ALAM: Algebraically Consistent Latent Action Model for Vision-Language-Action Models

Zuojin Tang[1,2] **Haoyun Liu**[3,4,2] **Xinyuan Chang**[2] **Changjie Wu**[2] **Dongjie Huo**[5,2]
Yandan Yang[2] **Bin Liu**[6] **Zhejia Cai**[7] **Feng Xiong**[2] **Mu Xu**[2]
Jiachen Luo[8] **De Ma**[1] **Zhiheng Ma**[4*] **Gang Pan**[1*]
[1]Zhejiang University [2]Amap, Alibaba Group [3]Nanjing University
[4]Faculty of Computility Microelectronics, Shenzhen University of Advanced Technology
[5]Beijing University of Chemical Technology [6]PaXini AI
[7]Tsinghua University [8]Queen Mary University of London
zuojintang@zju.edu.cn; mazhiheng@suat-sz.edu.cn; gpan@zju.edu.cn
*Corresponding Author

Abstract

Vision-language-action (VLA) models remain constrained by the scarcity of action-labeled robot data, whereas action-free videos provide abundant evidence of how the physical world changes. Latent action models offer a promising way to extract such priors from videos, but reconstruction-trained latent codes are not necessarily suitable for policy generation: they may predict future observations while lacking the structure needed to be reused or generated coherently with robot actions. We introduce **ALAM** (**A**lgebraic **L**atent **A**ction **M**odel), an Algebraically Consistent Latent Action Model that turns temporal relations in action-free video into structural supervision. Given frame triplets, ALAM learns latent transitions that are grounded by reconstruction while being regularized by composition and reversal consistency, encouraging a locally additive transition space. For downstream VLA learning, we freeze the pretrained encoder and use its latent transition sequences as auxiliary generative targets, co-generated with robot actions under a joint flow-matching objective. This couples structured latent transitions with flow-based policy generation, allowing the policy to exploit ALAM's locally consistent transition geometry without requiring latent-to-action decoding. Representation probes show that ALAM reduces additivity and reversibility errors by 25-85$\times$ over unstructured latent-action baselines and improves long-horizon cumulative reconstruction. When transferred to VLA policies, ALAM raises the average success rate from 47.9% to 85.0% on MetaWorld MT50 and from 94.2% to 98.1% on LIBERO, with consistent gains on real-world manipulation tasks. Ablations further confirm that the strongest improvements arise from the synergy between algebraically structured latent transitions and joint flow matching. Our code is available at: *https://github.com/Mark-zjtang/ALAM/tree/main*.

1 Introduction

Vision-language-action (VLA) models have emerged as a promising paradigm for robot learning, integrating visual perception, language understanding, and action generation within a unified policy. Recent systems [6, 63, 25, 24, 14, 5, 21] show that large multimodal backbones, when fine-tuned on robot demonstrations, can perform a wide range of language-conditioned manipulation tasks. However, the scalability of this paradigm remains limited by the scarcity and cost of action-labeled robot data. In contrast, action-free videos contain abundant evidence of how the physical world

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

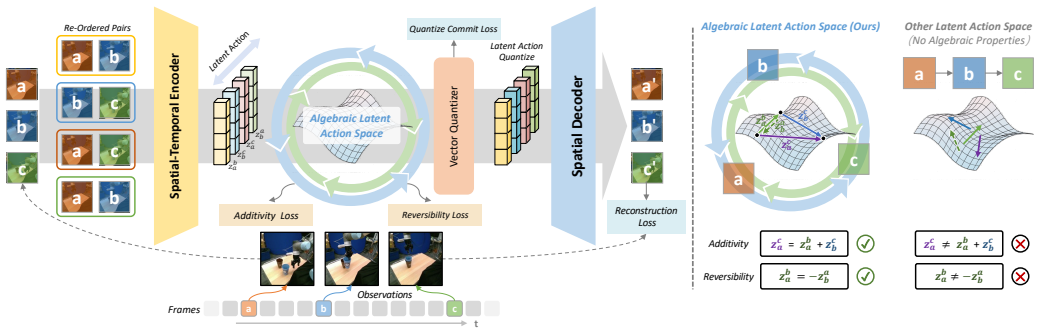

Figure 1: **Left: Pretraining pipeline.** From a sampled frame triplet (o_a, o_b, o_c), a relational encoder maps each pair (o_i, o_j) to a continuous latent transition z_i^j, and a decoder reconstructs $\hat{o}_j$ from o_i and z_i^j. Beyond conventional pair-based latent action models, ALAM further regularizes the latent space with additivity and reversal consistency losses. **Right: Algebraically structured latent action space.** Unlike conventional pair-based latent actions that only capture forward transitions, ALAM organizes transitions into two cycles representing forward and reverse temporal directions, with explicit additivity ($z_a^c \approx z_a^b + z_b^c$) and sign-reversibility ($z_b^a \approx -z_a^b$).

changes: objects are pushed, grasped, opened, stacked, and rearranged across diverse scenes and embodiments. A central challenge is therefore to extract behavior-relevant priors from such videos and transfer them effectively to downstream VLA policies.

Latent action models provide a natural route toward this goal by inferring transition variables from observation sequences without explicit action labels. Most existing methods [7, 11, 55, 12, 17] learn such variables through reconstruction or prediction objectives, encouraging the latent code to preserve information useful for recovering future observations. While reconstruction is an important grounding signal, it does not by itself guarantee that the learned latent is useful for policy generation. A reconstruction-trained code may capture appearance changes, camera motion, background dynamics, or other predictive factors that help recover nearby frames, while still lacking the structure needed to serve as a reliable auxiliary trajectory for control. In other words, a latent representation can be predictive without being easy for a generative policy to model jointly with actions.

This limitation suggests that latent-action pretraining should not be viewed merely as learning a compact code for future-frame reconstruction. Even without action labels, videos provide more than paired examples of before-and-after observations; they also contain temporal relations among changes. A longer visual transition can be compared with the composition of shorter consecutive transitions, and a transition observed in reverse should be consistent with undoing the corresponding forward change. These relations offer a form of self-supervised structural signal that is independent of explicit robot actions. The key idea of our work is to use such temporal relations to shape the latent space, so that the learned transitions are not only predictive of visual change but also more consistent when reused over time.

We propose **ALAM** (**A**lgebraic **L**atent **A**ction **M**odel), an Algebraically Consistent Latent Action Model that learns structured latent transitions from action-free video. During pretraining, ALAM samples frame triplets from unlabeled videos and constructs both forward and reversed observation pairs. A relational encoder maps each pair to a continuous latent transition, while a decoder reconstructs the target observation from the source observation and the inferred transition. Reconstruction grounds the latent variable in visual change, whereas composition and reversal losses regularize the latent space: consecutive transitions are encouraged to agree with the longer transition they span, and a reversed transition is encouraged to cancel its forward counterpart. In this way, ALAM turns temporal relations in action-free video into structural supervision for learning reusable latent transitions.

