LITEC-WAM: A LIGHTWEIGHT CROSS-TASK AND CROSS-EMBODIMENT WORLD ACTION MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

World action models (WAMs) aim to generate executable actions and predict their effects on the environment. However, key challenges remain in maintaining a lightweight model footprint, transferring learned dynamics across tasks, and adapting to new robots without forgetting previously learned capabilities. We introduce LiteC-WAM, a lightweight cross-task and cross-embodiment WAM comprising three key components: (1) a lightweight shared backbone that couples a light perception module, sparse Top-2 dynamics experts, and a flow-matching action head refined by world-model feedback; (2) cross-task adapters that transfers shared dynamics without backbone retraining; and (3) low-rank observation adapters paired with robot-specific action encoders and decoders. For lightweight deployment, LiteC-WAM achieves a 96.0% success rate with only 47.57M parameters on 5 representative LIBERO tasks, outperforming other lightweight policies such as Diffusion Policy (∼65M, 72.4%), Octo-Base (93M, 75.1%), and SmolVLA-240M (240M, 82.75%). Its success is also comparable to billion-parameter models, such as π_0 (3.3B, 94.2%), LingBot-VA (5.3B, 98.5%), and OpenVLA-OFT (7B, 97.1%). For cross-task transfer, LiteC-WAM achieves 86.67% across 15 RLBench manipulation tasks using cross-task adapters guided by language instructions, without retraining the shared backbone. For cross-embodiment transfer, LiteC-WAM achieves 82.09% closed-loop success on dual-arm SO-ARM101 and a 91.86% average offline pass rate when comparing predicted and recorded actions across seven RoboChallenge ARX5 tasks. Together, these results validate LiteC-WAM as an effective lightweight model for cross-task and cross-embodiment transfer. Evaluation code and protocols are available at `https://anonymous.4open.science/r/LITEC-WAM-A-LIGHTWEIGHT-CROSS-TASK-AND-EMBODIMENT-WORLD-ACTION-MODEL-632F/`

1 INTRODUCTION

World Action Models (WAMs), which integrate forward environment prediction with downstream action execution, represent a promising frontier in robotic manipulation(Ye et al., 2026; Li et al., 2026c). However, realizing deployable WAMs still presents two critical challenges. For practical deployment, these models should support closed-loop control with low inference cost and transfer across robots with different cameras, robot state measurements, and action spaces.

Three obstacles make this difficult. **First, model size and inference cost**: scaling a generative backbone can improve representation but makes frequent closed-loop replanning expensive (Li et al., 2026d). **Second, cross-task transfer**: shared dynamics must support different tasks while allowing small task-specific updates (Ye et al., 2026; Jiang et al., 2026). **Third, cross-embodiment transfer**: conventional fine-tuning changes shared weights and can cause catastrophic forgetting on previously learned robots (Chen et al., 2026c). These problems are coupled. A deployable WAM must predict action effects efficiently, reuse these predictions across tasks, and adapt to new robots while retaining previously learned capabilities.

To address these challenges, we introduce LiteC-WAM, a lightweight cross-task and cross-embodiment world action model built around three goals: (1) **lightweight modeling**, using a small shared model to generate actions and predict their effects on the environment; (2) **cross-task trans-**

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.

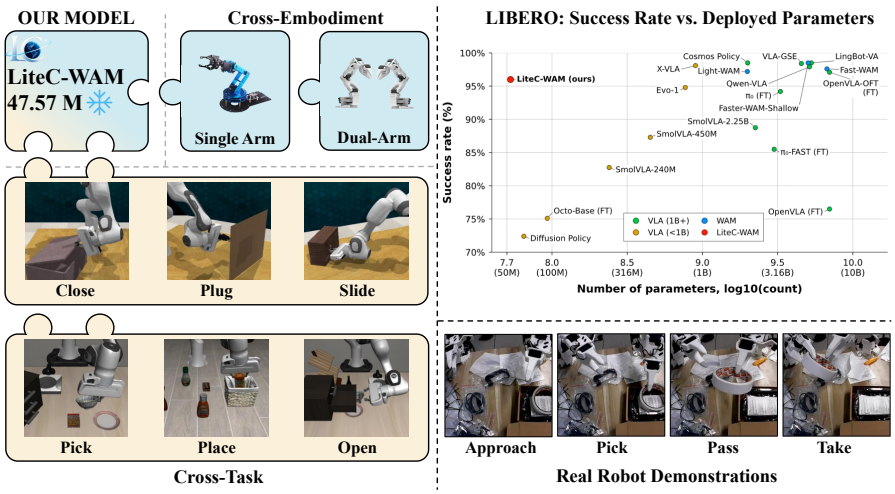


Figure 1: **LiteC-WAM: a cross-task and cross-embodiment world action model.** Simulation examples and real robot demonstrations illustrate cross-task and cross-embodiment transfer. With a 47.57M-parameter shared backbone, our model achieves a 96.0% success rate (48/50) across a representative subset of five LIBERO tasks, departing from the standard protocol that evaluates and aggregates across the full suite. Sources and notes on parameter counts appear in Appendix I.

fer, reusing the shared model across diverse manipulation tasks, including closing, inserting, picking, placing, opening, and sliding, without training a separate full model for each task; and (3) **cross-embodiment transfer**, transferring the model to robot platforms with different observations and action spaces, including single-arm and dual-arm systems. We evaluate these three capabilities through simulation benchmarks, closed-loop real-robot trials, and offline action-agreement evaluation as shown in Figure 1.

Our contributions are:

- **A lightweight WAM design.** We propose the WAM, featuring a lightweight 47.57M-parameter backbone that jointly represents multimodal context, eight-step flow-matching action chunks, and action-conditioned future features. When augmented with robot-specific adapters for RLBench tasks, the full architecture requires only 54.55M parameters.
- **Cross-task transfer with shared dynamics.** We demonstrate effective cross-task transfer by using cross-task adapters on the shared backbone, achieving an 86.67% success rate across 15 RLBench tasks. Furthermore, a paired ablation on LIBERO demonstrates that world-model feedback yields a +7.08-point performance gain across 480 rollouts.
- **Cross-embodiment transfer with a frozen backbone.** We introduce low-rank visual/state adapters, robot-specific action codecs, and task-conditioned sparse experts to transfer to ARX5 and dual-arm platforms while keeping the shared backbone frozen. Both transfer performance and capability retention are validated through task success rates and action-trajectory consistency.

2 Related Work

Lightweight World Action Models. Existing work makes world action models lighter in two complementary ways. One line of work reduces model size and the cost of learning future dynamics: Efficient-WAM uses a compact video expert and fewer denoising steps, turning future state prediction from "generating high-resolution frames" into "providing coarse guidance" (Li et al., 2026b); Light-WAM uses a compact video backbone, predicts the future in a compressed latent space, and decodes actions directly (Li et al., 2026d). The other line of work changes how future state prediction is used at inference time: Fast-WAM still does video co-training during training, but skips future state prediction at inference and generates actions directly from the world representation learned

during training (Yuan et al., 2026); Faster-WAM argues that future information at inference time is important for generalization, so it keeps future state prediction and only reduces inference cost through sparser interaction (Zhao et al., 2026). LiteC-WAM takes a different approach: it combines a small shared backbone with sparse dynamics experts—its 47.57M-parameter backbone supports both action generation and future state prediction, selecting only 2 of 8 dynamics experts per prediction. At inference time, the prediction is fed directly to the action head as latent features and events, without decoding future frames. In this way, LiteC-WAM can combines lightweight modeling with explicit future feedback in an efficient manner.

Cross-Task World Action Models. A common direction in existing work on cross-task transfer is transferring learned dynamics under different language instructions (Appendix Figure 6). DreamZero uses pretrained video dynamics to jointly generate future video and actions (Ye et al., 2026). Motubrain combines policy learning and world modeling in a unified video-action model (Motubrain Team et al., 2026). RepWAM learns aligned visual and action representations for language-conditioned prediction (Wang et al., 2026a). These approaches connect shared dynamics with task instructions to support diverse manipulation behaviors. LiteC-WAM takes a different approach: during cross-task transfer, the shared backbone remains frozen, different cross-task adapters are selected by the instructions, and 2 of 8 dynamics experts are sparsely activated, whose predictions are then fed back to the action head. In this way, LiteC-WAM achieves cross-task transfer without retraining the shared backbone.

Cross-Embodiment World Action Models. A common focus of existing work on cross-embodiment transfer is sharing dynamics knowledge across robots with different observations and action spaces (Appendix Figure 6). LDA-1B jointly learns dynamics, policies, and visual prediction from human and robot trajectories (Lyu et al., 2026). LAC-WM learns a unified latent action space shared across embodiments (Huang et al., 2026). Hydra-0 represents robot actions as pixel motion, providing a common visual interface across robots (Li et al., 2026a). These methods emphasize shared representations; a related question is how to adapt to a new robot while preserving previously learned capabilities. To this end, LiteC-WAM combines a unified latent action space with embodiment-specific action adapters trained during cross-embodiment transfer.

3 METHOD

LiteC-WAM combines a 47.57M shared backbone, cross-task sparse routing and cross-embodiment adapters as shown in Figure 2. The model and training details are shown in Appendix C.3.

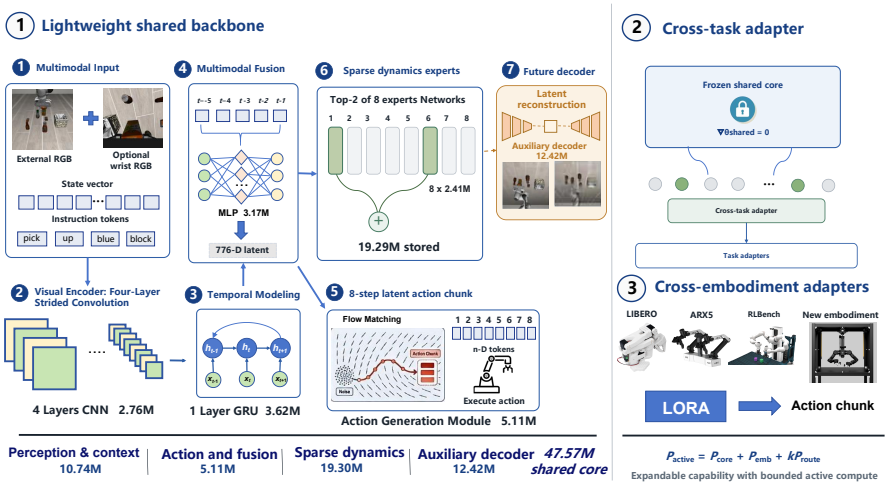


Figure 2: **LiteC-WAM architecture.** The shared backbone jointly generates action chunks and predicts future states. Cross-task adapters transfer contextual latent representations across tasks, while cross-embodiment adapters interface the backbone with specific robot platforms. Selectively updated components vary depending on the cross-task or cross-embodiment configuration.

3.1 LIGHTWEIGHT SHARED BACKBONE

Multimodal Context Encoding. At each timestep t, the shared backbone encodes a five-step history of 64×64 RGB observations and robot proprioceptive states, alongside a language task instruction. Specifically, a convolutional encoder coupled with a gated recurrent unit (GRU) extracts temporal visual features h_t^{vis}, while dedicated linear encoders process the state history into h_t^{state} and the instruction for task j into h_j^{lang}. A lightweight multi-layer perceptron (MLP) F_{ctx} fuses these representations into a unified context vector:

$$c_t = F_{\text{ctx}}\Big([h_t^{\text{vis}}; h_t^{\text{state}}; h_j^{\text{lang}}]\Big) \in \mathbb{R}^{776}. \tag{1}$$

Action Chunking via Flow Matching. To generate smooth action trajectories, the backbone incorporates a conditional flow matching head (Lipman et al., 2023). Given a normalized demonstration action sequence a, Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$, and flow time $\tau \sim \mathcal{U}[0, 1]$, the target trajectory is formed via linear interpolation $x_\tau = (1 - \tau)\epsilon + \tau a$. A velocity network v_θ conditions on the multimodal context c_t and is optimized via the flow matching objective:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{a,\epsilon,\tau}\left[\|v_\theta(x_\tau, \tau, c_t) - (a - \epsilon)\|_2^2\right]. \tag{2}$$

At inference, beginning from sampled noise $x_0 \sim \mathcal{N}(0, I)$, an 8-step Euler solver numerically integrates along the vector field predicted by v_θ from $\tau = 0$ to 1. This process generates an action sequence (e.g. an 8×7 chunk in LIBERO), which serves as the candidate actions for future prediction.

