0.209	0.211	0.194
Coefficient of determination R^2	0.994	0.999	0.997	0.998	0.990

Figure 4: On RoboCasa, WorldToken's RMSE decreases approximately as a power law with dataset size, while gains from increasing model capacity diminish. (a) RMSE versus dataset size for each model capacity. (b) RMSE versus model capacity for each dataset size. Both panels use log axes. Markers show individual training seeds, and solid lines connect the means across the two seeds. Dashed lines in (a) show fits of $\mathrm{RMSE} \propto D^{-\alpha}$ to these means. The table reports the fitted scaling exponent α and the coefficient of determination R^2 for each model, with R^2 computed in log–log space.

4.2 Scaling with data and model capacity

Increasing the amount of training data consistently improves both RMSE and SR across all five model sizes (Table 1). From D50 to D2900, RMSE decreases by 47.0–56.8% and SR increases by 32.7–39.3 percentage points, after averaging the two training seeds. Additional trajectories may improve coverage of scene and object configurations, helping the policy generalize across task states.

The dependence of holdout RMSE on dataset size also varies with model capacity (Figure 4(a)). Across D50–D2900, RMSE decreases approximately as a power law. After averaging the two training seeds at each model capacity, we fit

$$\mathrm{RMSE}_P(D) \approx a_P D^{-\alpha_P}, \tag{6}$$

where P indexes model capacity. The fitted exponents range from 0.151 to 0.211, with R^2 values between 0.990 and 0.999. The exponent increases from 0.151 for N1 to approximately 0.21 for N3 and N4, indicating that these higher-capacity models make more effective use of additional data. This interaction between data and capacity resembles the coordinated scaling observed in language modeling (Kaplan et al., 2020; Hoffmann et al., 2022).

Figure 4(b) shows that the relative RMSE improvement from N1 to N3 increases with dataset size. Further increases to N4 and N5 yield no consistent reduction. These patterns suggest that the benefits of additional model capacity may be constrained by available data. However, the SR gap between smaller and larger models does not show the same pattern (Table 1). Larger models therefore gain a clearer advantage in action fitting without a corresponding increase in their advantage in task

Table 2: Token-interface comparisons with a context of $C = 10$ policy timesteps. K is the number of temporal tokens contributed by each timestep. Relative FLOPs (RelPs) are the leading analytic full-prefix temporal-backbone counts (Appendix D.3), normalized to the N2, $K = 1$ reference. Encoder and action-head costs are excluded. Each SR entry is the mean over three complete evaluations.

				SR (%) ↑					**RMSE ↓**				
K	Seed	Params	RelPs	D50	D100	D300	D1000	D2900	D50	D100	D300	D1000	D2900
1	0	85.3M	1.0×	21.4	31.2	47.5	55.7	59.1	0.195	0.172	0.137	0.109	0.091
1	1	85.3M	1.0×	18.9	32.4	46.2	52.2	59.8	0.197	0.169	0.135	0.110	0.091
4	0	92.4M	4.0×	20.3	**38.1**	48.8	55.6	59.9	0.202	0.166	0.138	0.110	0.090
50	0	83.0M	55.3×	**26.5**	37.6	**49.6**	**57.7**	**60.7**	**0.191**	**0.165**	**0.132**	**0.107**	**0.089**

completion. This may reflect limits in data quantity and quality, particularly behavioral diversity. RoboCasa generates trajectories by transforming a fixed set of human demonstrations (Nasiriany et al., 2024). Additional data may therefore reinforce similar behavior patterns, allowing larger models to fit demonstrated actions more accurately without comparable gains in closed-loop execution.

5 Do more tokens per timestep improve performance?

The encoder processes each observation using multiple tokens, while the temporal backbone receives only one token per timestep. This raises a central question: how does passing more tokens per timestep to the temporal backbone affect control performance and computational cost?

We compare $K \in \{1, 4, 50\}$ across all five data settings, using the N2 policies from the scaling grid in Section 4 as the $K = 1$ references. The $K = 1$ and $K = 4$ variants use the same encoder and differ only in the final projection. The $K = 50$ variant passes all 50 fused observation tokens directly to the temporal backbone without compression. All variants share the training data and evaluation episodes at each data setting, and comparisons use seed 0.

Increasing K from 1 to 4 yields no consistent RMSE improvement across data scales, while SR improves at some settings, most clearly at D100 (Table 2). The two variants use the same encoder apart from the final projection, so adding tokens per timestep brings no clear overall benefit.

Passing all 50 fused observation tokens directly to the temporal backbone reduces RMSE by 1.7–4.2% and improves SR by 1.6–6.4 percentage points across the five data scales (Table 2). These modest gains suggest that a single token already preserves the vast majority of the information needed for action prediction and control. However, the SR advantage of $K = 50$ over $K = 1$ is larger at D50 and D100 than at D300 and above, suggesting that learning this compact representation may be harder with limited training data. Retaining all 50 tokens avoids this compression, while more training data may help the encoder learn which information to retain in a single token.

Table 2 shows that, at $C = 10$, $K = 4$ and $K = 50$ require 4.03 and 55.31 times the temporal-backbone FLOPs of $K = 1$, respectively. With KV caching, the rate at which temporal-backbone FLOPs increase with history length is proportional to K^2. Although CNN visual stems account for roughly 80–90% of total policy FLOPs at $C = 10$, using more tokens per timestep causes temporal computation to grow rapidly as history becomes longer (Appendix D.3–D.5). Using one token per timestep therefore balances control performance and the cost of modeling long histories. These results support the policy timestep as a suitable sequence unit for modeling robot interaction histories.

6 Does WorldToken use recent history?

WorldToken predicts actions using current and past observations. To examine how much trained policies rely on recent history, we evaluate each of the 50 policies from Section 4 once at each shorter evaluation history length $C_{\text{test}} \in 1, 2, 5$, keeping all other evaluation settings fixed and using the reported results at $C_{\text{test}} = 10$ as the reference. Table 3 shows that reducing the evaluation history from ten to one or two policy timesteps lowers SR for all 50 policies, while using five timesteps retains most of their performance. These policies therefore make substantial use of recent observations, which may help them track motion.

Table 3: SR changes when the same policies are evaluated with shorter histories. Each cell reports $\Delta\text{SR} = \text{SR}(C_{\text{test}}) - \text{SR}(10)$ in percentage points for one, two, and five visible policy timesteps.

		$C_{\text{test}} = 1$					$C_{\text{test}} = 2$					$C_{\text{test}} = 5$				
		D50	D100	D300	D1000	D2900	D50	D100	D300	D1000	D2900	D50	D100	D300	D1000	D2900
Seed 0	N1	−3.5	−10.1	−18.1	−14.9	−13.0	−4.1	−3.3	−11.0	−8.6	−6.8	−0.9	+0.4	−0.5	−0.1	−0.6
	N2	−7.1	−8.7	−17.8	−18.0	−15.3	−5.3	−6.7	−12.3	−6.5	−6.2	+0.3	−0.1	−1.1	+0.8	+1.5
	N3	−7.3	−11.1	−20.2	−20.3	−18.9	−4.4	−9.9	−13.2	−11.7	−5.6	+0.3	−0.3	−1.1	−0.6	0.0
	N4	−10.5	−15.0	−21.3	−18.9	−22.2	−8.9	−11.3	−11.9	−10.1	−9.2	−1.4	−1.9	−1.3	−0.6	−1.0
	N5	−6.4	−13.6	−19.7	−17.3	−18.6	−5.7	−8.5	−12.7	−9.4	−8.8	+0.5	−0.4	−1.5	+1.5	−1.0
Seed 1	N1	−5.6	−5.7	−10.4	−14.0	−14.1	−2.6	−4.8	−1.4	−7.4	−5.6	+0.6	−0.2	+0.8	−0.1	+0.6
	N2	−8.0	−10.7	−16.6	−17.8	−17.2	−5.4	−7.4	−9.7	−6.6	−6.6	−0.8	0.0	+0.2	+0.5	−1.1
	N3	−8.4	−11.5	−20.6	−20.2	−19.2	−7.6	−7.3	−12.6	−10.1	−7.2	−2.1	+0.5	−0.2	−0.6	−2.1
	N4	−7.6	−18.9	−22.1	−19.0	−18.9	−9.0	−11.2	−13.2	−10.0	−5.8	−0.6	−1.7	−1.5	+1.0	+1.9
	N5	−6.9	−14.7	−21.5	−18.9	−18.8	−5.6	−10.2	−15.1	−9.4	−10.7	−2.3	0.0	−1.2	+1.0	+0.1

Table 4: RMSE and SR at different history lengths, with $C_{\text{train}} = C_{\text{test}} = C$.

Context C	RMSE, seed 0	RMSE, seed 1	SR (%) , seed 0	SR (%) , seed 1
1	0.150260	0.147767	46.75 ± 1.02	47.68 ± 1.50
2	0.149383	0.147528	46.87 ± 0.84	48.52 ± 1.36
5	0.137165	0.139603	**51.48 ± 0.26**	**50.00 ± 0.92**
10	0.131710	0.133098	48.12% ± 1.70 pp	48.38 ± 1.09

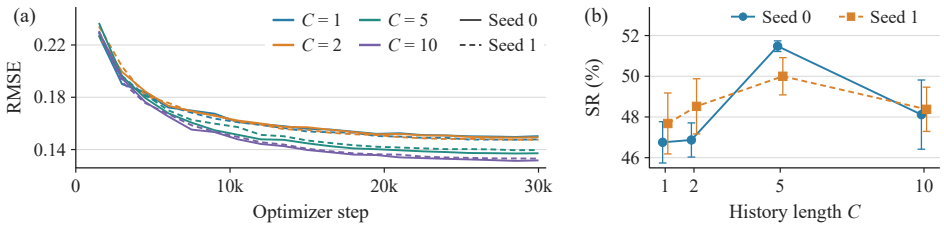


Figure 5: History length for N3 on D300 ($C_{\text{train}} = C_{\text{test}} = C$). (a) Training RMSE. (b) Mean final SR over three fixed-seed repeats per checkpoint. Error bars show sample standard deviations.

We next examine the effect of shorter training histories by training and evaluating N3 policies at D300 with $C_{\text{train}} = C_{\text{test}} \in \{1, 2, 5\}$. Training with these shorter histories recovers much of the SR lost when history is shortened only at evaluation (Tables 3 and 4). At $C = 1$, these policies achieve 47.22% mean SR, compared with 27.87% for policies trained with $C_{\text{train}} = 10$ and evaluated with $C_{\text{test}} = 1$. This recovery suggests that training can change how much a policy relies on history, even when the task itself may not require it.

Although RMSE decreases with longer training and evaluation contexts, SR peaks at $C = 5$ and falls at $C = 10$ (Table 4 and Figure 5). Longer histories may help the policy imitate expert actions, but also make it more sensitive to differences between expert demonstrations and its own execution.

7 Can WorldToken use longer histories?

The RoboCasa results suggest that a policy's reliance on recent history can reflect how it was trained, rather than a requirement of the task itself. We next turn to RMBench Blocks Ranking, where the current observation alone does not determine the next swap in the demonstrated sequence, which allows us to examine whether WorldToken can use longer histories to complete successive swaps.

Three colored blocks occupy left, middle, and right slots (L, M, R), with target order $(1, 2, 3)$. In the demonstrations, the robot first presses the button, then follows the reference swap sequence

$$\text{swap}(M, R) \to \text{swap}(L, R) \to \text{swap}(L, M) \to \text{swap}(M, R) \to \text{swap}(L, R), \tag{7}$$

Table 5: Blocks Ranking outcomes stratified by the number of swaps S in the reference solution. Each entry gives evaluator successes / strict behavioral successes / evaluator-only successes.

$C_{\text{test}} \backslash S$ (episodes)	1 (24)	2 (14)	3 (28)	4 (18)	5 (16)	Total (100)
608	24/24/0	14/14/0	25/25/0	17/16/1	15/15/0	95/94/1
288	24/24/0	14/14/0	23/23/0	16/16/0	15/15/0	92/92/0
128	24/24/0	14/14/0	13/3/10	5/5/0	3/3/0	59/49/10
64	24/24/0	3/3/0	2/0/2	9/0/9	0/0/0	38/27/11
32	24/24/0	3/0/3	0/0/0	1/0/1	0/0/0	28/24/4

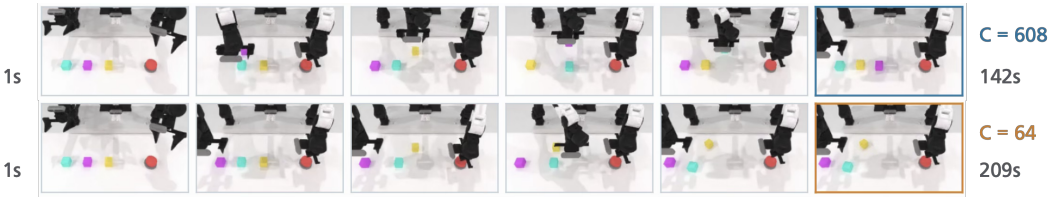


Figure 6: One episode evaluated with two history lengths. The checkpoint, initial state, and evaluation protocol are fixed, only C_{test} changes. With $C_{\text{test}} = 608$, the policy completes the five-swap reference sequence and succeeds at 142.6 seconds. With $C_{\text{test}} = 64$, it completes the first swap but fails to finish the task within the 210-second horizon. Frame timestamps are simulated seconds.

pressing after each swap and stopping at the target. The five non-target orders require 1–5 swaps.

We report evaluator success when the official checks for final block positions, an open right gripper, and a button press are satisfied. These checks do not require following the demonstrated swap sequence. We therefore also report *strict behavioral success*, which additionally requires following the reference swap sequence and placing the blocks accurately in the target slots. Evaluator successes that fail either additional criterion are counted as *evaluator-only successes* (Appendix F.5).

The policy uses the N2 temporal-backbone configuration from Section 3 and a maximum training history of $C_{\text{train}} = 608$. Model and training details are provided in Appendices F.2 and F.3. We evaluate the same checkpoint on 100 initial conditions at $C_{\text{test}} \in \{608, 288, 128, 64, 32\}$, keeping other evaluation settings fixed. Each policy timestep spans 0.24 seconds, so these history windows cover approximately 8–146 seconds. A complete swap followed by a button press takes about 28 seconds. The tested windows therefore cover less than one to about five such cycles.

7.1 Longer histories help complete multiple swaps

Increasing the evaluation history from 32 to 608 timesteps raises evaluator success from 28% to 95% and strict behavioral success from 24% to 94% (Table 5). All 24 episodes requiring one swap succeed at every tested history length, while shorter histories reduce success in episodes requiring multiple swaps. Figure 6 illustrates this difference. With $C_{\text{test}} = 64$, the policy completes the first swap but fails to finish the task, whereas $C_{\text{test}} = 608$ supports all five swaps. The main failures arise from errors in carrying out the swaps rather than an incorrect swap order. A block may be placed away from its intended position, after which the policy stalls without correcting the placement. These results suggest that longer histories help the policy complete successive swaps more reliably.