To transfer ALAM to downstream VLA policies, we freeze the pretrained encoder and use it to extract view-specific latent transition sequences from demonstration videos, temporally aligned with robot actions. Rather than decoding these latents into actions through a separate module, we train the policy to co-generate latent transition trajectories and robot action trajectories under a shared flow-matching objective. This coupling is important: ALAM turns reconstruction-grounded transition codes into locally consistent, approximately additive auxiliary trajectories, and flow matching can exploit this structure by modeling latents and actions along compatible interpolation paths. At inference time, the latent stream is generated jointly with the action stream but is not executed directly. It instead

provides an internal auxiliary trajectory that helps structure action generation over the horizon. This allows the downstream policy to exploit the locally consistent transition space learned by ALAM within the same flow-matching process used to generate robot actions.

We evaluate ALAM at three levels. At the representation level, algebraic probes show that ALAM substantially reduces additivity and reversibility errors compared with unstructured latent-action baselines, indicating that the proposed constraints reshape the latent transition space rather than merely improving reconstruction. At the prediction level, ALAM improves reconstruction fidelity at the training horizon and degrades more slowly under long-horizon cumulative composition, supporting the claim that local consistency improves temporal reuse. At the policy level, transferring the frozen encoder to VLA training yields large gains on MetaWorld MT50 and consistent improvements on LIBERO and real-world robotic tasks. Ablations further show that algebraic regularization benefits multiple generative frameworks, while the joint flow-matching formulation produces the strongest improvement, confirming the synergistic effect between structured latent transitions and flow-based policy generation. In summary, our contributions are as follows:

- We identify the lack of policy-friendly structure in reconstruction-trained latent actions as a key bottleneck for transferring action-free video pretraining to VLA control.
- We propose ALAM, a pretraining framework that turns temporal relations in action-free video into structural supervision through composition and reversal consistency.
- We introduce a joint flow-matching transfer mechanism that co-generates structured latent transition trajectories and robot actions, allowing the downstream policy to exploit the structure learned during pretraining.
- We validate ALAM through representation-level algebraic probes, long-horizon cumulative reconstruction, and downstream VLA policy learning on simulation and real-world manipulation benchmarks.

2 Related Work

2.1 Vision-Language-Action Models

Driven by the rapid progress of Large Language Models [16, 1, 2, 58, 52, 30] and Multimodal Large Language Models [48, 3, 44, 45, 46], together with the emergence of large-scale robot datasets, Vision-Language-Action (VLA) models have become a dominant paradigm in robot learning. Following the RT series [6, 63] that pioneered fine-tuning MLLMs on robot demonstrations, a line of work [25, 24, 38, 61, 42, 18, 32, 28] further improves scalability, controllability, and embodiment coverage. To better capture the multi-modal nature of robot actions, another line of work [14, 5, 21, 62, 36, 40, 53] replaces deterministic decoding with diffusion or flow matching, sampling actions from noise conditioned on observations, instructions, and proprioceptive states. Since directly mapping observations to actions lacks the explicit reasoning loop of LLM-style systems, recent methods [50, 19, 57, 20, 9, 41, 30, 60, 10, 54, 27, 51, 33] introduce an auxiliary forecasting step, either generating future frames externally or coupling pixel-level prediction with action prediction. However, such methods inherit the limitations of dense pixel reconstruction and are easily dominated by appearance details irrelevant to control. In contrast, our work bypasses pixel-level forecasting and instead learns algebraically structured latent transitions from action-free video, providing a compact, control-relevant prior for downstream policy learning.

2.2 Latent Action Models from Videos

Latent Action Models (LAMs) learn action-like representations directly from observation sequences to reduce the reliance on labeled robot actions. Early work [7, 11, 55] shows that unlabeled videos alone can supervise latent actions via inverse-dynamics or world-model objectives. Building on this, more recent methods couple forecasting with control by asking the policy to predict compact latent futures or latent actions alongside robot actions [12, 17, 8, 4, 15, 22, 56, 26, 23, 43], while a complementary line [13] grounds the latent space through a proprioceptive forward-dynamics module. Most of these methods do not impose explicit algebraic constraints on the latent space. A concurrent and independent preprint, AC-LAM [49], also studies algebraic structure in latent action spaces. While AC-LAM uses scene-wise additivity primarily to improve standalone latent-action

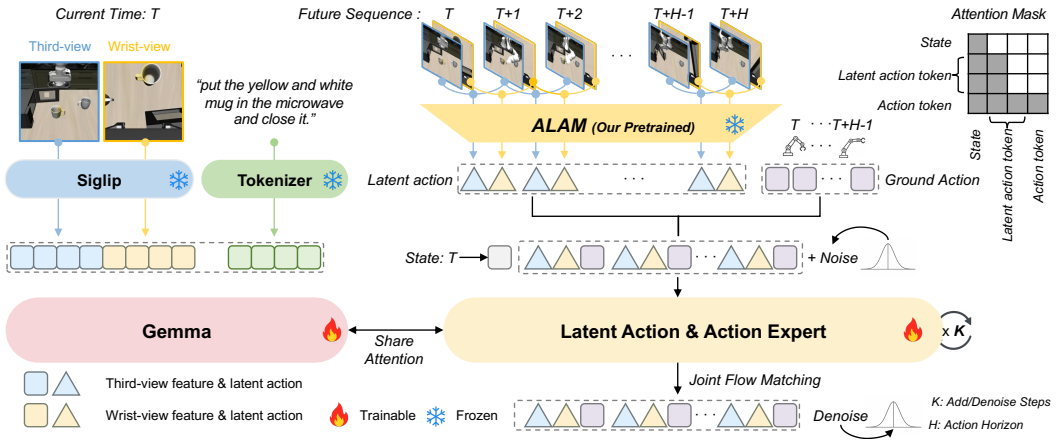

Figure 2: **Downstream transfer of ALAM.** The frozen ALAM encoder turns frame pairs over horizon $T{:}T{+}H$ into latent transition tokens, interleaved with action tokens. Conditioned on visual and language context, a shared expert co-generates both streams via K-step joint flow matching with the Gemma backbone; only the action stream is executed on the robot.

labels, ALAM uses algebraically structured transitions as auxiliary generative trajectories within a flow-based VLA policy.

3 Method

We introduce **ALAM**, a two-stage framework that learns latent transitions from action-free video and transfers them to vision-language-action (VLA) policy learning. In pretraining (Sec. 3.2), a relational encoder is trained with future-frame reconstruction together with two algebraic regularizers, composition consistency and reversal consistency, that encourage the latent transitions to be approximately additive and sign-reversible. At transfer (Sec. 3.3), the encoder is frozen and its latent transitions are jointly modeled with robot actions under a shared flow-matching objective on top of an off-the-shelf VLA backbone.