3.2 CROSS-TASK TRANSFER

Task Adaptation and Query Construction. To generalize across diverse tasks while preserving base capabilities, LiteC-WAM reuses its shared backbone and dynamics experts with the backbone parameters frozen. Before policy execution, an instruction selects a corresponding lightweight cross-task adapter. For forward dynamics prediction, the model constructs action-conditioned queries across a prediction horizon $h \in \{1, \ldots, H\}$ (where $H = 8$). At step h, the query vector combines multimodal context, candidate action embeddings, and temporal step embeddings:

$$q_{t,h} = c_t + E_a(a_{t+h}) + p_h, \tag{3}$$

where $E_a(a_{t+h})$ embeds the candidate action and p_h represents a step embedding.

Sparse Dynamics Experts. To model environment transitions efficiently, a routing layer with parameters W_r computes gating probabilities over eight candidate dynamics experts $\{f_m\}_{m=1}^{8}$:

$$g_{t,h} = \text{softmax}(W_r q_{t,h}). \tag{4}$$

For each query $q_{t,h}$, the router selects the top-2 dynamics experts and updates the query representation via a residual mixture:

$$\widetilde{q}_{t,h} = q_{t,h} + \sum_{m \in \text{TopK}(g_{t,h}, 2)} \bar{g}_{t,h,m} f_m(q_{t,h}), \tag{5}$$

where $\bar{g}_{t,h,m}$ are the gating weights renormalized over the two chosen experts. The refined features $\widetilde{q}_{t,h}$ are then routed to lightweight prediction heads to predict future latent states and task-relevant event logits $\widehat{y}_{t,h}$.

World-Model Feedback. Rather than using forward predictions solely as passive training signals, LiteC-WAM leverages them for active closed-loop action correction. We construct a feedback summary vector w_t that aggregates latent feature discrepancies, expert routing distributions, and predicted event probabilities averaged across all H horizon steps:

$$\begin{aligned} w_t &= \left[\text{mean}_h(\widetilde{q}_{t,h} - q_{t,h});\ \text{mean}_h\, g_{t,h};\ \text{mean}_h\, \sigma(\widehat{y}_{t,h})\right], \\ a^{(1)} &= a_{\text{ctx}} + \alpha \tanh\left(R_w(w_t)\right), \end{aligned} \tag{6}$$

where $\sigma(\cdot)$ denotes the sigmoid function. A lightweight feedback head R_w predicts a residual correction to refine the preliminary flow-matched action sequence a_{ctx}, with the maximum correction magnitude bounded by hyperparameter α.

Inference and Execution. Training the feedback head R_w requires updating only its own parameters while keeping the shared backbone frozen. Crucially, as w_t only operates in the latent space, world-model feedback requires no computationally expensive RGB pixel decoding during inference. (In our experiments, the representative five-task in LIBERO benchmark executes the base policy a_{ctx} directly to maintain minimal latency, while the 20-task suite ablates the performance gains of R_w). During closed-loop rollouts, the robot executes the action sequence according to the benchmark protocol before receiving new observations to replan.

3.3 Cross-Embodiment Transfer

To transfer the pre-trained shared backbone across distinct robotic platforms, LiteC-WAM decouples embodiment-specific input/output dimensions from the shared representations using modular adapters.

Low-Rank Feature Adaptation. To accommodate different observation dimensions and sensor configurations without retraining the backbone, a low-rank adapter aligns input features for robot embodiment e:

$$\phi_e(x) = \phi_0(x) + B_e\,\mathrm{SiLU}(A_e x), \qquad A_e \in \mathbb{R}^{r\times d_{\text{in}}}, \quad B_e \in \mathbb{R}^{776\times r}, \tag{7}$$

where $x \in \mathbb{R}^{d_{\text{in}}}$ denotes embodiment-specific raw input features, and $\phi_0(x) \in \mathbb{R}^{776}$ is the base feature representation. We set the adapter bottleneck rank to $r = 16$ with SiLU activations. By initializing B_e to zero, $\phi_e(x)$ strictly preserves the base backbone representation at the onset of adaptation.

Unified Latent Action Space. Different robots operate with distinct degrees of freedom (DoF) and control spaces. To bridge these disparities, we introduce robot-specific action encoders C_e^{enc} and decoders C_e^{dec} that project embodiment-specific actions a^e to and from a shared 16-dimensional space:

$$z_a = C_e^{\text{enc}}(a^e) \in \mathbb{R}^{16}, \qquad \widehat{a}^e = C_e^{\text{dec}}(z_a), \tag{8}$$

where $\widehat{a}^e$ is the reconstructed action. During closed-loop execution, the policy generates an 8-step trajectory within this unified 16-D latent space, which C_e^{dec} subsequently maps into executable robot commands. Cross-embodiment fine-tuning optimizes these action codecs alongside the low-rank feature and task adapters, while keeping the core backbone frozen.

Bimanual Adaptation (SO-ARM101). Transferring to the dual-arm SO-ARM101 platform introduces substantial distribution shifts due to coordinated bimanual dynamics and multi-view inputs. For this embodiment, we adapt the model with two specific modifications:

1. *Multi-Camera Adaptation and State Alignment:* We employ rank-32 visual adapters ($r = 32$) to process dual RGB streams (global and wrist views), and apply a fixed zero-padding mapping to project the 12-DoF bimanual joint and gripper coordinates into a 16-dimensional state vector.
2. *Forward-Conditioned Policy Refinement:* Unlike the LIBERO setup—which applies a residual correction R_w to preliminary actions—SO-ARM101 directly injects the predicted future feature differences as conditioning tokens into the action generation head. During this refinement, the visual encoder remains frozen while updating the shared dynamics experts and embodiment-specific modules.

Training recipes and parameter distributions across embodiments are detailed in Appendix C.

4 Experiments

We evaluate model size and performance, cross-task transfer, and cross-embodiment transfer, in that order. Closed-loop results report successes/trials and two-sided 95% Wilson intervals; offline action agreement is reported separately.

4.1 Model Size and Performance Comparison

We evaluate five LIBERO tasks on a Franka Panda arm (Liu et al., 2023): Spatial 0/3, Object 1/9, and LIBERO-10 task 5, each from ten fixed states with at most 300 steps. Success follows the

benchmark criteria. Parameter counts come from checkpoint tensors. The 47.57M-parameter policy succeeds in 48/50 trials (96.0%, 95% CI 86.5–98.9; Table 1). This five-task result excludes the world-model feedback head, tested separately in Section 4.2.

Efficiency of sparse expert routing. A separate comparison uses six seeds, identical demonstrations, and 120 matched rollouts per variant. Selecting two experts activates 30.6% fewer parameters than using all experts (32.40M versus 46.68M), with 50/120 versus 49/120 successes ($p = 1.0$). Selecting one expert gives 48/120 ($p = 0.75$). These differences are not significant: sparse routing reduces active computation but does not demonstrate a success-rate gain. Appendix Table 12 gives the full comparison, including the nearly eight-fold prediction-error increase without future-prediction losses. A separate Spatial-0 test uses 50 matched states and yields 43/50 successes versus 38/50 for a larger specialist ensemble, as shown in Appendix B.

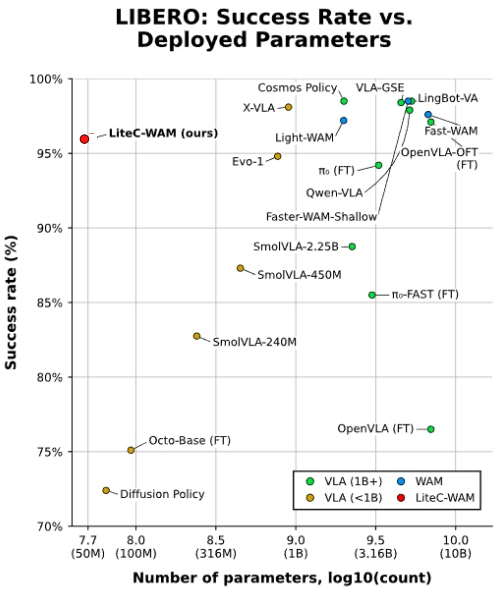
Limitations. The five-task LIBERO evaluation and differing baseline protocols limit direct comparison. Future work should cover all four LIBERO suites under matched conditions and measure latency on the target hardware.

4.2 Cross-Task Evaluation

We evaluate 15 RLBench tasks, each from eight fixed initial states (120 rollouts), using benchmark success criteria (James et al., 2020). The shared backbone remains fixed; cross-task adapters are selected from instructions before evaluation, never from outcomes. Like LIBERO, RLBench uses a Franka Panda arm, but its observations and action conventions differ. Exact seeds and task-level results appear in Appendix B.

Table 1: **LIBERO external-baseline comparison.** All models are listed in one continuous column. The right panel visualizes the same relationship between success rate and parameter count. LiteC-WAM uses the marked five-task protocol rather than an official four-suite macro average.

Method	Params.	Success
Diffusion Policy (Chi et al., 2023)	∼65M*	72.4
Octo-Base (FT) (Octo Model Team et al., 2024)	93M	75.1
OpenVLA (FT) (Kim et al., 2025b)	7B	76.5
SmolVLA-240M	240M	82.75
π_0-FAST (FT) (Kim et al., 2025a)	3B	85.5
SmolVLA-450M (Shukor et al., 2025)	450M	87.3
SmolVLA-2.25B (Shukor et al., 2025)	2.25B	88.75
π_0 (FT) (Black et al., 2024)	3.3B	94.2
Evo-1	770M	94.8
OpenVLA-OFT (FT) (Kim et al., 2025a)	7B	97.1
Light-WAM (Li et al., 2026d)	1.99B	97.2
Fast-WAM	6.73B	97.6
Qwen-VLA	5.15B	97.9
X-VLA	900M	98.1
VLA-GSE	4.55B	98.4
LingBot-VA	5.30B	98.5
Faster-WAM-Shallow (Zhao et al., 2026)	5.05B	98.5
Cosmos Policy	2.00B	98.5
LiteC-WAM (ours)	**47.57M**	**96.0**†



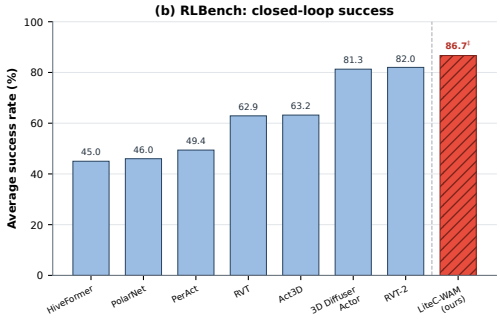
Success versus deployed parameters; higher and further left is better. Protocols differ.

* Diffusion Policy count is a configuration-level estimate because the cited work does not specify a single total parameter count for deployment. † LiteC-WAM: five selected tasks, 48/50. Other rows report their published LIBERO aggregates and are not protocol-matched. Sources for every model and variant are listed in Appendix I.

LiteC-WAM succeeds in 104/120 trials (86.67%, 95% CI 79.4–91.6) with a 54.55M-parameter task configuration, as shown in Table 2. Recorded policy-call latency is 20.61–24.40 ms.

Table 2: **RLBench external-baseline comparison.** Published rows use the standard 18-task/249-variation protocol. The right panel visualizes the same results; the marked LiteC-WAM row uses the stated 15-task evaluation set used during development.

Method	Params.	Success
HiveFormer (Guhur et al., 2023)	–	45.0
PolarNet (Chen et al., 2023)	14.1M	46.0
PerAct (Shridhar et al., 2023)	33.0M	49.4
RVT (Goyal et al., 2023)	36.4M	62.9
Act3D (Gervet et al., 2023)	–	63.2
3D Diffuser Actor (Ke et al., 2025)	–	81.3
RVT-2 (Goyal et al., 2024)	72.7M	**82.0**
LiteC-WAM (ours)	**54.55M**	**86.67**‡



Average closed-loop success (%).

Parameter counts are complete policy counts for publicly reported or reproducibly counted RLBench configurations; "–" denotes no comparable disclosed total. ‡ LiteC-WAM: 15 tasks and eight evaluation episodes with fixed initial states per task, 104/120. The protocol differs from the published RLBench rows and is therefore not presented as a matched leaderboard claim.

Task execution and failures. Eleven RLBench tasks reach 8/8; remaining failures concentrate near contact. Figure 3 shows representative LIBERO and RLBench executions. Full sequences appear in Appendix Figures 7 and 8.