7.2 Continuing beyond demonstrations and the history window

We also disable termination at first success to examine whether the policy can continue beyond the demonstrated sequences. This exploratory test uses nine initial conditions covering all five non-target initial arrangements. With $C_{\text{test}} = 608$ fixed, the history window begins to slide after 145.92 seconds.

Table 6 shows that five of the nine trajectories continue the reference sequence after the history window begins to slide. The longest completes 31 correctly ordered swaps, with the last correct swap occurring at 856.44 seconds. These results suggest that the policy has learned a repeating cycle of three swaps, $\text{swap}(M, R) \to \text{swap}(L, R) \to \text{swap}(L, M)$. By repeating this cycle, it can continue

Table 6: Nine exploratory Blocks Ranking stress trajectories with rollouts allowed to continue after first success. S is the number of swaps required to first reach the target. "Swaps" counts correctly ordered reference-sequence swaps. "Last" is the time of the final correct swap.

Env. seed	S	Swaps	Last (s)	Env. seed	S	Swaps	Last (s)	Env. seed	S	Swaps	Last (s)
100000	2	13	350.52	100004	3	8	222.12	100015	5	5	139.38
100001	3	5	139.20	100003	2	9	250.14	100016	1	5	139.32
100002	4	17	470.28	100007	1	5	139.08	100008	1	31	**856.44**

beyond the maximum of five swaps in the demonstrations, even after the initial observations have left the history window and the policy encounters history prefixes not seen during training.

8 Related Work

Sequence modeling and vision-language-action policies. Sequence modeling is well established in decision making and generalist agents (Chen et al., 2021; Janner et al., 2021; Zheng et al., 2022; Reed et al., 2022). Robotic Transformer policies further explore multimodal conditioning and different ways to organize observations and actions (Brohan et al., 2023b; Jiang et al., 2023; Ghosh et al., 2024; Wang et al., 2024; Fu et al., 2024). More recently, VLA models have adopted pretrained vision-language backbones for manipulation, combining transferred perceptual representations with discrete or continuous action generation (Brohan et al., 2023a; Kim et al., 2025; Black et al., 2025; Li et al., 2024a; Bjorck et al., 2025). For history-conditioned control, inheriting a perceptual representation leaves an additional design choice: how observations should enter cross-timestep sequence modeling. We argue that the basic unit of temporal context should be examined as an explicit architectural choice, separately from the perceptual tokenization used within each timestep.

Temporal modeling and memory in robotics. Incorporating interaction history is a longstanding approach to control under partial observability, spanning recurrent imitation policies and attention-based models of observation histories (Mandlekar et al., 2022; Guhur et al., 2023; Li et al., 2024b). Recent memory-augmented VLAs extend this direction through historical compression, temporal aggregation, and memory retrieval and consolidation (Koo et al., 2026; Shi et al., 2026; Torne et al., 2026; Wang et al., 2026). Related approaches encode past interactions as visual traces (Zheng et al., 2025) or organize longer histories through hierarchical and recurrent architectures (Shah et al., 2026a; Zhou et al., 2026). Beyond architectural design, research also examines history relevance, training across context lengths, and the evaluation of memory-dependent behavior (Shah et al., 2026b; Agarwal et al., 2026; Chen et al., 2026; Dai et al., 2026). Within this literature, WorldToken studies time-first sequence modeling for robotic imitation learning, focusing on the relationship between within-timestep perceptual granularity and the organization of temporal context. See Appendix H.

9 Conclusion

In this work, we propose WorldToken, which provides a time-first answer to how robot interaction histories should be organized. WorldToken encodes each policy timestep into one world token and separates within-timestep perception from cross-timestep history modeling. In this way, observations can be organized just like word tokens in an LLM, and hence benefit from scaling and support long-history tasks. Extensive experiments demonstrate the feasibility of this organization for multitask robotic manipulation and extended use of history. Beyond feasibility, our analysis indicates that a single world token per policy timestep preserves most decision-relevant information, since retaining all fused observation tokens improves closed-loop success only marginally at substantially higher backbone cost, and that history dependence is shaped largely by the training context, which provide a reference for future work. Although we only test our method on manipulation tasks, our method provides a clear, general modeling principle and a modular architecture that facilitates ablation studies, attribution of performance changes, and targeted improvements. We plan to compare our method against other baselines using larger datasets with greater data diversity in both simulated and real-world tasks in the future.

REPRODUCIBILITY STATEMENT

The main paper specifies the model architectures, training setups, and experimental protocols, and the appendix provides additional details on implementations, training, rollout, aggregation, context analyses, task specifications, and qualitative results. The planned public release includes training code, evaluation code, logs, and models.

AI USE STATEMENT

Large language models were used throughout this work under author direction: for literature collection, experimental statistics, code development, and manuscript drafting, with additional assistance in experimental design. All core ideas, the experimental program, and the conclusions originate from the authors. The authors verified the paper's numerical claims against the underlying configurations, logs, evaluation records, and analysis artifacts, reviewed all AI-assisted text, code, and figures, and take full responsibility for the paper's content.

REFERENCES

Abhinav Agarwal, Adam Wei, Taylan Kargin, Michael Zeng, Cole Becker, Arif Kerem Dayi, Pablo Parrilo, Asuman Ozdaglar, and Russ Tedrake. Training and evaluating diffusion policies with long context lengths. *arXiv preprint arXiv:2606.16447*, 2026. URL https://arxiv.org/abs/2606.16447.

ARISE Initiative. RoboCasa BC-Transformer implementation in the robocasa branch of robomimic. Official software repository, 2024. URL https://github.com/ARISE-Initiative/robomimic/tree/robocasa.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. URL https://arxiv.org/abs/2301.08243.

Mido Assran, Adrien Bardes, David Fan, Quentin 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 preprint arXiv:2506.09985*, 2025. URL https://arxiv.org/abs/2506.09985.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. URL https://arxiv.org/abs/2503.14734.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.010. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of the 7th Conference on Robot*

Learning, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023a. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan H. Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems*, 2023b. doi: 10.15607/RSS.2023.XIX.025. URL https://arxiv.org/abs/2212.06817.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://arxiv.org/abs/2106.01345.

Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, et al. RMBench: Memory-dependent robotic manipulation benchmark with insights into policy design. *arXiv preprint arXiv:2603.01229*, 2026. URL https://arxiv.org/abs/2603.01229.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin C. M. Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://arxiv.org/abs/2303.04137.

Yinpei Dai, Hongze Fu, Jayjun Lee, Yuejiang Liu, Haoran Zhang, Jianing Yang, Chelsea Finn, Nima Fazeli, and Joyce Chai. RoboMME: Benchmarking and understanding memory for robotic generalist policies. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2603.04639.

Letian Fu, Huang Huang, Gaurav Datta, Lawrence Yunliang Chen, William Chung-Ho Panitch, Fangchen Liu, Hui Li, and Ken Goldberg. In-context imitation learning via next-token prediction. *arXiv preprint arXiv:2408.15980*, 2024. URL https://arxiv.org/abs/2408.15980.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.090. URL https://arxiv.org/abs/2405.12213.

Pierre-Louis Guhur, Shizhe Chen, Ricardo Garcia Pinel, Makarand Tapaswi, Ivan Laptev, and Cordelia Schmid. Instruction-driven history-aware policies for robotic manipulations. In *Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 175–187, 2023. URL https://arxiv.org/abs/2209.04899. CoRL 2022 proceedings.

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*, volume 97 of *Proceedings of Machine Learning Research*, 2019. URL https://arxiv.org/abs/1811.04551.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1912.01603.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025. doi: 10.1038/s41586-025-08744-2. URL https://arxiv.org/abs/2301.04104.

Jonathan Ho, Ajay N. Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022. URL https://arxiv.org/abs/2203.15556.

Michael Janner, Qiyang Li, and Sergey Levine. Offline Reinforcement Learning as One Big Sequence Modeling Problem. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://arxiv.org/abs/2106.02039.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. VIMA: Robot manipulation with multimodal prompts. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 14975–15022. PMLR, 2023. URL https://proceedings.mlr.press/v202/jiang23b.html.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. URL https://arxiv.org/abs/2001.08361.

Dongyoung Kim, Huiwon Jang, Myungkyu Koo, Suhyeok Jang, Taeyoung Kim, Beomjun Kim, Byungjun Yoon, Changsung Jang, Daewon Choi, Dongsu Han, et al. RLDX-1 technical report. *arXiv preprint arXiv:2605.03269*, 2026. URL https://arxiv.org/abs/2605.03269.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713, 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Myungkyu Koo, Daewon Choi, Taeyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. HAMLET: Switch your vision-language-action model into a history-aware policy. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.00695.

Yann LeCun. A path towards autonomous machine intelligence. OpenReview position paper, 2022. URL https://openreview.net/forum?id=BZ5a1r-kVsf.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. CogACT: A Foundational Vision-Language-Action Model for Synergizing Cognition and Action in Robotic Manipulation. *arXiv preprint arXiv:2411.19650*, 2024a. URL https://arxiv.org/abs/2411.19650.

Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, Hang Li, and Tao Kong. Vision-Language Foundation Models as Effective Robot Imitators. In *International Conference on Learning Representations*, 2024b. URL https://arxiv.org/abs/2311.01378.

Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.18647.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual Instruction Tuning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/6dcf277ea32ce3288914faf369fe6de0-Abstract-Conference.html.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 1678–1690, 2022. URL https://arxiv.org/abs/2108.03298. Conference held in 2021; introduces the robomimic framework.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems*, 2024. URL https://arxiv.org/abs/2406.02523.

Open X-Embodiment Collaboration et al. Open x-embodiment: Robotic learning datasets and RT-X models. In *IEEE International Conference on Robotics and Automation*, 2024. URL https://arxiv.org/abs/2310.08864.

Tim Pearce, Tabish Rashid, David Bignell, Raluca Georgescu, Sam Devlin, and Katja Hofmann. Scaling laws for pre-training agents and world models. In *International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48542–48562, 2025. URL https://arxiv.org/abs/2411.04434.

William Peebles and Saining Xie. Scalable diffusion models with Transformers. In *IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Peebles_Scalable_Diffusion_Models_with_Transformers_ICCV_2023_paper.html.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763, 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *Transactions on Machine Learning Research*, 2022. URL https://arxiv.org/abs/2205.06175.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *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.

Rutav Shah, Rajat Kumar Jenamani, Xiaohan Zhang, Lingfeng Sun, Roberto Martín-Martín, Yuke Zhu, Deva Ramanan, and Karl Schmeckpeper. Scaling short-term memory of visuomotor policies for long-horizon tasks. *arXiv preprint arXiv:2606.16178*, 2026a. URL https://arxiv.org/abs/2606.16178.

Rutav Shah, Yisu Li, Femi Bello, Yuke Zhu, and Roberto Martín-Martín. Memory retrieval in visuomotor policies for long-horizon robot control. *arXiv preprint arXiv:2606.25136*, 2026b. URL https://arxiv.org/abs/2606.25136.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2508.19236.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced Transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021. URL https://arxiv.org/abs/2104.09864.

Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z. Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, Karan Dhabalia, Michael Equi, Quan

Vuong, Jost Tobias Springenberg, Sergey Levine, Chelsea Finn, and Danny Driess. MEM: Multi-Scale Embodied Memory for Vision Language Action Models, 2026. URL https://www.pi.website/download/Mem.pdf.

Aleksandar Vujinovic and Aleksandar Kovacevic. ACT-JEPA: Novel joint-embedding predictive architecture for efficient policy representation learning. *IEEE Access*, 14:78895–78906, 2026. doi: 10.1109/ACCESS.2026.3696039. URL https://arxiv.org/abs/2501.14622.

Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling Proprioceptive-Visual Learning with Heterogeneous Pre-trained Transformers. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2409.20537.

Ziye Wang, Modi Shi, Chaojun Ni, Jiazhi Yang, Mengdi Li, Zhizhong Su, Tianwei Lin, and Hongyang Li. NativeMEM: Native Memory Compression for Long-Horizon Robotic Manipulation. *arXiv preprint arXiv:2607.06678*, 2026. URL https://arxiv.org/abs/2607.06678.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2312.13139.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024. URL https://arxiv.org/abs/2407.10671.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023. URL https://arxiv.org/abs/2304.13705.

Qinqing Zheng, Amy Zhang, and Aditya Grover. Online Decision Transformer. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 27042–27059. PMLR, 2022. URL https://proceedings.mlr.press/v162/zheng22c.html.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé, III, Andrey Kolobov, Furong Huang, and Jianwei Yang. TraceVLA: Visual Trace Prompting Enhances Spatial-Temporal Awareness for Generalist Robotic Policies. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2412.10345.

Yulin Zhou, Yimeng Wang, Nengyu Wang, Shaojia Xing, Shiyun Tu, Xiang Li, Jingkai Zhang, Ningbo Jiang, Yuankai Lin, Hua Yang, et al. Chronos: A physics-informed full-history framework for non-markovian long-horizon manipulation. *arXiv preprint arXiv:2606.30318*, 2026. URL https://arxiv.org/abs/2606.30318. Submitted to IEEE Transactions on Robotics.

Minjie Zhu, Yichen Zhu, Jinming Li, Junjie Wen, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, and Jian Tang. Scaling diffusion policy in transformer to 1 billion parameters for robotic manipulation. *arXiv preprint arXiv:2409.14411*, 2024. URL https://arxiv.org/abs/2409.14411.

