WAM-TTT: Steering World-Action Models by Watching Human Play at Test Time

Yusen Feng[1,2,*] **Bingchen Han**[1,2,*] **Jiangran Lyu**[1,2,*]
Kai Liu[2,3] Yixin Zheng[2,3] Yuxuan Wan[1,2] Weiheng Liu[2,3] Sun Han[1,2] Ruiqin Li[1,2]
Yulong Zhang[1] Fangfu Liu[4] Xuesong Shi[2] Libin Liu[1,†] Yizhou Wang[1,†] Zhizheng Zhang[2,†] He Wang[1,2,†]

[1]Peking University [2]Galbot [3]CASIA [4]Tsinghua University
[*]Equal contribution [†]Corresponding authors

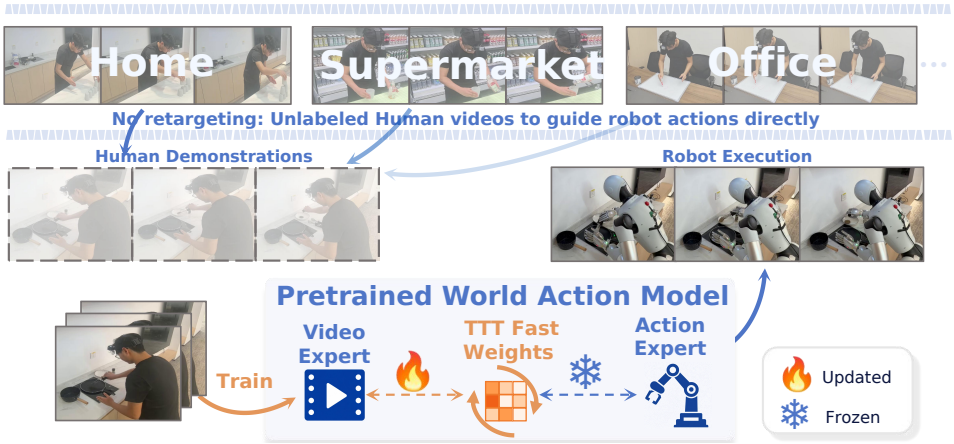

Figure 1: Overview of WAM-TTT. Given unlabeled human demonstrations from diverse environments, WAM-TTT steers a pretrained World Action Model (WAM) without retargeting, robot actions, or human-side annotations. During deployment, human videos are absorbed into lightweight TTT fast weights through self-supervised video prediction, while the pretrained action model remains frozen. The adapted memory then guides robot execution through the WAM's shared visual-action dynamics, enabling efficient and reusable steering from human demonstrations.

Abstract: Steering robot foundation models (RFMs) toward new task variants or user-preferred behaviors remains challenging, often requiring additional robot demonstrations, task-specific fine-tuning, or long-context conditioning. We present WAM-TTT, a test-time training framework for steering world action models from raw human videos. Rather than treating human videos as trajectories to imitate, WAM-TTT absorbs them into a lightweight adaptive memory inside a frozen WAM through self-supervised video prediction. To make this memory useful for control, we introduce a meta-training stage that aligns human demonstrations with robot behaviors using paired human-robot data and a key–value memory reconstruction objective. At test time, only unlabeled human videos are required to adapt the memory, while the pretrained WAM remains frozen. This enables efficient and reusable steering without robot actions, human-side annotations, or task-specific fine-tuning, while preserving the generalization ability of the foundation model. Extensive experiments show that WAM-TTT consistently outperforms in-context human-video conditioning baselines across diverse manipulation tasks and generalization settings.

Keywords: World Action Model, Test-time Training, Human Videos

1 Introduction

Recently, the robotics community has increasingly pursued general-purpose robot foundation models through large-scale pretraining. However, most existing RFMs primarily absorb knowledge into fixed model parameters. Once deployed, their behavior is largely determined by the pretrained weights and a limited conditioning interface, such as language instructions, goal images, or short observation histories[1, 2, 3, 4, 5, 6]. As a result, steering RFMs toward new task variants, object interactions, or user-preferred strategies typically requires collecting additional robot demonstrations or fine-tuning the full model. This limits the flexibility and reusability of RFMs in open-ended deployment settings, where users may wish to quickly specify new behaviors without retraining a robot policy.

Human demonstrations offer a natural and scalable interface for steering RFMs[7, 8, 9, 10, 11, 12]: users can simply show how objects should be handled, without specifying robot actions. Existing methods typically leverage human videos through co-training or fine-tuning with robot data [7, 8, 9, 10, 13, 11, 12], often relying on additional supervision such as hand poses, 3D motion, or retargeted trajectories [2, 14, 3, 15, 4, 6, 16, 17]. Such supervision can be noisy and costly to obtain, while task-specific fine-tuning may cause catastrophic forgetting and reduce the reusability of the pretrained model. A more direct alternative is to condition robot policies on raw human videos [18], but this requires learning such capabilities during large-scale pretraining and incurs rapidly growing context lengths as demonstrations accumulate.

To address these challenges, we propose WAM-TTT, a test-time training framework for steering world action models (WAMs) with human demonstrations. Rather than treating human videos as trajectories to imitate, WAM-TTT uses them as deployment-time memory learned through video prediction. Since WAMs jointly model visual dynamics and actions, the adapted memory can steer action generation through the model's shared video-action representation. To make this adaptation useful for control, we introduce a meta-training stage that aligns human demonstrations with robot behaviors using paired human-robot data and a key–value memory reconstruction loss. At deployment, only human videos are required: the memory is updated through video prediction, while the pretrained WAM remains frozen. As a result, WAM-TTT enables efficient and reusable steering of WAMs toward new task variants while preserving the generalization ability of the foundation model.

Extensive experiments show that WAM-TTT consistently outperforms in-context-learning-based human-video conditioning baselines. Our ablations further demonstrate the importance of test-time memory adaptation, the video prediction objective, and the key–value memory reconstruction loss, highlighting the effectiveness of our design for steering pretrained WAMs. Our contributions are threefold.

1. We formulate human-video-based steering of world action models as a test-time training problem, enabling deployment-time adaptation from raw human demonstrations without robot actions.
2. We propose WAM-TTT, a plug-and-play TTT memory that absorbs human videos into a frozen WAM through self-supervised video prediction, together with a human-robot alignment objective that makes the learned memory useful for robot control.
3. We demonstrate that WAM-TTT enables efficient and reusable steering from human demonstrations, outperforming in-context video conditioning while avoiding additional human-side annotations and full-model fine-tuning.

2 Related Work

World Action Models. World models have become an increasingly important interface for robot learning, as they provide a predictive substrate for reasoning about how actions change future observations[19, 20]. Recent world-action models (WAMs) go one step further by coupling future visual prediction with action generation, enabling policies to be grounded in imagined state transi-

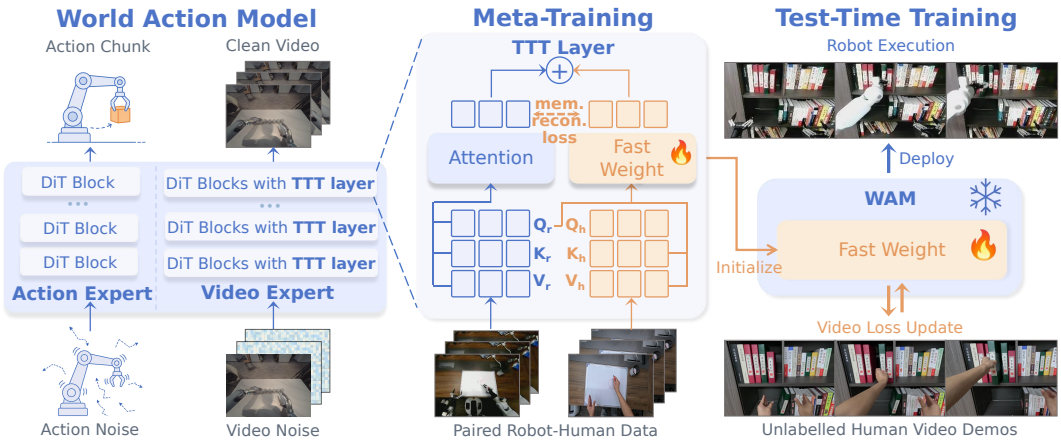

Figure 2: Pipeline of WAM-TTT. We first meta-train a fast-weight memory using paired human-robot demonstrations, encouraging human visual cues to align with robot behaviors through a key–value memory reconstruction objective. At test time, the memory is adapted from unlabeled human videos via video prediction, while the pretrained WAM remains frozen. The adapted memory then steers robot execution through the WAM's shared visual-action dynamics.

tions rather than in purely reactive action prediction [14, 17, 16, 21, 22, 23, 24, 25, 26, 23, 24, 27]. This coupling also makes video a natural supervision signal: action-free videos can improve visual-dynamics representations, while robot trajectories can anchor those representations to executable actions [14, 17, 16, 21]. However, existing world action models primarily focus on pretraining but ignoring the steerability during deployment.

Test-time training and adaptive memory. Test-time training adapts models using signals derived from test inputs, typically through self-supervised or entropy-based objectives under distribution shift [28, 29, 30, 31, 32]. Recent work reframes this idea as memory: TTT layers and related fast-weight mechanisms store information from the current sequence in adaptive parameters rather than relying only on fixed activations or explicit KV caches [32, 33, 34]. Robotics has also explored test-time adaptation through auxiliary losses, visual model-based objectives, or online environment feedback [5, 35, 36, 37]. These methods typically adapt from robot observations, rewards, or interaction rollouts. WAM-TTT instead uses human demonstrations as the test-time information source. Rather than updating the full policy online, we calibrate a TTT branch during pre-training so that, at inference, human-video Key/Value features can act as a residual skill memory inside the WAM.

Learning from human video. Human videos provide scalable evidence about objects, contacts, and task progress, but transferring them to robots is difficult because human motion is not directly executable by a robot. Prior work addresses this gap by using human data for training-time representation learning, cross-domain alignment, or policy supervision [7, 8, 9, 10, 13, 11, 12]. Other methods use human videos more directly, but often require hand pose, RGB-D motion, retargeted trajectories, object flow, latent plan extraction, generated videos, robot demonstrations, or online interaction [2, 14, 3, 15, 4, 6, 16, 17, 18]. In contrast, WAM-TTT treats a small set of unseen and unlabelled human play videos as deployment-time skill memory. The videos are not converted into robot actions or explicit human poses; instead, they provide Key/Value context to a calibrated TTT cross-attention branch, enabling skill transfer without retargeting, generated demonstrations, robot-context examples, interaction rollouts, or full deployment-time fine-tuning.

3 Method

3.1 Architecture

World Action Model. We build WAM-TTT on top of LDA [22], a pretrained world-action model (WAM). Each diffusion transformer block in LDA contains two coupled experts: a *video expert* operating on visual latent tokens, and an *action expert* operating on robot action tokens. The two experts communicate through joint attention. We denote the video and action tokens at block ℓ as $\boldsymbol{z}^{(\ell)}$ and $\boldsymbol{x}^{(\ell)}$, and the original LDA block output as $\hat{\boldsymbol{z}}^{(\ell+1)}$ and $\hat{\boldsymbol{x}}^{(\ell+1)}$.

Video TTT layer. We keep the pretrained WAM architecture unchanged except for adding TTT residual branches to the video expert. For a TTT-augmented block, the output is

$$\boldsymbol{z}^{(\ell+1)} = \hat{\boldsymbol{z}}^{(\ell+1)} + \Delta \boldsymbol{z}^{(\ell)}_{\mathrm{TTT}}, \qquad \boldsymbol{x}^{(\ell+1)} = \hat{\boldsymbol{x}}^{(\ell+1)}. \tag{1}$$

Each TTT layer follows the fast-weight memory formulation in prior TTT layers [28, 29, 32, 34, 33]. It contains slow projections $\theta^{(\ell)}_K, \theta^{(\ell)}_V, \theta^{(\ell)}_Q, \theta^{(\ell)}_O$ and a fast-weight network $f_{W^{(\ell)}}$. Given context tokens, the layer constructs Keys and Values; given the current video tokens, it constructs queries. After the fast weights are updated by the stage-specific TTT objective, the layer applies the fast-weight network to the video Queries:

$$\Delta \boldsymbol{z}^{(\ell)}_{\mathrm{TTT}} = \theta^{(\ell)}_O f_{W^{(\ell)}}\left(\theta^{(\ell)}_Q(\boldsymbol{z}^{(\ell)})\right). \tag{2}$$

3.2 Human-Robot Meta-Training

Meta-training objective. Given a paired human-robot demonstration, we denote the action-free human video clip by $\boldsymbol{u}_h$ and the synchronized robot trajectory by its actions and observations. Since each TTT block's video output (Eq. 1–2) is shifted by the residual $\theta^{(\ell)}_O f_{W^{(\ell)}}(\theta^{(\ell)}_Q(\boldsymbol{z}^{(\ell)}))$, the fast weights $\{W^{(\ell)}\}$ enter the per-block video stream and therefore propagate through to both (i) the final-block video latent $\boldsymbol{z}^{(L)}$, on which the human video prediction loss $\mathcal{L}^{\mathrm{human}}_{\mathrm{vg}}$ is computed, and (ii) the per-layer key–value memory reconstruction loss $\mathcal{L}^{(\ell)}_{\mathrm{KVM}}$ defined below that probes how well f_W maps human Keys to human Values. We adapt the fast weights on the combined inner-loop signal of these two objectives.