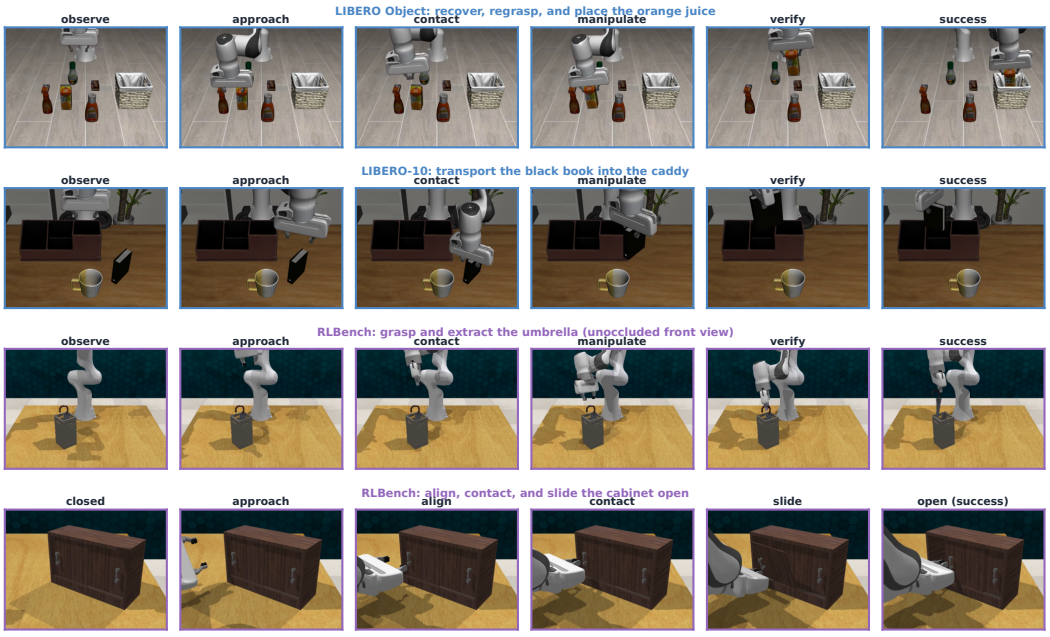


Figure 3: **Actual successful long-horizon rollouts.** Six ordered frames per episode: LIBERO regrasp, LIBERO-10 book transport, RLBench umbrella extraction, and RLBench cabinet sliding.

Effect of future predictions on actions. We conduct a separate LIBERO study to evaluate whether learning forward dynamics directly benefits policy control. Training with future-prediction losses alone yields 427/1080 successes compared to 432/1080 without predictive losses ($p = 0.793$). This indicates that improving future prediction via auxiliary losses alone does not confer a measurable control advantage.

In contrast, we evaluate whether *actively feeding* predicted dynamics back into policy execution improves control via the feedback head R_w. This ablation spans all 20 LIBERO-Spatial and LIBERO-

Object tasks across six independent seeds and four fixed initial states per task (yielding 480 paired rollouts per condition). We freeze the shared backbone, base policy, and context-based residual action head, training solely the 108,088-parameter feedback head.

To isolate the effect of this feedback, we ablate the module by setting its input vector $w_t = 0$. As shown in Table 3, active feedback yields a **+7.08 percentage point gain** (reaching an overall success rate of 73.54% on this 20-task suite, distinct from our primary five-task benchmark). At the rollout level, 64 rollouts succeed only with feedback compared to 30 only without it, demonstrating statistical significance under an exact McNemar's test ($p = 5.88 \times 10^{-4}$). Across seeds, five of the six show positive gains, though the seed-level sign test is not statistically significant ($p = 0.219$). Finally, because w_t jointly encapsulates dynamics residuals, sparse routing weights, and predicted event probabilities, this evaluation reflects the collective benefit of world-model feedback rather than isolated future-state prediction.

Table 3: **Effect of world-model feedback.** Twenty tasks, six seeds, and four fixed states give 480 paired closed-loop rollouts per condition. Weights, tasks, and initial states are matched; subsequent observations and actions may diverge.

Condition	Successes / trials	Rate	95% Wilson CI
w/o future prediction feedback	319/480	66.46%	62.12–70.54
w/ future prediction feedback	**353/480**	**73.54%**	**69.42–77.29**

Figure 4 shows predicted future frames. The example is selected by temporal change for display only; the full sequence appears in Appendix Figures 10.

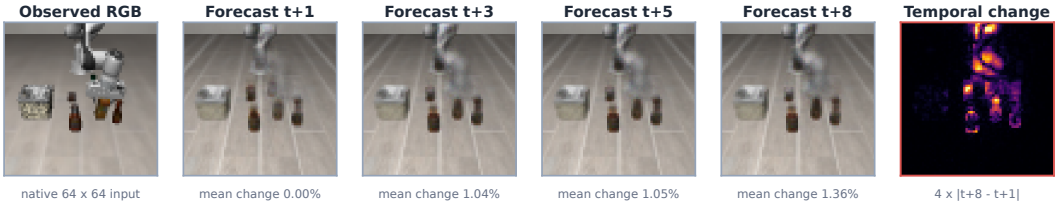


Figure 4: **Future-frame predictions during a LIBERO rollout.** Observed RGB, predicted future frames at four horizons, and a $4\times$ absolute temporal-difference overlay. RGB panels are direct checkpoint outputs without generative post-processing.

Limitations. The RLBench 15-task development set uses a different protocol from published baselines and does not test unseen-task generalization. Future work should use the standard benchmark protocol and evaluate tasks excluded from training and model selection.

4.3 Cross-Embodiment Evaluation

Real-robot evaluation. We evaluate on a dual-arm SO-ARM101 platform (two 5-joint arms with grippers) using global and wrist RGB inputs, keeping the 47.57M shared backbone frozen. Across 67 physical trials over three tasks, success requires: completing tape handover and release (*handover*), depositing the tape into a box (*placement*), or capping before setting down the bottle without tightening (*bottle-cap placement*). Trajectory analyses and the 0.34M-parameter auxiliary decoder are detailed in Appendix G.

Task completion. Across the three tasks, LiteC-WAM succeeds in 55/67 physical trials (82.09%). Figure 5 shows grasping, transport, handover, and placement. Per-task successes are 18/20 for tape removal and handover (90.0%), 26/30 for tape placement (86.7%), and 11/17 for bottle-cap placement (64.7%); criteria appear in Appendix Table 14.

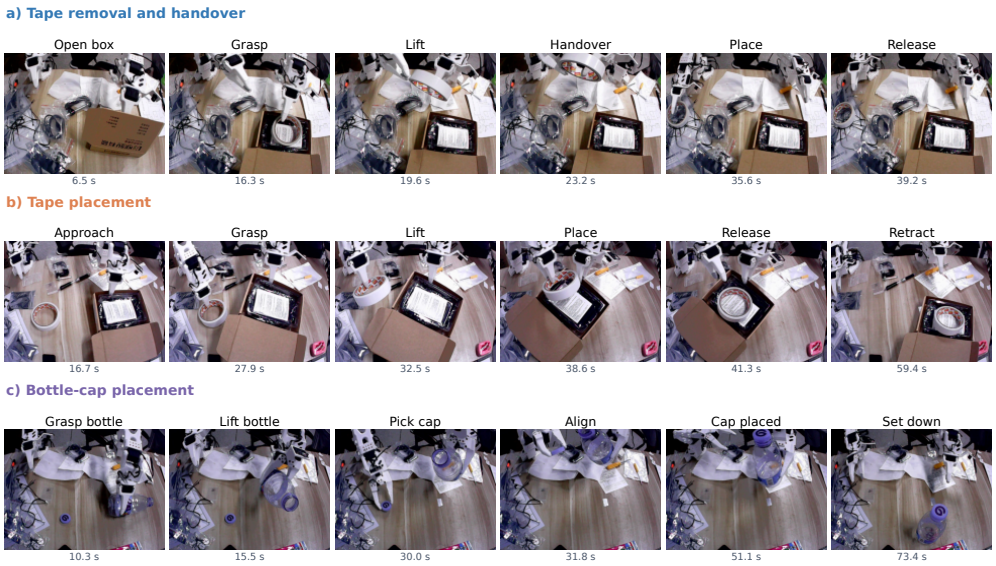


Figure 5: **Closed-loop real-robot executions.** Six stages from each recorded policy execution, using original global-camera frames. Labels describe visible progression.

Analysis of executed actions. On executed trajectories, future prediction feedback lowers the task-averaged action MAE from 1.518 to 1.453, approaching the 1.413 MAE achieved by an oracle using ground-truth recorded futures (Figure 11). Furthermore, expert routing distributions shift dynamically across manipulation phases (e.g., grasping vs. transfer), confirming phase-dependent expert specialization (Figure 12). Appendix G provides extended evaluations on future-image prediction quality and dual-view sensitivity across identical stages.

RoboChallenge evaluation. We evaluate offline action agreement on seven RoboChallenge ARX5 tasks (Yakefu et al., 2025), where an episode passes if joint and gripper errors satisfy fixed thresholds in $\geq$95% of sampled windows (Appendix B). The mean validation pass rate across tasks reaches 91.86%, but drops to 0% without the low-rank adapters, underscoring their critical role in cross-embodiment alignment.

Performance after adaptation. Adapting to ARX5 and Franka introduces only 5.53M and 6.98M parameters, respectively (Appendix Table 10). Both preserve the frozen shared weights and the base 48/50 LIBERO success rate, exhibiting only a minor closed-loop shift under Franka stress-testing (43/50 to 42/50). Notably, unfreezing the visual GRU alters eight paired rollout trajectories without changing aggregate success, demonstrating that invariant success rates do not imply identical trajectory execution.

Limitations. Physical evaluation covers three SO-ARM101 tasks, while ARX5 measures offline action agreement. Broader task and robot coverage and independent trials are needed; Further limitation details are given at Appendices G–H.

5 Conclusion

We introduced LiteC-WAM, a lightweight cross-task and cross-embodiment world action model. By unifying a 47.57M shared backbone with sparse dynamics experts, world-model feedback, and modular adapters, LiteC-WAM transfers learned dynamics to new tasks and robotic platforms while keeping the backbone frozen. Simulation experiments, closed-loop bimanual trials, and offline RoboChallenge evaluations demonstrate its strong performance and parameter efficiency. Future work includes expanding to full benchmark suites, transferring to unseen tasks and novel embodiments, and optimizing runtime latency on embedded hardware.

Ethics statement

This work includes robot simulation, physical experiments on a bimanual SO-ARM101 platform, and analyses of recorded robot trajectories. It involves no human-subject or animal experimentation and no personally identifiable information. Real-robot deployment should retain workspace exclusion, actuator limits, and emergency-stop procedures; offline agreement must not be treated as evidence of physical safety.

Reproducibility statement

The main text reports task counts, rollout denominators, fixed evaluation seeds, parameter accounting, confidence intervals, and intervention definitions. Appendix B gives task-level outcomes and evaluation protocols; Appendix D reports controlled model comparisons, and Appendix C specifies selected training settings and gradient flow. Public statistics utilities, reported aggregate counts, and policy-interface documentation are available in an anonymous repository and the accompanying anonymized supplementary code archive. The public toolkit supports recomputation of the reported statistical summaries; the learned model implementation, training pipeline, checkpoints, raw recordings, and hardware-control code are not included. The archive therefore does not independently reproduce policy training or closed-loop rollouts.

AI use statement

Generative AI tools were used to assist with language editing, figure prototyping, code development, and document organization. All AI-assisted text, code, experimental results, citations, and visual artifacts were manually reviewed and verified by the authors. The authors take full responsibility for the final content and claims.

References

AgiBot Research Team, Renhang Liu, Wenzhi Zhao, et al. GE-Act 2.0: Pretraining and Scaling a World-Action Model for Robotic Manipulation. *arXiv preprint arXiv:2609.05588*, 2026. URL https://arxiv.org/abs/2609.05588.

Kevin Black et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Jialei Chen, Kai Wang, Kang Chen, et al. LaWAM: Latent World Action Models for Efficient Dynamics-Aware Robot Policies. *arXiv preprint arXiv:2606.15768*, 2026a. URL https://arxiv.org/abs/2606.15768.

Ronghan Chen, Yandan Yang, Zuojin Tang, et al. ABot-M0.5: Unified Mobility-and-Manipulation World Action Model. *arXiv preprint arXiv:2607.00678*, 2026b. URL https://arxiv.org/abs/2607.00678.

Shizhe Chen, Ricardo Garcia Pinel, Cordelia Schmid, and Ivan Laptev. PolarNet: 3d point clouds for language-guided robotic manipulation. In *Conference on Robot Learning*, 2023. URL https://proceedings.mlr.press/v229/chen23b.html.

Yixiang Chen, Jiabing Yang, Yuan Xu, Qisen Ma, Keji He, Peiyan Li, Kai Wang, Ziheng He, Xiangnan Wu, Jing Liu, Nianfeng Liu, Yan Huang, and Liang Wang. XEWorld: Can action-conditioned world models generalize to unseen robot embodiments? *arXiv preprint arXiv:2608.05799*, 2026c. URL https://arxiv.org/abs/2608.05799.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems XIX*, 2023. URL https://www.roboticsproceedings.org/rss19/p026.html.