APPENDIX CONTENTS

A Detailed WorldToken Architecture **17**

A.1 Within-Timestep Multimodal Encoding 17

A.2 Temporal Backbone and History Window 17

A.3 DiT Action Head 18

B Detailed Experimental Setup for RoboCasa **18**

B.1 Tasks and Datasets 18

B.2 Model Configurations and Training 18

B.3 Experimental Configurations 20

B.4 Evaluation Details 20

C Data and Model Scaling on RoboCasa **21**

C.1 Success Rates and RMSE for All 50 Policies 21

C.2 RMSE at 10% Training Intervals 23

C.3 Success Rates for Individual Tasks 25

C.4 Local BC-Transformer Comparison 26

C.5 Comparison with the Reported $\pi_{0.5}$ Result 26

D Token Interfaces, Policy Performance, and Computational Cost **26**

D.1 Single- and Multi-Token Interfaces 26

D.2 Policy Performance across Data Scales 27

D.3 Why Short-Context Cost Is Nearly Linear in Token Count 27

D.4 Module Costs with Cached History 28

D.5 Computational Growth with History Length 28

E History Truncation and Short-Context Training on RoboCasa **29**

E.1 Inference-Time History Truncation 29

E.2 Training and Evaluation with Matched Context Lengths 29

F History Use and Extended Execution in RMBench Blocks Ranking **30**

F.1 Task and Demonstrations 30

F.2 Detailed Experimental Setup 30

F.3 Task-Specific Training Objective 31

F.4 Comparison of Training Objectives 32

F.5 History Intervention and Success Criteria 32

F.6 History Length, Block Placement, and Swap Order 34

F.7 Extended Execution beyond Demonstrations 35

G Relationship between Offline Action RMSE and Closed-Loop Success **35**

G.1 Agreement and Differences between RMSE and Success Rate 35

G.2 Definitions of Offline RMSE and Closed-Loop Success 36

G.3 Variation across Training Seeds and Repeated Evaluations 37

H Extended Related Work on Robotic Sequence Modeling **37**

H.1 Sequence organization in robot policies 37

H.2 Temporal context, memory, and partial observability 38

H.3 Latent representations and predictive world models 39

H.4 Scaling in robot learning 39

I Qualitative RoboCasa Rollouts **40**

A Detailed WorldToken Architecture

This appendix details the attention mask, history window, and diffusion objective of the three modules in Section 2.

A.1 Within-Timestep Multimodal Encoding

The within-timestep encoder fuses multiview images, proprioception, and the task condition into a fixed-width temporal representation z_t.

For each camera, a visual stem maps I_t^i to visual patch tokens, which are augmented with embeddings that identify spatial position, camera view, and modality before multimodal fusion. If camera i produces M_i tokens,

$$V_t^i = \big(v_{t,1}^i, \ldots, v_{t,M_i}^i\big). \tag{8}$$

Proprioception and task conditioning are projected into u_t^{prop} and u_t^{task}. The within-timestep sequence is

$$X_t = \big[V_t^1; \ldots; V_t^{K_{\text{view}}}; u_t^{\text{prop}}; u_t^{\text{task}}\big]. \tag{9}$$

We append R learned readout tokens, denoted by Q, with $R = 4$ by default. Their output states are concatenated and projected into one temporal token. As shown in Figure 2(a), observation tokens attend only to observation tokens, while readout tokens attend to both observation and readout tokens. The readouts therefore aggregate observation features without feeding information back to the observation tokens.

Let $r_t^1, \ldots, r_t^R$ denote the final outputs at the readout positions:

$$(r_t^1, \ldots, r_t^R) = F_{\text{enc}}([X_t; Q])_{\text{readout}}. \tag{10}$$

Concatenation, RMSNorm, and a linear projection produce the only token passed to the temporal backbone:

$$z_t = W_z\, \text{RMSNorm}([r_t^1; \ldots; r_t^R]). \tag{11}$$

A.2 Temporal Backbone and History Window

We define the world-token trajectory as

$$Z_{1:T} = (z_1, z_2, \ldots, z_T). \tag{12}$$

The causal temporal backbone T_ϕ uses the Qwen2 decoder architecture (Yang et al., 2024) to produce history-conditioned states:

$$(h_1, \ldots, h_T) = T_\phi(z_1, \ldots, z_T), \qquad h_t = T_\phi(z_{\le t})_t. \tag{13}$$

Equation (13) gives the full-prefix form; finite-window evaluation uses $z_{s_t:t}$ as in Eq. (4). One-dimensional RoPE (Su et al., 2021) is applied along the world-token sequence to encode policy time. Causal self-attention ensures that h_t depends only on $z_{\le t}$.

Within each episode, each policy timestep adds one world token to the history. The action decoder reads only the last valid backbone output h_t.

Let C be the maximum visible-history length. At policy timestep t the policy reads

$$z_{\max(1,t-C+1):t}. \tag{14}$$

If C covers the episode so far, the policy receives full-episode context; once the episode exceeds C, the earliest tokens are evicted.

When the window slides, we reindex the retained tokens contiguously and recompute the backbone outputs. The backbone has no absolute position embeddings. Subtracting the same offset from all retained positions preserves the relative position differences used by standard RoPE attention.

A.3 DiT Action Head

At policy timestep t, WorldToken generates an H-step action chunk

$$\hat{A}_t = (\hat{a}_{t,0}, \hat{a}_{t,1}, \ldots, \hat{a}_{t,H-1}) \in \mathbb{R}^{H \times d_a}. \tag{15}$$

Each action dimension is normalized to $[-1, 1]$ using statistics from the training demonstrations and unnormalized before execution.

We model the conditional action distribution with diffusion. For a normalized ground-truth chunk A_t, training samples diffusion step k and Gaussian noise ϵ:

$$A_t^{(k)} = \sqrt{\bar{\alpha}_k}\, A_t + \sqrt{1 - \bar{\alpha}_k}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I). \tag{16}$$

Here $\bar{\alpha}_k$ is the cumulative product of the diffusion schedule's per-step signal-retention factors. A DiT denoiser (Peebles & Xie, 2023) predicts the injected noise conditioned on the current history representation h_t:

$$\hat{\epsilon} = \epsilon_\psi\big(A_t^{(k)}, k, h_t\big), \qquad L_{\text{act}} = \mathbb{E}_{t,k,\epsilon}\left[\left\|\epsilon - \epsilon_\psi\big(A_t^{(k)}, k, h_t\big)\right\|_2^2\right]. \tag{17}$$

Equation (17) is the default diffusion objective; Appendix F.3 specifies the task-specific modification for Blocks Ranking. The diffusion timestep and h_t condition each DiT block through adaLN-Zero modulation. At inference, action chunks are generated from Gaussian noise using stochastic DDPM sampling (Ho et al., 2020).

Control follows a receding-horizon scheme. Each query generates H actions but executes only the first H_{exec}:

$$\hat{A}_t \sim \mathcal{D}_\psi(\cdot \mid h_t), \qquad \hat{A}_t^{\text{exec}} = (\hat{a}_{t,0}, \ldots, \hat{a}_{t,H_{\text{exec}}-1}). \tag{18}$$

After executing this prefix, the policy observes the environment again, appends the next world token, and replans.

B Detailed Experimental Setup for RoboCasa

This appendix specifies the RoboCasa data, models, training, and evaluation. Settings specific to the token and history comparisons are given in Appendices D and E.

B.1 Tasks and Datasets

The RoboCasa sweep uses a fixed set of 23 tasks. We exclude `OpenDoubleDoor` because its official image-based generated-demonstration dataset has only 1,500 demonstrations, fewer than the common $D = 2900$ endpoint. For each retained task, we reserve 100 demonstrations for a shared holdout set, giving 2,300 holdout demonstrations in total. None appear in any training set.

The $D = 300$ training set is the official `300_demos` subset. The $D \in \{50, 100, 1000\}$ sets are independently fixed, hash-recorded samples from the non-holdout pool; they may overlap but are not nested. The $D = 2900$ set uses the full non-holdout pool. At each D, all model capacities and training seeds use the same demonstrations.

B.2 Model Configurations and Training

Each data scale is trained for approximately 100 loader epochs, using the optimizer-step budgets in Table 8. Each epoch visits eight sample slots per training demonstration, drawing a random ten-timestep observation window for each slot. This samples windows rather than enumerating all valid windows. We evaluate the final scheduled checkpoint from every run, without rollout-based checkpoint selection. Appendix C.2 reports holdout RMSE at 10% intervals of the training budget.

All training and evaluation used two single-node servers: one with eight NVIDIA A100-80GB GPUs and one with eight NVIDIA H100-80GB GPUs.

Table 7 specifies the five model configurations, including the within-timestep fusion encoder, causal temporal backbone, and DiT action decoder. Each model uses a separate CNN stem for each camera

Table 7: RoboCasa N1–N5 architecture and peak learning rates. Parameters include all visual stems and policy modules, excluding the frozen CLIP text encoder. FF denotes feed-forward hidden width; Q and KV denote query and key/value attention heads.

Setting	N1	N2	N3	N4	N5
Parameters	44.3M	85.3M	218.8M	648.9M	1490.3M
Encoder/temporal LR (10^{-4})	6.00	4.25	3.00	1.50	0.50
Action-decoder LR (10^{-4})	3.00	3.00	3.00	3.00	3.00
Within-timestep fusion encoder (SwiGLU)					
Width	512	768	1024	1536	2048
Layers	1	2	4	6	8
Q heads	4	6	8	12	16
KV heads	4	6	8	12	16
FFN width	2048	3072	4096	6144	8192
Causal temporal backbone (Qwen2, SwiGLU)					
Width	512	768	1024	1536	2048
Layers	2	4	8	12	16
Q heads	4	6	8	12	16
KV heads	4	6	8	12	16
FFN width	1408	2048	2816	4096	5504
DiT action decoder (GELU, adaLN-Zero)					
Width	128	192	256	384	512
Layers	1	2	4	6	8
Attention heads	4	6	8	12	16
FFN width	512	768	1024	1536	2048

Table 8: RoboCasa training and closed-loop execution recipe. The main sweep uses $C_{\text{train}} = 10$; separately trained context variants are identified where they are analyzed.

Component	Protocol
Inputs	Three 128×128 RGB views, 16-D proprioception, and a frozen 768-D CLIP task embedding
Visual stems	One five-stage CNN per camera; 5×5 convolutions with channels $(96, 144, 192, 384, 768)$, each followed by 2×2 max pooling, group normalization, and SiLU; output grid $4 \times 4 \times 768$ per view
Token readout	48 visual tokens plus proprioception and task tokens; four learned readouts are concatenated, RMS-normalized, and projected to one temporal world token
Temporal sampling	20 Hz environment control; one observation, world token, and replanning decision every four control steps (5 Hz)
Actions	Same-frame-aligned 12-D commands; $H = 10$ predicted, $H_{\text{exec}} = 4$ executed before replanning
Context and diffusion	$C_{\text{train}} = 10$ world tokens; 20-stage cosine schedule and stochastic 20-step DDPM evaluation
Optimization	AdamW, $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, zero weight decay, global gradient-norm clipping at 1.0, BF16, and global batch size 192 observation windows
Training budget	$D = 50/100/300/1000/2900$: 5k/10k/30k/100k/280k optimizer steps, respectively
LR schedule	Encoder/temporal rates in Table 7; action-decoder rate 3.0×10^{-4}; 5% linear warmup, then cosine decay to 10% of each peak rate

and the frozen `openai/clip-vit-large-patch14` text tower with projection (Radford et al., 2021) for task conditioning. The shared visual and training settings appear in Table 8. The learning rates in Table 7 were selected at $D = 300$ and reused across the data scales for each capacity.

Width denotes the hidden dimension d of each module. The feed-forward network (FFN) width denotes its intermediate dimension. The fusion encoder and action decoder use an FFN width of $4d$. The temporal backbone uses $8d/3$, rounded up to a multiple of 128. Temporal attention dropout is 0.1. Dropout is disabled in the fusion encoder and action decoder.

B.3 Experimental Configurations

Table 9 lists training seeds separately from repeated evaluations of a fixed checkpoint. Comparisons share demonstration identities, holdout crops, and evaluation episode identities where these are specified as controlled.

Table 9: RoboCasa experiment configurations. "Executions" counts complete 1,150-episode evaluations per checkpoint and stated inference setting, not independent training seeds.

Experiment	Changed / held fixed	Training seeds	Executions
Data and model scaling	Five D values $\times$ five capacities; $C_{\text{train}} = C_{\text{test}} = 10$	0, 1	3
Token count	$K = 1, 4, 50$ at all five D; $d = 768$, $C = 10$	0	3
Same-checkpoint history	$C_{\text{test}} = 1, 2, 5$ for every scaling checkpoint; $C_{\text{train}} = 10$	0, 1	1
Matched training history	$C_{\text{train}} = C_{\text{test}} = 1, 2, 5, 10$; 218.8M, $D = 300$	0, 1	3

B.4 Evaluation Details

Closed-loop evaluation. A low-level control step is one environment command, distinct from a simulator integration step. Each closed-loop evaluation contains 50 episodes for each of the same 23 tasks, or 1,150 episodes in total. At episode startup, we left-pad the history with copies of the first observation to reach C_{test} positions. Only RoboCasa uses this padding; RMBench does not.

Repeated evaluations and reporting. Each main-sweep final checkpoint is evaluated three times at $C_{\text{test}} = 10$, using the same episode identities, environment seeds, rollout seed, and protocol. Reported standard deviations are sample standard deviations across the stated repeats. Because stochastic diffusion and simulation are not bitwise deterministic, these values measure variation across fixed-seed executions, not uncertainty over independently sampled environments. To average across training seeds, we first average each checkpoint's repeated evaluations and then give each trained policy equal weight.

Action alignment. Let j_t be the stored control-frame index for policy timestep t. Policy timesteps are four control frames apart, but each target chunk contains H consecutive low-level commands: `action[`j_t`:`$j_t + H$`]`. The first command is applied after conditioning on the observation at j_t. Padded terminal slots repeat the final action and are masked. A query contributes to the loss only when all H targets are real, so no padded terminal targets enter training. Appendix F.2 gives the RMBench indexing convention.

Holdout RMSE. For the main sweep, `eval_seed=0` selects eight fixed ten-timestep crops from each of the 2,300 holdout demonstrations. Evaluation uses these crops rather than all possible trajectory windows. Separately trained context variants use the same procedure with crops of their own training length C. At each query with a complete target, we evaluate one $H = 10$ action chunk. Query i sees only the first i observations of its crop, so the first two queries see one and two observations, respectively. No query reads observations before the crop begins. We compute RMSE in the unnormalized 12-D command space and average the 23 taskwise RMSE values equally.

Deterministic and stochastic DDPM evaluations start from the same fixed-seed Gaussian x_T for each query. The deterministic sampler follows the DDPM posterior mean without reverse-step noise. The stochastic sampler adds Gaussian noise with the variance specified by the DDPM schedule.

Observation fields and action normalization. The RGB views are ordered as left agent view, right agent view, and eye-in-hand. Proprioception concatenates end-effector position (3) and quaternion (4) relative to the base, base position (3) and quaternion (4), and gripper joint positions (2), preserving the stored coordinate order. With zero-based action indices, 0–2 are arm-translation commands, 3–5 arm-rotation commands, 6 the gripper command, 7–10 the base/torso commands, and 11 the base-mode command. These controller inputs have different meanings, so the full 12-D RMSE has no single physical unit.

For each action dimension, the normalizer scans all actions in the selected training demonstrations once. It sets $\ell = (a_{\max} + a_{\min})/2$ and $r = (a_{\max} - a_{\min})/2$, then applies $(a - \ell)/r$, replacing r

with 1 when $r < 10^{-4}$. These fixed statistics are stored in the checkpoint. Generated actions are transformed back before RMSE computation and execution.