3.1 Structured Latent Transitions

A useful representation of how a scene changes between two observations should behave like a difference: composing two consecutive transitions should match the longer one they span, and reversing a transition in time should flip its sign. For any function f defined on a temporal index, the difference operator $\Delta(t_i, t_j) = f(t_j) - f(t_i)$ satisfies $\Delta(t_a, t_c) = \Delta(t_a, t_b) + \Delta(t_b, t_c)$ and $\Delta(t_b, t_a) = -\Delta(t_a, t_b)$. We use these two identities as soft constraints on a learned transition encoder. Given two observations O_i and O_j, we represent the transition from O_i to O_j by

$$z_i^j = E(O_i, O_j), \tag{1}$$

where $E(\cdot, \cdot)$ is a relational encoder that takes both endpoints as input. For three temporally ordered observations (O_a, O_b, O_c) with $a < b < c$, we impose

$$z_a^c \approx z_a^b + z_b^c, \qquad z_b^a \approx -z_a^b, \tag{2}$$

which we call composition consistency and reversal consistency. Both are imposed as soft regularizers, and we do not require any further structure on the latent space. See Fig. 1 for the pretraining pipeline.

3.2 Pretraining on Action-Free Video

The pretraining stage trains a latent transition encoder on unlabeled video so that its outputs are predictive of future frames and approximately respect the two consistency relations of Sec. 3.1. Concretely, given an unlabeled video corpus $\mathcal{D}$, we draw a temporally ordered triplet and assemble four transition pairs that span both temporal directions and two horizons:

$$(O_a, O_b, O_c) \sim \mathcal{D}, \quad a < b < c, \qquad \mathcal{T} = \underbrace{\{(a,b), (b,c), (a,c)\}}_{\mathcal{T}_{\text{fwd}}} \cup \underbrace{\{(b,a)\}}_{\mathcal{T}_{\text{rev}}}, \tag{3}$$

where (a, b) and (b, c) are the two short-horizon legs, (a, c) provides the long-horizon target for composition consistency, and (b, a) provides the time-reversed counterpart for reversal consistency.

Each pair $(O_i, O_j) \in \mathcal{T}$ is encoded into a latent transition by a relational encoder. A patch tokenizer f_{patch} first turns the two frames into patch sequences $X_i^{\text{patch}}, X_j^{\text{patch}}$, which we concatenate with K learnable query tokens $Q \in \mathbb{R}^{K \times d}$ along the sequence axis (denoted $[\cdot \,\|\, \cdot]$). A spatiotemporal Transformer f_θ processes the joint sequence; we read out the query positions, $(\cdot)^{\text{query}}$, and project them through a linear head g_ϕ:

$$z_i^{\,j} \;=\; g_\phi\Big(f_\theta\big([\,Q \,\|\, f_{\text{patch}}(O_i, O_j)\,]\big)^{\text{query}}\Big), \tag{4}$$

yielding the four latents $\big[z_a^{\,b},\, z_b^{\,c},\, z_a^{\,c},\, z_b^{\,a}\big]$. Each latent is then passed through a learned codebook $q(\cdot)$, with $z_{q,i}^{\,j} = q(z_i^{\,j})$ and the standard commitment loss

$$\mathcal{L}_{\text{vq}} \;=\; \big\|z_i^{\,j} - \text{sg}[z_{q,i}^{\,j}]\big\|_2^2, \tag{5}$$

where $\text{sg}[\cdot]$ is the stop-gradient operator and the codebook entries are updated by EMA [47].

For each forward pair $(i, j) \in \mathcal{T}_{\text{fwd}}$, two linear heads f_α, f_β, a cross-attention block h_ψ, and a pixel decoder u_ω produce the prediction

$$\hat{O}^{\,i,j} \;=\; u_\omega\Big(h_\psi\big([\,f_\alpha(z_{q,i}^{\,j}),\; f_\beta(X_i^{\text{patch}})\,]\big)\Big), \tag{6}$$

i.e., the decoder reconstructs O_j from $z_{q,i}^{\,j}$ together with the source patches. Conditioning on O_i in this way encourages the latent to carry only what is needed to map O_i to O_j, rather than the full content of O_j.

The two algebraic regularizers are imposed on the continuous latents prior to quantization, so that gradients reach the encoder unimpeded by the codebook lookup:

$$\mathcal{L}_{\text{add}} \;=\; \big\|z_a^{\,c} - (z_a^{\,b} + z_b^{\,c})\big\|_2^2, \qquad \mathcal{L}_{\text{rev}} \;=\; \big\|z_a^{\,b} + z_b^{\,a}\big\|_2^2, \tag{7}$$

which are squared relaxations of the two identities in Eq. (2). Pixel and perceptual reconstruction terms are averaged over $\mathcal{T}_{\text{fwd}}$,

$$\mathcal{L}_{\text{rec}} = \tfrac{1}{|\mathcal{T}_{\text{fwd}}|} \sum_{(i,j)\in\mathcal{T}_{\text{fwd}}} \big\|\hat{O}^{\,i,j} - O_j\big\|_2^2, \qquad \mathcal{L}_{\text{perc}} = \tfrac{1}{|\mathcal{T}_{\text{fwd}}|} \sum_{(i,j)\in\mathcal{T}_{\text{fwd}}} d_{\text{perc}}\big(\hat{O}^{\,i,j},\, O_j\big), \tag{8}$$

with d_{perc} the LPIPS distance [59]. Combining the five terms gives the total pretraining objective,

$$\mathcal{L}_{\text{ALAM}} \;=\; \lambda_{\text{vq}}\mathcal{L}_{\text{vq}} \;+\; \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} \;+\; \lambda_{\text{perc}}\mathcal{L}_{\text{perc}} \;+\; \lambda_{\text{add}}\mathcal{L}_{\text{add}} \;+\; \lambda_{\text{rev}}\mathcal{L}_{\text{rev}}. \tag{9}$$

At transfer time, we discard the quantizer and the decoder, and freeze the encoder $E = g_\phi \circ f_\theta$ as a fixed transition extractor on robot trajectories (Sec. 3.3).

3.3 Transfer via Joint Flow Matching

At transfer time, the goal shifts from reconstructing video frames to generating robot action sequences. We reuse the pretrained latent transitions as auxiliary generation targets in a flow-matching policy, without any fine-tuning of the encoder. As shown in Fig. 2, we apply the frozen ALAM encoder to a clip of $H{+}1$ consecutive frames $V^m = (I_1^m, \ldots, I_{H+1}^m)$ for each view $m \in \{\text{th}, \text{wr}\}$, where th and wr denote the third-person and wrist views. Applying the encoder to every adjacent frame pair gives H continuous latent transitions per view, time-aligned with the H ground-truth actions $u_{1:H}$,

$$z_t^m \;=\; \mathcal{F}_{\text{ALAM}}(I_t^m,\, I_{t+1}^m), \qquad t = 1, \ldots, H. \tag{10}$$

We take z_t^m to be the continuous output of the encoder, before the codebook lookup. The codebook in Sec. 3.2 is used only as a pretraining-time bottleneck; for joint flow matching we prefer a continuous, differentiable target with higher fidelity. Each z_t^m is then linearly projected to the dimension of u_t, and we co-generate the two latent streams together with the action stream under a shared flow-matching objective. To let the latent transitions condition the action stream, we order the three modalities at

Table 1: **Per-difficulty success rates on MetaWorld MT50. Avg.** is the macro-average over the four tiers. AR: autoregressive, FM: flow matching, VA: video-augmented, LA: latent action. To our knowledge, no prior latent-action VLA reports MT50 results under this protocol.