Key–value memory reconstruction loss. For each TTT layer ℓ, synchronized human tokens are projected into keys and values, while robot video tokens are projected into queries:

$$\boldsymbol{K}^{(\ell)}_h = \theta^{(\ell)}_K(\boldsymbol{h}^{(\ell)}_\phi), \qquad \boldsymbol{V}^{(\ell)}_h = \theta^{(\ell)}_V(\boldsymbol{h}^{(\ell)}_\phi), \qquad \boldsymbol{Q}^{(\ell)}_r = \theta^{(\ell)}_Q(\boldsymbol{z}^{(\ell)}_r).$$

The per-layer memory reconstruction loss measures how well the current fast weights reconstruct the human values from the human keys:

$$\mathcal{L}^{(\ell)}_{\mathrm{KVM}}(W_i) = \frac{1}{BL_h d}\left\| f_{W^{(\ell)}_i}(\boldsymbol{K}^{(\ell)}_h) - \boldsymbol{V}^{(\ell)}_h \right\|^2_2. \tag{3}$$

Inner-loop adaptation. Propagating $\boldsymbol{u}_h$ through the L TTT-augmented blocks via Eq. 1–2 produces the final video latents $\boldsymbol{z}^{(L)}(\boldsymbol{u}_h;\, \Theta_{\mathrm{WAM}},\, \theta_{\mathrm{TTT}},\, \{W^{(\ell)}\})$, on which $\mathcal{L}^{\mathrm{human}}_{\mathrm{vg}}$ is the standard LDA video-prediction loss; the W-dependence of $\mathcal{L}_{\mathrm{adapt}}$ below is exactly this propagation. Starting from $W^{(\ell)}_0 = W^{(\ell)}_{\mathrm{init}}$, the fast weights are updated by inner SGD on the combined human-side objective:

$$\mathcal{L}_{\mathrm{adapt}}(W_i) = \mathcal{L}^{\mathrm{human}}_{\mathrm{vg}}\big(\boldsymbol{u}_h;\, \Theta_{\mathrm{WAM}},\, \theta_{\mathrm{TTT}},\, W_i\big) + \lambda \sum_\ell \mathcal{L}^{(\ell)}_{\mathrm{KVM}}(W_i), \tag{4}$$

$$W^{(\ell)}_{i+1} = W^{(\ell)}_i - \eta\, \nabla_{W^{(\ell)}_i} \mathcal{L}_{\mathrm{adapt}}(W_i), \tag{5}$$

where $i \in \{0, 1, \ldots, N\}$ indexes the inner SGD iteration and λ weights the memory reconstruction term (see Table B.1). The adapted weight $W^{(\ell)}_N$ is what the residual readout in Eq. 2 uses. Both terms in $\mathcal{L}_{\mathrm{adapt}}$ depend only on the action-free human side, so the same inner-loop signal remains available at test time (Section 3.3).

Outer loss. The updated fast weights $W_N^{(\ell)}$ are queried by $\boldsymbol{Q}_r^{(\ell)}$ on the robot side and produce the residual in Eq. 2. The outer training objective is the standard WAM multitask loss on the paired robot data, which combines a video diffusion target on the robot video latents and an action diffusion target on the robot action chunks, inherited from the underlying LDA backbone [22]:

$$\mathcal{L}_{\text{meta}} = \mathcal{L}_{\text{WAM}}^{\text{robot}}. \tag{6}$$

Gradients are backpropagated through the TTT residual and the inner fast-weight update. The optimized parameters are the WAM parameters, the TTT slow projections $\theta_{\{K,V,Q,O\}}$, and the initialization W_{init}. The adapted fast weights are discarded after each training example and reinitialized from W_{init}.

Human-robot data synchronization. To support training with alignment, we conduct offline synchronization for human-robot data pairs. For a robot timestep t in an episode of length T_r, we compute the normalized phase $\phi = t/T_r$ and select the nearest-phase frame from the paired human video of length T_h.

3.3 Test-Time Training from Human Video

At deployment, the WAM parameters, TTT slow projections, and W_{init} are frozen. The input to test-time training is a small batch of action-free human videos $\mathcal{B}_h$ from the target domain. We run the model in video-generation mode and optimize only the video-side TTT fast weights on the same combined objective form used at meta-training:

$$\mathcal{L}_{\text{TTT}}(W_i) = \frac{1}{|\mathcal{B}_h|} \sum_{\boldsymbol{u} \in \mathcal{B}_h} \left[\mathcal{L}_{\text{vg}}\big(\boldsymbol{u};\, \Theta_{\text{WAM}},\, \theta_{\text{TTT}},\, W_i\big) + \lambda \sum_{\ell} \mathcal{L}_{\text{KVM}}^{(\ell)}(\boldsymbol{u};\, W_i) \right], \tag{7}$$

$$W_{i+1}^{(\ell)} = W_i^{(\ell)} - \eta\, \nabla_{W_i^{(\ell)}} \mathcal{L}_{\text{TTT}}(W_i). \tag{8}$$

Both $\mathcal{L}_{\text{vg}}$ and $\mathcal{L}_{\text{KVM}}^{(\ell)}$ are computed from the human side alone (Eq. 3), so no robot-side supervision is needed. No WAM parameter, TTT slow projection, initialization parameter, or action-expert parameter is updated. After N test-time updates (the same step budget as in Eq. 5; see Table B.1), the adapted fast weights W_N are fixed during robot rollout:

$$\boldsymbol{a}_{t:t+k} \sim p_{\Theta_{\text{WAM}},\, \theta_{\text{TTT}},\, W_N}\big(\boldsymbol{a}_{t:t+k} \mid \boldsymbol{o}_t,\, \boldsymbol{g}\big). \tag{9}$$

4 Experiments

We evaluate WAM-TTT on real-robot manipulation across three embodiments. The protocol matches Section 3: a WAM is pre-trained, then the WAM is undergone human-robot meta training, and at deployment the TTT branch's fast weights adapt online via inner SGD on a small set of unseen-task human demonstrations while the WAM and slow weights stay frozen.

4.1 Experimental Setup

Robot and Tasks. We evaluate WAM-TTT across three real-robot embodiments—**Unitree G1** (humanoid), **Galbot gripper** (bimanual two-finger), and **Galbot sharpa** (bimanual dexterous)—on a total of **9 manipulation tasks**: *Transfer Bottle*, *Table Bussing*, *Deliver Drink*, *Swap Place*, *Pour Water*, *Stamp Paper*, *Flip Steak*, *Pyramid Stacking*, and *Multi-step Steak*. Each task is assigned to a single embodiment and is evaluated under two settings. The *Orig.* setting collects evaluation trials inside the **standardized robot cubicle** that was also used to record the training data, with matching lighting, table height, and object instances. The *New* setting deploys the robot in previously unseen **household environments** where lighting, table height, and the manipulated objects all change jointly relative to training—i.e., a combined out-of-distribution perturbation rather than a single-factor shift. We report *progress* (%) over 25 trials per (task, setting) cell. Progress is the standard partial-credit metric used in recent VLA evaluations: each trial receives 1.0 for full task completion and a fractional score in $[0, 1]$ proportional to the number of pre-defined subgoals reached.

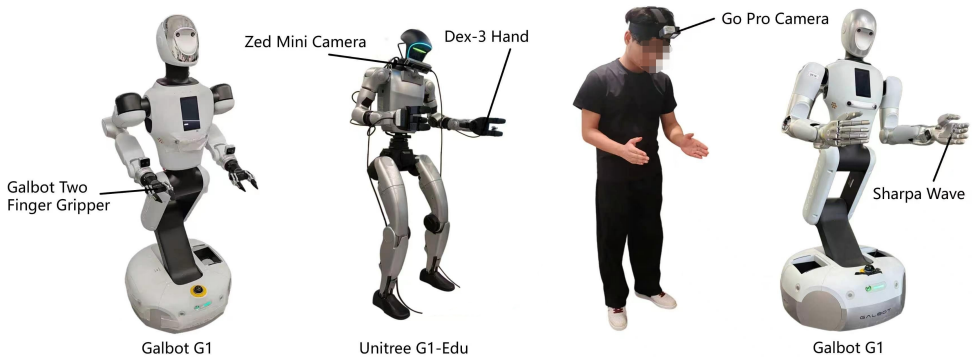

Figure 3: Experimental setup.

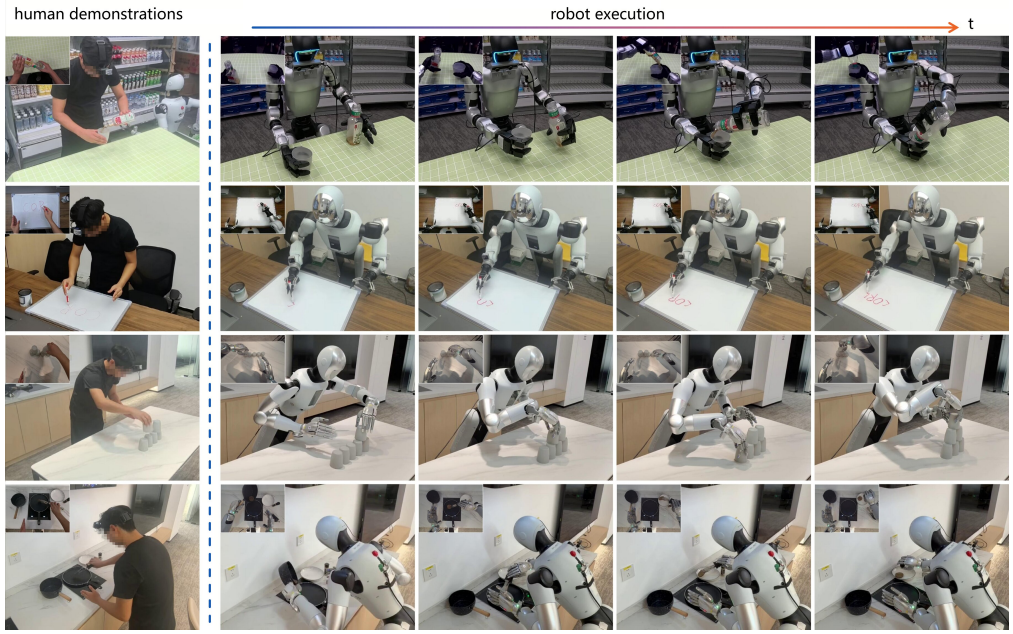

Figure 4: **Qualitative rollouts.** For each unseen task we show a robot rollout filmstrip (right) and the paired human demonstration used as deployment-time Key/Value (left).

Dataset and Metric. We collect a meta-training dataset consisting of 2,286 paired human and robot episodes, which broadly covers 9 distinct manipulation tasks. Both robot and human data are captured from an egocentric perspective. Specifically, human demonstrations are recorded using a GoPro camera, without any form of pose estimation.

4.2 Compared with baselines

Baselines. We compare against five baselines plus our WAM-TTT. LDA [22]: the pretrained WAM backbone, no human data, no TTT branch. WAM-COTRAIN: the same WAM further trained with paired human play data via the WAM multitask objectives (co-training; no TTT branch). WAM-ICL: the same WAM that ingests deployment-time human videos as in-context demonstrations, with no fast-weight adaptation. EGOSCALE [10]: a recent VLA scaled on diverse egocentric human data; as the original model is not open-source, we evaluate our re-implementation. $\pi_{0.5}$ [38]: Physical Intelligence's open-world-generalization VLA.

Quantitative Results. Table 1 reports per-task progress in the *New* household setting; the full table including the in-cubicle *Orig.* numbers is deferred to Appendix C. WAM-TTT averages **46.2%**

Table 1: **Main results.** Progress (%) on 9 manipulation tasks evaluated in previously unseen home environments. All cells averaged over 25 trials. The full table including the in-cubicle *Orig.* setting is in Appendix C.