Rollout settings. Task horizons are measured in low-level control steps: 300 for `CoffeePressButton`; 600 for `CoffeeServeMug`, `CoffeeSetupMug`, and `PnPCounterToMicrowave`; 700 for `CloseDoubleDoor` and `PnPCounterToSink`; and 500 for each of the other 17 tasks. The reporting runtime uses robosuite 1.5.0 at commit `dc7fcf9fa6cd`, environment-bound action clipping with margin 10^{-4}, and unit action scale.

C Data and Model Scaling on RoboCasa

This appendix provides detailed results supporting the RoboCasa analysis in Section 4, including the full 50-policy sweep, holdout RMSE throughout training, task-level success rates, and baseline comparisons. These results allow the aggregate trends reported in the main text to be examined across training seeds, repeated evaluations, training progress, and individual tasks. We also document the settings and limitations of the baseline comparisons to clarify how their results should be interpreted.

C.1 Success Rates and RMSE for All 50 Policies

Descriptive power-law fits. For each capacity, the five data-scale points are the arithmetic means of the two training-seed RMSE values. We fit $\log(\mathrm{RMSE}) = b - \alpha \log D$ by unweighted ordinary least squares over these five points and compute R^2 in the same log-RMSE space.

Table 10: Closed-loop SR (%) for all 50 scaling policies trained with $C_{\text{train}} = 10$. Each checkpoint is evaluated once at $h = 1, 2, 5$ and three times at $h = 10$, where $h = C_{\text{test}}$ counts policy timesteps. Each evaluation contains 1,150 episodes; mean ± sample SD is reported only for $h = 10$. Final-checkpoint stochastic full-chunk RMSE follows Appendix B.4. N1–N5 are defined in Table 7.

				Single execution			$h = 10$ repeats			$h = 10$
D	Model	Seed	RMSE	$h = 1$	$h = 2$	$h = 5$	1	2	3	Mean ± SD
50	N1	0	0.20765	10.61	10.00	13.22	14.52	13.91	14.00	14.14 ± 0.33
		1	0.20301	12.52	15.57	18.78	18.17	18.96	17.30	18.14 ± 0.83
	N2	0	0.19514	14.26	16.09	21.65	20.96	20.96	22.26	21.39 ± 0.75
		1	0.19682	10.96	13.57	18.09	20.35	17.65	18.78	18.93 ± 1.35
	N3	0	0.19532	14.70	17.57	22.26	22.70	22.00	21.22	21.97 ± 0.74
		1	0.19411	14.35	15.13	20.61	23.39	22.70	22.09	22.72 ± 0.65
	N4	0	0.19279	16.61	18.26	25.74	27.13	28.17	26.09	27.13 ± 1.04
		1	0.19518	16.43	15.04	23.48	24.17	23.74	24.17	24.03 ± 0.25
	N5	0	0.18934	16.70	17.39	23.57	22.52	23.22	23.48	23.07 ± 0.49
		1	0.18610	14.78	16.09	19.39	21.48	22.00	21.48	21.65 ± 0.30
100	N1	0	0.17477	19.83	26.70	30.35	29.48	29.57	30.87	29.97 ± 0.78
		1	0.17701	21.65	22.52	27.13	27.30	26.87	27.91	27.36 ± 0.52
	N2	0	0.17212	22.52	24.52	31.13	30.17	31.74	31.65	31.19 ± 0.88
		1	0.16947	21.65	24.96	32.35	34.43	31.83	30.87	32.38 ± 1.85
	N3	0	0.16618	24.70	25.91	35.48	37.30	34.87	35.30	35.83 ± 1.30
		1	0.16491	21.74	25.91	33.74	31.91	34.00	33.83	33.25 ± 1.16
	N4	0	0.16913	21.48	25.13	34.61	36.09	36.35	36.96	36.46 ± 0.45
		1	0.17028	17.39	25.04	34.61	36.43	35.74	36.70	36.29 ± 0.49
	N5	0	0.17117	20.09	25.22	33.30	33.48	33.30	34.35	33.71 ± 0.56
		1	0.17098	19.91	24.43	34.61	33.13	34.35	36.35	34.61 ± 1.62
300	N1	0	0.15265	22.96	30.09	40.52	41.04	40.87	41.22	41.04 ± 0.17
		1	0.15193	30.96	39.91	42.17	41.13	41.74	41.22	41.36 ± 0.33
	N2	0	0.13680	29.65	35.22	46.35	48.00	46.17	48.26	47.48 ± 1.14
		1	0.13544	29.57	36.43	46.35	46.52	46.35	45.65	46.17 ± 0.46
	N3	0	0.13171	27.91	34.96	47.04	46.70	50.00	47.65	48.12 ± 1.70
		1	0.13310	27.83	35.74	48.17	48.87	49.13	47.13	48.38 ± 1.09
	N4	0	0.13550	29.91	39.30	49.91	50.78	51.22	51.74	51.25 ± 0.48
		1	0.13487	29.30	38.17	49.91	50.52	51.91	51.74	51.39 ± 0.76
	N5	0	0.14404	30.17	37.13	48.35	50.00	49.39	50.09	49.83 ± 0.38
		1	0.14376	28.35	34.78	48.61	49.83	49.83	49.91	49.86 ± 0.05
1000	N1	0	0.12933	34.17	40.52	48.96	48.96	49.91	48.43	49.10 ± 0.75
		1	0.12888	35.83	42.43	49.74	50.26	49.91	49.39	49.86 ± 0.44
	N2	0	0.10873	37.65	49.22	56.43	55.83	56.35	54.87	55.68 ± 0.75
		1	0.10998	34.43	45.57	52.70	52.87	52.78	50.96	52.20 ± 1.08
	N3	0	0.09937	38.17	46.78	57.91	59.39	58.26	57.74	58.46 ± 0.84
		1	0.10051	37.57	47.74	57.22	58.35	56.96	58.09	57.80 ± 0.74
	N4	0	0.10117	38.78	47.57	57.04	56.78	58.43	57.74	57.65 ± 0.83
		1	0.10044	37.65	46.70	57.74	56.78	56.26	57.04	56.70 ± 0.40
	N5	0	0.10853	38.87	46.78	57.74	56.43	56.70	55.48	56.20 ± 0.64
		1	0.10806	37.57	47.04	57.48	56.87	56.52	55.91	56.43 ± 0.48
2900	N1	0	0.10905	41.39	47.57	53.83	55.04	55.48	52.70	54.41 ± 1.50
		1	0.10851	40.61	49.04	55.22	54.09	56.00	53.91	54.67 ± 1.16
	N2	0	0.09083	43.74	52.87	60.61	58.96	58.17	60.09	59.07 ± 0.96
		1	0.09069	42.61	53.22	58.70	59.83	59.74	59.91	59.83 ± 0.09
	N3	0	0.08363	40.43	53.74	59.30	61.04	56.96	60.00	59.33 ± 2.12
		1	0.08447	42.00	54.00	59.13	61.74	62.00	59.83	61.19 ± 1.19
	N4	0	0.08370	36.61	49.65	57.83	57.65	60.52	58.26	58.81 ± 1.51
		1	0.08436	38.87	52.00	59.74	58.09	57.83	57.48	57.80 ± 0.31
	N5	0	0.08662	41.04	50.87	58.70	59.48	60.43	59.04	59.65 ± 0.71
		1	0.08615	40.43	48.61	59.39	58.61	60.87	58.35	59.28 ± 1.39

C.2 RMSE AT 10% TRAINING INTERVALS

The final checkpoints lie in the flatter part of the recorded training curves. Across all 50 runs, the relative RMSE reduction over the last 20% of the scheduled budget (80% to 100%) has a median of 0.53% and a maximum of 2.21%. Over the final 10% (90% to 100%), the median is 0.08%.

Table 11: RoboCasa holdout stochastic full-chunk RMSE at 10% training intervals, seed 0, using the metric in Appendix B.4. Progress is optimizer steps divided by the scheduled budget: 5k/10k/30k/100k/280k for $D = 50/100/300/1000/2900$. Entries use exact logged steps, without interpolation.

D	Params	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
50	44.3M	0.41821	0.31303	0.27148	0.25973	0.23128	0.21949	0.21252	0.21222	0.20983	0.20765
	85.3M	0.41246	0.35694	0.24800	0.22547	0.21357	0.20352	0.20087	0.19598	0.19501	0.19514
	218.8M	0.40335	0.34023	0.27017	0.23318	0.21947	0.20875	0.19826	0.19802	0.19648	0.19532
	648.9M	0.33931	0.23841	0.22386	0.20937	0.20262	0.19702	0.19359	0.19386	0.19313	0.19279
	1.49B	0.27697	0.21628	0.20776	0.19756	0.19285	0.19263	0.19124	0.19109	0.19044	0.18934
100	44.3M	0.39791	0.27210	0.22569	0.20335	0.19464	0.18634	0.18098	0.17686	0.17595	0.17477
	85.3M	0.30601	0.22293	0.20872	0.19259	0.18493	0.17955	0.17515	0.17277	0.17237	0.17212
	218.8M	0.26028	0.21184	0.19617	0.18517	0.17863	0.17552	0.16948	0.16763	0.16724	0.16618
	648.9M	0.23599	0.20562	0.19045	0.18188	0.17736	0.17374	0.17132	0.16982	0.16926	0.16913
	1.49B	0.22077	0.18924	0.18223	0.17932	0.17597	0.17370	0.17270	0.17192	0.17148	0.17117
300	44.3M	0.21602	0.18548	0.16730	0.16072	0.15789	0.15515	0.15375	0.15248	0.15204	0.15265
	85.3M	0.20004	0.17128	0.15749	0.14874	0.14546	0.14227	0.13900	0.13765	0.13699	0.13680
	218.8M	0.19457	0.16529	0.15336	0.14426	0.13967	0.13615	0.13402	0.13292	0.13191	0.13171
	648.9M	0.18732	0.16346	0.15164	0.14691	0.14215	0.13926	0.13726	0.13627	0.13541	0.13550
	1.49B	0.17897	0.16058	0.15385	0.15074	0.14796	0.14645	0.14523	0.14438	0.14394	0.14404
1000	44.3M	0.16689	0.14403	0.13747	0.13414	0.13272	0.13176	0.13019	0.12972	0.12950	0.12933
	85.3M	0.15205	0.13051	0.12170	0.11857	0.11469	0.11205	0.11029	0.10969	0.10879	0.10873
	218.8M	0.15245	0.12478	0.11399	0.10965	0.10525	0.10286	0.10128	0.09993	0.09921	0.09937
	648.9M	0.14708	0.12308	0.11645	0.10929	0.10659	0.10402	0.10254	0.10128	0.10092	0.10117
	1.49B	0.14283	0.12695	0.11988	0.11528	0.11266	0.11105	0.10942	0.10916	0.10836	0.10853
2900	44.3M	0.13699	0.12244	0.11685	0.11388	0.11233	0.11033	0.10989	0.10918	0.10921	0.10905
	85.3M	0.12267	0.10732	0.10211	0.09790	0.09498	0.09335	0.09205	0.09155	0.09076	0.09083
	218.8M	0.11636	0.09999	0.09447	0.08980	0.08727	0.08615	0.08465	0.08417	0.08348	0.08363
	648.9M	0.11615	0.10074	0.09301	0.09021	0.08774	0.08586	0.08458	0.08391	0.08351	0.08370
	1.49B	0.11378	0.10132	0.09558	0.09188	0.09055	0.08857	0.08758	0.08646	0.08636	0.08662

Table 12: RoboCasa holdout stochastic full-chunk RMSE at 10% training intervals, seed 1, using the metric in Appendix B.4. Progress is optimizer steps divided by the scheduled budget: 5k/10k/30k/100k/280k for $D = 50/100/300/1000/2900$. Entries use exact logged steps, without interpolation.

D	Params	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
50	44.3M	0.41265	0.36703	0.30551	0.24915	0.24223	0.22526	0.21346	0.20761	0.20684	0.20301
	85.3M	0.41460	0.31348	0.25889	0.23951	0.22562	0.21208	0.20414	0.19972	0.19819	0.19682
	218.8M	0.39254	0.28892	0.23680	0.21865	0.20694	0.20339	0.19930	0.19517	0.19357	0.19411
	648.9M	0.37310	0.25907	0.22671	0.21701	0.20702	0.19907	0.20138	0.19804	0.19506	0.19518
	1.49B	0.26374	0.21731	0.20156	0.19317	0.19170	0.18972	0.18823	0.18731	0.18720	0.18610
100	44.3M	0.28741	0.23559	0.20988	0.19770	0.18795	0.18145	0.17921	0.17774	0.17699	0.17701
	85.3M	0.31299	0.21829	0.19867	0.18964	0.18305	0.17601	0.17273	0.17065	0.16928	0.16947
	218.8M	0.25743	0.21477	0.19513	0.18202	0.17580	0.16951	0.16723	0.16519	0.16534	0.16491
	648.9M	0.22760	0.19918	0.18997	0.18003	0.17849	0.17363	0.17196	0.17081	0.16950	0.17028
	1.49B	0.21579	0.18919	0.18121	0.18160	0.17432	0.17193	0.17251	0.17158	0.17115	0.17098
300	44.3M	0.21696	0.17824	0.16642	0.15908	0.15538	0.15390	0.15282	0.15188	0.15173	0.15193
	85.3M	0.20141	0.16884	0.15676	0.14888	0.14439	0.14156	0.13848	0.13707	0.13578	0.13544
	218.8M	0.19170	0.16711	0.15352	0.14558	0.14062	0.13714	0.13610	0.13394	0.13323	0.13310
	648.9M	0.19037	0.16538	0.15334	0.14652	0.14100	0.13936	0.13678	0.13567	0.13489	0.13487
	1.49B	0.17863	0.15991	0.15371	0.14901	0.14836	0.14607	0.14560	0.14432	0.14388	0.14376
1000	44.3M	0.16650	0.14604	0.13806	0.13510	0.13208	0.13021	0.12925	0.12887	0.12872	0.12888
	85.3M	0.15905	0.13433	0.12191	0.11875	0.11418	0.11301	0.11119	0.11089	0.11019	0.10998
	218.8M	0.14750	0.12548	0.11522	0.11001	0.10641	0.10479	0.10232	0.10158	0.10057	0.10051
	648.9M	0.14540	0.12131	0.11420	0.11018	0.10559	0.10327	0.10179	0.10062	0.10044	0.10044
	1.49B	0.14269	0.12519	0.11899	0.11489	0.11216	0.11054	0.10963	0.10837	0.10810	0.10806
2900	44.3M	0.13605	0.12162	0.11687	0.11390	0.11160	0.10998	0.10947	0.10889	0.10885	0.10851
	85.3M	0.12315	0.10721	0.10169	0.09778	0.09553	0.09331	0.09166	0.09082	0.09084	0.09069
	218.8M	0.11918	0.10115	0.09508	0.09166	0.08923	0.08669	0.08583	0.08471	0.08454	0.08447
	648.9M	0.11520	0.10078	0.09431	0.08996	0.08768	0.08639	0.08498	0.08463	0.08463	0.08436
	1.49B	0.11576	0.09991	0.09588	0.09259	0.08945	0.08781	0.08729	0.08659	0.08630	0.08615

C.3 Success Rates for Individual Tasks

Table 13: Task-level RoboCasa SR (%) for the 218.8M policy. Each data-scale entry averages three executions of 50 episodes per task, with training seeds separate. The final column records successes in the selected single execution: seed 1, $D = 2900$, repeat 02 (713/1,150 overall).