Method	Type	Size	Easy	Medium	Hard	Very Hard	Avg.
Diffusion Policy	DP	0.3B	23.1	10.7	1.9	6.1	10.5
TinyVLA-H	AR	1.3B	77.6	21.5	11.4	15.8	31.6
RT-2*	AR	7B	75.5	35.3	30.7	15.2	39.2
RoboTron Mani	AR	4B	85.5	67.7	76.7	81.0	77.7
GR-1*	VA	0.2B	76.6	35.3	46.0	44.0	50.5
PAD	VA	–	81.8	65.1	56.7	87.2	72.7
Evo-1	VA	0.8B	89.2	76.8	77.2	79.2	80.6
π_0	FM	3B	71.8	48.2	41.7	30.0	47.9
SmolVLA	FM	2B	87.1	51.8	70.0	64.0	68.2
π_0 + ALAM **(ours)**	LA	3B	**89.3**	**83.6**	**85.0**	**82.0**	**85.0**

* GR-1 and RT-2* scores are reproduced from PAD [39].

every timestep so that the two latent tokens come first and the action token last; the full sequence concatenates these per-timestep blocks,

$$\left[\, z_1^{\mathrm{th}},\, z_1^{\mathrm{wr}},\, u_1,\ \ \ldots,\ \ z_H^{\mathrm{th}},\, z_H^{\mathrm{wr}},\, u_H \,\right]. \tag{11}$$

Let y^m denote the clean sequence of modality $m \in \{u, \mathrm{th}, \mathrm{wr}\}$, with $y^{\mathrm{th}} = z_{1:H}^{\mathrm{th}}$, $y^{\mathrm{wr}} = z_{1:H}^{\mathrm{wr}}$, and $y^u = u_{1:H}$. All three modalities share the same linear data-to-noise interpolation,

$$x_\tau^m \;=\; \tau\, \epsilon^m + (1-\tau)\, y^m, \qquad \epsilon^m \sim \mathcal{N}(0, I), \qquad v_{\mathrm{tgt}}^m \;=\; \epsilon^m - y^m, \tag{12}$$

sampled at a common time $\tau = 0.999\,\xi + 0.001$ with $\xi \sim \mathrm{Beta}(1.5, 1)$ [5]. We further apply a structured attention mask over the interleaved sequence: at each timestep, the two view-specific latent tokens attend to each other but are isolated from action tokens, while each action token attends causally to all preceding tokens. Conditioned on context c (visual observations, language instructions, and proprioceptive states), the model predicts a per-modality velocity field $\hat{v}_\theta^m(x_\tau^m \mid \tau, c)$, trained with the per-modality flow-matching loss

$$\mathcal{L}^m \;=\; \mathbb{E}\big[\big\|\hat{v}_\theta^m(x_\tau^m \mid \tau, c) - v_{\mathrm{tgt}}^m\big\|_1\big], \qquad m \in \{u, \mathrm{th}, \mathrm{wr}\} \tag{13}$$

and the total transfer objective is the weighted sum

$$\mathcal{L}_{\mathrm{transfer}} \;=\; \lambda_{\mathrm{th}}\mathcal{L}^{\mathrm{th}} + \lambda_{\mathrm{wr}}\mathcal{L}^{\mathrm{wr}} + \lambda_u\mathcal{L}^u. \tag{14}$$

This design connects the two stages without retraining: the latent transitions used downstream come directly from the frozen encoder, and the policy is asked to generate them jointly with the actions, along the same interpolation schedule. The action stream is therefore not predicted in isolation, but co-denoised with a structured visual signal already shaped during pretraining.

At inference time, we initialize all modalities from Gaussian noise, $x_1^m \sim \mathcal{N}(0, I)$ for $m \in \{u, \mathrm{th}, \mathrm{wr}\}$, and jointly integrate from $\tau{=}1$ to $\tau{=}0$ with the learned velocity fields,

$$x_{\tau-\Delta_\tau}^m \;=\; x_\tau^m \;-\; \Delta_\tau\, \hat{v}_\theta^m(x_\tau^m \mid \tau, c), \tag{15}$$

following the same interleaved token order and attention mask as in training. Both streams are denoised jointly, but only u is executed on the robot.

4 Experimental Setup

Pretraining. ALAM is pretrained on 11 action-free video sources, mostly real-robot manipulation data from Open-X-Embodiment [37] and CALVIN [35]. We use AdamW on 128×H20 GPUs (90G) and run for 39 epochs (about 4 days), at which point the pretraining loss has plateaued. The full dataset configuration and hyperparameters are in Appendix A.

Table 2: **Success rates on LIBERO. Avg.** is the mean over four tasks. AR: autoregressive, FM: flow matching, VA: video-augmented, LA: latent action. For latent-action methods we annotate the underlying backbone in parentheses, so improvements over the corresponding backbone (gray row) directly reflect the contribution of the latent-action design.

Method	Type	Size	Spatial	Object	Goal	Long	Avg.
OpenVLA	AR	7B	84.7	88.4	79.2	53.7	76.5
SpatialVLA	AR	4B	88.2	89.9	78.6	55.5	78.1
CoT-VLA	AR	7B	87.5	91.6	87.6	69.0	83.9
WorldVLA	VA	7B	87.6	96.2	83.4	60.0	81.8
DreamVLA	VA	0.4B	97.5	94.0	89.5	89.5	92.6
SmolVLA	FM	2B	93.0	94.0	91.0	77.0	88.8
GR00T-N1	FM	2B	94.4	97.6	93.0	90.6	93.9
π_0	FM	3B	96.8	98.8	95.8	85.2	94.2
$\pi_{0.5}$	FM	3B	98.8	98.2	98.0	92.4	96.9
LAPA	LA	7B	87.4	91.2	90.0	65.4	83.5
UniVLA	LA	9B	96.5	96.8	95.6	92.0	95.2
JALA	LA	3B	96.0	98.2	97.4	**96.0**	96.9
π_0 + ALAM **(ours)**	LA	3B	**99.2**	**99.6**	**99.0**	94.4	**98.1**