Method	Transfer Bottle	Table Bussing	Deliver Drink	Swap Place	Pour Water	Stamp Paper	Flip Steak	Pyramid stacking	Multi-step Steak	Avg.
$\pi_{0.5}$[38]	33.4	36.0	15.0	7.4	10.0	24.4	2.0	4.7	0.3	14.8
LDA [22]	56.0	70.0	55.0	44.4	20.0	**33.3**	10.0	0.6	3.0	32.5
EGOSCALE [10]	34.4	44.0	33.3	6.0	7.5	1.1	5.0	2.0	2.0	15.0
WAM-COTRAIN	10.0	10.0	44.3	48.1	24.0	21.7	34.2	**12.0**	23.8	25.3
WAM-ICL	10.0	10.0	14.2	10.0	0.0	5.0	10.0	2.0	2.5	7.1
WAM-TTT	**55.6**	**100.0**	**66.7**	**66.7**	**30.0**	8.3	**34.3**	10.4	**43.8**	**46.2**

across the 9 tasks, against 32.5% for the no-TTT LDA[22] backbone (**+13.7** pts), 25.3% for WAM-COTRAIN (**+20.9** pts), 15.0% for EGOSCALE[10] (**+31.2** pts), 14.8% for $\pi_{0.5}$[38] (**+31.4** pts), and 7.1% for WAM-ICL (**+39.1** pts). Three observations follow. (i) The gap against WAM-ICL is the strongest piece of evidence for the design hypothesis: feeding the same human videos as in-context tokens fails to transfer skill to unseen home environments, whereas absorbing them as fast-weight memory does. (ii) The gap against LDA (same WAM, no human data, no TTT) quantifies the contribution of human play data; the gap against $\pi_{0.5}$ and EGOSCALE (no test-time human videos at all) quantifies the contribution of test-time adaptation itself. (iii) Across the 9 tasks WAM-TTT wins 7 outright and ties on Flip Steak (10.0); the single exception is *Stamp Paper* (8.3 vs. LDA's 33.3), where the in-cubicle stamp pose is geometrically tight and the household-scene perturbation breaks an alignment that the human videos do not visibly correct.

Qualitative Results. Figure 4 shows robot rollouts on three representative unseen tasks alongside the human demonstrations used as deployment-time Key/Value. Additional ablations (data-ratio sweep, model architecture) are in Appendix E.

4.3 Ablation Study

We conduct ablations to isolate the contribution of each component in WAM-TTT. Table 2 reports progress over 10 trials on *Table Bussing* and *Swap Place*. The full WAM-TTT model combines human-robot meta-training, a key–value memory reconstruction objective, and test-time adaptation of the video-side TTT layers from human videos. **WAM-LoRA** replaces the TTT fast-weight mechanism with a generic parameter-efficient adaptation baseline. **w/o Meta Training** removes the human-robot meta-training stage, so the TTT branch is not explicitly trained to align human Keys/Values with robot Queries. **w/o Memory Recon.** removes the inner key–value memory reconstruction loss, disabling the structured write mechanism into fast weights. **w/o TTT** removes human-video adaptation entirely and evaluates the frozen WAM.

This ablation separates the effects of three design choices: using human videos at deployment, representing them through TTT fast weights, and meta-training the Q/K/V interface with paired human-robot data. The comparison between WAM-TTT and **w/o TTT** measures the value of test-time human-video adaptation. The comparison with **WAM-LoRA** tests whether the improvement comes specifically from the TTT memory structure rather than generic low-rank adaptation. The drops from **w/o Meta Training** and **w/o Memory Recon.** further quantify the importance of learning a human-to-robot memory interface before deployment.

4.4 Generalization Preservation

A potential concern is that test-time adaptation from a short human video may overfit the WAM to the demonstrated trajectory, sacrificing the broad generalization inherited from the pretrained foundation model. We evaluate this by first adapting WAM-TTT with human play videos from the target task and then testing the adapted policy under perturbations that change the execution condition,

Table 2: **Protocol ablation** on training and test-time inference choices. Progress(%) on *Table Bussing* and *Swap Place* under the *New* setting; 10 trials per cell.

Task	WAM-TTT	WAM-LoRA	w/o Meta Training	w/o Memory Recon.	w/o TTT
Table Bussing	**100.0**	30.0	9.0	66.7	40.0
Swap Place	**88.9**	0.0	0.0	72.0	74.1

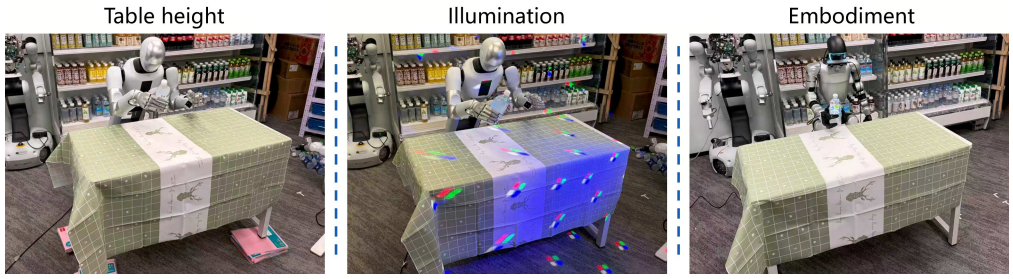

Figure 5: Generalization Setup.

including lighting, object position, and embodiment-related appearance shifts. Table 3 compares WAM-TTT with the pretrained LDA[22] backbone, a policy baseline $\pi_{0.5}$[38] , and WAM-ICL, which uses the same human videos only as in-context demonstrations without fast-weight adaptation.

The key comparison is between WAM-ICL and WAM-TTT. Both methods receive the same human demonstration, but they use it in different ways: WAM-ICL conditions the frozen model on the demonstration at inference time, while WAM-TTT converts the human video into video-side fast weights through test-time training. Because the WAM backbone and action expert remain frozen, WAM-TTT steers the visual dynamics used for action generation without overwriting the pretrained action prior. As shown in Table 3, WAM-TTT maintains strong performance across all perturbation types on bimaual task *Diliver Drink*. This indicates that the proposed TTT mechanism does not merely memorize the human demonstration; instead, it provides task- and domain-specific adaptation while preserving the foundation model's original robustness to visual and spatial shifts.

5 Conclusion, Limitation and Future Direction

We presented WAM-TTT, a two-stage adaptation pipeline for World-Action Models. *Human-robot meta-training* attaches Spatial-TTT-style[34] fast-weight branches to the WAM's video expert and jointly updates the WAM and the branches' slow projections via the WAM multitask outer loss, while the fast weights adapt online via inner SGD on self-supervised video-prediction and key–value memory reconstruction objectives derived from synchronized human–robot pairs. At *test time*, the WAM, the action expert, and all slow projections are frozen; only the video-side fast weights update, via inner SGD on the user's unseen-task human videos. The recipe yields deployment-time skill absorption without any gradient step on the WAM, and matches or exceeds online-adaptation baselines on a real-robot manipulation suite (Section 4).

Limitations. Three caveats. (1) Meta-training phase pairing assumes the paired human episode covers the same skill phase distribution as the robot episode; mis-aligned manifests degrade the inner adaptation signal in a way the loss does not flag (Section 4.3). (2) The deployment-time fast-weight adaptation is bounded by the expressiveness of the fast-weight network and by the slow projections fixed at meta-training; the further the deployment task drifts from the meta-training pairing distribution, the weaker the adaptation. We have not characterised the boundary empirically. (3) Our deployment-time interface accepts only egocentric human RGB frames; it does not exploit hand-pose, contact, or 3-D scene cues that related work has shown useful [39, 11].

Table 3: **Evaluation of Generalization Ability.**

Task	Perturbation	$\pi_{0.5}$[38]	LDA[22]	WAM-ICL	**WAM-TTT**
Deliver Drink	Lighting	28.0	54.0	12.0	**66.0**
	Spatial	0.0	28.0	20.0	**56.0**

Outlook. The meta-training / test-time TTT interface generalises beyond human Key/Value: any auxiliary modality with a phase-pairable training signal could in principle drive a parallel fast-weight branch under the same loss-only-then-residual regime. We see WAM-TTT as a step toward WAM-based foundation model backbones whose attention structure carries explicit "adaptation seats" that downstream practitioners can drive with whatever side information they have at hand.

References

[1] T. Yu et al. One-shot imitation from observing humans via domain-adaptive meta-learning. In *RSS*, 2018.

[2] S. Bahl, A. Gupta, and D. Pathak. Human-to-robot imitation in the wild. In *RSS*, 2022.

[3] M. Xu, Z. Xu, C. Chi, M. Veloso, and S. Song. Xskill: Cross embodiment skill discovery. In *CoRL*, 2023.

[4] H. Bharadhwaj, A. Gupta, V. Kumar, and S. Tulsiani. Towards generalizable zero-shot manipulation via translating human interaction plans. In *ICRA*, 2024.

[5] N. Hansen et al. Self-supervised policy adaptation during deployment. In *ICLR*, 2021.

[6] M. Xu, Z. Xu, C. C. Pan, X. Zhu, C. Tomei, Y. Shen, Z. Wu, S.-R. Chen, J. B. Tenenbaum, T. Lozano-Perez, and S. Song. Flow as the cross-domain manipulation interface. In *CoRL*, 2024.

[7] S. Kareer, D. Patel, R. Punamiya, P. Mathur, S. Cheng, C. Wang, J. Hoffman, and D. Xu. Egomimic: Scaling imitation learning via egocentric video. *arXiv preprint arXiv:2410.24221*, 2024.

[8] R. Hoque, P. Huang, D. J. Yoon, M. Sivapurapu, and J. Zhang. Egodex: Learning dexterous manipulation from large-scale egocentric video. *arXiv preprint arXiv:2505.11709*, 2025.

[9] K. Grauman et al. Ego-exo4d: Understanding skilled human activity from first- and third-person perspectives. In *CVPR*, 2024.

[10] R. Zheng et al. Egoscale: Scaling dexterous manipulation with diverse egocentric human data. *arXiv preprint arXiv:2602.16710*, 2026.

[11] H. Chen et al. Vidbot: Learning generalizable 3d actions from in-the-wild 2d human videos for zero-shot robotic manipulation. In *CVPR*, 2025.

[12] H. Kim et al. Uniskill: Imitating human videos via cross-embodiment skill representations. In *CoRL*, 2025.

[13] Z. Chen, S. Chen, E. Arlaud, I. Laptev, and C. Schmid. Vividex: Learning vision-based dexterous manipulation from human videos. In *ICRA*, 2025.

[14] C. Wang, L. Fan, J. Sun, R. Zhang, L. Fei-Fei, D. Xu, Y. Zhu, and A. Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. In *CoRL*, 2023. arXiv:2302.12422.

[15] H. Bharadhwaj, A. Gupta, S. Tulsiani, and V. Kumar. Zero-shot robot manipulation from passive human videos. *arXiv preprint arXiv:2302.02011*, 2023.

[16] V. Jain, M. Attarian, N. J. Joshi, A. Wahid, D. Driess, Q. Vuong, P. R. Sanketi, P. Sermanet, S. Welker, C. Chan, I. Gilitschenski, Y. Bisk, and D. Dwibedi. Vid2robot: End-to-end video-conditioned policy learning with cross-attention transformers. *arXiv preprint arXiv:2403.12943*, 2024.

[17] H. Bharadhwaj, D. Dwibedi, A. Gupta, S. Tulsiani, C. Doersch, T. Xiao, D. Shah, F. Xia, D. Sadigh, and S. Kirmani. Gen2act: Human video generation in novel scenarios enables generalizable robot manipulation. *arXiv preprint arXiv:2409.16283*, 2024.

[18] R. Shah, S. Liu, Q. Wang, Z. Jiang, S. Kumar, M. Seo, R. Martín-Martín, and Y. Zhu. Mimicdroid: In-context learning for humanoid robot manipulation from human play videos. *arXiv preprint arXiv:2509.09769*, 2025.

[19] X. Chi, C.-K. Fan, H. Zhang, X. Qi, R. Zhang, A. Chen, C.-m. Chan, W. Xue, Q. Liu, S. Zhang, et al. Eva: An embodied world model for future video anticipation. *arXiv preprint arXiv:2410.15461*, 2024.

[20] X. Chi, P. Jia, C.-K. Fan, X. Ju, W. Mi, K. Zhang, Z. Qin, W. Tian, K. Ge, H. Li, et al. Wow: Towards a world omniscient world model through embodied interaction. *arXiv preprint arXiv:2509.22642*, 2025.