	Training seed 0: D					Training seed 1: D					Selected run
Task	50	100	300	1000	2900	50	100	300	1000	2900	Successes
CloseDoubleDoor	23.33	67.33	88.00	91.33	94.67	36.67	68.67	92.67	90.67	91.33	45/50
CloseDrawer	88.00	98.67	100.00	100.00	98.67	87.33	96.00	99.33	100.00	97.33	48/50
CloseSingleDoor	57.33	67.33	82.67	83.33	87.33	50.67	76.00	76.00	84.00	84.00	45/50
CoffeePressButton	44.67	56.67	78.00	91.33	94.67	38.00	51.33	84.67	86.00	94.67	49/50
CoffeeServeMug	12.67	20.00	54.00	62.00	64.67	6.00	24.00	61.33	67.33	69.33	38/50
CoffeeSetupMug	1.33	3.33	19.33	28.00	32.67	0.00	8.67	22.67	32.67	42.00	23/50
OpenDrawer	24.00	52.67	74.00	92.00	79.33	22.67	57.33	79.33	86.00	86.67	43/50
OpenSingleDoor	28.67	32.67	46.00	61.33	62.00	21.33	26.67	50.67	57.33	66.00	30/50
PnPCabToCounter	0.67	4.00	11.33	12.00	14.67	1.33	7.33	14.67	14.00	16.67	9/50
PnPCounterToCab	0.00	10.67	32.67	28.67	42.67	3.33	5.33	26.67	39.33	42.67	23/50
PnPCounterToMicrowave	0.67	4.00	3.33	17.33	20.00	0.00	4.67	8.67	14.67	32.00	16/50
PnPCounterToSink	1.33	6.67	16.67	26.67	28.00	3.33	5.33	6.00	26.00	32.00	18/50
PnPCounterToStove	0.67	2.00	4.00	21.33	22.00	0.00	0.67	1.33	20.67	32.00	16/50
PnPMicrowaveToCounter	0.67	3.33	10.67	18.67	18.00	0.00	0.67	9.33	21.33	18.00	10/50
PnPSinkToCounter	1.33	4.00	22.67	47.33	52.67	4.67	8.00	14.00	34.00	52.00	25/50
PnPStoveToCounter	0.00	1.33	8.00	45.33	42.00	0.00	0.67	10.00	41.33	48.67	22/50
TurnOffMicrowave	39.33	66.00	93.33	98.67	99.33	63.33	65.33	95.33	94.00	100.00	50/50
TurnOffSinkFaucet	54.00	85.33	83.33	84.67	84.00	65.33	58.67	72.67	89.33	88.00	41/50
TurnOffStove	6.67	6.67	16.67	20.67	21.33	1.33	6.00	18.00	18.67	14.00	9/50
TurnOnMicrowave	32.67	84.67	86.67	90.00	95.33	42.00	47.33	89.33	93.33	92.00	46/50
TurnOnSinkFaucet	27.33	43.33	62.00	76.00	70.00	18.00	60.67	51.33	69.33	65.33	32/50
TurnOnStove	14.67	30.67	48.00	69.33	62.00	12.67	21.33	47.33	66.67	66.00	36/50
TurnSinkSpout	45.33	72.67	65.33	78.67	78.67	44.67	64.00	81.33	82.67	76.67	39/50
All tasks	21.97	35.83	48.12	58.46	59.33	22.72	33.25	48.38	57.80	61.19	713/1,150

C.4 LOCAL BC-TRANSFORMER COMPARISON

The local BC-Transformer comparison uses the same 23 tasks, $D = 300$ demonstrations, and final 1,150 evaluation episodes as WorldToken. It follows the official native training recipe: one training seed (123), batch size 16, 500,000 optimizer steps, and AdamW weight decay 0.01. It observes and replans after every action, whereas WorldToken observes every four control steps and executes four actions per query. The comparison therefore matches the data and evaluation episodes but not the optimizer settings, training compute, seed count, observation cadence, or action-execution cadence.

Table 14 lists all repeated evaluations underlying the 31.28% local reference in the main text.

Table 14: Local BC-Transformer trained with the official native recipe (Appendix C.4): training seed 123, $D = 300$, final 500k-step checkpoint. All three repeats use the same 1,150 episodes across 23 tasks. SR is in percent; SD is the sample standard deviation across repeats, in percentage points.

Policy	Training seed	Repeat 1	Repeat 2	Repeat 3	Mean $\pm$ SD
BC-Transformer	123	32.00	30.78	31.04	31.28 ± 0.64

C.5 COMPARISON WITH THE REPORTED $\pi_{0.5}$ RESULT

Table 15 gives the public comparison used for context in the main text. WorldToken uses a frozen CLIP text encoder and trains its visual and policy modules from scratch. The reported $\pi_{0.5}$ policy uses multi-source robot and Web pretraining. **Although our 23-task evaluation excludes `OpenDoubleDoor`, which is included in the 24-task benchmark underlying the reported $\pi_{0.5}$ result (Kim et al., 2026), WorldToken N2 at $D = 2900$ achieves over 80% mean SR across the other three tasks in RoboCasa's official "Opening and closing doors" family: `OpenSingleDoor`, `CloseSingleDoor`, and `CloseDoubleDoor` (training seed 0, averaged over three evaluations). WorldToken strictly follows the official RoboCasa evaluation protocol for all 23 evaluated tasks (Nasiriany et al., 2024).**

Table 15: WorldToken and the reported $\pi_{0.5}$ result on RoboCasa Kitchen. Training data counts are generated demonstrations per task. The WorldToken result averages two training seeds, each with three evaluations. The $\pi_{0.5}$ result is reported by Kim et al. (2026).

Policy	Training data	Parameters	Tasks	SR (%)
WorldToken	2,900	85.3M	23	59.4
$\pi_{0.5}$	300	$\sim$3.35B	24	62.1

D TOKEN INTERFACES, POLICY PERFORMANCE, AND COMPUTATIONAL COST

D.1 SINGLE- AND MULTI-TOKEN INTERFACES

The 85.3M-parameter reference passes one learned world token per timestep. For $K = 4$, we concatenate the same four internal readout states, jointly project them to $4d$ dimensions, and split the result into four d-dimensional tokens. The number of internal readouts remains four. For $K = 50$, we pass all 48 post-fusion CNN feature tokens, followed by the proprioception and task tokens. These features are already fused, rather than raw pixels. In every variant, tokens are grouped by timestep, and the decoder reads the final contextualized token of the current timestep. The history always spans ten policy timesteps, giving $N = 10K$ temporal positions.

The temporal backbone applies a standard lower-triangular causal mask over the flattened token sequence, including within each timestep. RoPE assigns consecutive positions to all CK tokens, so tokens from the same timestep have distinct positions. The final token can attend to all earlier tokens in its timestep and all earlier timesteps. For $K = 4$, this final token is the last block of the joint projection and has no predefined modality. For $K = 50$, it is the contextualized task-language token.

At each D, all variants use the same training demonstrations, holdout demonstrations, and final 1,150 evaluation episodes. The added variants use only training seed 0. Because parameter count and temporal compute are not jointly matched, the results describe these token interfaces; they do not isolate the causal effect of token count or establish an optimal interface.

D.2 Policy Performance across Data Scales

Table 16 reports results across the five data scales. Relative to $K = 1$, $K = 50$ reduces full-chunk RMSE by 1.7–4.2%. Its SR gain is larger at $D = 50$ and 100, at 5.10 and 6.38 percentage points, respectively, and is 2.09, 2.06, and 1.59 points at $D = 300$, 1000, and 2900. These differences use the exact success counts before rounding the displayed means.

Increasing the learned interface to $K = 4$ gives less uniform results: RMSE improves at two of five data scales, and the SR change ranges from -1.13 to $+6.96$ points. The largest gain again occurs at $D = 100$. These seed-0 results show that both the interface construction and data scale matter; increasing token count alone does not give a uniform gain. Panel (b) reports full-window temporal-backbone FLOPs. Appendix D.4 separately reports per-query costs with cached history, including the encoder and action decoder.

Table 16: Token-count experiments at $C = 10$, training seed 0. (a) Final-checkpoint results for all 15 (D, K) settings. Stochastic full-chunk RMSE follows Appendix B.4. SR entries give three evaluation percentages (1,150 episodes each) and their mean $\pm$ sample SD. The $K = 1, 4, 50$ interfaces are defined in Appendix D. (b) Temporal dimensions and analytic full-prefix FLOPs (Equation 19), excluding encoder and decoder. Relative FLOPs use the 85.3M, $K = 1$ reference; parameter count and compute are not jointly matched.

(a) Complete token-interface results

D	K	Params	N	Rel. FLOPs	RMSE	SR: repeats 1 / 2 / 3	SR: mean $\pm$ SD
50	1	85.3M	10	1.00×	0.19514	20.96/20.96/22.26	21.39 ± 0.75
	4	92.4M	40	4.03×	0.20163	20.00/19.48/21.30	20.26 ± 0.94
	50	83.0M	500	55.31×	0.19086	25.83/27.04/26.61	26.49 ± 0.62
100	1	85.3M	10	1.00×	0.17212	30.17/31.74/31.65	31.19 ± 0.88
	4	92.4M	40	4.03×	0.16586	38.17/38.00/38.26	38.14 ± 0.13
	50	83.0M	500	55.31×	0.16497	37.57/38.09/37.04	37.57 ± 0.52
300	1	85.3M	10	1.00×	0.13680	48.00/46.17/48.26	47.48 ± 1.14
	4	92.4M	40	4.03×	0.13826	49.22/47.83/49.22	48.75 ± 0.80
	50	83.0M	500	55.31×	0.13230	49.74/49.13/49.83	49.57 ± 0.38
1000	1	85.3M	10	1.00×	0.10873	55.83/56.35/54.87	55.68 ± 0.75
	4	92.4M	40	4.03×	0.10972	55.83/56.00/54.96	55.59 ± 0.56
	50	83.0M	500	55.31×	0.10686	57.57/58.09/57.57	57.74 ± 0.30
2900	1	85.3M	10	1.00×	0.09083	58.96/58.17/60.09	59.07 ± 0.96
	4	92.4M	40	4.03×	0.08997	60.00/59.83/60.00	59.94 ± 0.10
	50	83.0M	500	55.31×	0.08851	60.26/59.83/61.91	60.67 ± 1.10

(b) Temporal dimensions and full-prefix FLOPs

Configuration	d	f	L	K	N	Temporal FLOPs
85.3M, one token	768	2048	4	1	10	0.567G (1.00×)
92.4M, four tokens	768	2048	4	4	40	2.285G (4.03×)
83.0M, fifty tokens	768	2048	4	50	500	31.384G (55.31×)

D.3 Why Short-Context Cost Is Nearly Linear in Token Count

We first count temporal-backbone computation for a full window, without KV caching. For temporal width d, feed-forward width f, depth L, and sequence length N, the leading matrix-multiplication count is

$$F(N) = L(8d^2 + 6df)N + 4LdN^2, \tag{19}$$

using two FLOPs per multiply–accumulate. The $d = 768$, $f = 2048$, $L = 4$ backbone gives $56{,}623{,}104N + 12{,}288N^2$, with $N = 10K$ at $C = 10$. Table 16 lists the resulting counts. They exclude the within-timestep encoder, action decoder, and temporal wrapper's output projection. They also assume full rectangular attention products; causal kernels may skip masked work. These FLOPs are not end-to-end latency measurements. The cached operator counts in Appendix D.4 include the output projection and measure the incremental work per policy query.

The linear term includes the attention projections and feed-forward layers. The quadratic term comes from attention between token positions. The two terms are equal when $N = 2d + 1.5f = 4{,}608$ for this backbone. At $C = 10$, the $K = 1, 4, 50$ interfaces contain $N = 10, 40, 500$ tokens. The linear term therefore dominates at these lengths, giving the approximately 4.03- and 55.31-fold costs in Table 16.

D.4 Module Costs with Cached History

We next count incremental computation for the N2 $K = 1, 4, 50$ interfaces with a growing KV cache and no eviction. A context of C observations, including the current one, contains $N = KC$ tokens. Each query encodes only the current observation and processes its K new tokens alongside $(C-1)K$ cached tokens. At $C = 1$, there is no prior history. This cached calculation differs from the RoboCasa success-rate evaluations, which recompute the full window.

Table 17: Cached computation in GFLOPs per timestamp, where one timestamp denotes one policy query. (a) Baseline with one observation, $C = 1$. Fusion/projections include encoder readouts, and encoder total is a subtotal. Action-head costs include all 20 diffusion steps; totals are computed before rounding. (b) Additional cost per query for 100 more retained observations. Relative slopes compare the cost of added history, not total policy costs.

(a) One-observation baseline: $C = 1$

K	CNN	Fusion + projections	Encoder total	Temporal backbone	Action head	Total
1	19.1103	2.1189	21.2292	0.0578	0.4502	21.7372
4	19.1103	2.1330	21.2433	0.2314	0.4502	21.9250
50	19.1103	1.9606	21.0709	2.9209	0.4502	24.4420

(b) Cost of 100 additional historical observations

K	$\Delta_{100,K}$ (GFLOPs/timestamp)	Slope relative to $K = 1$
1	0.0012288	1×
4	0.0196608	16×
50	3.0720000	2500×

The three CNNs dominate encoder computation. Fusion/projections cover the remaining encoder operations and readouts; the temporal and action-head architectures are shared across variants. At $C = 10$, total cached computation is 21.7373, 21.9268, and 24.7185 GFLOPs/query for $K = 1, 4, 50$, respectively. Thus, including all modules, $K = 50$ costs $1.14\times$ as much as $K = 1$ at this history length.