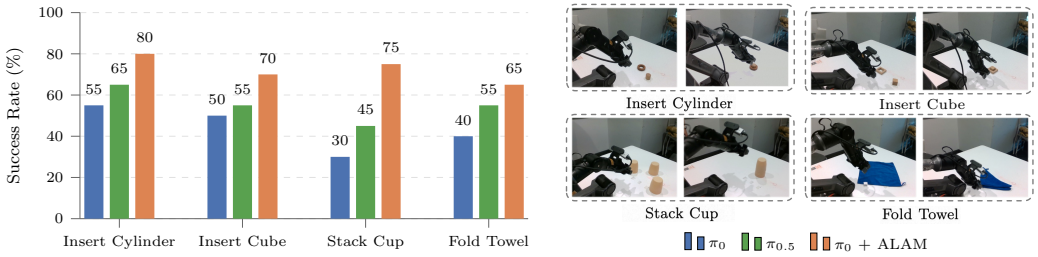

Figure 3: **Real-world results on a Piper 6-DoF manipulator.** ALAM is compared with π_0 and $\pi_{0.5}$ on four tasks: *insert cylinder*, *insert cube*, *stack cup*, and *fold towel*.

Downstream control. The transfer stage is built on π_0 [5]. The frozen ALAM encoder takes $H+1$ third-person and wrist frames and produces H latent transitions, which are linearly projected and interleaved with the action stream (Sec. 3.3). Policies are finetuned with flow matching on PaliGemma-2B [45] and Gemma-300M [45] on 8×H20 GPUs. The full configuration is in Appendix A.2.

Evaluation benchmarks. We use three settings. **(i) MetaWorld MT50** [34]: a single policy is trained on all 50 tabletop tasks. Following Evo-1 [29], we report per-task success rates averaged over 10 trials per task and group tasks into Easy, Medium, Hard, and Very Hard tiers. All methods are trained for 30k steps. **(ii) LIBERO** [31]: four suites (Spatial, Object, Goal, Long), with the Long suite focusing on long-horizon and cross-scene generalization. We report success rates over 500 evaluation episodes per suite, again with 30k training steps. Training and inference details are in Appendix 9. **(iii) Real-world**: a Piper 6-DoF manipulator evaluated on four tasks (*insert cylinder*, *insert cube*, *stack cup*, *fold towel*); task descriptions and evaluation protocols are in Appendix D.

Baselines. We use π_0 [5] without latent actions or joint generation as the backbone reference, and compare against four families of VLA models: autoregressive (RT-2 [63], OpenVLA [25]), flow-matching ($\pi_{0.5}$ [21]), video-augmented (Evo-1 [29], PAD [19], GR-1 [50], DreamVLA [60], WorldVLA [9]), and latent-action (UniVLA [8], JALA [33], LAPA [55]). We also include LAM [7] as an unstructured latent-action baseline at the same encoder capacity: it is trained on image pairs without any algebraic constraint, while ALAM uses consecutive triplets with the reversal ($\mathcal{L}_{\text{rev}}$) and additivity ($\mathcal{L}_{\text{add}}$) losses. Per-benchmark protocols are in Appendix B.

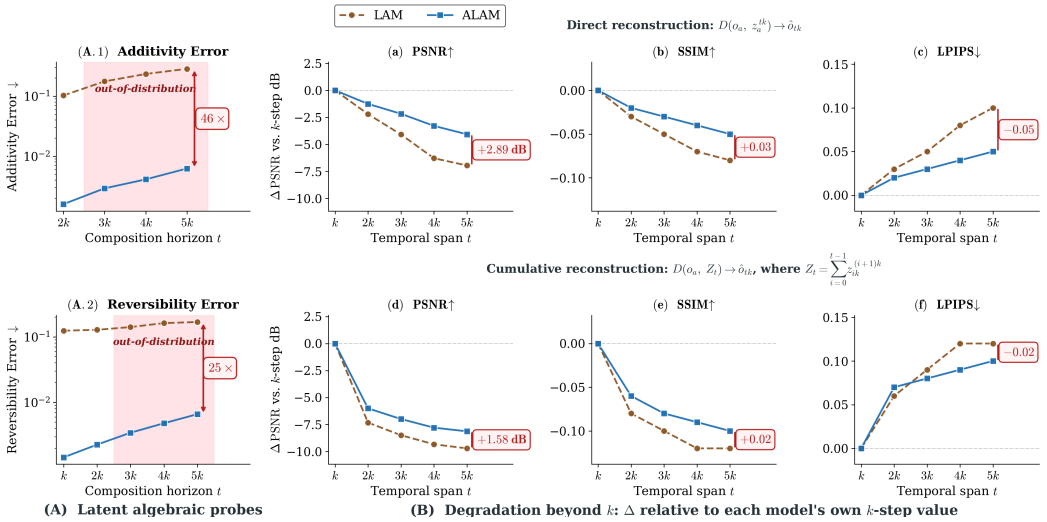

Figure 4: **LAM vs. ALAM on the 5% held-out split. (A)** Algebraic probes (log y, shaded $t \geq 3k$ not seen during training): **(A.1)** additivity error and **(A.2)** reversibility error, with LAM/ALAM ratios at $t{=}5k$ shown in red. **(B)** Per-horizon Δ from each model's own k-step score, for direct (top, (a)–(c)) and cumulative (bottom, (d)–(f)) reconstruction. See Sec. 4.1 for definitions and k-step scores.

4.1 Evaluating the algebraic structure of latent actions

We split the pretraining video corpus into $95\%/5\%$ train/test partitions, train ALAM and the LAM baseline on the 95% split, and evaluate the frozen encoder E and decoder D on the held-out 5%. All probes use anchor and target frames at horizons $t \in \{k, 2k, 3k, 4k, 5k\}$ with stride $k{=}5$ frames; training only supervises $t \in \{k, 2k\}$, so $t \in \{3k, 4k, 5k\}$ are not seen during pretraining. We write $z_a^b = E(o_a, o_b)$ for the latent transition between two frames, and $Z_t = \sum_{i=0}^{t-1} z_{ik}^{(i+1)k}$ for the sum of t consecutive short-horizon latents. We look at three things: how well each model reconstructs the target frame at long horizons, how closely the latent space respects the algebraic identities, and whether the same additivity carries over to videos from a different domain.

Reconstruction quality across horizons. For each anchor and horizon we report PSNR, SSIM, and LPIPS between the ground-truth target o_{tk} and two reconstructions: the direct one, $D(o_a, z_a^{tk}) \to \hat{o}_{tk}$, and the cumulative one, $D(o_a, Z_t) \to \hat{o}_{tk}$. At the training horizon $t{=}k$, ALAM and LAM reach PSNR 28.87 vs. 26.74 dB, SSIM 0.91 vs. 0.90, and LPIPS 0.08 vs. 0.08. Fig. 4(B) plots the per-horizon Δ relative to each model's own k-step score. Under both regimes ALAM keeps a smaller Δ than LAM at every $t{>}k$, and the PSNR gap at $t{=}5k$ reaches $+2.89$ dB (direct) and $+1.58$ dB (cumulative).