[21] C. Zhu, R. Yu, S. Feng, B. Burchfiel, P. Shah, and A. Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *arXiv preprint arXiv:2504.02792*, 2025.

[22] J. Lyu, K. Liu, X. Zhang, H. Liao, Y. Feng, W. Zhu, T. Shen, J. Chen, J. Zhang, Y. Dong, W. Cui, S. Qi, S. Wang, Y. Zheng, M. Yan, X. Shi, H. Li, D. Zhao, M.-Y. Liu, Z. Zhang, L. Yi, Y. Wang, and H. Wang. LDA-1B: Scaling latent dynamics action model via universal embodied data ingestion. *arXiv preprint arXiv:2602.12215*, 2026.

[23] H. Bi, H. Tan, S. Xie, Z. Wang, S. Huang, H. Liu, R. Zhao, Y. Feng, C. Xiang, Y. Rong, et al. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 35101–35113, 2026.

[24] M. Team, C. Xiang, F. Bao, H. Liu, H. Tan, H. Bi, J. Li, J. Liu, J. Pang, K. Jing, et al. Motubrain: An advanced world action model for robot control. *arXiv preprint arXiv:2604.27792*, 2026.

[25] Y. Zhang, W. Zhang, Z. Qi, H. Zhang, H. Lin, J. Zhang, Y. Mu, X. Yang, W. Zeng, and X. Jin. Imagewam: Do world action models really need video generation, or just image editing?, 2026. URL https://arxiv.org/abs/2606.19531.

[26] H. Yu, H. Lin, J. Zhang, W. Zhang, C. Gu, H. Li, and P. Tan. Maskwam: Unifying mask prompting and prediction for world-action models. *arXiv preprint arXiv:2606.13515*, 2026.

[27] B. Peng, W. Zhang, L. Xu, Z. Qi, J. Zhang, H. Liu, W. Zeng, and X. Jin. Reworld: Multi-dimensional reward modeling for embodied world models. *arXiv preprint arXiv:2601.12428*, 2026.

[28] Y. Sun, X. Wang, Z. Liu, J. Miller, A. A. Efros, and M. Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020.

[29] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021.

[30] Y. Liu, P. Kothari, B. van Delft, B. Bellot-Gurlet, T. Mordan, and A. Alahi. Ttt++: When does self-supervised test-time training fail or thrive? In *NeurIPS*, 2021.

[31] Y. Gandelsman, Y. Sun, X. Chen, and A. A. Efros. Test-time training with masked autoencoders. In *NeurIPS*, 2022.

[32] Y. Sun, X. Li, K. Dalal, J. Xu, A. Vikram, G. Zhang, Y. Dubois, X. Chen, X. Wang, S. Koyejo, T. Hashimoto, and C. Guestrin. Learning to (learn at test time): Rnns with expressive hidden states. In *ICML*, 2025.

[33] A. Behrouz, P. Zhong, and V. Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2025.

[34] F. Liu, D. Wu, J. Chi, Y. Cai, Y.-H. Hung, X. Yu, H. Li, H. Hu, Y. Rao, and Y. Duan. Spatial-ttt: Streaming visual-based spatial intelligence with test-time training. *arXiv preprint arXiv:2603.12255*, 2026.

[35] S. Yang, Y. Ze, and H. Xu. Movie: Visual model-based policy adaptation for view generalization. In *NeurIPS*, 2023.

[36] Z. Bai, C. Gao, and M. Z. Shou. Evolve-vla: Test-time training from environment feedback for vision-language-action models. *arXiv preprint arXiv:2512.14666*, 2025.

[37] C. Liu, Y. Liu, T. Wang, Q. Zhuang, J. C. Liang, W. Yang, R. Xu, Q. Wang, D. Liu, and C. Han. On-the-fly vla adaptation via test-time reinforcement learning. *arXiv preprint arXiv:2601.06748*, 2026.

[38] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[39] J. Mu, S. Yang, Y. Bao, H. Bae, T. Wei, L. Xu, B. Li, H. Xu, and J. Pang. Deximit: Learning bimanual dexterous manipulation from monocular human videos. *arXiv preprint arXiv:2602.10105*, 2026.

[40] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning (ICML)*, 2020.

[41] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann. MediaPipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*, 2019.

[42] J. Romero, D. Tzionas, and M. J. Black. Embodied hands: Modeling and capturing hands and bodies together. In *ACM Transactions on Graphics (SIGGRAPH Asia)*, 2017.

[43] O. Siméoni, H. V. Vo, M. Seitzer, F. Baldassarre, M. Oquab, C. Jose, V. Khalidov, M. Szafraniec, S. Yi, M. Ramamonjisoa, F. Massa, D. Haziza, L. Wehrstedt, J. Wang, T. Darcet, T. Moutakanni, L. Sentana, C. Roberts, A. Vedaldi, J. Tolan, J. Brandt, C. Couprie, J. Mairal, H. Jégou, P. Labatut, and P. Bojanowski. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025.

A Meta-training algorithm

Notation: fast vs. slow weights. Throughout, $W_{\text{init}}^{(\ell)}$ denotes the learnable initialization of the fast-weight MLP at layer ℓ and is a *slow* parameter trained by the outer optimizer; the projections $\theta_{\{K,V,Q,O\}}^{(\ell)}$ and the WAM parameters Θ_{WAM} are likewise slow. The symbol $W_i^{(\ell)}$ (with $W_0^{(\ell)} \equiv W_{\text{init}}^{(\ell)}$) denotes the *fast* weights of layer ℓ at inner iteration i. The adjective "fast" refers to the time scale of updates per Spatial-TTT terminology [34]: $W_i^{(\ell)}$ updates N times *within* a single forward pass via inner SGD, while $W_{\text{init}}^{(\ell)}$ and the slow projections update only once per outer optimizer step (and are frozen at deployment).

Notation: token streams. We use $\boldsymbol{z}^{(\ell)}$ for the video latent tokens and $\boldsymbol{x}^{(\ell)}$ for the robot action tokens at the input of LDA block ℓ. These are the two streams that LDA's video expert and action expert process jointly via cross-stream attention (Section 3.1). The hatted symbols $\hat{\boldsymbol{z}}^{(\ell+1)}$ and $\hat{\boldsymbol{x}}^{(\ell+1)}$ denote the block's intrinsic outputs *before* any TTT residual is added; the unhatted $\boldsymbol{z}^{(\ell+1)}, \boldsymbol{x}^{(\ell+1)}$ are what is actually fed into block $\ell+1$, which equals $\hat{\boldsymbol{z}}^{(\ell+1)} + \Delta\boldsymbol{z}_{\text{TTT}}^{(\ell)}$ on the video stream and equals $\hat{\boldsymbol{x}}^{(\ell+1)}$ on the action stream (Eq. 1). The TTT residual modifies only the video stream, leaving the action expert's output untouched; this places test-time human-video adaptation entirely on the video side, in the modality where the action-free human videos can naturally supervise.

Notation: the outer-loop robot multitask loss $\mathcal{L}_{\text{WAM}}^{\text{robot}}$. We use $\mathcal{L}_{\text{WAM}}^{\text{robot}}$ as a shorthand for the WAM's standard multitask training loss evaluated on the paired robot side. It is the sum of two diffusion targets inherited verbatim from the LDA backbone [22]: a video-side flow-matching / diffusion denoising loss on the robot video latents $\{\boldsymbol{z}_r^{(\ell)}\}$ output by the video expert, and an action-side flow-matching / diffusion denoising loss on the robot action chunks $\{\boldsymbol{x}_r^{(\ell)}\}$ output by the action expert. We adopt LDA's exact loss formulation, weighting, and noise schedule without modification; the only WAM-TTT contribution at this outer level is the TTT residual that shifts $\hat{\boldsymbol{z}}^{(\ell+1)}$ to $\boldsymbol{z}^{(\ell+1)}$ on the video stream and is then back-propagated through together with both diffusion targets.

The runtime target: cross-attention from robot queries to human keys/values. At deployment, what we want each TTT layer to do is conceptually simple: let the robot token stream $\boldsymbol{z}_r$ *read information from* the in-scene human token stream $\boldsymbol{h}$ through the standard attention interface. Letting $\boldsymbol{q} = \theta_Q(\boldsymbol{z}_r)$, $\boldsymbol{k}_i = \theta_K(\boldsymbol{h}_i)$, $\boldsymbol{v}_i = \theta_V(\boldsymbol{h}_i)$ collect the per-token projections of one query and of every human-side key/value, classical softmax cross-attention defines the desired readout

$$\text{Out}(\boldsymbol{q}) \;=\; \sum_{i=1}^{L_h} \frac{\exp(\boldsymbol{q}^\top \boldsymbol{k}_i)}{\sum_j \exp(\boldsymbol{q}^\top \boldsymbol{k}_j)}\, \boldsymbol{v}_i. \tag{A.1}$$

Doing this literally would require materializing the full $(\boldsymbol{K}_h, \boldsymbol{V}_h)$ cache, whose length scales with the human-episode token count and which is awkward to re-update with each test-time gradient step.

The parametric substitute: an MLP that returns "the value of the closest key". Instead of carrying the explicit cache, the TTT layer stores the human side inside the *weights* W of the fast-weight MLP f_W. The runtime readout is then the parametric expression already given by Eq. 2, namely $\Delta\boldsymbol{z}_{\text{TTT}} = \theta_O\, f_W(\theta_Q(\boldsymbol{z}_r))$, with the slow θ_O projecting the d-dimensional output of f_W back to the LDA hidden dimension so the residual can be added to $\hat{\boldsymbol{z}}^{(\ell+1)}$. The claim that this MLP-based readout behaves like cross-attention is purely a claim about the weights W: querying f_W at $\boldsymbol{q}$ has to return the value associated with the key in the stored set that most resembles $\boldsymbol{q}$.

What makes f_W act like attention: a linear-attention witness. The deployed f_W is a small nonlinear MLP, but the linear special case $f_W(\boldsymbol{x}) = W\boldsymbol{x}$ with $W \in \mathbb{R}^{d\times d}$ is already a tractable witness for what minimizing the key–value memory reconstruction loss $\mathcal{L}_{\text{KVM}}$ does to the weights, and the nonlinear MLP is the smooth, normalized analog of the same retrieval pattern. Stack the human keys

and values from the synchronized frame as $\boldsymbol{K}_h, \boldsymbol{V}_h \in \mathbb{R}^{L_h \times d}$ in the row-token convention of the notation paragraph above; the denominator $BL_h d$ in Eq. 3 makes $\mathcal{L}_{\mathrm{KVM}}$ a per-element mean-squared error that is invariant to mini-batch size, human-sequence length, and embedding dimension, and is the form analyzed here. Minimizing the linear-case loss has a closed-form solution

$$\min_{W \in \mathbb{R}^{d \times d}} \ \frac{1}{BL_h d} \left\| \boldsymbol{K}_h W^\top - \boldsymbol{V}_h \right\|_F^2 \ \Longrightarrow \ W^* \ = \ \boldsymbol{V}_h^\top \, \boldsymbol{K}_h \, (\boldsymbol{K}_h^\top \boldsymbol{K}_h)^{-1}. \tag{A.2}$$

Under the standard linear-attention / modern-Hopfield isotropy hypothesis $\boldsymbol{K}_h^\top \boldsymbol{K}_h \approx (L_h/d)\, \boldsymbol{I}_d$, which holds for whitened or random-projection-style features and which serves here as a sanity check rather than a strict modeling assumption, the solution collapses to the Hebbian / outer-product memory

$$W^* \ \propto \ \boldsymbol{V}_h^\top \, \boldsymbol{K}_h \ = \ \sum_{i=1}^{L_h} \boldsymbol{v}_i \, \boldsymbol{k}_i^\top. \tag{A.3}$$

Querying with the robot side $\boldsymbol{Q}_r = \theta_Q(\boldsymbol{z}_r)$ then yields

$$f_{W^*}(\boldsymbol{Q}_r) \ = \ W^* \, \boldsymbol{Q}_r \ \propto \ \sum_{i=1}^{L_h} (\boldsymbol{k}_i^\top \boldsymbol{Q}_r) \, \boldsymbol{v}_i, \tag{A.4}$$

which is exactly a kernel-free, softmax-free linear-attention readout against $(\boldsymbol{K}_h, \boldsymbol{V}_h)$ [40]. In other words, $\mathcal{L}_{\mathrm{KVM}}$ is not an auxiliary regularizer next to the cross-attention behaviour; in the linear case it is the variational definition of that behaviour. Equations (A.2)–(A.4) hold as exact equalities only in this linear special case; the nonlinear MLP we actually deploy obeys the same training target $f_W(\boldsymbol{K}_h) \approx \boldsymbol{V}_h$, but the closed-form $W^* Q_r = \sum_i (\boldsymbol{k}_i^\top \boldsymbol{Q}_r) \boldsymbol{v}_i$ decomposition is replaced by the MLP's smooth, learned attention-like readout, in which the layer's nonlinearity and normalization play the role of the softmax kernel. The residual $\theta_O \, f_W(\boldsymbol{Q}_r)$ then injects the resulting human-derived value back into the video stream of Eq. 1.