D.5 Computational Growth with History Length

The encoder and action sampler have fixed input shapes; only temporal computation grows with C. Per 100 additional observations,

$$F_K(C) = F_K(1) + \frac{C-1}{100}\Delta_{100,K}, \tag{20}$$

where F_K is total GFLOPs/query and Table 17(b) gives $\Delta_{100,K}$. With four temporal layers of width 768, the $K \times CK$ attention products make the cost of added history proportional to K^2. For equal increases in observation count, the added costs have ratio $1 : 16 : 2500$. This ratio describes the growth with history, not the total policy cost.

FLOPs counting. Counts use archived architectures with random weights, synthetic batch-one inputs, CPU FP32, and no gradients in evaluation mode. Inputs are three 128×128 RGB views, 16-D proprioception, and a precomputed 768-D language embedding; outputs are ten-step, 12-D action chunks. `FlopCounterMode` counts convolution and dense matrix products at two FLOPs per multiply–accumulate, including temporal projections and all 20 denoising steps. Normalization, softmax, pooling, other elementwise operations, and data movement are excluded.

E History Truncation and Short-Context Training on RoboCasa

E.1 Inference-Time History Truncation

For each scaling checkpoint, we vary only $C_{\text{test}} \in \{1, 2, 5, 10\}$, keeping the initial conditions, environment seeds, rollout seed, and execution settings fixed. Each shorter-history setting uses one evaluation; the $C_{\text{test}} = 10$ reference is the mean of three evaluations.

Table 10 in Appendix C.1 reports the complete history-truncation results for all 50 scaling policies.

E.2 Training and Evaluation with Matched Context Lengths

We also train separate 218.8M policies at $D = 300$ with matching training and evaluation contexts, $C \in \{1, 2, 5, 10\}$. Each setting uses two training seeds and three evaluations per final checkpoint.

The global batch sizes are $1920, 960, 384, 192$ windows for $C = 1, 2, 5, 10$, respectively, with 30k optimizer steps. Multiplying batch size by C gives 1,920 observation-query positions per update and 57.6M over training. These budgets are matched before excluding queries without all ten action targets. All variants sample crops from the same training demonstrations, using their own context length C. The exact queries and the number of valid action chunks after masking can therefore differ across variants.

Table 18: D300 matched-context training with N3 (218.8M), $C_{\text{train}} = C_{\text{test}} = C$. Each final 30k-step checkpoint is evaluated three times on 1,150 episodes. Stochastic full-chunk RMSE uses the context-specific crops in Appendix B.4. SR entries give the repeated evaluation percentages and their mean ± sample SD. $C = 10$ is the scaling reference.

			SR (%): repeats			SR (%)
C	Seed	RMSE	1	2	3	Mean ± SD
1	0	0.150260	45.65	47.65	46.96	46.75 ± 1.02
	1	0.147767	48.17	46.00	48.87	47.68 ± 1.50
2	0	0.149383	47.83	46.52	46.26	46.87 ± 0.84
	1	0.147528	47.74	50.09	47.74	48.52 ± 1.36
5	0	0.137165	51.74	51.22	51.48	51.48 ± 0.26
	1	0.139603	50.87	49.04	50.09	50.00 ± 0.92
10	0	0.131710	46.70	50.00	47.65	48.12 ± 1.70
	1	0.133098	48.87	49.13	47.13	48.38 ± 1.09

The truncation experiment measures a fixed policy's reliance on history; the separately trained policies show how training with shorter contexts changes that reliance.

Separately trained short-context policies recover much of the success lost when long-context checkpoints are truncated. Across the two training seeds, mean SR is 47.22%, 47.70%, 50.74%, and 48.25% at $C = 1, 2, 5, 10$, respectively. The $C = 5$ policies have the highest observed SR, while reported RMSE decreases with C under the context-specific crop protocol in Appendix B.4.

F History Use and Extended Execution in RMBench Blocks Ranking

F.1 Task and Demonstrations

Blocks Ranking uses three colored blocks in left, middle, and right slots, with target order $(1, 2, 3)$. The reference procedure begins with a button press, then follows the positional swaps in Eq. (7), pressing after each swap and stopping at the target. The five non-target initial permutations require one through five swaps.

Initial training uses 50 `demo_clean` demonstrations for each of `battery_try`, `blocks_ranking_try`, `cover_blocks`, `observe_and_pickup`, `press_button`, `put_back_block`, `rearrange_blocks`, `swap_T`, and `swap_blocks`: 450 demonstrations in total, with no initial holdout.

We continue from the seed-1 nine-task checkpoint at step 5,000 using 45 Blocks Ranking demonstrations. Episode IDs 6, 15, 19, 25, and 46 are held out from this continuation (split seed 4). All five were seen during initial training, so their offline loss monitors further training on previously seen demonstrations.

F.2 Detailed Experimental Setup

Model architecture. The RMBench model has 54,300,238 trainable parameters and a 768-D world token. Its within-timestep encoder has two 768-D layers, six attention heads, and SwiGLU FFN width 3,072. The temporal backbone has four 768-D layers, six heads, and FFN width 2,048. The diffusion action decoder has two 192-D layers, six heads, and FFN width 768. Four readout tokens pass through both fusion layers together with the observation tokens, using the asymmetric mask and shared FFNs in Appendix A.1 (Eq. (10)).

The encoder shares a 20×20 patch projection across the three native 240×320 RGB camera fields `head_camera`, `left_camera`, and `right_camera`. It also receives the 14-D `joint_action/vector` proprioceptive field and a nine-way task one-hot condition. Each action chunk contains $H = 8$ absolute 14-D joint-position commands. Reconstruction, predictive auxiliary losses, and action-decoder cross-attention to observation tokens are disabled.

Sequence construction and supervision. Initial training uses a maximum history of $C_{\text{train}} = 288$ and samples one observation sequence per episode in each dataset pass. Observations are four raw frames apart. For an episode of length L, let $j_{\max} = L - H - 1$. If $j_{\max} \leq 4(C_{\text{train}} - 1)$, we uniformly sample a phase from $\{0, 1, 2, 3\}$ and take its complete subsequence through $j_{\max}$. Otherwise, we uniformly sample a legal start and take a capped window.

Continuation increases the cap to $C_{\text{train}} = 608$ and constructs four sequences per episode, starting at raw frames 0, 1, 2, and 3 with stride four. All 50 Blocks Ranking demonstrations fit within this cap: the maximum length is $L = 2,436$, giving at most 607 queries with complete future action chunks per phase. Sequences are shuffled during training. The five continuation holdout demonstrations use the same four phases and are evaluated with the ordinary diffusion objective.

Every sampled query with a complete future chunk contributes loss and sees only the causal prefix of its sampled sequence. Each optimizer update averages eight single-window microbatches; the batch count therefore refers to windows, not individual action queries.

AdamW uses separate parameter groups for the within-timestep encoder, temporal backbone, and action decoder. Continuation preserves optimizer and action-normalization state and extends the cosine schedule to step 5,500 without restarting warmup.

We construct a fixed set of 100 initial conditions using RMBench's `demo_clean` setup, accepting candidates when the scripted expert passes both planning and official success checks. The first 100 candidate seeds, 100000–100099, all passed without rejection.

All history interventions keep the same final checkpoint, the initial conditions, success predicate, and horizon fixed to measure the policy's use of visible history.

Table 19: Recorded Blocks Ranking continuation and action-generation settings. The same step-5,500 checkpoint is used for every visible-history intervention.

Setting	Value
Initial / continuation budget	5,000 / 500 optimizer steps; continuation uses 45 training / 5 continuation-holdout demonstrations
Training context cap	Initial $C_{\text{train}} = 288$; continuation $C_{\text{train}} = 608$
Microbatch / accumulation	1 / 8 steps
Optimizer	AdamW, $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, zero weight decay
Gradient clipping / precision	Unit gradient norm / BF16
Reference learning rates	Within-timestep encoder and temporal backbone: 4.25×10^{-4}; action decoder: 3.0×10^{-4}
Schedule	Initial 250-step warmup, then cosine decay to 10% of reference; continuation restores optimizer/scheduler state at global step 5,000
Action normalization	Inherited per-dimension min–max statistics from the nine-task training split; nondegenerate dimensions mapped to $[-1, 1]$
Diffusion head	Epsilon prediction; 20 squared-cosine DDPM steps; fixed-small variance; denoised clipping to $[-1, 1]$
Rollout sampling	Stochastic ancestral reverse steps; four executed actions per replanning query

Action alignment. Let j_t be the raw data-frame index of policy query t. The observation is `vector[`j_t`]`, and the target contains the next H absolute 14-D joint-position commands, `vector[`$j_t + 1$`:`$j_t + H + 1$`]`. The first target is the first command applied after the current observation. The one-index offset comes from the dataset's storage convention and adds no open-loop delay. We sample only queries with complete future target chunks, so training uses no padded terminal targets.

F.3 Task-Specific Training Objective

During Blocks Ranking continuation, a training-only loss modification favors a small additional offset along the expert's local joint-space descent direction, motivated by the button press. Orthogonal components and targets without a usable descent direction retain the ordinary diffusion loss.

Let $i \in \{0, \dots, H-1\}$ be an action position, $r = j_t + i + 1$ its target raw frame, $q_r = \texttt{vector}[r]$, and z_r^{EE} the recorded left-end-effector height. Define $\Delta z_i = z_r^{\text{EE}} - z_{r-1}^{\text{EE}}$ and the 14-D direction $d_i = (q_r^{1:6} - q_{r-1}^{1:6}, 0, \dots, 0)$, retaining only the six left-arm joints. A descent candidate requires $\Delta z_i < -0.2$ mm and $\|d_i\|_2 > 10^{-6}$. Both increments use adjacent raw frames, rather than the stride-four observations. The recorded end-effector displacement selects frames and scales the joint-space direction. The loss applies no forward kinematics to predicted actions, and end-effector pose is not a policy input.

For dimension a, action normalization is $x_a = (q_a - \mu_a)/s_a$, with $\mu_a = (q_a^{\max} + q_a^{\min})/2$ and $s_a = (q_a^{\max} - q_a^{\min})/2$ from initial nine-task training. A half-range below 10^{-4} is replaced by $s_a = 1$. Directions use only this linear scale, without subtracting μ:

$$v_i = d_i \oslash s, \qquad u_i = \frac{v_i}{\|v_i\|_2}, \qquad c_i = \left\langle \frac{0.004}{-\Delta z_i} v_i, u_i \right\rangle, \qquad p_i = \frac{1.5}{4} c_i. \tag{21}$$

Here $\oslash$ denotes componentwise division, and distances used in c_i are in meters. A direction is usable when $\|v_i\|_2 > 10^{-8}$ and $c_i > 0$. Otherwise, the original loss is retained; we set $u_i = 0$ when $v_i = 0$.

Let $k \in \{0, \dots, 19\}$ denote the diffusion step, sampled once per query chunk, and let $x_i^{(k)}$ be its noisy normalized target. The loss uses the *unclipped* reconstruction

$$\hat{x}_{0,i} = \frac{x_i^{(k)} - \sqrt{1 - \bar{\alpha}_k}\, \hat{\epsilon}_i}{\sqrt{\bar{\alpha}_k}}, \qquad e_i = \langle \hat{x}_{0,i} - x_{0,i}, u_i \rangle. \tag{22}$$

Clipping is used during sampling (Table 19), but not for this training reconstruction. We replace the DDPM prediction-loss component parallel to u_i with

$$\ell_{\parallel}(e_i) = 3\kappa_k \begin{cases} 16(p_i - e_i)^2, & e_i < p_i, \\ 0, & p_i \le e_i \le c_i, \\ (e_i - c_i)^2, & e_i > c_i, \end{cases} \tag{23}$$

where $\kappa_k = \bar{\alpha}_k / \max(1 - \bar{\alpha}_k, 10^{-8})$ maps the denoised-action distance to epsilon-prediction space. Writing $\eta_i = \hat{\epsilon}_i - \epsilon_i$ and $m_i \in \{0, 1\}$ for the usable-direction mask, the loss for one query is

$$\mathcal{L}_{\text{query}} = \frac{1}{14H} \sum_{i=0}^{H-1} \left[\|\eta_i\|_2^2 + m_i \left\{ \ell_{\parallel}(e_i) - \langle \eta_i, u_i \rangle^2 \right\} \right]. \tag{24}$$

The factors 3 and 16 apply only to the replaced parallel epsilon-error component. The preferred 1.5–4 mm range scales the expert's local joint-space direction as above, using no button-contact or press-stage label. We average query losses within each window, then average the eight microbatch window losses per optimizer update.

The standard-loss and modified-loss continuations share the nine-task step-5,000 checkpoint, 45/5 split, 500 additional optimizer steps, architecture, and evaluation settings. They use different recorded code revisions, so the comparison does not isolate the effect of the objective alone. All history interventions use the same modified-loss checkpoint across C.

F.4 Comparison of Training Objectives

The standard-loss and modified-loss policies succeed in 13/100 and 95/100 evaluation episodes, respectively. Both frequently reach the evaluator's target block geometry, but success also requires the open-gripper and button conditions. The matched-seed example shows how a millimeter-scale difference in button position can change the binary outcome. As detailed in Appendix F.3, the full success-rate gap compares two training procedures whose differences are not limited to the loss.

Diagnostic definitions and records. Target geometry means the evaluator's block-order and pairwise-position conditions, reached at any check regardless of the gripper and button conditions. For each failure that reaches this geometry, we record the minimum button position while it holds. There are 86 such standard-loss failures and three modified-loss failures. Button positions are in mm, with a -5 mm threshold.

Standard-loss summaries and paired outcome counts come from saved evaluation records and were not recomputed from raw rollouts. Modified-loss summaries and the matched-seed example use retained rollout and diagnostic records. Paired outcome counts compare separate stochastic rollouts. The last row uses seed 100084 and a separate standard-loss diagnostic execution.

F.5 History Intervention and Success Criteria

Our standard 100-episode Blocks Ranking evaluations strictly follow the official RMBench rollout protocol (Chen et al., 2026), using `demo_clean`, expert-validated initial conditions, the unmodified official success checks, and the official 3,500-action horizon. The official task initializer samples uniformly from the five non-target block permutations, corresponding to one through five swaps in the reference procedure. Our evaluation contains 24, 14, 28, 18, and 16 episodes in these categories, respectively: 76% require multiple reference swaps and 62% require at least three. The complete categories, episode counts, and stratified success rates are reported in Table 5 in the main text, and Figure 6 illustrates a five-swap episode.

Each history condition evaluates the same 100 initial conditions using the official success predicate and a 3,500-action (210-second) horizon. Queries are separated by four 0.06-second actions. We report a nominal history budget of $0.24C$ seconds: 145.92, 69.12, 30.72, 15.36, and 7.68 seconds for $C = 608, 288, 128, 64, 32$, respectively. This budget counts C observations sampled at 0.24-second intervals.