Algebraic consistency on unseen horizons. On the same latents we compute two consistency probes, the additivity error $\mathrm{Add}(t) = \mathbb{E}\,\|z_a^{tk} - Z_t\|$ and the reversibility error $\mathrm{Rev}(t) = \mathbb{E}\,\|z_a^{tk} + z_{tk}^{a}\|$. Both are zero for an ideally additive encoder, and $\mathrm{Add}(1)$ is zero by construction. Fig. 4(A) shows that LAM's errors grow with horizon, while ALAM stays one to two orders of magnitude lower across the whole range. The largest LAM/ALAM ratio is about $85\times$ on reversibility at $t{=}k$, and at $t{=}5k$ the ratio is $46\times$ for additivity and $25\times$ for reversibility. The same gap is preserved on the horizons $t \in \{3k, 4k, 5k\}$ that were never used during training.

Cross-domain additivity of latent actions. We also look at this additivity qualitatively, on samples from the pretraining mixture and on game videos that the encoder never saw during training. Given an anchor o_a and two endpoints o_b, o_c, decoding $z_a^b + z_b^c$ gives a frame that looks similar to the direct long-horizon reconstruction $D(o_a, z_a^c)$, on both the test backgrounds and the unseen game footage. Additional examples are in Appendix E.

4.2 Downstream Policy Learning

Main results. We now turn to downstream control and evaluate whether the pretrained structured transitions translate into better policies. On MetaWorld MT50, adding ALAM latent transitions to the π_0 backbone raises the average success rate from 47.9% to $\mathbf{85.0}\%$, a $+37.1$-point absolute gain, and also exceeds the strongest video-augmented baseline Evo-1 [29] by 4.4 points on the same protocol.

Table 3: **Generative framework ablation on MetaWorld MT50.** *LAM*/*ALAM*: unstructured/structured latents. **J.**: ✓ joint, ✗ two-stage. DP: diffusion, FM: flow matching.

Method	J.	Gen.	Easy	Med.	Hard	V.H.	Avg.
π_0	–	FM	71.8	48.2	41.7	30.0	47.9
+ LAM	✓	DP	70.0	37.3	20.0	26.0	38.3
+ ALAM	✓	DP	83.2	63.6	41.7	74.0	65.6
+ LAM	✓	FM	82.5	44.5	55.0	78.0	65.0
+ ALAM	✗	FM	83.2	53.6	56.7	82.0	68.9
+ ALAM **(ours)**	✓	FM	**89.3**	**83.6**	**85.0**	**82.0**	**85.0**

Table 4: **Loss ablation on MetaWorld MT50.** $\mathcal{L}_{\text{rev}}$ and $\mathcal{L}_{\text{add}}$ are added to the reconstruction loss to form the full ALAM objective. All variants use the epoch-39 checkpoint.

Variant	Easy	Med.	Hard	V.H.	Avg.
π_0	71.8	48.2	41.7	30.0	47.9
w/o VQ	84.6	71.8	66.7	60.0	70.8
w/o $\mathcal{L}_{\text{rev}}+\mathcal{L}_{\text{add}}$	82.9	60.0	51.7	88.0	70.7
w/o $\mathcal{L}_{\text{rev}}$	87.1	60.9	48.3	**90.0**	71.6
w/o $\mathcal{L}_{\text{add}}$	84.6	60.0	61.7	**90.0**	74.1
ALAM **(ours)**	**89.3**	**83.6**	**85.0**	82.0	**85.0**

On LIBERO, ALAM lifts π_0 from 94.2% to $\mathbf{98.1}\%$ averaged across the four suites, with the largest relative gain on the long-horizon LIBERO-Long split, the suite most sensitive to compositional reasoning. On the real-world Piper robot (Fig. 3), ALAM improves over π_0 and $\pi_{0.5}$ [21] on all four tasks, with the largest gain reaching 45 percentage points over π_0 on the *Stack Cup* task. The three settings together indicate that the pretrained transitions transfer from action-free video to physical manipulation, rather than fitting a particular simulator.

ALAM and flow matching combine well. Tab. 3 varies the latent structure (LAM vs. ALAM) and the policy coupling (two-stage vs. joint), under both diffusion and flow matching, on MetaWorld MT50. Three patterns stand out. (i) Adding LAM with joint diffusion reaches 38.3%, below π_0 alone (47.9%), so joint coupling does not help when the latent stream is unstructured. (ii) Replacing LAM with ALAM at the same coupling raises the average from 38.3% to 65.6% under diffusion and from 65.0% to $\mathbf{85.0}\%$ under flow matching, the largest single change in either column. (iii) With ALAM fixed under flow matching, switching from two-stage to joint coupling raises the average from 68.9% to $\mathbf{85.0}\%$ ($+16.1$ pts), and the joint flow-matching variant is also the best column overall. The pattern is consistent with the additivity and reversibility errors reported in Sec. 4.1.

Ablation on the algebraic losses. A natural confound when comparing ALAM with LAM is that ALAM additionally consumes triplet inputs at pretraining, so its gain over LAM could in principle reflect added input capacity rather than the algebraic constraints. To control for this, we train five encoder variants on the same single-domain pretraining subset and transfer each to π_0 with joint flow matching under matched downstream conditions (Tab. 4; epoch-39 checkpoints). Removing both algebraic losses (*w/o* $\mathcal{L}_{\text{rev}}+\mathcal{L}_{\text{add}}$) reaches 70.7%, only $+5.7$ pts over the LAM baseline (65.0%) and well below the full ALAM at $\mathbf{85.0}\%$ ($+20.0$ pts over LAM). Each loss on its own recovers most of this gap (*w/o* $\mathcal{L}_{\text{rev}}$: 71.6%; *w/o* $\mathcal{L}_{\text{add}}$: 74.1%), and using both gives the highest score. Within this controlled comparison, the structured latent transitions, rather than the additional inputs, account for most of the downstream gain.

5 Conclusion and Discussion

We presented ALAM, a pretraining framework that learns structured latent transitions from action-free video by regularizing a relational encoder with two algebraic constraints, composition consistency and reversal consistency, motivated by elementary identities of difference operators. The resulting latents are trained to be approximately additive and sign-reversible, and transfer best when the latent is co-generated with the robot action under a shared flow-matching objective rather than decoded through a separate module. On a held-out video split, ALAM reduces the additivity and reversibility errors by up to 25–85$\times$ relative to an unstructured latent-action baseline, and the cumulative reconstruction quality degrades more slowly with horizon. Transferring ALAM into a π_0 backbone matches or exceeds the best previously reported numbers we are aware of on MetaWorld MT50, with positive transfer on LIBERO and on a four-task real-world Piper setup. Composition and reversal consistency each improve over an unstructured LAM under both diffusion and flow matching, and the joint flow-matching variant gives the largest improvement in our ablations.