Dyna Robotics. Dyna-2: A 1-Million-Hour Scaling Law for World-Action Models. Technical report, 2026. URL https://www.dyna.co/dyna-2. August 2026; accessed September 21, 2026.

Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. Act3D: 3d feature field transformers for multi-task robotic manipulation. In *Conference on Robot Learning*, 2023. URL https://proceedings.mlr.press/v229/gervet23a.html.

Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. RVT: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, 2023. URL https://proceedings.mlr.press/v229/goyal23a.html.

Ankit Goyal, Valts Blukis, Jie Xu, Yijie Guo, Yu-Wei Chao, and Dieter Fox. RVT-2: Learning precise manipulation from few demonstrations. *arXiv preprint arXiv:2406.08545*, 2024. URL https://arxiv.org/abs/2406.08545.

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*, 2023. URL https://proceedings.mlr.press/v205/guhur23a.html.

Jun Guo, Qiwei Li, Peiyan Li, et al. Unified 4D World Action Modeling from Video Priors with Asynchronous Denoising. *arXiv preprint arXiv:2604.26694*, 2026. URL https://arxiv.org/abs/2604.26694.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Yucheng Hu, Jianke Zhang, Yuanfei Luo, et al. BagelVLA: Enhancing Long-Horizon Manipulation via Interleaved Vision-Language-Action Generation. *arXiv preprint arXiv:2602.09849*, 2026. URL https://arxiv.org/abs/2602.09849.

Huang Huang, Sriram Yenamandra, Arjun Majumdar, et al. Cross-Embodiment Robot Foundation World Models with Latent Actions. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/pdf/5c8fa1532215f2ea0717ed9db11ee805185dc7d3.pdf.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. RLBench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 2020. URL https://arxiv.org/abs/1909.12271.

Yuhua Jiang, Junjie Lu, Xinyao Qin, et al. VLA-GSE: Boosting Parameter-Efficient Fine-Tuning in VLA with Generalized and Specialized Experts. *arXiv preprint arXiv:2605.06175*, 2026. URL https://arxiv.org/abs/2605.06175.

Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. In *Conference on Robot Learning*, 2025. URL https://proceedings.mlr.press/v270/ke25a.html.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025a. URL https://arxiv.org/abs/2502.19645.

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 *Conference on Robot Learning*, 2025b. URL https://proceedings.mlr.press/v270/kim25c.html.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, et al. Cosmos Policy: Fine-Tuning Video Models for Visuomotor Control and Planning. *arXiv preprint arXiv:2601.16163*, 2026. URL https://arxiv.org/abs/2601.16163.

Hongyu Li, Bowen Wen, Xinghao Zhu, et al. Hydra-0: Action Flow for Generalist World Modeling and Control. *arXiv preprint arXiv:2608.18077*, 2026a. URL https://arxiv.org/abs/2608.18077.

Jiajun Li, Tiecheng Guo, Yifan Ye, et al. Efficient-WAM: A 1B-Parameter World-Action Model with Low-Cost Future Imagination. *arXiv preprint arXiv:2606.10040*, 2026b. URL https://arxiv.org/abs/2606.10040.

Lin Li, Qihang Zhang, Yiming Luo, et al. Causal World Modeling for Robot Control. *arXiv preprint arXiv:2601.21998*, 2026c. URL https://arxiv.org/abs/2601.21998.

Ziang Li et al. Light-WAM: Efficient world action models with state-fusion action decoding. *arXiv preprint arXiv:2606.08242*, 2026d. URL https://arxiv.org/abs/2606.08242.

Tao Lin, Yilei Zhong, Yuxin Du, et al. Evo-1: Lightweight Vision-Language-Action Model with Preserved Semantic Alignment. *arXiv preprint arXiv:2511.04555*, 2025. URL https://arxiv.org/abs/2511.04555.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.02747.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2023. URL https://arxiv.org/abs/2306.03310.

Hao Luo, Wanpeng Zhang, Yicheng Feng, et al. Being-H0.7: A Latent World-Action Model from Egocentric Videos. *arXiv preprint arXiv:2605.00078*, 2026. URL https://arxiv.org/abs/2605.00078.

Jiangran Lyu, Kai Liu, Xuheng Zhang, et al. LDA-1B: Scaling Latent Dynamics Action Model via Universal Embodied Data Ingestion. *arXiv preprint arXiv:2602.12215*, 2026. URL https://arxiv.org/abs/2602.12215.

Motubrain Team, Chendong Xiang, Fan Bao, et al. Motubrain: An Advanced World Action Model for Robot Control. *arXiv preprint arXiv:2604.27792*, 2026. URL https://arxiv.org/abs/2604.27792.

NVIDIA. Cosmos 3: Omnimodal World Models for Physical AI. *arXiv preprint arXiv:2606.02800*, 2026. URL https://arxiv.org/abs/2606.02800.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, et al. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems*, 2024. URL https://arxiv.org/abs/2405.12213.

OneRobotics. OneModel 1.7 FrontoStria-RL: A World Action Model Proven in Real-World Tests. Official model release, 2026. URL https://www.onerobot.com/OneModel. Accessed September 21, 2026.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, et al. FAST: Efficient Action Tokenization for Vision-Language-Action Models. *arXiv preprint arXiv:2501.09747*, 2025. URL https://arxiv.org/abs/2501.09747.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1701.06538.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, 2023. URL https://proceedings.mlr.press/v205/shridhar23a.html.

Mustafa Shukor et al. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025. URL https://arxiv.org/abs/2506.01844.

Haofeng Sun, Jiangbo Pei, Fei Kang, et al. Riemann-1.0: An Embodied World Action Model for Physical AI. *arXiv preprint arXiv:2608.27033*, 2026. URL https://arxiv.org/abs/2608.27033.

Junke Wang, Qihang Zhang, Shuai Yang, et al. RepWAM: World Action Modeling with Representation Visual-Action Tokenizers. *arXiv preprint arXiv:2606.13674*, 2026a. URL https://arxiv.org/abs/2606.13674.

Qiuyue Wang, Mingsheng Li, Jian Guan, et al. Qwen-VLA: Unifying Vision-Language-Action Modeling across Tasks, Environments, and Robot Embodiments. *arXiv preprint arXiv:2605.30280*, 2026b. URL https://arxiv.org/abs/2605.30280.

Adina Yakefu, Bin Xie, Chongyang Xu, et al. RoboChallenge: Large-scale Real-robot Evaluation of Embodied Policies. *arXiv preprint arXiv:2510.17950*, 2025. URL https://arxiv.org/abs/2510.17950.

Yi Yang, Zhihong Liu, Siqi Kou, et al. World-Language-Action Model for Unified World Modeling, Language Reasoning, and Action Synthesis. *arXiv preprint arXiv:2606.05979*, 2026. URL https://arxiv.org/abs/2606.05979.

Seonghyeon Ye et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026. URL https://arxiv.org/abs/2602.15922.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, et al. Fast-WAM: Do World Action Models Need Test-time Future Imagination? *arXiv preprint arXiv:2603.16666*, 2026. URL https://arxiv.org/abs/2603.16666.

Weiheng Zhao et al. Faster-WAM: Efficient inference-time future conditioning for robust world action models. *arXiv preprint arXiv:2608.04404*, 2026. URL https://arxiv.org/abs/2608.04404.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, et al. X-VLA: Soft-Prompted Transformer as Scalable Cross-Embodiment Vision-Language-Action Model. *arXiv preprint arXiv:2510.10274*, 2025. URL https://arxiv.org/abs/2510.10274.

Jian Zhu, Jianjun Zhang, Taiyi Su, et al. DSWAM: A Dual-System World Action Foundation Model for Fine-Grained Robot Manipulation. *arXiv preprint arXiv:2607.04927*, 2026. URL https://arxiv.org/abs/2607.04927.

Appendix to "LiteC-WAM: A Lightweight Cross-Task and Cross-Embodiment World Action Model"

A Related Works Figure . . . 15

B Evaluation Protocols and Full Task Results . . . 15

B.1 RLBench . . . 15

B.2 LIBERO . . . 15

B.3 ARX5 Offline Protocol . . . 16

B.4 Successful Rollout Sequences . . . 16

C Training, Adaptation, and Parameter Efficiency . . . 19

C.1 Trainable Components at Each Stage . . . 19

C.2 Stored, Selected-Task, and Active Parameters . . . 19

C.3 Model and Training Details . . . 20

C.4 Training Algorithm of LiteC-WAM . . . 22

D Controlled Component Analysis . . . 23

D.1 Cumulative Model Variants . . . 23

D.2 Sparse Dynamics and Predictive Objectives . . . 23

E Paired World-Model Feedback Ablation . . . 24

F Additional World-model Diagnostics . . . 24

G Real-Robot Results and Analysis . . . 25

G.1 Task Completion and Stage Alignment . . . 25

G.2 World-Model Feedback across the Task Sequence . . . 26

G.3 Dynamic Expert Selection . . . 27

G.4 Future Images at Interaction Stages . . . 28

G.5 Global and Wrist Visual Sensitivity . . . 28

H Limitations and Scope . . . 29

I Sources and Scope of the Overview Figures . . . 29

- Appendix B gives evaluation protocols and complete task results.
- Appendix C details the model, training, adaptation, parameter counts, and training algorithm.
- Appendices D–G report controlled comparisons, feedback ablations, world-model diagnostics, and real-robot analyses.
- Appendices H and I state limitations and document the sources used in the overview figures.

A Related Works Figure

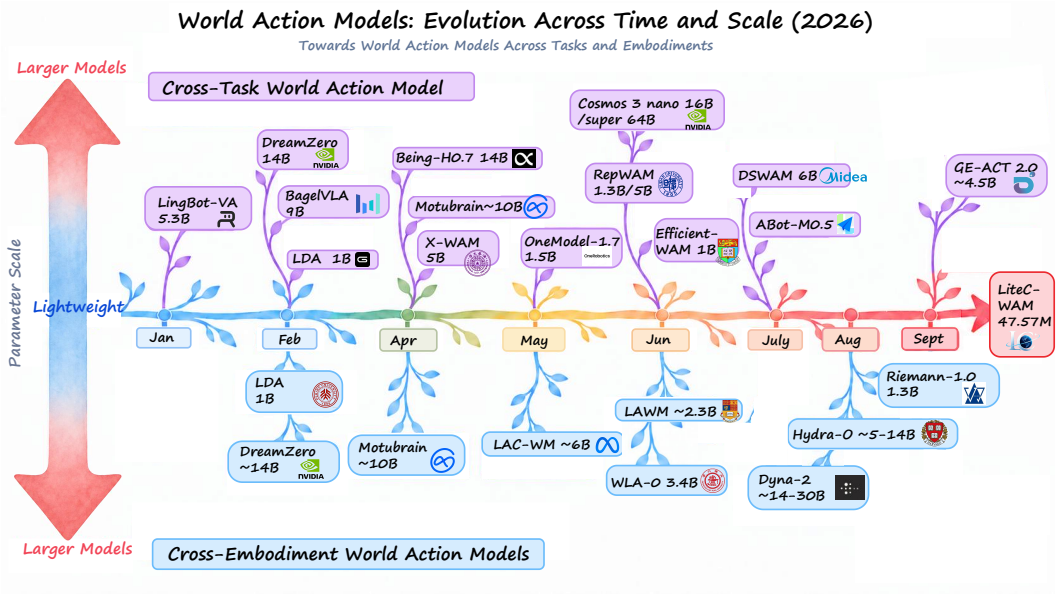


Figure 6: **Cross-task and cross-embodiment world action models.** Upper and lower branches organize multi-task and cross-embodiment work. Dates and parameter labels are schematic and are not used for quantitative comparisons; model sources appear in Appendix I.

B Evaluation Protocols and Full Task Results

This section reports the complete task-level results behind the aggregate numbers in the main paper. Closed-loop simulation comparisons use fixed initial states and benchmark-defined success criteria; the ARX5 offline metric is defined separately below. A task-specific adapter is selected once from the task instruction and remains fixed throughout the episode; we never choose adapters after observing an outcome.

B.1 RLBench

We evaluate 15 manipulation tasks using eight fixed initial states per task (seeds 730072–730079), for a total of 120 closed-loop rollouts. Table 4 reports every task rather than only the successful examples shown later.

Table 4: Task-level results on the fixed 15-task RLBench evaluation set.