The official evaluator checks final block positions, an open right gripper, and the button press, without checking the reference swap sequence. We group episodes by the number of reference swaps required

Table 20: Standard-loss and modified-loss results at $C = 608$ on the same fixed evaluation set of 100 initial conditions. Panels report episode outcomes, paired outcome counts, and button-position diagnostics (mm). Definitions and source records are given in Appendix F.4.

	Standard loss	Modified loss
(a) Episode outcomes		
Evaluator successes	13	95
Evaluator failures	87	5
Target block geometry reached	99	98
Failures reaching target block geometry	86	3
Failures without target block geometry	1	2
(b) Paired outcome counts		
Both policies succeed	13	
Only standard-loss policy succeeds	0	
Only modified-loss policy succeeds	82	
Both policies fail	5	
(c) Button-position diagnostics (mm)		
Deepest button position among geometry-reaching failures	-4.65	-0.004
Median of per-episode deepest positions in those failures	-0.54	-0.004
Matched seed 100084: deepest position with target geometry	-1.485	-5.020

by their initial permutation, determined before rollout. The five groups contain 24, 14, 28, 18, and 16 episodes requiring one through five swaps, respectively. Each group has a single initial permutation, so differences between groups also reflect initial geometry.

Stable-order reconstruction and strict success. We retain all official evaluator labels and add a stricter behavioral classification. Strict success also requires the observed stable-order path through its first target state to match the reference, and final placement to satisfy Table 21. An evaluator-only success passes the official checks but fails at least one of these additional criteria.

Table 21: Criteria for stable block order and final placement. Stability criteria are shared with the extended-rollout sequence monitor. The final-placement cutoff is used only to classify strict behavioral success.

Quantity	Operational rule
Stable order	Same left-to-right block order for eight consecutive action-level records satisfying the motion, row/height, and right-gripper criteria below
Motion	Maximum block displacement between consecutive sampled positions ≤ 1.5 mm
Row and height	Each block within 6 cm of $y = -0.10$ m and $z \in [0.740, 0.785]$ m
Right gripper	Open-command value > 0.8
Final nominal-slot placement	For blocks $1, 2, 3$, $x^* = (0.04, 0.16, 0.28)$ m and $y^* = -0.10$ m; maximum absolute x/y error ≤ 4 cm at evaluator success
Sensitivity cutoffs	Recompute the final-placement classification at 2, 3, 4, and 5 cm; keep sequence and evaluator labels fixed

We reconstruct stable orders from the final position snapshot before each action's evaluator check. These action-level traces cover all 500 episodes. We visually reviewed flagged evaluator successes and two orderly control episodes.

Failures and transitions at the wrong phase. For each failed episode, we find the first action at which the target-geometry and open-gripper conditions hold simultaneously, if any. We then check for a button position below -5 mm during that action or any later action. We flag a *candidate button-only failure* if there is no such button event and, at that first action, the maximum absolute block-center x/y error is at most 4 cm and all block heights lie in $[0.740, 0.785]$ m. Separately, we count consecutive reference swaps from the initial state, stopping at the first mismatch.

A *wrong-phase candidate* is the first stable-order transition that departs from the reference path but matches a transition used at another reference phase. It must also meet two conditions: each block is within 4 cm in absolute x/y error of the nominal slot assigned by its observed left-to-right order; and the button is below -5 mm during the confirming action or a later action before the next stable transition. These candidate labels describe observed behavior, without identifying the policy's intended subgoal or the cause of an error.

F.6 History Length, Block Placement, and Swap Order

All 24 one-swap episodes succeed at every tested history length (Table 5). Shorter histories produce more failures in episodes requiring multiple swaps. Table 22 adds placement accuracy, progress along the reference sequence, and wrong-phase diagnostics for the same 100 evaluation episodes per C.

Placement error is the maximum absolute block-center x/y error at first evaluator success. In the table, "displaced" means an error above 4 cm, and "signature" denotes the wrong-phase flag. Panel (c) varies only the placement cutoff, keeping official labels fixed. Panel (d) counts path conditions among all 100 episodes per context length; panel (e) counts the listed conditions among evaluator failures. Panel (f) covers 266 cycles from the 94 strict successes at $C = 608$. Each cycle runs from the first press associated with one stable order to the first press associated with the next.

Table 22: Placement, sequence, and failure diagnostics for the 100 evaluation episodes at each history length. Aggregate and swap-stratified outcomes appear in Table 5; success criteria are defined in Appendix F.5. Panels (a)–(e) describe history length, placement, and behavioral outcomes. Panel (f) reports swap-and-press cycle durations at $C = 608$.

	$C = 608$	$C = 288$	$C = 128$	$C = 64$	$C = 32$
(a) History windows and strict-success durations					
Visible window (s)	145.92	69.12	30.72	15.36	7.68
Longest strict success (s)	142.62	142.50	142.26	59.94	31.98
(b) Block placement and swap order in successful episodes					
Strict-success placement: minimum (cm)	0.36	0.36	0.36	0.45	0.25
Strict-success placement: median (cm)	1.63	1.66	0.84	0.62	0.53
Strict-success placement: maximum (cm)	2.39	2.23	1.85	2.13	0.84
Evaluator-only successes	1	0	10	11	4
Expected path, displaced	0	0	8	2	3
Displaced, unexpected path; signature absent	0	0	0	9	1
Displaced, unexpected path; signature present	0	0	2	0	0
Unexpected path, within 4 cm	1	0	0	0	0
(c) Evaluator-only successes at each placement cutoff					
2 cm	18	17	10	13	4
3, 4, and 5 cm (each)	1	0	10	11	4
(d) Signatures across all 100 episodes per history					
Unexpected stable-order path	4	4	29	40	9
Wrong-phase candidate	1	1	6	1	0
(e) Block placement, button presses, and swap order in failed episodes					
Evaluator failures	5	8	41	62	72
Target geometry and open gripper never co-occurred	4	8	40	62	71
Target geometry with open gripper reached; button never below -5 mm from that action onward	1	0	1	0	1
Candidate button-only failure	0	0	0	0	0
Complete expected path, evaluator failure	0	0	1	0	1
Exactly one reference swap matched, evaluator failure	0	0	0	53	61

(f) Swap-and-press cycle duration, $C = 608$				
Unit	Minimum	Median	Mean	Maximum
World-token equivalents	112.50	116.50	116.07	118.75
Simulated seconds	27.00	27.96	27.86	28.50

F.7 Extended Execution beyond Demonstrations

The exploratory stress test disables termination at first evaluator success while keeping the checkpoint, stochastic sampler, four-action execution interval, and $C = 608$ window fixed. We record one trajectory for each of the nine seeds used for the exploratory stress test in Table 6, covering all five non-target initial permutations. Rollouts have no fixed action-step limit; they end under the stopping conditions below.

The monitor identifies stable block orders using Table 21 and checks transitions against the repeating sequence $\mathrm{swap}(M, R)$, $\mathrm{swap}(L, R)$, $\mathrm{swap}(L, M)$. A swap counts as correct when the next stable order matches the prescribed positional swap; a subsequent button press is not separately required for every counted swap. We record first evaluator success independently, so the target arrangement may be reached before the success event. The history window starts sliding at 145.92 seconds.

Each trajectory ends at the first confirmed stable-order deviation or after 1,500 actions (90 seconds) without a new confirmed reference transition. Event times equal action indices multiplied by 0.06 seconds. The test measures how long ordered behavior continues after removing the demonstrations' stopping rule.

The nine trajectories complete a median of eight confirmed swaps and a maximum of 31. Five continue the reference sequence after the $C = 608$ window starts sliding. Ordered behavior can therefore persist beyond the initial episode prefix, although its duration varies substantially across seeds.

G Relationship between Offline Action RMSE and Closed-Loop Success

G.1 Agreement and Differences between RMSE and Success Rate

Across the RoboCasa sweep, lower expert-action RMSE generally accompanies higher closed-loop success. All 40 comparisons between adjacent data sizes at fixed model capacity and training seed show lower full-chunk stochastic holdout RMSE and higher mean SR with a ten-step context. The same holds for all 10 same-data comparisons between the 44.3M- and 218.8M-parameter models. Figure 7 shows this association across all 50 policies.

This overall trend does not reliably rank nearby checkpoints. In the separately trained context comparison, increasing the training and evaluation context from five to ten steps lowers both RMSE and SR for both training seeds. Mean RMSE falls from 0.138384 to 0.132404, while mean SR falls from 50.74% to 48.25%. Across tasks, RMSE improves on 20/23; SR decreases on 15, is unchanged on one, and increases on seven. Thus, lower RMSE under these context-specific evaluation protocols does not imply higher closed-loop SR.

From $D = 1000$ to $D = 2900$, all 10 matched training-seed/model-capacity comparisons improve in both metrics. Using full-precision final measurements, the mean relative RMSE reduction is 17.10% with a 1.76-percentage-point sample standard deviation. SR increases by 3.39 percentage points on average, with a 2.12-point sample standard deviation. RMSE also improves in all 230 task-level comparisons across ten seed/capacity pairs and 23 tasks. Task-level SR can increase or decrease, as the 218.8M results show (Table 13). Relative RMSE reductions and absolute SR changes use different units and aggregation, so their numerical magnitudes are not directly comparable.

Holdout RMSE measures sampled action-chunk similarity on fixed expert observations and histories. Closed-loop SR measures task completion along trajectories generated by the policy and environment. Two differences help explain why the metrics can disagree, although the present experiments do not isolate the cause of that disagreement.

First, expert demonstrations have limited state coverage. A small policy error can change later images, proprioception, object poses, and contacts. Better predictions on recorded trajectories therefore need not improve recovery after a deviation, as in the covariate-shift problem of behavior cloning (Ross et al., 2011). A recorded action may also be only one of several valid approaches, contact patterns, timings, or recovery paths. RoboCasa uses 50 human demonstrations per task to guide MimicGen in generating roughly 3,000 trajectories (Nasiriany et al., 2024). Additional generated trajectories

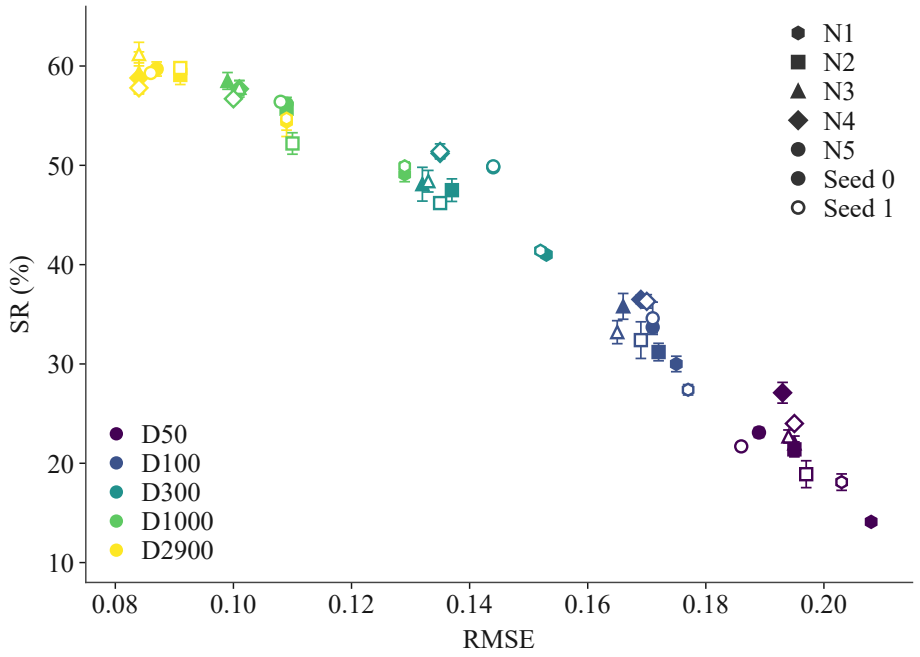


Figure 7: Holdout stochastic RMSE versus closed-loop SR for all 50 trained policies. The x-axis is task-averaged RMSE over all 12 action dimensions and all ten steps in each sampled action chunk; the y-axis is mean SR over three complete evaluations with a ten-step context. Color, marker, and fill encode demonstrations per task D, capacity, and training seed. Error bars are fixed-seed repeatability sample standard deviations (Appendix B.4). Descriptive Spearman correlation is $\rho = -0.980$, summarizing the sweep-level trend rather than ranking nearby checkpoints.

provide denser coverage of this expert distribution, but not necessarily recovery states induced by the policy. Identifying this mechanism would require failure-state data, recovery demonstrations, or on-policy data collection.

Second, success depends on receding-horizon execution, controllers, contact dynamics, termination, and the evaluator's discrete success predicate. In the Blocks Ranking diagnostic (Appendix F.4), the standard-loss record associates a millimeter-scale difference in final button depression with a different binary outcome, while sorting remains largely intact. This example shows that SR reflects the full execution and evaluation process, rather than model outputs alone.

These observations concern the behavior-cloning policies trained on the benchmark data studied here. RMSE provides a dense diagnostic on fixed expert-trajectory crops and has a smaller relative range across training seeds than SR in this grid. Repeated rollout evaluation remains necessary to assess closed-loop capability. Large, consistent changes in both metrics provide complementary evidence. When the metrics disagree, training-state coverage, recovery behavior, controller effects, and evaluation rules deserve closer examination; small metric differences alone offer a weak basis for ranking models.

G.2 Definitions of Offline RMSE and Closed-Loop Success

For the main RoboCasa sweep, let $p_E^{\mathrm{ho}}(\tau)$ denote the finite holdout expert-trajectory distribution. RMSE uses the fixed crops described in Appendix B.4: eight ten-timestep crops from each task's 100 holdout demonstrations, rather than all trajectory windows. For a crop starting at policy index s, query t receives only $o_{s:t}$. Let j_t be its stored control-frame index and $A_t = (a_{j_t}, \ldots, a_{j_t+H-1})$ the target of consecutive low-level commands. The indexed collection $\mathcal{Q}_m$ contains all pairs $(o_{s:t}, A_t)$

with a complete H-step target for task m. Then

$$\mathrm{RMSE}_m = \left[\frac{1}{|\mathcal{Q}_m|\, H d_a} \sum_{(o_{s:t}, A_t) \in \mathcal{Q}_m} \left\| \hat{A}(o_{s:t}) - A_t \right\|_2^2 \right]^{1/2}, \qquad \mathrm{RMSE}_E = \frac{1}{M} \sum_{m=1}^{M} \mathrm{RMSE}_m, \tag{25}$$

with $M = 23$ tasks and one stochastic ten-step, 12-dimensional chunk $\hat{A}(o_{s:t})$ per query under `eval_seed=0`. Targets and predictions are in the original, unnormalized controller-command space. We average squared error over all scalar action elements for each task, take the square root, and then average equally across tasks. The deterministic variant uses the same initial x_T and the posterior-mean reverse transitions in Appendix B.4. RMSE measures imitation on holdout expert observations and histories; the training objective is diffusion noise-prediction loss.