Why the inner loop drives the fast weights to this witness. The inner SGD step (Eq. 5) directly minimizes $\mathcal{L}_{\mathrm{KVM}}$ alongside the human video-prediction loss $\mathcal{L}_{\mathrm{vg}}^{\mathrm{human}}$, so each inner update of the fast weights W is, by construction, a gradient step toward a W' for which the linear-attention witness above applies. The outer loop only optimizes the slow parameters $\Theta_{\mathrm{WAM}}, \theta_{\{K,V,Q,O\}}$, and W_{init} via the robot multitask loss $\mathcal{L}_{\mathrm{WAM}}^{\mathrm{robot}}$ (Eq. 6), and does so by backpropagating through the analytical, differentiable inner update of Spatial-TTT [34]. There is therefore no indirection between "do well on human video prediction" and "encode a human key–value memory": both are simultaneously and explicitly part of the inner-loop signal that shapes W.

Why the witness is preserved at deployment. The test-time inner loop (Eq. 8) optimizes the same combined objective form as meta-training: human video prediction plus per-layer key–value memory reconstruction. Since $\boldsymbol{K}_h^{(\ell)} = \theta_K^{(\ell)}(\boldsymbol{h}_\phi^{(\ell)})$ and $\boldsymbol{V}_h^{(\ell)} = \theta_V^{(\ell)}(\boldsymbol{h}_\phi^{(\ell)})$ are derivable from action-free human videos alone, no robot-side supervision is required to evaluate either term at deployment. The same inner SGD that produces the linear-attention witness during meta-training (Eqs. (A.2)–(A.4)) therefore continues to produce it at deployment on $\mathcal{B}_h$, and the residual $\theta_O \, f_W(\theta_Q(\boldsymbol{z}_r))$ remains the human-key/value cross-attention readout that the meta-training stage promised.

Algorithm. Algorithm A.1 below realizes one meta-training step. Stage (ii), test-time TTT (Section 3.3), uses the same per-block forward and the same combined inner-loop objective form, but freezes the WAM, the TTT slow projections, and W_{init} while only the fast weights W adapt via Eq. 8.

B Hyperparameters and datasets

Hyperparameters. See Table B.1.

Algorithm A.1: One meta-training step on a paired robot–human batch.

Input: Paired batch $\mathcal{B}$ of robot trajectories with action-free human videos; LDA-based WAM [22] with L diffusion transformer blocks; TTT slow projections $\theta^{(\ell)}_{\{K,V,Q,O\}}$; fast-weight initializations $W^{(\ell)}_{\text{init}}$; inner iterations N, inner LR η; memory reconstruction weight λ.

`// 1. phase-aligned human-robot sync (Section 3.2)`

For each robot timestep t, pick the nearest-phase human frame $\boldsymbol{h}_\phi$;

`// 2. inner adaptation: N full-network SGD steps on the combined human-side objective`

$W^{(\ell)} \leftarrow W^{(\ell)}_{\text{init}}$ for all ℓ;

for $i = 1, \ldots, N$ **do**

- Run a full WAM forward on $\boldsymbol{u}^h$ with current $\{W^{(\ell)}\}$ and TTT residuals (Eq. 2);
- Compute per-layer $\mathcal{L}^{(\ell)}_{\text{KVM}}$ on the synchronized human frame (Eq. 3);
- Assemble the inner-loop loss $\mathcal{L}_{\text{adapt}} = \mathcal{L}^{\text{human}}_{\text{vg}} + \lambda \sum_\ell \mathcal{L}^{(\ell)}_{\text{KVM}}$; backprop and update all layers simultaneously by Eq. 5;

`// 3. robot forward with adapted` W_N

for $\ell = 1, \ldots, L$ **do**

- LDA block forward gives $(\hat{\boldsymbol{z}}^{(\ell+1)}, \hat{\boldsymbol{x}}^{(\ell+1)})$;
- Apply TTT residual to the video stream by Eq. 2 and Eq. 1;

`// 4. outer loss and backprop`

Compute $\mathcal{L}_{\text{meta}} = \mathcal{L}^{\text{robot}}_{\text{WAM}}$ (Eq. 6); backprop through the analytical inner update [34] into the WAM, $\theta_{\{K,V,Q,O\}}$, and W_{init}; optimizer step;

Table B.1: Hyperparameters for the main WAM-TTT runs.

Setting	Value
WAM backbone	LDA [22] (Qwen3-VL-4B-Instruct VLM + DiT-L MMDiT action head)
MMDiT blocks L / hidden dim D / heads H	16 / 1536 / 32
TTT head dim d / fast-weight hidden width f_h	48 / 128
Inner SGD iterations N (meta-training and test-time)	1
Inner LR η at meta-training	0.1
Inner LR η at test time	0.01
Memory reconstruction weight λ	4×10^{-2}
Outer optimizer	AdamW ($\beta_1=0.9$, $\beta_2=0.999$), weight decay 10^{-8}
Outer LR (DiT action head / VLM interface)	1×10^{-4} / 1×10^{-5}
LR schedule	cosine with min 5×10^{-7}, 5 k-step warmup
Meta-training steps	100 k
Batch size (per device / global)	16 / 128
GPUs	8× NVIDIA H800, DeepSpeed ZeRO-2

Embodiments and datasets. Three robot embodiments. **Unitree G1** (humanoid bimanual, three-finger dex hand) covers *Table Bussing*, *Pour Water*, and *Deliver Drink*. **Galbot gripper** (bimanual two-finger) covers *Transfer Bottle* and *Stamp Paper*. **Galbot sharpa** (bimanual dexterous, 22-DoF per side, 58-dim total) covers *Swap Place*, *Pyramid Stacking*, *Flip Steak*, and the long-horizon *Multi-step Steak*. Across all three embodiments we collect 2,286 paired robot-human episodes spanning these 9 manipulation tasks: 600 on Unitree G1, 544 on Galbot gripper, and 1,142 on Galbot sharpa. Robot data is captured via teleoperation inside a standardized cubicle (Figure B.1), while the paired human demonstrations are recorded with a GoPro camera in egocentric view *directly in the actual household environments* that we later evaluate as the *New* setting (Figure B.2), without any hand-pose, joint-angle, or motion-retargeting annotation. The two views are paired by phase alignment (Section 3.2) for meta-training, and the human side is re-recorded in the deployment scene for test-

time TTT (Section 3.3). Each figure uses the same 5×2 grid (read left-to-right, top-to-bottom), with *Multi-step Steak* occupying two panels (panels 8 and 10) so the 9 tasks fill the 10-cell layout.

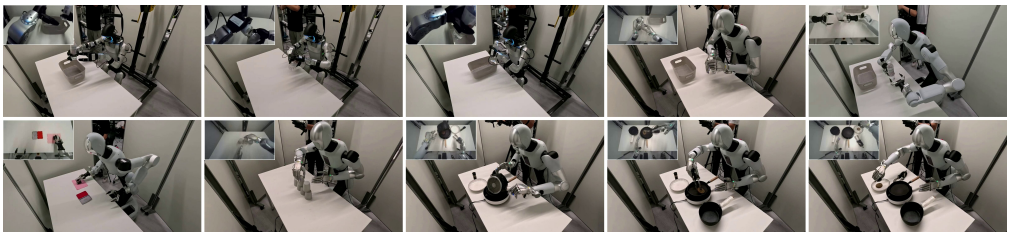

Figure B.1: **Robot data collection in the standardized cubicle.** Representative teleoperation frames in the 5×2 layout, read left-to-right, top-to-bottom. **Top row:** (1) *Table Bussing*, (2) *Pour Water*, (3) *Deliver Drink* on the **Unitree G1**; (4) *Swap Place* on the **Galbot sharpa**; (5) *Transfer Bottle* on the **Galbot gripper**. **Bottom row:** (6) *Stamp Paper* on the **Galbot gripper**; (7) *Pyramid Stacking*, (8) *Multi-step Steak* (grasping the pan and pouring the beef in), (9) *Flip Steak*, (10) *Multi-step Steak* (sprinkling pepper), all on the **Galbot sharpa**. The long-horizon *Multi-step Steak* occupies panels 8 and 10.

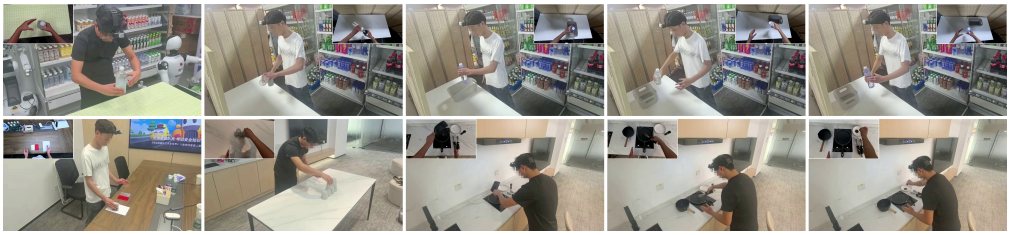

Figure B.2: **Human data collection in the actual New household scenes.** Same 10 task slots and panel order as Figure B.1, but the human demonstrator is in the actual household environment used as the *New* evaluation setting (lighting, clutter, tableware, and target instances all differ from the cubicle). The hand pose varies per panel according to the paired robot end-effector: parallel-jaw mimic on the **Galbot gripper** panels (5, 6), three-finger dex-hand grasp on the **Unitree G1** panels (1–3), and unconstrained dexterous use on the **Galbot sharpa** panels (4, 7–10). The task identity and panel order are the same as in Figure B.1. Demonstrations are recorded with a GoPro camera in egocentric view, without any hand-pose, joint-angle, or motion-retargeting annotation.

C Full main results: Orig. and New settings

Table C.1 is the full version of the main-paper result table (Table 1 in Section 4.2), including both the in-cubicle *Orig.* setting and the unseen-household *New* setting for each of the 9 manipulation tasks. The *New* block is the one promoted to the main paper; the *Orig.* block is provided here for completeness so the reader can verify how each baseline degrades under household-scene perturbation.

Analysis: *Orig.* (standardized cubicle) results. Under the *Orig.* setting (top row of each block in Table C.1), WAM-TTT leads at **61.1%**, against 50.2% for the no-TTT LDA backbone (**+10.9** pts), 48.4% for WAM-ICL (**+12.7** pts), 31.8% for $\pi_{0.5}$ (**+29.3** pts), 30.1% for EGOSCALE (**+31.0** pts), and 29.8% for WAM-COTRAIN (**+31.3** pts). Notably, WAM-COTRAIN, which mixes paired human data into the WAM multitask outer loss without our TTT mechanism, drops *below* the no-human $\pi_{0.5}$ and EGOSCALE baselines: simply diluting robot supervision with human data without an explicit human-to-robot alignment mechanism actively damages in-distribution performance. WAM-TTT instead absorbs human data into a fast-weight memory that does not perturb the policy stream for unrelated robot trajectories, so it strictly improves over the WAM backbone in the same standardized setting.

Table C.1: **Full main results.** Progress (%) on 9 tasks under the *Orig.* (standardized robot cubicle) and *New* (unseen household, combined OOD shift) settings. All cells averaged over 25 trials.