Task	Success	Task	Success
Reach target	8/8	Close box	8/8
Close microwave	8/8	Plug charger	2/8
Toilet seat down	8/8	Take umbrella out	8/8
Put umbrella in	3/8	Slide cabinet open	8/8
Close fridge	8/8	Pick and lift	8/8
Open box	8/8	Open microwave	8/8
Unplug charger	8/8	Toilet seat up	6/8
Open fridge	5/8	**Total**	**104/120**

B.2 LIBERO

The representative LIBERO evaluation contains five tasks spanning spatial reasoning, object identity, and long-horizon manipulation. Each task is evaluated from ten fixed initial states. As shown in Table 5, LiteC-WAM succeeds in 48 of 50 rollouts.

Table 5: Task-level results on the representative five-task LIBERO evaluation set.

Task	Success
LIBERO-Spatial task 0	8/10
LIBERO-Spatial task 3	10/10
LIBERO-Object task 1	10/10
LIBERO-Object task 9	10/10
LIBERO-10 task 5	10/10
Total	**48/50**

Table 6 isolates the effect of the final task-specific adaptation on LIBERO-Spatial task 0. All variants are evaluated on the same 50 initial states. The final LIBERO update changes only two 128-dimensional instruction embeddings (256 scalar parameters), before freezing the checkpoint for ARX5/Franka adaptation. Restoring their values from the preceding model reduces success from 48/50 to 46/50 on the five-task evaluation. Both discordant episodes favor the adapted model, although the exact McNemar test is not significant at this sample size ($p = 0.5$).

Table 6: Matched comparison on LIBERO-Spatial task 0 using the same 50 initial states and success criterion. These are model variants, not external baselines.

Model variant	Successes	Rate	Stored params.
Compact base model	25/50	50.0%	47.57M
Improved base model	35/50	70.0%	47.57M
Seven-model specialist ensemble	38/50	76.0%	∼333M
LiteC-WAM	**43/50**	**86.0%**	**47.57M**

B.3 ARX5 Offline Protocol

The seven-task report uses the dataset's validation split, with 4,096 sampled windows per task. State inputs use joint and gripper velocities estimated from successive recorded commands and timestamps, then adjusted by calibrated scale and bias. The reported configuration uses regression-anchor weight 1.0, which bypasses the latent flow sampler. An episode passes when at least 95% of its evaluated windows have first-step errors no greater than 0.08 radians for every arm joint and 0.002 meters for the gripper. This is a sampled-window action-agreement metric, not a requirement that every trajectory step or all eight predicted actions pass. The validation split was used for model selection, so these results are not an independent blind test.

Table 7: ARX5 offline episode agreement on seven validation tasks. Each task contributes equally to the macro average.

Task	Passing / evaluated episodes	Rate
Turn on the light switch	81/82	98.78%
Hang the cup	88/101	87.13%
Press the button	97/97	100.00%
Wipe the table	99/120	82.50%
Water the flowers	88/95	92.63%
Pick out the green blocks	115/125	92.00%
Arrange flowers	99/110	90.00%
Equal-task macro average	–	91.86%

B.4 Successful Rollout Sequences

Figures 7 and 8 show six temporally ordered frames from one successful rollout for every evaluated task. The LIBERO examples include recovery after an unsuccessful first grasp in Object-9. Policy inputs remain at the trained resolution of 64×64; the higher-resolution views are recorded separately for visualization and are not provided to the policy. For the charger and refrigerator tasks, we use a clearer shoulder camera only for visualization, without changing the initial state or action sequence. These examples complement, but do not replace, the complete results in Tables 4 and 5.

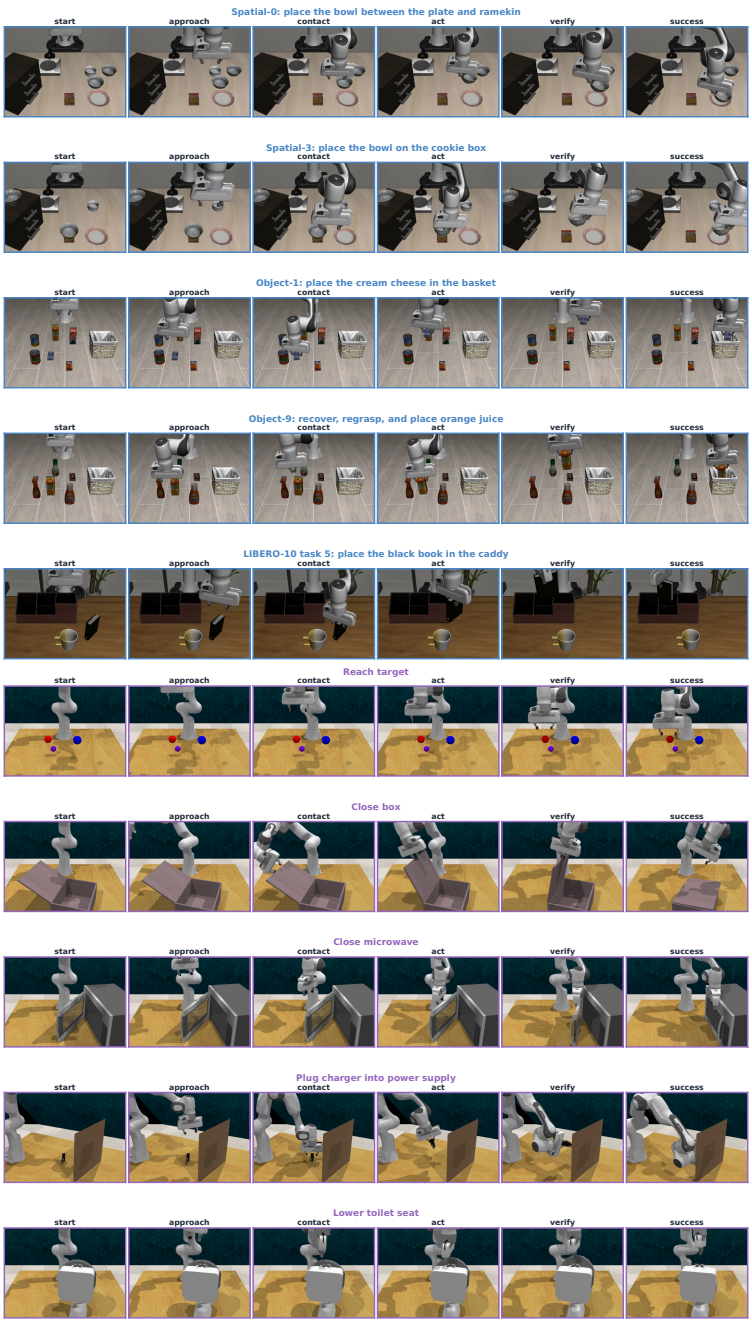


Figure 7: **Successful rollout sequences I.** Top: the five representative LIBERO tasks. Bottom: RLBench tasks 1–5 (reach target, close box, close microwave, plug charger, and lower toilet seat). Each row progresses from initialization to benchmark-defined success.

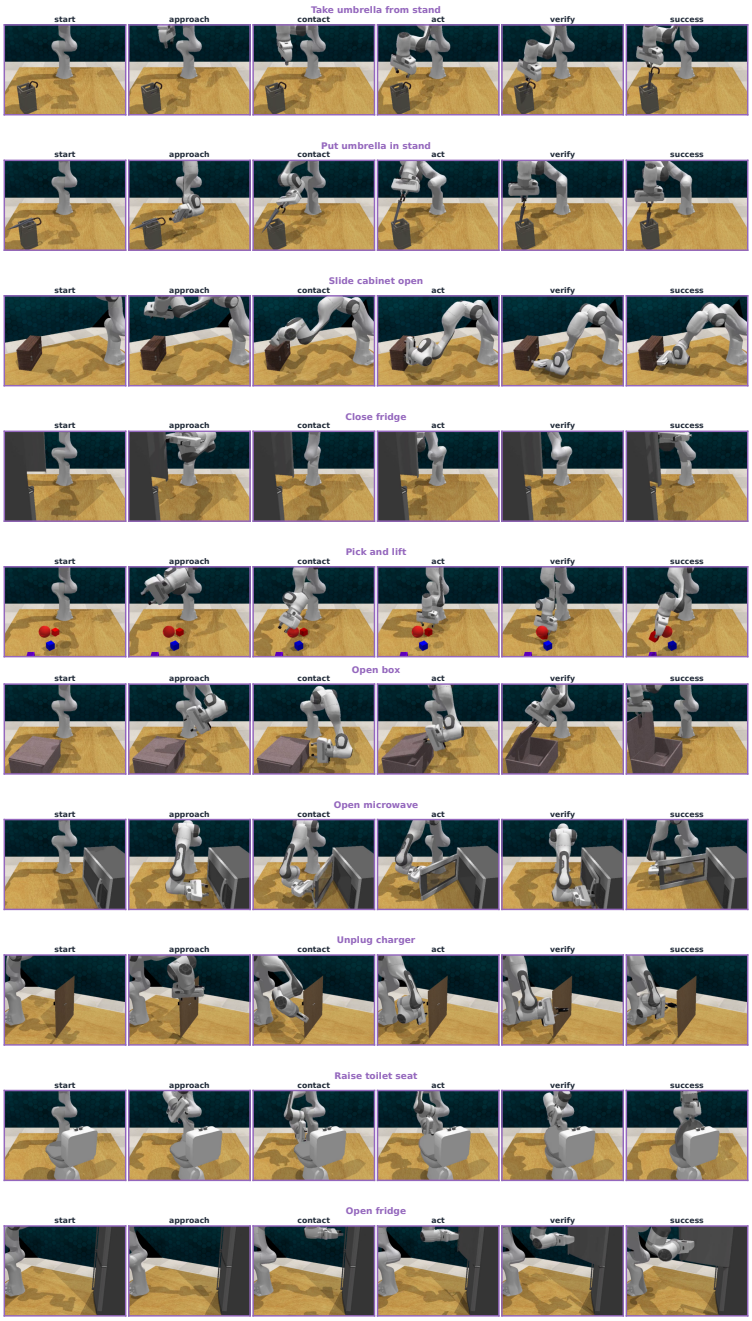


Figure 8: **Successful rollout sequences II.** Top: RLBench tasks 6–10 (take umbrella, return umbrella, slide cabinet, close refrigerator, and pick-and-lift). Bottom: tasks 11–15 (open box, open microwave, unplug charger, raise toilet seat, and open refrigerator).

Across the two benchmarks, the successful trajectories consistently show approach, contact, object motion, and task verification rather than a single favorable endpoint. The Object-9 and umbrella examples also show that closed-loop execution can recover from an imperfect first contact. Failure statistics, however, are taken from the full evaluation sets rather than inferred from these examples.

C Training, Adaptation, and Parameter Efficiency

C.1 Trainable Components at Each Stage

LiteC-WAM separates general visuomotor learning from embodiment- and task-specific adaptation. Table 8 states exactly which components receive gradients at each stage. During ARX5 and Franka adaptation, the shared backbone, including its router and experts, is frozen and future-state targets are produced by its frozen visual encoder with stop-gradient, whether computed online or loaded from a cache.

Table 8: Trainable components and objectives at each stage. A dash denotes that no additional component or loss is used.

Stage	Trainable components	Frozen components	Training objectives
Shared pretraining	visual/language encoders, GRU, action head, router, dynamics experts, prediction heads	–	flow matching, future RGB, events, phase, load balancing
ARX5 / Franka adaptation	visual/state adapters, action encoder/decoder, embodiment-specific action and dynamics heads	shared backbone	flow matching, first action, gripper state, action reconstruction, latent dynamics
LIBERO final update	two instruction embeddings (256 scalars)	all other parameters	task-specific action loss
RLBench task adaptation	selected task embedding and residual adapter	shared backbone and other task adapters	first action and task-specific action loss
LIBERO feedback training	feedback head (108,088 parameters)	shared backbone, base and context residual heads	action residual loss
SO-ARM101 refinement	embodiment modules, shared dynamics (router and experts), auxiliary decoder	visual encoder, base action encoder and decoder, remaining base modules	action, latent future, auxiliary image losses
Deployment	none	all parameters	–

Shared pretraining uses $(\lambda_{\text{rgb}}, \lambda_{\text{evt}}, \lambda_{\text{bal}}, \lambda_{\text{phase}}) = (1, 0.1, 0.01, 0.02)$. The controlled routing experiment uses 100,000 sampled LIBERO-Spatial/Object trajectory segments, 12,500 optimization steps, batch size 8, hidden width 776, and the same data order for every variant.

C.2 Stored, Selected-Task, and Active Parameters

The shared backbone contains 47,570,534 parameters. The ARX5 embodiment adapter adds 5,527,845 parameters, giving 53,098,379 in total. The Franka embodiment adapter adds 6,972,327 parameters. Its task-specific adapter bank stores another 21,568 parameters, of which 7,232 belong to the selected task; the corresponding RLBench configuration contains 54,550,093 parameters, while the full stored model contains 54,564,429. These configuration counts include all shared experts; the approximately 33.10M Top-2 active backbone count is a separate measure. The world-model feedback head contains 108,088 additional parameters, equivalent to 0.23% of the shared backbone; it is excluded from the five-task LIBERO checkpoint count. The 256 adapted instruction scalars are already included in the backbone and must not be added again. The SO-ARM101 policy contains 53,753,814 parameters; its approximately 0.34M auxiliary image decoder is counted separately.