For closed-loop evaluation, the initial-state distribution ρ_0, policy, and environment/evaluation protocol $\mathcal{M}$ induce the trajectory distribution

$$p_{\pi,\mathcal{M}}(\tau \mid \rho_0). \tag{26}$$

Let $E_{\mathcal{M}}(\tau)$ be the evaluator's binary outcome for trajectory τ. The set of successful trajectories is

$$T_{\mathcal{M}}^{\mathrm{succ}} = \{\tau \mid E_{\mathcal{M}}(\tau) = 1\}. \tag{27}$$

Then

$$\mathrm{SR}(\pi; \mathcal{M}, \rho_0) = \Pr_{\tau \sim p_{\pi,\mathcal{M}}(\cdot \mid \rho_0)} \left[\tau \in T_{\mathcal{M}}^{\mathrm{succ}} \right]. \tag{28}$$

Successful trajectories may follow expert paths, alternative solutions, or recovery paths. They may also be very short if the success criterion permits this. Offline RMSE on p_E^{ho} and closed-loop SR on $p_{\pi,\mathcal{M}}$ therefore evaluate different distributions and outcomes, as discussed in Appendix G.1.

G.3 Variation across Training Seeds and Repeated Evaluations

We summarize variation in metrics with different units using the relative range

$$R_{\mathrm{rel}}(x) = \frac{\max_i x_i - \min_i x_i}{\bar{x}}. \tag{29}$$

For each of the 50 policies, we use three evaluations with a ten-step context, totaling 150 executions of 1,150 episodes. Each checkpoint's repeats use the same episodes, environment seeds, and rollout seed, as specified in Appendix B.4. They measure fixed-seed repeatability rather than variation across independently sampled rollout seeds. SR relative range has median 3.40%, third quartile 5.09%, and maximum 14.24%; the corresponding absolute ranges are 1.39, 2.09, and 4.09 percentage points. Many small differences between nearby high-performing configurations fall within this range of variation.

Across training seeds, we apply the same formula to the two results for each of the 25 $(D, \text{capacity})$ pairs. RMSE relative range has median 0.76%, third quartile 1.14%, and maximum 2.26%; the SR values are 1.74%, 6.35%, and 24.78%, respectively. These summaries compare different trained policies, so they do not isolate the contributions of individual sources of variation. Together, the repeatability and cross-seed results support using RMSE and repeated SR as complementary evidence for broad trends, rather than selecting models by small local differences.

H Extended Related Work on Robotic Sequence Modeling

H.1 Sequence organization in robot policies

Sequence modeling is widely used in robot learning and generalist agents. Decision Transformer arranges returns, states, and actions in a causal sequence. Gato serializes data across tasks, modalities, and embodiments into one token stream, using a shared Transformer to generate text, actions, and other outputs from context (Chen et al., 2021; Reed et al., 2022). These works formulate decision making as sequence modeling while leaving open the choice of a basic unit for temporal context.

Robot Transformers use different sequence units. RT-1 compresses each visual frame into several tokens with TokenLearner. ICRT combines vision and proprioception into a state token and interleaves

it with action features in a causal sensorimotor sequence. GR-1 places language, image sequences, robot state, and future-image prediction in one GPT-style sequence (Brohan et al., 2023b; Fu et al., 2024; Wu et al., 2024). All three track physical time, but their top-level positions can differ in modality, input source, or generation role.

Another design choice separates history modeling from action generation. Diffusion Policy models continuous, multimodal action sequences with conditional diffusion; ACT uses a conditional VAE to generate action chunks; Octo summarizes context with learned readout tokens and uses a lightweight diffusion head; and π_0 adds a separately parameterized flow-matching action expert (Chi et al., 2023; Zhao et al., 2023; Ghosh et al., 2024; Black et al., 2025). These methods establish precedents for separate action-generation modules. Here, we focus on how temporal context is organized and accessed before action generation.

Compact timestep representations also have direct precedents. The BC-Transformer baseline in RoboCasa encodes each observation timestep, applies non-causal self-attention within a fixed ten-observation window, and lets every window position emit an action prediction through a Gaussian mixture head (Nasiriany et al., 2024; ARISE Initiative, 2024). HAMLET introduces moment tokens that compress each timestep of a pretrained VLA and aggregates them with a lightweight memory module, and Chronos represents each control step with one state-representative token propagated through a selective state-space model (Koo et al., 2026; Zhou et al., 2026).

WorldToken contributes one observation-derived token per policy timestep to a causal sequence. We study data and model scaling, vary the number of tokens per timestep across data scales, and test the effect of visible history. Token-count comparisons keep tokens grouped by timestep and preserve causal order.

H.2 Temporal context, memory, and partial observability

History-conditioned control predates modern Transformers: recurrent imitation policies such as BC-RNN maintain a hidden state across timesteps, and Hiveformer jointly models language, multiview observations, and full observation/action history for multitask manipulation (Mandlekar et al., 2022; Guhur et al., 2023).

History availability, necessity, and utilization are distinct. Availability concerns whether the architecture can receive past observations; necessity concerns whether the task is partially observable without them; utilization concerns whether the trained policy's behavior depends on history. A large input window establishes only availability. Truncating or perturbing history while fixing policy weights primarily tests utilization (Chen et al., 2026; Shi et al., 2026; Zhou et al., 2026).

Existing designs retain history in three broad ways. Recurrent state compression summarizes the past in a continually updated hidden state, as in BC-RNN and Chronos's selective state-space model. Explicit sequence context keeps past interactions available for attention. Examples include Hiveformer's early full-history Transformer, HALO's VQA-supervised relevance and sparse top-K attention to reduce spurious historical correlations, and PRISM's gated attention and hierarchical compression for minute-scale visuomotor memory. Controlled Diffusion Policy studies also show that context-length gains depend on conditioning, denoising architecture, variable-length training, and data conditions (Shah et al., 2026a,b; Agarwal et al., 2026).

Dedicated memory subsystems define write, update, or retrieval rules. RMBench proposes Mem-0 as an explicit-memory reference, MemoryVLA maintains a Perceptual-Cognitive Memory Bank, and RoboMME evaluates memory-augmented VLA integration across several dimensions (Chen et al., 2026; Shi et al., 2026; Dai et al., 2026). Across these approaches, longer context does not by itself ensure effective memory.

WorldToken uses explicit sequence context, without a separate memory bank or hand-designed rules for retrieval or updates. We measure history utilization through controlled context interventions on fixed trained checkpoints.

Memory design also affects computation. RoboMME compares memory budgets through performance–compute curves, while HAMLET and MEM compress observation history at different granularities (Dai et al., 2026; Koo et al., 2026; Torne et al., 2026). Appendix D.5 examines how the tokens contributed by each timestep affect computational growth with cached history.

H.3 Latent representations and predictive world models

Compressing observations into latent state and predicting the environment there is a classic world-model strategy. PlaNet learns latent dynamics for planning, Dreamer learns through imagined trajectories, and JEPA-style methods predict abstract representations rather than reconstructing all inputs (Hafner et al., 2019, 2020, 2025; LeCun, 2022; Assran et al., 2023). V-JEPA 2-AC adds an action-conditioned latent world model for robot planning, while GR-1 and ACT-JEPA combine behavior learning with future image or latent prediction (Assran et al., 2025; Wu et al., 2024; Vujinovic & Kovacevic, 2026).

WorldToken also forms a compact representation before decision making. Its tokens are policy representations aligned with physical time, rather than predictive world-model states. They are not defined through future-state prediction, though such objectives could be added.

H.4 Scaling in robot learning

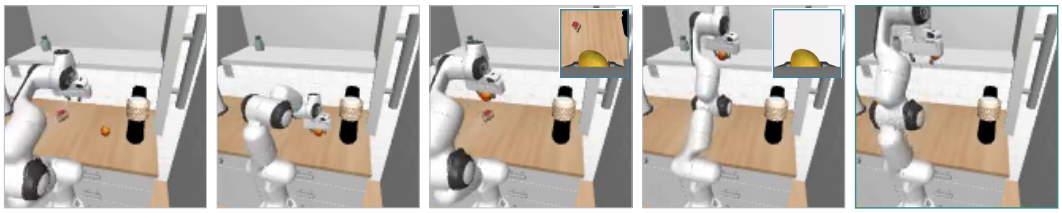
Scaling studies examine the effects of data and model size. RT-1 varies data size, model capacity, and diversity; Open X-Embodiment/RT-X studies cross-robot data aggregation; and OpenVLA combines vision–language pretraining with large robot datasets (Brohan et al., 2023b; Open X-Embodiment Collaboration et al., 2024; Kim et al., 2025). Other studies show that demonstration count and the diversity of training environments and objects affect policy generalization (Lin et al., 2025). ScaleDP and broader imitation-learning studies show that the benefits of model capacity depend on architecture, tokenization, data, and optimization (Zhu et al., 2024; Pearce et al., 2025).

Our sweep evaluates all combinations of five dataset sizes and five model capacities within one WorldToken family, with visual and policy modules trained from scratch. Given the architecture dependence found in prior work, our conclusions characterize this family within the tested range.

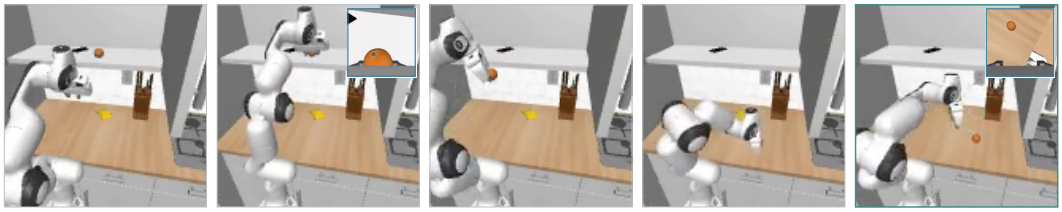
I QUALITATIVE ROBOCASA ROLLOUTS

Figures 8–13 show 22 successful WorldToken rollouts on RoboCasa, with one trajectory per task–scene configuration. The examples are organized into seven official atomic-task families. Episodes are selected across training runs, checkpoints, and evaluated history lengths, with success verified from evaluation logs.

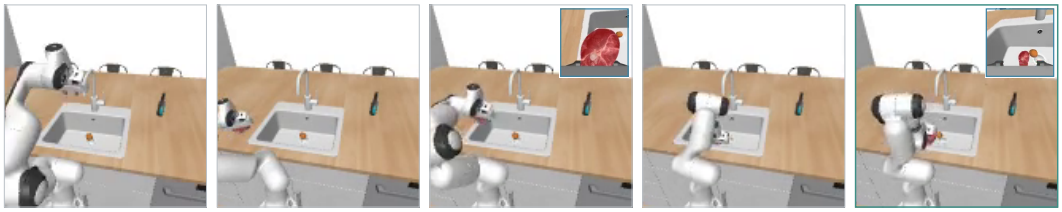
Each row presents five frames together with the complete original language instruction. Frames proceed from left to right at nonuniform intervals. Blue-bordered insets show wrist-camera observations from the same timestep as the corresponding main frame, and green borders mark the last displayed frame of each trajectory.



(a) pick the mango from the counter and place it in the cabinet



(b) pick the tangerine from the cabinet and place it on the counter



(c) pick the steak from the counter and place it in the sink



(d) pick the orange from the sink and place it on the plate located on the counter

Figure 8: Pick and place (I): cabinet and sink transfers. The robot moves objects between the counter and a cabinet (a,b), and between the counter and a sink (c,d).



(a) pick the garlic from the counter and place it in the microwave



(b) pick the egg from the plate and place it in the pan



(c) pick the apple from the pan and place it on the plate

Figure 9: Pick and place (II): microwave and stove transfers. The robot places an object in the microwave (a), moves an egg from a plate to a pan (b), and moves an apple from a pan to a plate (c).



(a) open the cabinet door



(b) close the cabinet door



(c) close the cabinet doors



(d) open the left drawer



(e) close the right drawer

Figure 10: Opening and closing doors and drawers. Panels (a–c) show opening a cabinet door, closing a cabinet door, and closing both cabinet doors, respectively. Panels (d,e) show opening and closing a drawer.



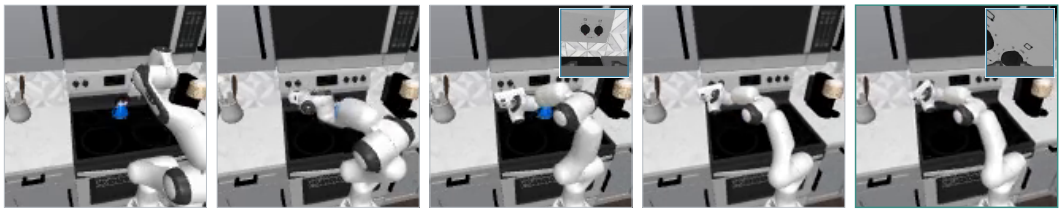
(a) turn on the sink faucet



(b) turn off the sink faucet



(c) turn the sink spout to the left



(d) turn on the front left burner of the stove



(e) turn off the front center burner of the stove

Figure 11: Sink and stove controls. Panels (a–c) show faucet and spout control in the Turning levers family. Panels (d,e) show switching stove burners on and off in the Twisting knobs family.

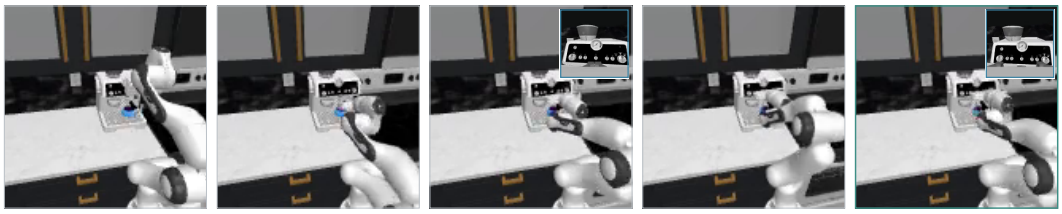


(a) pick the mug from the counter and place it under the coffee machine dispenser



(b) pick the mug from under the coffee machine dispenser and place it on the counter

Figure 12: Mug placement and retrieval at the coffee machine. The two tasks belong to the official Insertion family: placing a mug under the dispenser (a) and moving it from the dispenser to the counter (b).



(a) press the button on the coffee machine to serve coffee



(b) press the start button on the microwave



(c) press the stop button on the microwave

Figure 13: Pressing appliance buttons. The robot presses the button on the coffee machine (a), the microwave start button (b), and the microwave stop button (c).