Method	Setting	Transfer Bottle	Table Bussing	Deliver Drink	Swap Place	Pour Water	Stamp Paper	Flip Steak	Pyramid Stacking	Multi-step Steak	Avg.
$\pi_{0.5}$	Orig.	55.5	80.0	70.0	12.2	25.0	31.1	4.4	6.1	2.0	31.8
	New	33.4	36.0	15.0	7.4	10.0	24.4	2.0	4.7	0.3	14.8
LDA	Orig.	72.0	90.0	80.0	66.7	33.6	**50.0**	33.3	6.7	19.2	50.2
	New	**56.0**	70.0	55.0	44.4	20.0	**33.3**	10.0	0.6	3.0	32.5
EGOSCALE	Orig.	69.4	80.0	69.6	10.0	30.0	2.7	5.0	2.0	2.0	30.1
	New	34.4	44.0	33.3	6.0	7.5	1.1	5.0	2.0	2.0	15.0
WAM-COTRAIN	Orig.	11.6	40.0	59.4	74.1	26.3	10.0	21.0	9.4	16.8	29.8
	New	10.0	10.0	44.3	48.1	24.0	21.7	34.2	**12.0**	23.8	25.3
WAM-ICL	Orig.	60.0	89.0	70.3	68.7	55.3	36.0	33.3	4.7	18.2	48.4
	New	10.0	10.0	14.2	10.0	0.0	5.0	10.0	2.0	2.5	7.1
WAM-TTT	Orig.	**77.8**	**100.0**	**90.0**	**88.9**	**63.3**	35.0	**44.0**	**10.0**	**40.6**	**61.1**
	New	55.6	**100.0**	**66.7**	**66.7**	**30.0**	8.3	**34.3**	10.4	**43.8**	**46.2**

Analysis: *Orig.* $\rightarrow$ *New* transfer of human-data benefits. The summary table below is computed directly from Table C.1: the *Orig.* and *New* columns are the per-method 9-task averages (the **Avg** column of Table C.1, one number per row of each *Orig./New* block), and the retention ratio New/Orig and the gap $\Delta = \mathrm{Avg}_{\mathrm{Orig}} - \mathrm{Avg}_{\mathrm{New}}$ are derived from those two averages. The retention ratio thus measures how well each method's average standardized-cubicle competence survives the unseen-household perturbation, and Δ reports the average per-task progress lost in absolute points:

Method	Orig.	New	New/Orig	Δ (Orig $-$ New)
WAM-TTT (WAM-TTT)	61.1	46.2	**0.76**	-14.9
WAM-COTRAIN	29.8	25.3	0.85	-4.5
LDA	50.2	32.5	0.65	-17.7
EGOSCALE	30.1	15.0	0.50	-15.1
$\pi_{0.5}$	31.8	14.8	0.47	-17.0
WAM-ICL	48.4	7.1	0.15	-41.3

WAM-ICL's catastrophic collapse (15% retention, -41.3 pts) shows that feeding human videos as in-context tokens is fragile under scene perturbation: the same long-context conditioning that helps in distribution becomes a liability when visual statistics shift. WAM-COTRAIN's high retention ratio (85%) is largely artifactual because its starting point is already weak (29.8); in absolute terms it remains the second-worst method on *New* after WAM-ICL. WAM-TTT combines the highest *Orig.* performance (61.1%) with the highest meaningful retention ratio (76%), preserving most of the human-data benefit even when the deployment scene departs from the training cubicle.

Summary: form comparison among human-data methods. Among the three forms of injecting paired human play data into a WAM, only WAM-TTT achieves the dual goal of strong in-distribution accuracy and robust OOD transfer: (i) direct multitask co-training (WAM-COTRAIN) sacrifices in-distribution policy quality (lowest *Orig.* score) and yields only modest *New* gains; (ii) in-context conditioning (WAM-ICL) reaches reasonable *Orig.* performance but collapses under household-scene shift; (iii) WAM-TTT's fast-weight TTT memory adapts the WAM only along the human-derived task evidence, leaving the LDA backbone's pretrained visual reasoning intact, so the human-data signal survives the OOD shift. This is the empirical signature of a useful human-data-injected memory: task-specific adaptation *without* overwriting the WAM's transferable structure.

D Per-task progress definitions

We list the per-task subgoal decomposition used to compute the *progress* (%) reported in Tables 1 and C.1. Following the additive-scoring convention in recent VLA evaluations (e.g., [10]), each task is decomposed into a small set of pre-defined milestones with weights summing to 1.0; a trial's progress score is the sum of milestone weights reached, with 1.0 reserved for trials that satisfy the final goal. Weights below are taken directly from our production evaluation rubric and are scored automatically from the robot's end-effector pose and known scene-object poses captured during the rollout. Progress is averaged across the 25 trials per (task, setting) cell.

Design rationale of the per-task weights. Within each task we deliberately concentrate the bulk of the weight on the few critical milestones whose completion implies task success—e.g., *Stamping successful* ($+0.50$) in Stamp Paper, *Pouring successful* ($+0.60$) in Pour Water, *Flipping successful* ($+0.45$) in Flip Steak, and *Successfully place the 3rd cup* ($+0.36$) in Pyramid Stacking. This makes the metric reward primarily for finishing the task, in line with the binary success-rate interpretation a reader expects from a manipulation benchmark. We then assign small fractional weights to easier preliminary steps such as reaching toward an object before grasping it. These small weights extract useful signal from lower-quality trials that complete the early phases but stall later in the rollout, which we find informative for both behavioural-cloning training and downstream reinforcement-learning fine-tuning.

Transfer Bottle. Instruction: *"Hand the bottle from the left arm to the right arm and place it into the receiving box."* Additive rubric:

- $+0.05$: left hand reaches for the bottle.
- $+0.10$: left hand successfully grasps the bottle.
- $+0.05$: right hand reaches to receive the bottle.
- $+0.15$: right hand successfully grasps the bottle.
- $+0.10$: left hand releases the bottle.
- $+0.05$: right hand reaches to place the bottle.
- $+0.50$: right hand successfully places the bottle into the box.

Table Bussing. Instruction: *"Clear the tableware items from the table into the bin."* Additive rubric, with N items per trial (default $N = 1$):

- $+0.5/N$ per item: item grasped from the table.
- $+0.5/N$ per item: item released into the bin.

Deliver Drink. Instruction: *"Pick up the drink and hand it to a designated location."* Additive rubric:

- $+0.30$: drink (cup or bottle) grasped.
- $+0.30$: drink transported toward the recipient.
- $+0.40$: drink placed or released at the recipient.

Swap Place. Instruction: *"Pass the object from the left hand to the right hand, then place it at a designated location."* Additive rubric:

- $+0.20$: object A picked up.
- $+0.30$: object A staged in a buffer location.
- $+0.30$: object B picked up and placed at A's original position.
- $+0.20$: object A placed at B's original position.

Pour Water. Instruction: *"Pour water from the bottle into the cup."* Additive rubric:

- $+0.10$: cup successfully grasped.
- $+0.15$: bottle successfully grasped.
- $+0.05$: pouring posture reached.
- $+0.60$: pouring successful.
- $+0.10$: bottle successfully placed back.

Stamp Paper. Instruction: *"Stamp the paper at the marked location after applying the ink paste."* Additive rubric:

- $+0.05$: reach for the stamp.
- $+0.15$: stamp s[uccessfully grasped.
- $+0.05$: reach to the ink paste.
- $+0.20$: ink paste successfully applied.
- $+0.05$: reach to stamp the paper.
- $+0.50$: stamping successful.

Flip Steak. Instruction: *"Use the tongs to flip the steak in the pan."* Additive rubric:

- $+0.02$: reach for the tongs.
- $+0.20$: tongs successfully grasped.
- $+0.03$: approach the steak.
- $+0.20$: steak successfully clamped.
- $+0.45$: flipping successful.
- $+0.10$: tongs successfully put down.

Pyramid Stacking. Instruction: *"Stack six cups into a three-layer pyramid (a base layer of three cups, a middle layer of two, and a single top cup)."* Additive rubric (each "1st/2nd/3rd cup" entry below tracks the layer-defining placement of that layer):

- $+0.01$: reach for the 1st cup.
- $+0.05$: 1st cup successfully grasped.
- $+0.02$: reach to place the 1st cup.
- $+0.10$: 1st cup successfully placed.
- $+0.01$: reach for the 2nd cup.
- $+0.10$: 2nd cup successfully grasped.
- $+0.02$: reach to place the 2nd cup.
- $+0.15$: 2nd cup successfully placed.
- $+0.01$: reach for the 3rd cup.
- $+0.15$: 3rd cup successfully grasped.
- $+0.02$: reach to place the 3rd cup.
- $+0.36$: 3rd cup successfully placed.

Multi-step Steak. Instruction: *"Plate the steak from the pan, flip it during cooking, transfer it back to the plate, and season it with pepper."* Additive rubric:

- $+0.01$: reach for the pan.
- $+0.02$: pan successfully held.
- $+0.01$: reach to place the steak.
- $+0.07$: steak successfully poured into the pan.

- $+0.01$: reach to place the pan.
- $+0.01$: pan placed successfully.
- $+0.01$: reach for the tongs.
- $+0.07$: tongs successfully held.
- $+0.01$: reach to clamp the steak.
- $+0.17$: steak successfully clamped.
- $+0.20$: steak flipping successful.
- $+0.15$: steak successfully clamped again.
- $+0.07$: steak successfully placed onto the plate.
- $+0.01$: reach to place the tongs.
- $+0.01$: tongs put down.
- $+0.01$: reach for the pepper bottle.
- $+0.07$: pepper bottle successfully held.
- $+0.09$: sprinkling pepper successful.

E Additional results

E.1 Qualitative rollout gallery in unseen household, office, and kitchen scenes

Figure E.1 compiles 9 representative WAM-TTT rollouts in unseen scenes. Six of the rows correspond to evaluated tasks from the 9-task benchmark of Section 4.1 executed under the *New* setting (Multi-step Steak, Transfer Bottle, Deliver Drink, Pyramid Stacking, Stamp Paper, Table Bussing); the remaining three rows are additional in-the-wild demonstrations exercising perturbation axes that the benchmark does not test (whiteboard wiping, free-form circle drawing, and handwriting), included to convey the breadth of skills the model retains after meta-training and test-time TTT. Each row is a single rollout of one task, shown as 5 evenly-spaced keyframes read left-to-right, with the leftmost column showing the initial observation and the rightmost column showing the final scene at episode termination. All rollouts come from the same checkpoint used to produce Table C.1, after test-time TTT adaptation from in-scene human videos (Section 3.3); no per-task hyperparameter sweep is performed.

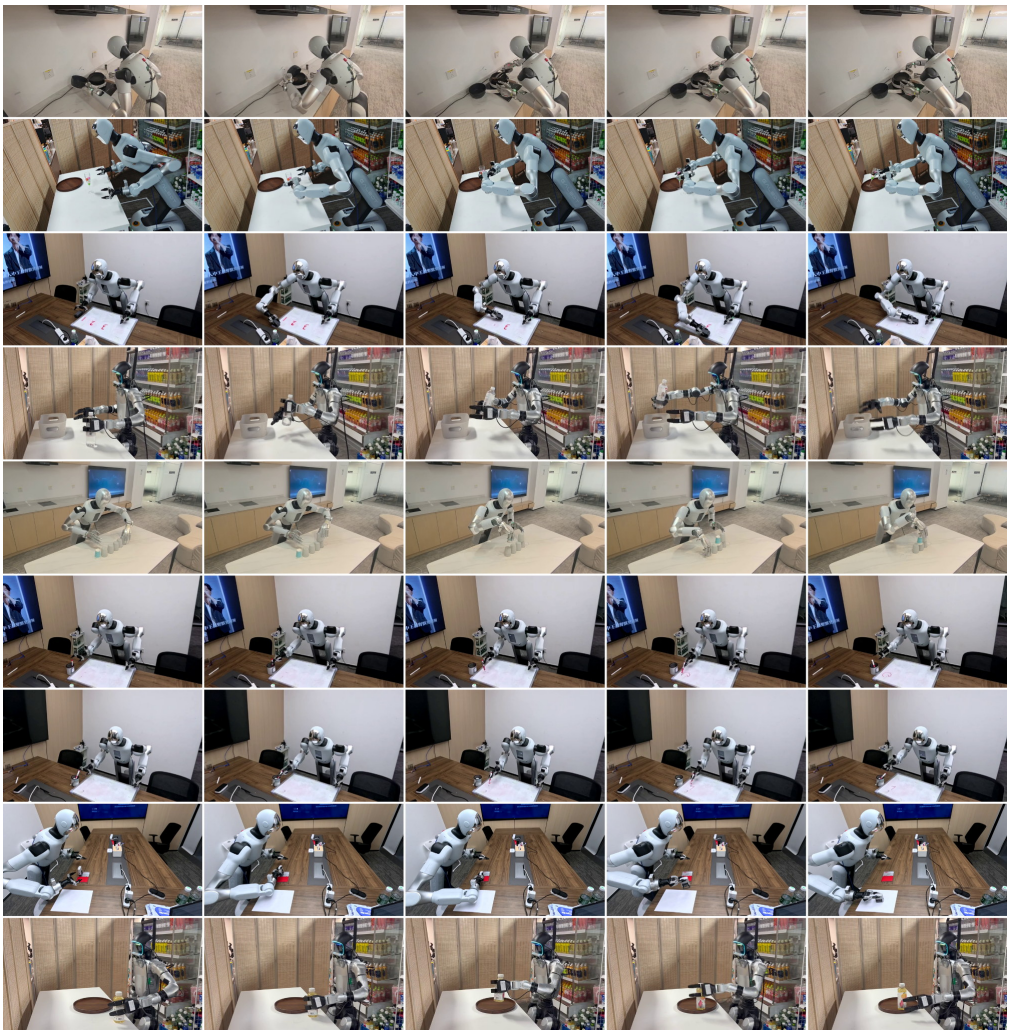

Figure E.1: **Qualitative gallery of WAM-TTT rollouts in unseen household, office, and kitchen scenes.** Each row is a single rollout of one task, shown as 5 evenly-spaced keyframes (left $\rightarrow$ right, initial $\rightarrow$ terminal frame). **Rows from top to bottom:** (1) *Multi-step Steak* in a completely new kitchen with the stovetop raised $+10$ cm relative to the meta-training cubicle (Galbot sharpa); (2) *Transfer Bottle* with a novel long-stem wine-glass instance (Galbot gripper); (3) *Wipe Blackboard* in a meeting room (Galbot gripper); (4) *Deliver Drink* (Unitree G1, dex-3 hand); (5) *Pyramid Stacking* on the Galbot sharpa, with the leftmost cup replaced by a novel paper-cup instance mid-task; (6) *Draw a circle* on the meeting-room whiteboard (Galbot gripper); (7) *Stamp Paper* in the meeting room with the target stamp position shifted away from the cubicle anchor (Galbot gripper); (8) free-form handwriting of the letter *"L"* (Galbot gripper); (9) *Table Bussing* (Unitree G1, dex-3 hand). Rows (1, 2, 4, 5, 7, 9) draw from the 9-task benchmark of Section 4.1 under the *New* perturbation, while rows (3, 6, 8) are additional in-the-wild demonstrations beyond the headline benchmark.