Table 9: Parameter breakdown. "Selected task" counts a task-specific configuration, including all shared experts; it is not a count of parameters activated by Top-2 dynamics routing.

Component	Stored	Selected task	Updated during
Shared LiteC-WAM backbone	47,570,534	47,570,534	Shared pretraining
ARX5 embodiment adapter	5,527,845	5,527,845	Embodiment adaptation
Franka embodiment adapter	6,972,327	6,972,327	Embodiment adaptation
RLBench task-adapter bank	21,568	7,232	Task-specific adaptation
World-model feedback head	108,088	108,088	Feedback-head training
LIBERO instruction adaptation	256	256	Task-specific adaptation

Table 10: **Adaptation cost and performance retention.** ARX5 changes the robot; RLBench changes the environment and interface while using the same Franka Panda arm as LIBERO. Each setting uses its own metric; success on the original five LIBERO tasks is reevaluated after loading each adapter.

Adapter	Total	Added	New-domain evidence	LIBERO retained
ARX5	53.10M	5.53M	91.86% offline pass	48/50
RLBench	54.55M	6.98M	104/120 closed loop	48/50

C.3 Model and Training Details

This section specifies the inputs, network components, action mappings, and losses summarized in the main Method section.

C.3.1 Inputs and Notation

For embodiment $e \in \mathcal{E}$ and task $j \in \mathcal{T}_e$, the policy receives

$$o_t^e = \left(\{I_{t-K+1:t}^{e,v}\}_{v\in\mathcal{V}_e},\ s_{t-K+1:t}^e,\ \ell_j\right), \tag{9}$$

where $I^{e,v}$ is RGB from view v, s^e is robot state, and ℓ_j is the instruction. It predicts the robot action chunk $a_{t:t+H-1}^e$, then future visual states $z_{t+1:t+H}$ and events $y_{t+1:t+H}$ from the same context and candidate action. We use $K = 5$, $H = 8$, 64×64 images, and shared width $d = 776$.

We assess *efficiency* by total and active parameter counts, and *world-model feedback* by intervening on the action head's feedback input. *ARX5* (for cross-embodiment transfer) and *RLBench* (for cross-task transfer) adaptation enforces $\nabla_\theta \mathcal{L}_e = 0$ for shared parameters θ, with empirical retention tests. SO-ARM101 uses a different set of frozen components, specified below.

C.3.2 Encoders and Action Generation

Multimodal context. A four-block convolutional encoder ($48, 96, 192, d$ channels), shared across views, encodes camera frames. A single-layer GRU summarizes visual history; a two-layer encoder embeds robot state; learned 128-D token embeddings represent instructions. A fusion MLP produces

$$c_t = F_{\text{ctx}}\left([h_t^{\text{vis}}; h_t^{\text{state}}; h_j^{\text{lang}}]\right) \in \mathbb{R}^d. \tag{10}$$

Flow-matching action chunks. For a normalized ground-truth chunk a, Gaussian noise ϵ, and $\tau \sim \mathcal{U}[0,1]$, flow matching (Lipman et al., 2023) trains a two-hidden-layer MLP v_θ at $x_\tau = (1-\tau)\epsilon + \tau a$ toward $u_\tau^\star = a - \epsilon$:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{a,\epsilon,\tau}\left[\|v_\theta(x_\tau, \tau, c_t) - u_\tau^\star\|_2^2\right]. \tag{11}$$

Euler integration generates $H \times 7$ Cartesian-delta/gripper chunks for LIBERO; embodiment-specific heads use $H \times 16$ latent chunks. The model executes the first action or a protocol-specified number of steps before replanning.

C.3.3 Dynamics Experts and Feedback

We use sparse expert routing (Shazeer et al., 2017): Equation 5 routes each action-conditioned query to two of eight dynamics experts. Each f_m is a two-layer SiLU MLP. Only selected experts

are evaluated; a version that evaluates all experts provides equivalence and efficiency checks. During training, a transposed-convolution decoder predicts future RGB; linear heads predict reward, termination, gripper state, and one of eight manipulation phases. Phase labels derive solely from demonstration actions and terminal flags, not evaluation-time simulator state.

Equation 6 combines dynamics changes, routing weights, and predicted events to refine the preliminary LIBERO action. Training the 108,088-parameter feedback head R_w freezes the shared backbone, base action head, and context-based residual action head. Zeroing w_t preserves weights and preliminary actions for identical observations. Paired rollouts match tasks and initial states; subsequent observations and preliminary actions may diverge. Default deployment omits future RGB decoding.

C.3.4 Task/Embodiment Adaptation

After shared pretraining and the final LIBERO instruction-embedding update, ARX5 and Franka adaptations freeze all shared parameters, including the dynamics router and experts. Inspired by low-rank adaptation (Hu et al., 2022), residual low-rank adapters align observations:

$$\phi_e(x) = \phi_0(x) + B_e\,\mathrm{SiLU}(A_e x), \qquad A_e \in \mathbb{R}^{r\times d_{\text{in}}},\ B_e \in \mathbb{R}^{d\times r}, \tag{12}$$

with rank $r = 16$ and zero-initialized B_e. Robot-specific encoders and decoders map commands to and from a shared 16-D action representation:

$$z_a = C_e^{\text{enc}}(a^e), \qquad \widehat{a}^e = C_e^{\text{dec}}(z_a). \tag{13}$$

Seven robot action coordinates are encoded into 16 dimensions. Their flow and regression heads predict $H \times 16$ chunks, combined and decoded to $H \times 7$ actions; the latent flow head does not receive LIBERO head outputs. ARX5 cross-embodiment adapter outputs six absolute joint positions plus gripper. RLBench cross-task adapter applies task-specific corrections after decoding, then converts to absolute position–quaternion–gripper commands. LIBERO feedback acts directly on seven-dimensional actions.

A task-specific residual adapter adds a bounded correction to the first action:

$$\Delta a_t^{e,j} = \alpha \tanh(U_{e,j}\sigma(D_e[c_t; \widehat{a}_t^e])), \qquad \widetilde{a}_t^e = \widehat{a}_t^e + \mathbb{I}[j]\Delta a_t^{e,j}. \tag{14}$$

The instruction selects one residual adapter before execution; all others output zero. It may use RGB, robot state, execution progress, and predicted phase, but not reward, success flags, segmentation, or simulator ground truth. This selection is separate from dynamics-expert routing.

For bimanual SO-ARM101, rank-32 adapters align global and wrist RGB. A fixed mapping preserves and pads 12 normalized coordinates (five joints and one gripper per arm) to 16 dimensions. A preliminary chunk conditions dynamics; predicted dual-view latent changes refine the flow head's context. Eight Euler steps generate a latent delta, decoded to 12 dimensions and added to the current state before bounding commands. Feedback thus precedes final flow generation, unlike LIBERO's additive residual.

C.3.5 Losses and Inference

Shared pretraining jointly optimizes actions and future prediction:

$$\mathcal{L}_{\text{shared}} = \mathcal{L}_{\text{FM}} + \lambda_{\text{rgb}}\|\widehat{I} - I\|_1 + \lambda_{\text{evt}}\mathcal{L}_{\text{BCE}} + \lambda_{\text{bal}}\mathcal{L}_{\text{bal}} + \lambda_{\text{phase}}\mathcal{L}_{\text{CE}}, \tag{15}$$

where $\mathcal{L}_{\text{bal}}$ discourages router collapse. ARX5 cross-embodiment adapter and RLBench cross-task adapter use stop-gradient targets from the shared encoder, updating only observation and state adapters, action encoders and decoders, embodiment-specific flow and preliminary-action regression heads, low-rank dynamics residuals, and the selected task residual adapter. Adaptation combines flow matching, first-step and gripper losses, action reconstruction, latent Huber and cosine losses, and available phase and event supervision; missing contact labels are not synthesized.

Deployment uses five observations to generate an eight-step chunk, optionally applies feedback, executes the first action, and replans, including after failed contact; no privileged retry controller is used. The backbone stores 47.57M parameters, with approximately 33.10M active under final Top-2 routing. The models used in ARX5 and selected-task RLBench store 53.10M and 54.55M parameters respectively, including all experts. Model in five-task LIBERO uses the 47.57M shared backbone; the 20-task ablation counts the additional 108,088-parameter feedback head separately. Appendix C details training settings and gradient flow.

C.4 Training Algorithm of LiteC-WAM

Algorithm 1 summarizes one training stage. Shared pretraining produces the base checkpoint; each later run loads the checkpoint for its configuration and updates only the components listed in Table 8. The later stages are configuration-specific, not a single sequence applied to every model. Loss definitions are given in Section C.3; the SO-ARM101 refinement is described further in Appendix G.

Algorithm 1 LiteC-WAM training and adaptation

Require: Demonstrations $\mathcal{D}$; training stage s; model parameters Θ (initialized for shared pretraining, otherwise loaded); number of updates N_s.
Notation: Θ_s denotes the parameters updated in stage s.

Select Θ_s and the training objective from Table 8.
if s is shared pretraining **then**
    Train the encoders, action head, router, experts, and prediction heads.
else if s is ARX5 or RLBench adaptation **then**
    Freeze the shared backbone, including the router and all experts.
    Train the modules for this stage; keep other task adapters frozen.
else if s is the final LIBERO instruction update **then**
    Update only the two selected instruction embeddings.
else if s is LIBERO feedback training **then**
    Update only R_w; freeze the backbone and existing action heads.
else if s is SO-ARM101 refinement **then**
    Train embodiment modules, shared dynamics, and the auxiliary decoder.
    Keep the visual encoder and remaining base modules frozen.
end if
for update $k = 1, \ldots, N_s$ **do**
    Sample a batch of observation histories, instructions, and future targets.
    Encode the histories and instructions into the context c_t.
    if s is shared pretraining **then**
        Compute $\mathcal{L}_{\mathrm{FM}}$ from demonstrated actions (Equation 2).
        Route queries conditioned on demonstrated actions to two experts.
        Predict future images, events, and stages; compute $\mathcal{L}_{\mathrm{shared}}$.
    else if s is LIBERO feedback training **then**
        Predict preliminary actions and their future effects; form w_t.
        Correct actions using Equation 6; compute the action residual loss.
    else
        Apply the selected robot/task modules and predict robot actions.
        Compute the action and available prediction losses for stage s.
        For ARX5 adaptation, RLBench task adaptation, and SO-ARM101 refinement, keep visual targets fixed
    end if
    Backpropagate the stage loss and take an optimizer step on Θ_s only.
end for
return The updated model and its configuration-specific modules.

Fixed visual targets are computed by the frozen visual encoder or loaded from its cache, with no gradient through the targets. The final LIBERO instruction update uses an action loss only. For ARX5 and RLBench, task instructions select adapters independently of dynamics-expert routing. The five-task LIBERO result omits feedback-head training; the separate 20-task ablation evaluates that head.

D Controlled Component Analysis

D.1 Cumulative Model Variants

Figure 9 and Table 11 summarize cumulative improvements under fixed evaluation conditions. LIBERO-Spatial task 0 always uses the same 50 initial states. The same 15 tasks and eight initial states per task are always used in RLBench. Each row therefore measures a complete model variant on the same episodes, rather than selecting the best adapter separately for each rollout. The largest remaining errors occur in precise plug insertion, umbrella placement, refrigerator opening, and toilet-seat lifting.