Limitations. ALAM provides a local and approximate algebraic regularizer rather than an exact group-structured latent space. Although the learned transitions reduce composition and reversal residuals and generalize to held-out temporal spans, the triplet-based constraints do not guarantee exact consistency or extrapolation to arbitrary horizons. In addition, joint flow matching introduces auxiliary latent tokens at inference time, increasing computation relative to action-only generation. However, these tokens are not decoded into future frames and are not executed by the robot, so the additional cost is confined to the denoising network.

Acknowledgements. This work is supported by the National Natural Science Foundation of China under Grant 62576224; the Guangdong Provincial Key Laboratory of Computility Microelectronics (2024B1212010007); the Guangdong Basic and Applied Basic Research Foundation Project (2026A1515011733); the Characteristic Innovation Project of Ordinary Universities in Guangdong Province (2024KTSCX026); the Research and Development Fund Project of China Huaneng Group Co., Ltd. (HNKJ24-H144).

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[3] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[4] Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. *arXiv preprint arXiv:2512.13030*, 2025.

[5] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026.

[6] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[7] Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

[8] Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.

[9] Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

[10] Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, et al. Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024.

[11] Xi Chen, Ali Ghadirzadeh, Tianhe Yu, Jianhao Wang, Alex Yuan Gao, Wenzhe Li, Liang Bin, Chelsea Finn, and Chongjie Zhang. Lapo: Latent-variable advantage-weighted policy optimization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:36902–36913, 2022.

[12] Xiaoyu Chen, Junliang Guo, Tianyu He, Chuheng Zhang, Pushi Zhang, Derek Cathera Yang, Li Zhao, and Jiang Bian. Igor: Image-goal representations are the atomic control units for foundation models in embodied ai. *arXiv preprint arXiv:2411.00785*, 2024.

[13] Xiaoyu Chen, Hangxing Wei, Pushi Zhang, Chuheng Zhang, Kaixin Wang, Yanjiang Guo, Rushuai Yang, Yucen Wang, Xinquan Xiao, Li Zhao, et al. Villa-x: enhancing latent action modeling in vision-language-action models. *arXiv preprint arXiv:2507.23682*, 2025.

[14] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[15] Zichen J Cui, Hengkai Pan, Aadhithya Iyer, Siddhant Haldar, and Lerrel Pinto. Dynamo: In-domain dynamics pretraining for visuo-motor control. *Advances in Neural Information Processing Systems*, 37:33933–33961, 2024.

[16] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and machines*, 30(4):681–694, 2020.

[17] Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. Adaworld: Learning adaptable world models with latent actions. *arXiv preprint arXiv:2503.18938*, 2025.

[18] Ankit Goyal, Hugo Hadfield, Xuning Yang, Valts Blukis, and Fabio Ramos. Vla-0: Building state-of-the-art vlas with zero modification. *arXiv preprint arXiv:2510.13054*, 2025.

[19] Yanjiang Guo, Yucheng Hu, Jianke Zhang, Yen-Jen Wang, Xiaoyu Chen, Chaochao Lu, and Jianyu Chen. Prediction with action: Visual policy learning via joint denoising process. *Advances in Neural Information Processing Systems*, 37:112386–112410, 2024.

[20] Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. *arXiv preprint arXiv:2412.14803*, 2024.

[21] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[22] Joel Jang, Seonghyeon Ye, Zongyu Lin, Jiannan Xiang, Johan Bjorck, Yu Fang, Fengyuan Hu, Spencer Huang, Kaushil Kundalia, Yen-Chen Lin, et al. Dreamgen: Unlocking generalization in robot learning through video world models. *arXiv preprint arXiv:2505.12705*, 2025.

[23] Hanjung Kim, Lerrel Pinto, and Seon Joo Kim. Hierarchical latent action model. *arXiv preprint arXiv:2603.05815*, 2026.

[24] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[25] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[26] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

[27] Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026.

[28] Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

[29] Tao Lin, Yilei Zhong, Yuxin Du, Jingjing Zhang, Jiting Liu, Yinxinyu Chen, Encheng Gu, Ziyan Liu, Hongyi Cai, Yanwen Zou, et al. Evo-1: Lightweight vision-language-action model with preserved semantic alignment. *arXiv preprint arXiv:2511.04555*, 2025.

[30] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[31] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

[32] Haoyun Liu, Jianzhuang Zhao, Xinyuan Chang, Tianle Shi, Chuanzhang Meng, Jiayuan Tan, Feng Xiong, Tong Lin, Dongjie Huo, Mu Xu, SongLin Dong, Zhiheng Ma, Yihong Gong, and Sheng Zhong. Neural implicit action fields: From discrete waypoints to continuous functions for vision-language-action models, 2026.

[33] Hao Luo, Ye Wang, Wanpeng Zhang, Haoqi Yuan, Yicheng Feng, Haiweng Xu, Sipeng Zheng, and Zongqing Lu. Joint-aligned latent action: Towards scalable vla pretraining in the wild, 2026.

[34] Reginald McLean, Evangelos Chatzaroulas, Luc McCutcheon, Frank Röder, Tianhe Yu, Zhanpeng He, K.R. Zentner, Ryan Julian, J K Terry, Isaac Woungang, Nariman Farsad, and Pablo Samuel Castro. Meta-world+: An improved, standardized, RL benchmark. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

[35] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

[36] Chaojun Ni, Cheng Chen, Xiaofeng Wang, Zheng Zhu, Wenzhao Zheng, Boyuan Wang, Tianrun Chen, Guosheng Zhao, Haoyun Li, Zhehao Dong, et al. Swiftvla: Unlocking spatiotemporal dynamics for lightweight vla models at minimal overhead. *arXiv preprint arXiv:2512.00903*, 2025.

[37] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[38] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

[39] Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked world models for visual control. In *Conference on Robot Learning*, pages 1332–1344. PMLR, 2023.

[40] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[41] Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. Vla-jepa: Enhancing vision-language-action model with latent world model. *arXiv preprint arXiv:2602.10098*, 2026.

[42] Zuojin Tang, Bin Hu, Chenyang Zhao, De Ma, Gang Pan, and Bin Liu. Vlascd: A visual language action model for simultaneous chatting and decision making. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 9223–9243, 2025.

[43] Zuojin Tang, Shengchao Yuan, Xiaoxin Bai, Zhiyuan Jin, De Ma, Gang Pan, and Bin Liu. One token per frame: Reconsidering visual bandwidth in world models for vla policy, 2026.

[44] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[45] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[46] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[47] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[48] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[49] Hangxing Wei, Xiaoyu Chen, Chuheng Zhang, Tim Pearce, Jianyu Chen, Alex Lamb, Li Zhao, and Jiang Bian. Learning additively compositional latent actions for embodied ai. *arXiv preprint arXiv:2604.03340*, 2026.

[50] Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. *arXiv preprint arXiv:2312.13139*, 2023.