E.2 Direct lab-scene generalization without scene-specific human data

We further stress-test WAM-TTT's deployment-time generalization by moving the robot into a previously unseen lab scene while keeping the model exactly as it left meta-training. Crucially, *no additional in-scene human videos* are collected for this evaluation: the TTT branch only uses the slow projections and W_{init} shaped by paired robot-human play during meta-training. This isolates the contribution of the meta-trained slow projections from any test-time human-video adaptation in the deployment scene.

Figure E.2 reports rollouts on **Galbot gripper** and **Galbot sharpa** across four perturbation axes simultaneously relative to the standardized robot cubicle: (i) lighting (warm vs. cool, side- vs. top-mounted), (ii) tablecloth pattern and colour, (iii) novel object instances of the same category, and (iv) target-pose offsets. Despite all four perturbations being applied jointly, the WAM together with the meta-trained slow projections retains a useful skill prior in the new lab scene, indicating that the calibration achieved during meta-training is not tied to the in-cubicle observation statistics. This direct-transfer result complements Table C.1: it shows that even without populating the Human-Key/Value cache from in-scene demonstrations, WAM-TTT can recover useful behaviour purely from the WAM-side knowledge that the meta-training stage has shaped.

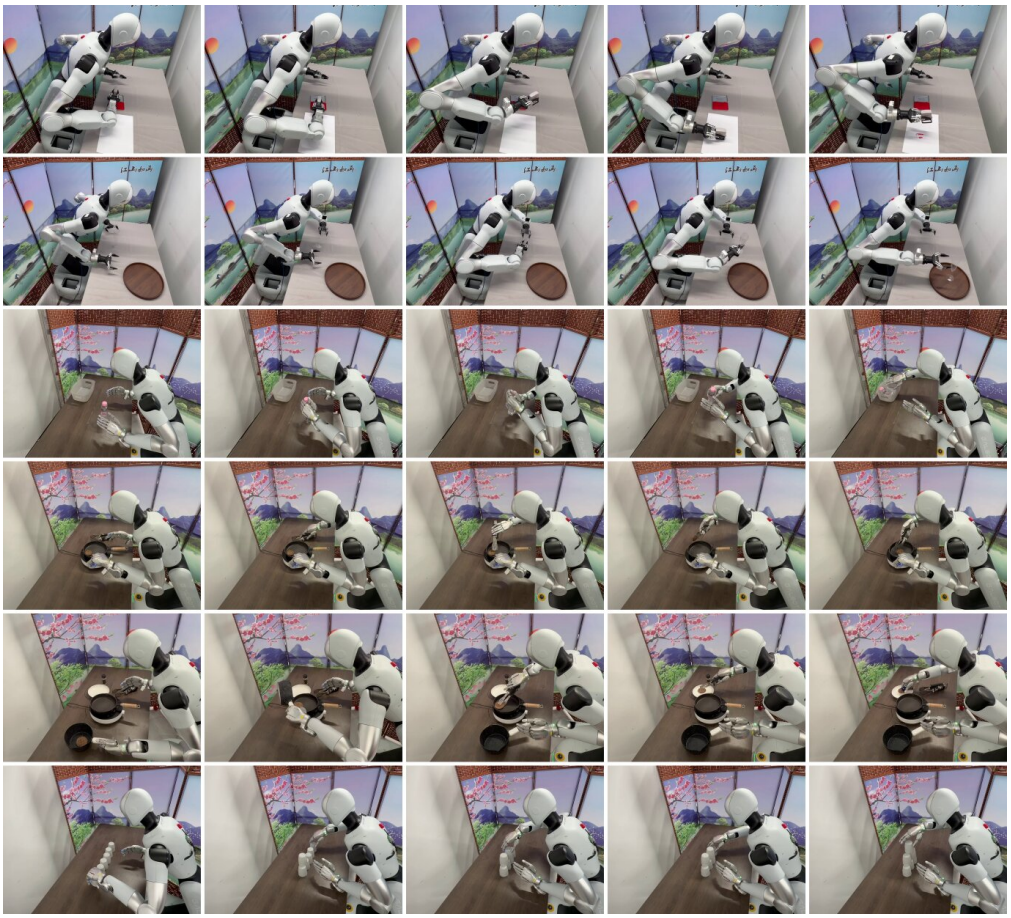

Figure E.2: **Direct generalization to an unseen lab scene without scene-specific human data.** WAM-TTT rollouts after meta-training only; no in-scene human videos are provided, so the TTT branch is driven entirely by the meta-trained slow projections and $W_{\text{init}}^{(\ell)}$. The lab scene differs from the standardized cubicle in lighting, tablecloth, object instance, and target-pose offset *simultaneously*. From top to bottom, the six rollout strips show: *Stamp Paper* and *Transfer Bottle* on the **Galbot gripper**; *Swap Place*, *Flip Steak*, *Multi-step Steak*, and *Pyramid Stacking* on the **Galbot sharpa**. The behaviour transfers despite the joint OOD shift, showing that the calibration achieved during meta-training is not tied to the in-cubicle observation statistics.

E.3 Robustness across six axes of in-scene distribution shift

Appendix E.2 stress-tested WAM-TTT with *no* in-scene human videos at all. Here we consider the complementary stress test: the deployment scene *does* provide in-scene human videos, but the scene itself (and therefore the human videos used by test-time TTT) departs from the meta-training cubicle along six different perturbation axes that we evaluate one at a time. Each axis is applied to

both the robot rollout scene and the paired human-side videos consumed by the test-time inner loop (Eq. 8), so each axis is a single-factor OOD perturbation of the entire deployment signal rather than a synthetic perturbation of only one stream.

This is the regime where standard adaptation pipelines tend to fail. Methods that *co-train* on the perturbed human data (e.g. the WAM-COTRAIN baseline of Section 4.1) propagate the human-side domain shift into the full backbone, so a few perturbed in-scene videos are enough to noticeably erode the pretrained policy. Methods that condition on human videos *in context* (e.g. WAM-ICL) likewise feed the perturbed observation statistics directly into the conditioning stream, which Table C.1 already shows collapses sharply on the household *New* split for the same reason. WAM-TTT's TTT branch avoids both failure modes by construction: at test time only the fast weights $W^{(\ell)}$ are updated, while Θ_{WAM}, the slow projections $\theta^{(\ell)}_{\{K,V,Q,O\}}$, and the fast-weight initialization $W^{(\ell)}_{\mathrm{init}}$ all remain frozen (Eq. 8). The human-side domain shift can therefore only rewrite the human-to-robot memory; it cannot rewrite the policy itself, and the WAM's pretrained visual reasoning and action prior are preserved.

The six perturbation axes shown in Figure E.3, in row order top-to-bottom:

- **Object generalization** (Unitree G1, dex-3 hand, *Table Bussing*). The tabletop is populated with novel object instances drawn from categories that the meta-training set never includes.
- **Lighting generalization** (Galbot sharpa, *Swap Place*). The dominant light source is changed in color temperature (warm $\leftrightarrow$ cool), intensity, and direction relative to the cubicle.
- **Height generalization** (Galbot sharpa, *Swap Place*). The table is raised relative to the standardized cubicle height, shifting the robot-to-target geometry.
- **Chassis (background) generalization** (Galbot gripper, *Stamp Paper*). The robot chassis is repositioned and the desk is cluttered with distractor items, perturbing the background statistics of the egocentric scene.
- **Brand-new stamp / affordance generalization** (Galbot gripper, *Stamp Paper*). The stamp is replaced by an unfamiliar instance whose shape and grasp affordance differ from any stamp seen at training.
- **Object-position generalization** (Galbot gripper). The target placement lies outside the meta-training distribution, requiring the policy to interpolate a manipulation pose it has never been demonstrated.

All six rows are produced from the same checkpoint used to fill Table C.1; the test-time inner loop is run on the perturbed in-scene human videos for that axis, with no checkpoint switching or per-axis hyperparameter tuning. For the full rollout videos at native frame rate, please refer to our accompanying supplementary video.

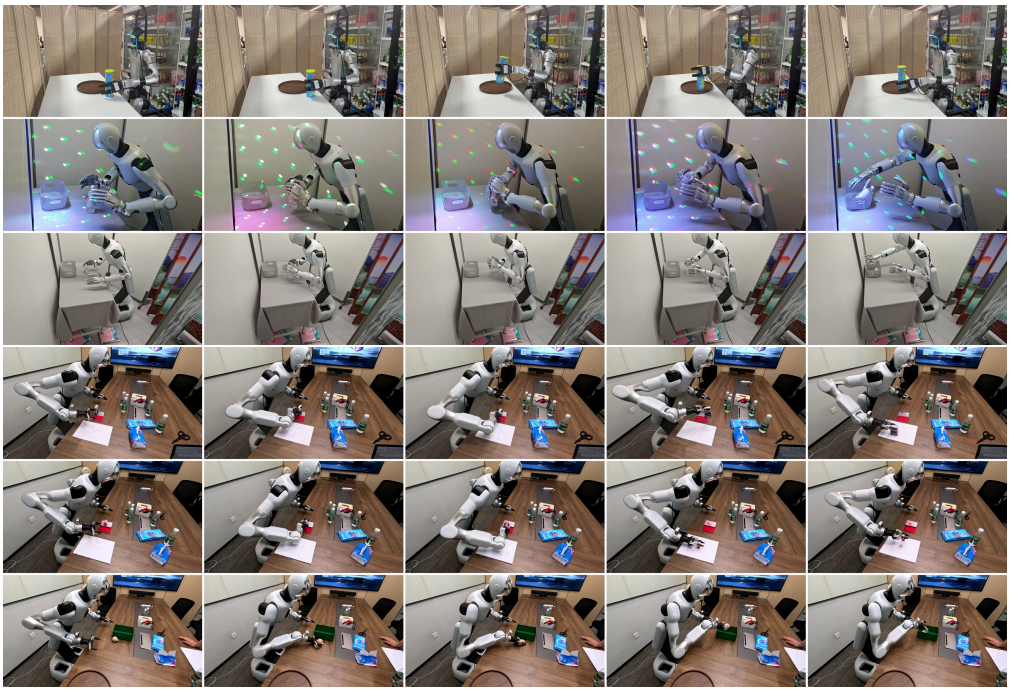

Figure E.3: **WAM-TTT across six axes of in-scene distribution shift.** Each row is a single rollout under one perturbation axis, shown as 5 evenly-spaced keyframes (left → right, initial → terminal frame). **Rows from top to bottom:** (1) *object* generalization, novel object instances on *Table Bussing* (Unitree G1, dex-3 hand); (2) *lighting* generalization, heavy color-temperature / intensity / direction shift on *Swap Place* (Galbot sharpa); (3) *height* generalization, raised-table on *Swap Place* (Galbot sharpa); (4) *chassis* generalization, chassis position shifted plus cluttered desk on *Stamp Paper* (Galbot gripper); (5) *brand-new stamp* / affordance shift on *Stamp Paper* (Galbot gripper); (6) *object-position* generalization, target placement outside the meta-training distribution (Galbot gripper). All rollouts come from the same checkpoint as Table C.1; the test-time inner loop is fed the perturbed in-scene human videos with no per-axis hyperparameter tuning.