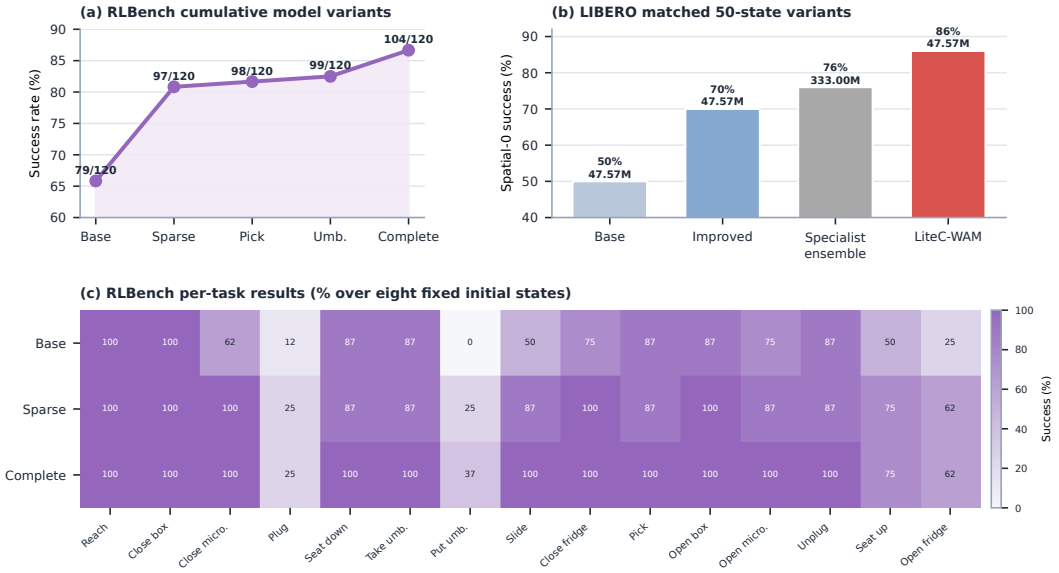


Figure 9: **Cumulative model improvements under fixed evaluation conditions.** (a) RLBench performance on 120 closed-loop rollouts, (b) matched LIBERO-Spatial task-0 performance with stored parameter counts, and (c) per-task RLBench performance. Each percentage in (c) is computed over eight fixed initial states.

Table 11: Cumulative RLBench model variants evaluated on the same 120 closed-loop rollouts.

Variant	Added capability	Success	Rate
Base policy	Shared policy with the RLBench embodiment adapter	79/120	65.83%
Sparse expert routing and task adaptation	Task-conditioned dynamics experts and contact-sensitive residual adapters	97/120	80.83%
Pick-task adaptation	Factorized pick actions and corrected terminal pose	98/120	81.67%
Umbrella adaptation	Task-specific terminal action for umbrella placement	99/120	82.50%
Complete task-adapter set	**Contact-sensitive adapters selected from task instructions**	**104/120**	**86.67%**
Unified inference model	All adapters integrated in one model with sparse expert routing	104/120	86.67%

D.2 Sparse Dynamics and Predictive Objectives

Table 12 compares dense dynamics with Top-1 and Top-2 sparse expert activation using identical trajectory segments, optimization steps, and data order. Top-2 routing reduces active parameters from 46.68M to 32.40M while maintaining comparable prediction error and control success. Removing the world-model objectives increases future-frame error from 0.0485 to 0.3798. Its higher small-sample success count (54/120) shows that this experiment alone does not establish a control benefit from predictive training; Section E separately tests the feedback input, without isolating the contribution of predictive training itself.

Table 12: Matched dynamics comparison using the same trajectory segments and optimization budget. Stored and active parameter counts refer to the model used in this experiment.

Dynamics variant	Stored	Active	Future L_1	Success
Dense	46.68M	46.68M	0.0479	49/120
Sparse Top-1	46.86M	29.99M	0.0483	48/120
Sparse Top-2	46.86M	**32.40M**	0.0485	50/120
Top-2 without world-model objectives	46.86M	32.40M	0.3798	54/120

E Paired World-Model Feedback Ablation

We investigate whether predicted forward dynamics enhance action selection under strictly controlled conditions—holding model weights, tasks, and initial environment states fixed. We first train a lightweight context-conditioned residual action head and freeze it. The world-model feedback head then receives a summary vector w_t (Equation 6), which aggregates three signals averaged over the eight predicted steps: (1) mean latent feature differences, (2) routing probabilities across all eight dynamics experts, and (3) predicted probabilities for reward, episode termination, and gripper closure.

During closed-loop evaluation, our paired comparison isolates this feedback: the two variants are otherwise identical, differing solely in whether w_t is provided or masked with zeros ($w_t = 0$). While both models produce identical preliminary action chunks from the initial observation, their closed-loop trajectories subsequently diverge as world-model feedback actively refines ongoing execution.

Table 13: Paired world-model feedback ablation across six independently trained models. Each row evaluates 20 tasks from four fixed initial states.

Seed	With world-model feedback	Feedback zeroed	Difference
20260829	65/80	62/80	+3
20260830	70/80	54/80	+16
20260831	61/80	52/80	+9
20260832	58/80	47/80	+11
20260833	42/80	49/80	−7
20260834	57/80	55/80	+2
Total	**353/480**	**319/480**	**+34**

World-model feedback improves five of the six independently trained models and raises aggregate success from 319/480 to 353/480. The seed-level sign test is not significant ($p = 0.219$), so we do not claim improvement for every training run. Instead, the paired episode analysis uses the 94 discordant rollout pairs, for which all model parameters and initial conditions are matched.

F Additional World-model Diagnostics

We further test whether the predictive branch captures action-conditioned task dynamics independently of the benchmark success signal. The following visualization uses the model's unmodified outputs, and the quantitative comparisons use held-out trajectory segments.

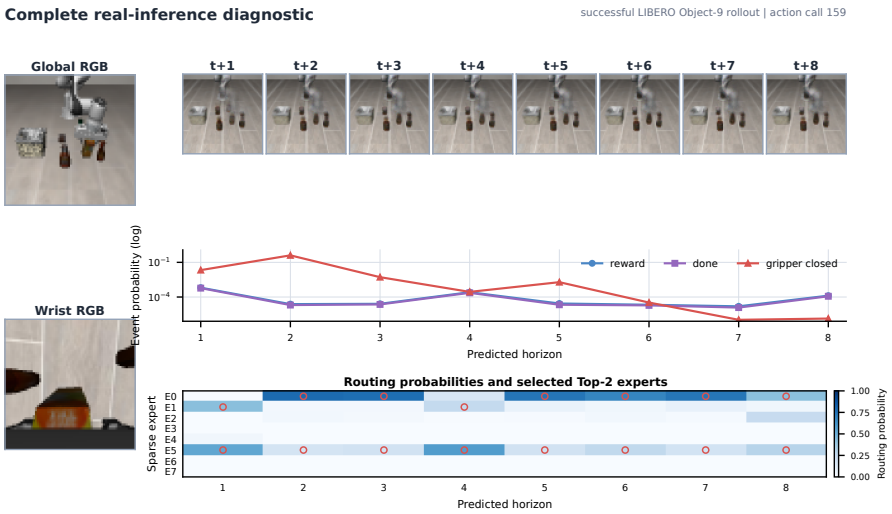


Figure 10: **World-model predictions during a successful LIBERO-Object task-9 rollout.** At control step 159, we show the current observations, eight raw future-frame predictions, event probabilities, and the selected Top-2 dynamics experts. No image enhancement or generative post-processing is applied.

Offline Discriminative Evaluation. On 1,920 held-out LIBERO trajectory segments, we evaluate the world model's discriminative ability by comparing ground-truth demonstrated actions against temporally shifted action chunks from the same task. Without access to privileged simulator success labels, the world model assigns higher predictive scores to the demonstrated action in 68.49% of future-frame comparisons, 70.00% of event predictions, 80.94% of manipulation-phase and routing assessments, and 60.99% of final-frame predictions.

Limitations of Naive Selection and Benchmark Reporting. While these predictions capture meaningful physical dynamics, they do not directly translate into policy improvements via naive action selection. Reranking four candidate action chunks based purely on predicted events yields 55.83% success (67/120 rollouts) using mean event scores, compared to 53.33% (64/120) using maximum predicted reward or termination probabilities (paired McNemar's test, $p = 0.736$).

Similarly, attaching a newly trained residual head conditioned on dynamics-expert features to the frozen five-task LiteC-WAM policy yields 47/50 successes at both residual scales (0.025 and 0.005), compared to 48/50 when this feature is zeroed out. Because performance on this five-task subset is already near saturation, we report the base 96.0% (48/50) success rate for the unmodified policy, and evaluate the full **+7.08 percentage-point gain** from closed-loop world-model feedback within the broader, un-saturated 20-task paired ablation (Section E).

G Real-Robot Results and Analysis

G.1 Task Completion and Stage Alignment

In cross-embodiment transfer using SO-ARM101, the three tasks require different forms of bimanual coordination. Tape removal involves opening the box, lifting the tape, transferring it to the other arm, and releasing it on the table. Tape placement requires moving the tape into the box and releasing it. Bottle-cap placement requires one arm to support the bottle while the other picks up and aligns the cap, followed by returning the bottle to the table. Table 14 reports completion counts; Figure 5 provides one continuous task sequence for each task.

Stages are labeled from the recorded images rather than predicted event classes. Global-camera frames are stored at 640×480 resolution; the policy encodes 64×64 views, while the auxiliary renderer outputs 128×128 images. Figure 5 retains full global frames, including occlusions. In the bottle task, the cap contact is partly outside the global view, so the sequence shows visible

Table 14: **Closed-loop real-robot task success.** Each task retains its own trial denominator. The illustrated recordings show task stages, rather than all repeated trials.

Task	Completion criterion	Successes / trials	Rate
Tape removal + handover	Tape transferred and released on the table	18/20	90.0%
Tape placement	Tape released inside the box	26/30	86.7%
Bottle-cap placement	Cap placed; bottle returned to the table	11/17	64.7%

The two-sided 95% Wilson intervals are 69.9–97.2% for tape removal and handover, 70.3–94.7% for tape placement, and 41.3–82.7% for bottle-cap placement.

before/after states rather than thread engagement. No super-resolution, image synthesis, or frame interpolation is used for the task-stage panels.

Table 15: **Real-robot analysis coverage.** A window contains an observation history and an eight-step action target. Frame indices identify the corresponding grasp, transfer/placement, and release/retraction phases.

Task	Recorded frames	Analysis windows	Selected frames
Tape removal + handover	491	63	170 / 240 / 385
Tape placement	805	102	300 / 415 / 640
Bottle-cap placement	941	118	100 / 320 / 540

Mechanism-analysis protocol. The following analyses use a single adapted 53.75M-parameter policy on recordings from the physical experiments. In this refinement, the visual encoder remains frozen, while shared dynamics (including router and experts) and embodiment modules are trainable; retention of the original LIBERO policy was not tested for this refinement. The recordings also enter adaptation: full-record summaries include training and validation windows, while the separately reported validation blocks have disjoint context and target frames within each recording. These blocks assess within-recording prediction, not generalization to independent episodes. The 0.34M image decoder is auxiliary, and recorded-future features are used only as a diagnostic input. Action and image errors supplement the physical success counts in Table 14; they do not define task success.

G.2 World-Model Feedback across the Task Sequence

We compare the same action head with predicted future features, zero feedback, and features extracted from the recorded future. The observation context, model weights, and integration method remain fixed. Action MAE averages absolute error over both arms and eight predicted steps; arm joints use the controller's $[-100, 100]$ scale and grippers use $[0, 100]$, not degrees. Tasks receive equal weight in the aggregate.

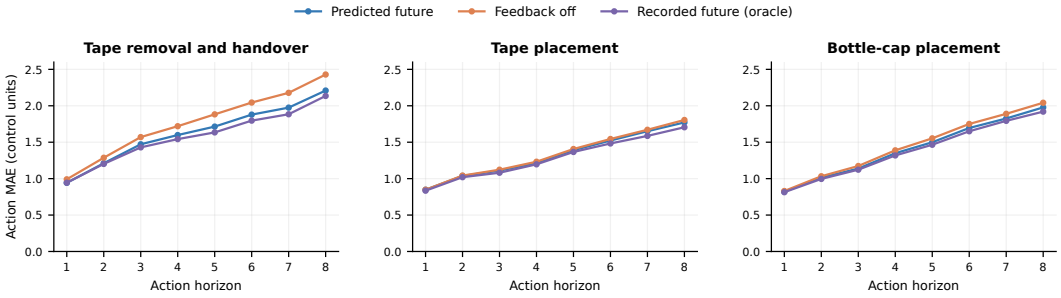


Figure 11: **World-model feedback analysis of real-robot tasks.** Action error over eight horizons for each task. Features from the recorded future provide a diagnostic reference; action targets are the commands recorded in the corresponding task sequence.

Predicted feedback improves the task mean by 4.3% relative to zero feedback, and recorded-future features reduce it by another 2.7%. The improvement is not uniform across every data split: on the fixed validation blocks, the corresponding errors are 3.200, 3.288, and 3.215 for predicted, zero, and recorded-future feedback. Thus recorded-future substitution is informative, but not a guaranteed upper bound. In the same validation comparison, adaptation reduces tape-placement MAE from

Table 16: **Action agreement on real-robot trajectories.** Lower MAE is better. The shaded row is the predicted-feedback condition; the recorded-future condition is a diagnostic reference.

World input	Tape removal	Tape placement	Bottle cap	Task mean
Feedback off	1.763	1.334	1.457	1.518
Predicted future	1.625	1.318	1.415	1.453
Recorded future	1.571	1.284	1.385	1.413

4.823 to 2.274 and the three-task mean from 4.242 to 3.200. The completed executions and model-response measurements answer different questions; MAE is not used as a success criterion.

G.3 Dynamic Expert Selection

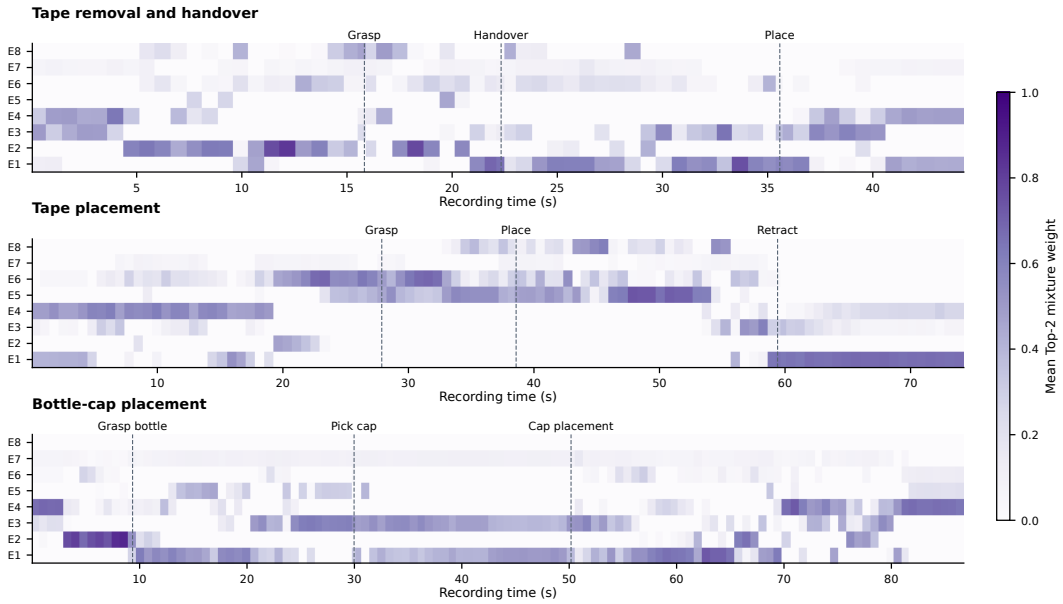


Figure 12: **Expert routing through real-robot task stages.** Each column averages the actual Top-2 mixture weights over the eight predicted steps. Dashed markers locate manually labeled task phases. More than two rows can be nonzero after averaging, although exactly two experts are active at each step.

The active mixture changes within each task and differs across tasks. Tape placement, for example, transitions from a sustained E4 contribution to E5/E6 and later E1; bottle-cap placement uses a different sequence. These patterns show dynamic selection rather than a single fixed expert pair. They do not assign semantic labels to experts: similar phases can use overlapping mixtures.

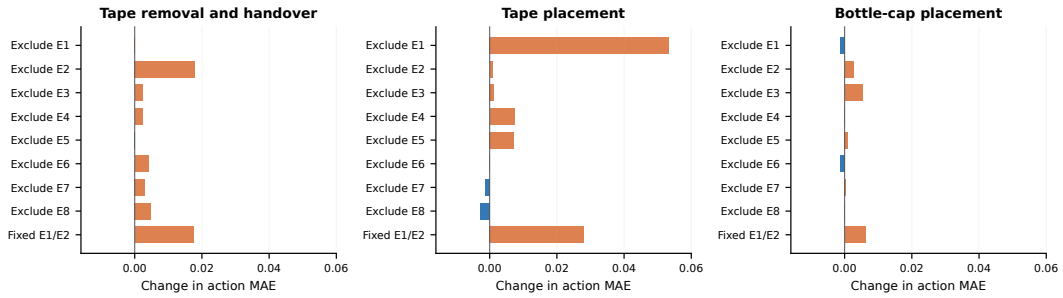


Figure 13: **Expert interventions on real-robot trajectories.** Change in action MAE when an expert is excluded from selection or routing is fixed to E1/E2. Exclusion masks its router logit before reselecting the Top-2 experts. Positive values indicate degradation; negative values indicate improvement. All tasks share one horizontal scale.

The intervention effects are small and task dependent. Excluding E2 increases tape-removal MAE by 0.018, whereas excluding E1 increases tape-placement MAE by 0.053. Some exclusions slightly improve prediction. Fixing E1/E2 increases error for all three tasks, by 0.017, 0.028, and 0.006, respectively. This supports the usefulness of the selected mixture for action agreement, without showing distinct expert skills or higher task success than using all experts.

G.4 Future Images at Interaction Stages

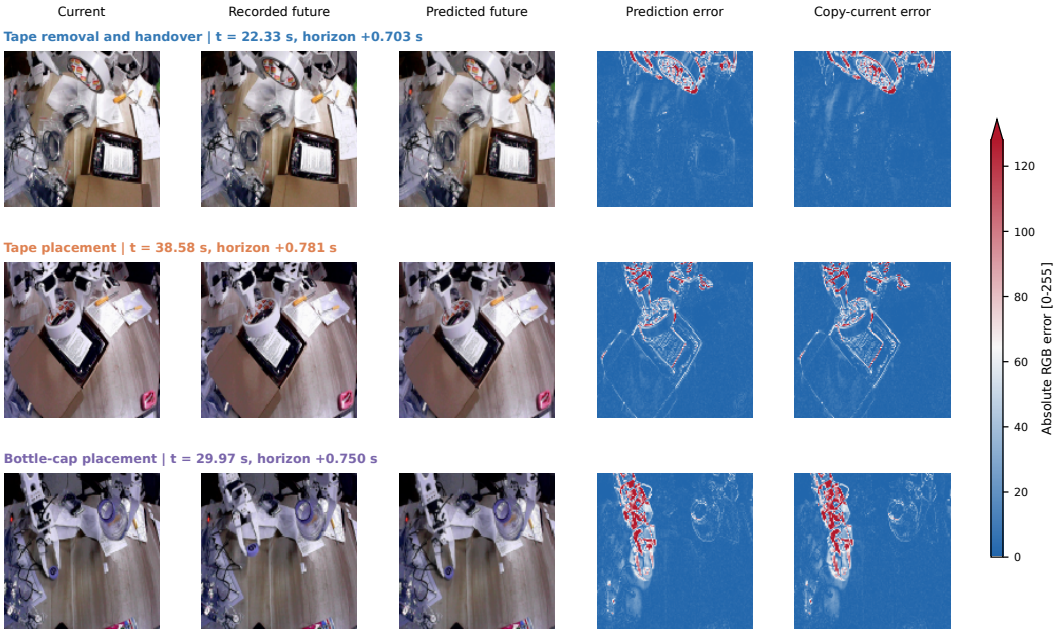


Figure 14: **Future-image analysis at real-robot interaction stages.** From left: current observation, recorded future, model prediction conditioned on the recorded actions, prediction error, and copy-current error. Horizons follow actual recording timestamps. All error panels use the same scale; values above 128 saturate.

We inspect tape handover, tape placement, and cap pickup at the selected frames in Table 15. The auxiliary renderer receives the current RGB image and predicted latent change, preserving visible texture while forecasting motion. Figure 14 uses recorded-action conditioning so that predictions and targets refer to the same action sequence. The output resolution is 128×128; no generated detail is added afterward.

The model preserves scene appearance, but fine motion remains difficult. On validation blocks, motion-weighted RGB error is 22.150 versus 22.988 for copying the current frame, while unweighted RGB error is 14.326 versus 14.221. Removing the future change from the same renderer gives motion-weighted error 22.126. Consequently, sharp appearance alone does not demonstrate accurate dynamics: current-image information explains much of the reconstruction, and precise contact prediction remains a limitation.

G.5 Global and Wrist Visual Sensitivity

Global and wrist observations influence the action head differently. The selected tape-handover frame is more sensitive to several global-view patches, whereas responses in the other two tasks are weaker under the same scale. This is a perturbation-based sensitivity measurement, not an attention map, segmentation mask, or object-location estimate. A response near an interaction region is compatible with visual dependence, but does not by itself establish spatial understanding or robustness to an occluded camera.

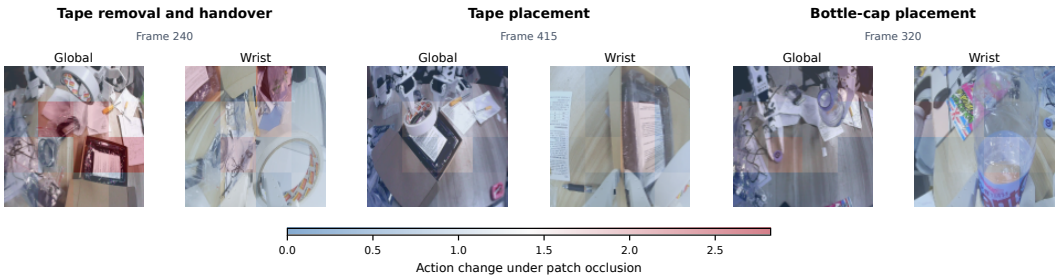


Figure 15: **Dual-view sensitivity during real-robot tasks.** Each heatmap measures the change in predicted actions when one cell of a 4×4 image grid is replaced by the frame's mean RGB value across the five context frames. Warmer cells indicate larger action changes; all panels share one scale.

H Limitations and Scope

The 96.0% LIBERO result is measured on a representative five-task subset rather than the official four-suite macro average. The RLBench result uses a fixed 15-task evaluation set, and the ARX5 study measures offline action-prediction agreement rather than closed-loop hardware success. On RLBench, most remaining errors occur in plug insertion, umbrella placement, refrigerator opening, and toilet-seat lifting, suggesting that narrow contact tolerances and articulated-object constraints remain challenging. Sparse routing reduces active computation, but the present 120-rollout comparison does not show a statistically significant improvement in task success. Finally, the rollout sequences are qualitative examples; all reported success rates are computed from the full evaluation sets. Future work should include the complete official benchmark suites and standardized on-device latency measurements, and extend the current closed-loop real-robot evaluation to more tasks, embodiments, and trial repetitions.

I Sources and Scope of the Overview Figures

LIBERO model comparison. Figure 1 and Table 1 include Diffusion Policy (Chi et al., 2023), Octo (Octo Model Team et al., 2024), OpenVLA (Kim et al., 2025b), π_0 (Black et al., 2024), π_0-FAST (Pertsch et al., 2025), OpenVLA-OFT (Kim et al., 2025a), and three SmolVLA sizes (Shukor et al., 2025). Other systems are Evo-1 (Lin et al., 2025), Qwen-VLA (Wang et al., 2026b), X-VLA (Zheng et al., 2025), VLA-GSE (Jiang et al., 2026), LingBot-VA (Li et al., 2026c), Light-WAM (Li et al., 2026d), Fast-WAM (Yuan et al., 2026), Faster-WAM-Shallow (Zhao et al., 2026), and Cosmos Policy (Kim et al., 2026). Diffusion Policy/Octo/OpenVLA scores follow the SmolVLA comparison; fine-tuned π_0/π_0-FAST/OpenVLA-OFT scores follow the OFT study. These results do not share our five-task protocol. As indicated in Figure 1, the default model architecture in LIBERO consists of 47.57M parameters. Appendix C gives configuration totals.

Cross-task branch. The upper branch of Figure 6 includes Cosmos 3 (NVIDIA, 2026), LingBot-VA (Li et al., 2026c), DreamZero (Ye et al., 2026), BagelVLA (Hu et al., 2026), Motubrain (Motubrain Team et al., 2026), DSWAM (Zhu et al., 2026), RepWAM (Wang et al., 2026a), Efficient-WAM (Li et al., 2026b), GE-Act 2.0 (AgiBot Research Team et al., 2026), Being-H0.7 (Luo et al., 2026), ABot-M0.5 (Chen et al., 2026b), X-WAM (Guo et al., 2026), OneModel 1.7 (OneRobotics, 2026), and LDA-1B (Lyu et al., 2026). BagelVLA is a related VLA that integrates visual forecasting with action generation; OneModel is cited to its official release page.

Cross-embodiment branch. The lower branch includes DreamZero (Ye et al., 2026), LDA-1B (Lyu et al., 2026), LAC-WM (Huang et al., 2026), LaWAM (Chen et al., 2026a), WLA-0 (Yang et al., 2026), Hydra-0 (Li et al., 2026a), Dyna-2 (Dyna Robotics, 2026), Riemann-1.0 (Sun et al., 2026), and Motubrain (Motubrain Team et al., 2026). Cross-robot video transfer and multi-embodiment world prediction are distinct from zero-shot control of an unseen robot. LaWM denotes a component of LaWAM; repeated entries refer to the same work.