This result is the architectural reason WAM-TTT is robust where vanilla adaptation is not: residual fast-weight TTT decouples the human-side update from the WAM-side parameters, so domain shift in the human videos cannot reach into and overwrite the WAM's pretrained capability.

E.4 Data-ratio ablation

Our default meta-training budget is $(r, h) = (100, 100)$ paired robot/human episodes per task. Table E.1 sweeps this ratio across three representative tasks. Rather than a full grid sweep, we keep one row per distinct question we want to answer:

- **(100, 0)** – no-human baseline at our robot budget; isolates the marginal value of paired human data.
- **Iso-budget triple at total** $= 200$ **episodes: (200, 0)**, **(100, 100)**, and **(10, 190)**. The three rows hold the total data-collection cost fixed and only vary the robot/human split, so any difference between them is attributable to the mix rather than to scale. **(200, 0)** is the robot-only upper bound and answers the natural challenge "why not just collect more robot data?". **(10, 190)** pushes the mix to the cheap-human extreme, testing whether human data can carry the policy when robot teleoperation is the bottleneck resource. **(100, 100)** is the deployed configuration used everywhere else in the paper.
- **(100, 200)** – adds 100 more paired human episodes on top of our default while keeping robot count fixed; probes whether the human-side gain has saturated at $h = 100$.

Table E.1: **Data-ratio ablation.** Progress (%) under the *New* setting at varying meta-training data budgets. r = robot demos per task, h = paired human demos per task. Our deployed setup is $(100, 100)$; each other row isolates a single question. All cells averaged over **25 trials per (configuration, task)**, matching the main-paper protocol of Section 4.1.

(robot, human)	Transfer Bottle	Table Bussing	Deliver Drink	Avg.
(100, 0)	44.1	90.0	44.4	59.5
Iso-budget triple: total $= 200$ *episodes per task*				
(10, 190)	42.1	68.0	44.2	51.4
(100, 100) *(ours)*	55.6	**100.0**	66.7	**74.1**
(200, 0)	47.9	**100.0**	**73.2**	73.7
(100, 200)	**58.9**	**100.0**	61.0	73.3

Takeaways. Three reads of the table line up with the design intent. *(i) Paired human data has a clear marginal value at fixed robot budget:* adding 100 paired human episodes on top of $(r, h) = (100, 0)$ raises the 3-task average from 59.5 to 74.1 ($+14.6$ pts), and is essentially free relative to the robot-side cost. *(ii) At the same total budget of 200 episodes per task, paired human data substitutes 1-for-1 for robot data:* the iso-budget triple shows $(100, 100)$ at 74.1 and $(200, 0)$ at 73.7 are statistically indistinguishable on the 3-task average, so the practitioner can halve the robot teleoperation cost without sacrificing performance. The cheap-human extreme $(10, 190)$ at 51.4, however, falls well below both, confirming that some robot grounding is required and that human data is a substitute, not a replacement, for the action-conditioned signal. *(iii) The human side has already saturated at our default:* doubling human episodes to $(100, 200)$ gives 73.3, marginally below $(100, 100)$ on the same 3-task average, so we use $h = 100$ as the deployed setup.

E.5 Model-architecture ablation

Whereas the main-paper baselines (Section 4.1) contrast WAM-TTT against alternative *training recipes* that all share a VLM-conditioned DiT backbone, Table E.2 instead ablates the *backbone composition* itself: every row carries our full meta-training and test-time TTT pipeline, and only the VLM side is varied. The goal is to justify the architectural choice of a pretrained, fully unfrozen VLM as the conditioning backend for the DiT.

Table E.2: **Model-architecture ablation.** Progress (%) on *Table Bussing* under the *New* setting. All variants retain the full meta-training plus test-time TTT pipeline of Section 3; only the VLM conditioning backend is changed. Each row averaged over **10 trials** per configuration; the main-paper 25-trial protocol of Section 4.1 is reduced here to keep the four-way architectural sweep tractable, so single-row figures should be read with a wider error margin than in the data-ratio ablation of Table E.1.

Configuration	Progress (%)
DiT only (no VLM backend)	72.0
DiT + VLM (no VLM pretrain)	80.0
DiT + VLM (VLM frozen)	54.0
DiT + VLM (VLM open) *(ours)*	**100.0**

Takeaways. Three reads of the table justify the architectural choice. *(i) The VLM backbone is load-bearing:* removing it entirely costs -28 pts (DiT-only at 72 vs. ours at 100). *(ii) The VLM's pretraining is load-bearing:* replacing the pretrained VLM with a randomly initialized one costs -20 pts (no-pretrain at 80 vs. ours at 100), so the visual-language prior is doing real work beyond contributing capacity. *(iii) Joint adaptation is not optional:* freezing the pretrained VLM costs -46 pts and drops below even the no-VLM configuration (54 vs. 72), indicating that a fixed pretrained representation, however general, becomes a bottleneck for the DiT once human-robot alignment requires adapting the conditioning features themselves. The joint meta-training of VLM and DiT is therefore the right setup for our pipeline.

E.6 Action-pseudolabel ablation

The original WAM-TTT design treats human videos as *action-free*: at test time only the video-generation loss $\mathcal{L}_{\text{vg}}^{\text{human}}$ drives the inner SGD on $W^{(\ell)}$ (Eq. 7), because no robot action stream is available on the human side. A natural alternative, used by several human-data pipelines [7, 10], is to extract a *pseudo-action* for each human frame and then train an action-conditioned objective on the human side as well. We test this alternative here.

Pipeline. For each human episode collected on the GoPro, we estimate the wrist 6-DoF pose with the MediaPipe hand tracker [41] combined with the EgoMimic-style estimation pipeline [7], fit a parametric MANO hand model [42] to recover the full hand pose, and retarget the resulting fingertip and palm targets to the target embodiment's joint configuration using the optimization-based retargeting protocol of EgoScale [10]. The output is a sequence of pseudo-qpos $\tilde{\boldsymbol{a}}_t^{\text{human}}$, the human-side analogue of robot teleop actions, available at the same frame rate as the egocentric video. Figure E.4 shows a representative 5-frame strip with the MediaPipe keypoints and MANO mesh overlaid on the original egocentric video, illustrating the typical quality of the single-view annotation that the FD pipeline consumes.

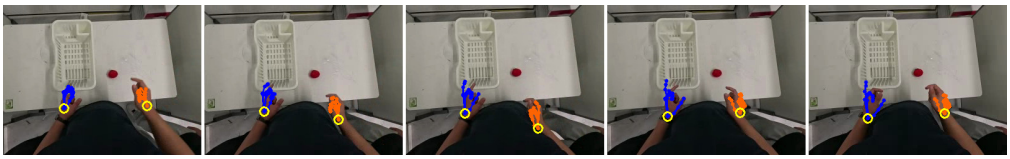

Figure E.4: **Representative single-view hand-pose annotation from the FD pipeline.** Five evenly-spaced frames of one human episode, overlaid with the MediaPipe [41] keypoints and the fitted MANO [42] mesh. The overlay is the input to the EgoScale-style [10] retargeter that produces the pseudo-qpos $\tilde{\boldsymbol{a}}_t^{\text{human}}$ used by the VG + FD variant in Table E.3. Visible imperfections of the monocular single-view fit (finger-tip drift, occluded thumb estimates, inconsistent palm normal across frames) propagate downstream into the retargeted pseudo-action and are the underlying reason the FD loss hurts.

Objective added on the human side. Given pseudo-actions, we can train one of the WAM-side objectives that the original WAM-TTT drops on human data: a *forward-dynamics* (FD) loss that, conditioned on the current observation $\boldsymbol{o}_t^{\text{human}}$ and the pseudo-action $\tilde{\boldsymbol{a}}_t^{\text{human}}$, predicts the next observation in the frozen DINOv3 [43] feature space. The human-data contribution at meta-training becomes $\mathcal{L}_{\text{vg}}^{\text{human}} + \lambda_{\text{FD}} \mathcal{L}_{\text{FD}}^{\text{human}}$ rather than $\mathcal{L}_{\text{vg}}^{\text{human}}$ alone; we use $\lambda_{\text{FD}} = 1$ throughout this ablation, so the FD term enters at the same scale as the video-generation term and is not artificially down-weighted. At test time both losses are available because both depend only on the in-scene human videos and their MANO-derived pseudo-actions.

Comparison. The two configurations compared in Table E.3 are: (i) VG ONLY (OURS), the deployed action-free WAM-TTT design, where the only human-side meta-training and test-time TTT loss is $\mathcal{L}_{\text{vg}}^{\text{human}}$, with no MANO retargeting and no pseudo-action; and (ii) VG + FD (PSEUDO-ACTION), the same backbone, the same meta-training schedule, the same paired robot-human dataset, and the same test-time TTT pipeline, but with the MANO retargeting pipeline (MediaPipe wrist + EgoMimic-style estimation $\rightarrow$ MANO hand $\rightarrow$ EgoScale-style retargeter $\rightarrow$ pseudo-qpos $\tilde{\boldsymbol{a}}_t^{\text{human}}$ matched to the target embodiment) producing pseudo-actions, and with the DINOv3-feature-space FD loss added to the human side at $\lambda_{\text{FD}} = 1$. The four tasks span all three embodiments and three end-effector families: *Transfer Bottle* on the **Galbot gripper** (two-finger), *Table Bussing* and *Deliver Drink* on the **Unitree G1** (dex-3 hand), and *Swap Place* on the **Galbot sharpa** (22-DoF dexterous). Both configurations are evaluated under the *New* household setting using the same main-paper 25-trial protocol of Section 4.1; the VG ONLY row reproduces the WAM-TTT entry of Table C.1 for these four tasks, so the cross-row delta directly isolates the contribution of the MANO retargeting and the FD loss.

Table E.3: **Action-pseudolabel ablation.** Progress (%) under the *New* setting; **25 trials per (configuration, task)**. Per-task embodiment: Transfer Bottle = Galbot gripper, Table Bussing / Deliver Drink = Unitree G1 (dex-3 hand), Swap Place = Galbot sharpa (dexterous). VG ONLY (OURS) is the deployed action-free WAM-TTT design (no MANO, no pseudo-action). VG + FD (PSEUDO-ACTION) adds the MANO retargeting pipeline (MediaPipe [41] + MANO [42] + EgoScale-style retargeter [10]) to produce an embodiment-matched pseudo-qpos $\tilde{a}_t^{\text{human}}$ per human frame, and adds the DINOv3 [43]-feature-space forward-dynamics loss $\mathcal{L}_{\text{FD}}^{\text{human}}$ on the human side at $\lambda_{\text{FD}} = 1$.

Configuration	Transfer Bottle	Table Bussing	Deliver Drink	Swap Place	Avg.
VG ONLY *(ours)*	**55.6**	**100.0**	**66.7**	**66.7**	**72.3**
VG + FD (pseudo-action)	14.2	33.3	26.8	41.2	28.9

Takeaways. Adding the retargeted pseudo-action and FD loss is uniformly harmful, dropping the 4-task average from 72.3 to 28.9 (-43.4 pts). The damage is largest on the two end-effector families where the MANO output does not map cleanly to the robot's actuation: -41.4 on *Transfer Bottle* (Galbot gripper, a one-DoF parallel jaw), -66.7 on *Table Bussing* and -39.9 on *Deliver Drink* (Unitree G1 dex-3 hand). For both the gripper and the dex-3 hand, the binary or near-binary closure command is not naturally present in the MANO pose, so a hand-engineered post-processor is needed to derive the open/close signal. This post-processor compounds on top of the already-noisy single-view monocular hand-pose estimate, and the resulting pseudo-action is too far from the true robot action distribution to provide a useful FD supervision signal. Even on *Swap Place* on the Galbot sharpa, where the dexterous robot is the most direct geometric target for the MANO output, FD still costs -25.5 pts ($66.7 \rightarrow 41.2$): the residual single-view retargeting noise alone is enough to corrupt the learned forward dynamics. The conclusion supports the design choice of WAM-TTT: under current single-view hand-tracking and retargeting maturity, injecting retargeted pseudo-actions into the human-side training signal is net-negative, and keeping human videos action-free (so that only the noise-tolerant video-prediction loss $\mathcal{L}_{\text{vg}}^{\text{human}}$ supervises the human side) is the right call.