[51] Junjin Xiao, Yandan Yang, Xinyuan Chang, Ronghan Chen, Feng Xiong, Mu Xu, Wei-Shi Zheng, and Qing Zhang. World-env: Leveraging world model as a virtual environment for vla post-training. *arXiv preprint arXiv:2509.24948*, 2025.

[52] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[53] Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, Junjin Xiao, Haoyun Liu, Ronghan Chen, Yuzhi Chen, Dongjie Huo, et al. Abot-m0: Vla foundation model for robotic manipulation with action manifold learning. *arXiv preprint arXiv:2602.11236*, 2026.

[54] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

[55] Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. *arXiv preprint arXiv:2410.11758*, 2024.

[56] Chuheng Zhang, Tim Pearce, Pushi Zhang, Kaixin Wang, Xiaoyu Chen, Wei Shen, Li Zhao, and Jiang Bian. What do latent action models actually learn? *arXiv preprint arXiv:2506.15691*, 2025.

[57] Jianke Zhang, Yanjiang Guo, Yucheng Hu, Xiaoyu Chen, Xiang Zhu, and Jianyu Chen. Up-vla: A unified understanding and prediction model for embodied agent. *arXiv preprint arXiv:2501.18867*, 2025.

[58] Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*, 2023.

[59] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018.

[60] Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, Fan Lu, He Wang, et al. Dreamvla: a vision-language-action model dreamed with comprehensive world knowledge. *arXiv preprint arXiv:2507.04447*, 2025.

[61] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[62] Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025.

[63] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

A Implementation Details

A.1 ALAM pretraining

The relational encoder E maps a frame pair (O_i, O_j) to a latent transition $z_i^j \in \mathbb{R}^{7\times128}$. Frames are tokenized with a 14×14 patch embedding, and the patch sequence is prepended with 7 learnable latent queries and processed by a 12-layer ViT. The query outputs are linearly projected to the latent transition space. During pretraining, z_i^j passes through a small VQ bottleneck and is fed to a symmetric 12-layer decoder that reconstructs $\hat{O}_j$ from O_i and the quantized latent. Architectural hyperparameters are listed in Tab. 5.

In the implementation summarized in Tab. 5, seven learnable transition-query tokens serve as action-query slots, producing a continuous transition tensor of shape [B, 7, 128], where B denotes the batch size. Separately, VectorQuantizer2 is configured with 256 shared codebook entries, resulting in an embedding table of shape [256, 128]. Here, the latent dimension of 128 specifies the feature dimension of each transition token and each codebook entry. The quantizer independently maps each of the seven tokens to one of these 256 shared entries and returns indices of shape [B, 7]. There is no intermediate pooling from 256 encoder queries to seven transition tokens.

Table 5: ALAM pretraining architecture.

Component	**Value**
Input resolution	$200\times200\times3$ (RGB)
Patch size	14
Hidden size / heads	768 / 12
Encoder / decoder depth	12 / 12
Shared VQ entries	256
Latent dimension	128
Learnable transition-query tokens	7

The pretraining loss combines pixel reconstruction, an LPIPS perceptual term [59], the two algebraic regularizers (composition and reversal consistency), and the standard VQ commitment loss. All terms are weighted equally at 1.0 (Tab. 6). For each video, we first apply temporal subsampling with a frame-skip of 5 by default and 10 for CALVIN. We then sample a temporally ordered triplet (O_a, O_b, O_c) from the subsampled sequence, with $a < b < c$. The four pairs $\{(a, b), (b, c), (a, c), (b, a)\}$ are encoded independently and supervised as in Sec. 3.2. We optimize with AdamW (lr 10^{-4}, weight decay 10^{-4}, β_1=0.9, β_2=0.95) and per-GPU batch size 32 for 39 epochs, on 128×H20 (90 GB) GPUs (about 4–5 days).

Table 6: Pretraining loss weights.

Term	**Weight**	**Role**
λ_{rec}	1.0	Pixel reconstruction of $\hat{O}_j$
λ_{perc}	1.0	LPIPS perceptual similarity
λ_{add}	1.0	Composition consistency
λ_{rev}	1.0	Reversal consistency
λ_{commit}	1.0	VQ commitment

The 11 video sources from Sec. 4 are mixed with the sampling weights in Tab. 7, and clips are drawn with probability proportional to their weight. Frames are decoded at the corresponding stride and resized to 200×200. No action, language, or proprioceptive labels are used at this stage.

A.2 Downstream adaptation

The transfer stage is built on π_0 [5], with PaliGemma-2B [45] as the vision-language encoder and Gemma-300M as the action expert. We do not add a latent-to-action decoder; latent transitions enter the policy through the input pathway in Sec. 3.3. For each view, the frozen ALAM encoder turns $H+1$ consecutive frames into H continuous latent transitions, which are linearly projected to the

Table 7: Pretraining data mixture and sampling weights. Stride is the frame-skip used for decoding.

Dataset	Stride	Weight
fractal20220817 (RT-1)	5	150
BridgeData-V2	5	50
TACO-Play	5	5
JaCo-Play	5	20
Berkeley Cable Routing	5	20
RoboTurk	5	10
NYU Door-Opening	5	5
VIOLA	5	3
Berkeley AutoLab UR5	5	5
TOTO	5	5
CALVIN (`task_ABC_D`)	10	200

action dimension and interleaved with the action stream. The encoder E is frozen throughout, and only the π_0 backbone, action expert, and the linear projection are updated. Policies are trained with flow matching on 8×H20 GPUs; default hyperparameters are in Tab. 8, with per-task overrides (e.g., training steps for short vs. long horizons) noted in the corresponding benchmark protocol.

Table 8: Downstream finetuning hyperparameters (defaults).

Hyperparameter	Value
Optimizer	AdamW (β_1=0.9, β_2=0.95)
Learning rate	5×10^{-5}
Weight decay	10^{-4}
Per-GPU batch size	32
GPUs	8×H20 (90 GB)
Action chunk H	16
Vision resolution	224×224
Frozen modules	ALAM encoder E
Trainable modules	PaliGemma-2B, Gemma-300M, linear projection

B Simulation Benchmark Protocols

On MetaWorld MT50 [34], we train a single multi-task policy on all 50 tasks and follow the Evo-1 [29] protocol: each task is evaluated with 10 rollouts of up to 200 environment steps, with success defined by the task-specific success flag. We report per-tier means (Easy / Medium / Hard / Very Hard, with the same partitioning as Evo-1) and the overall average; the full task split is in App. C. The action horizon is 5 at both training and inference for π_0 and π_0+ALAM.

On LIBERO [31], we use the four suites (Spatial, Object, Goal, Long) and follow the original protocol: each suite is evaluated with 500 episodes from unseen initial states, and we report per-suite success and the overall average. All four suites share a single trained checkpoint with action horizon H=20, and only the inference-time action horizon and replan interval differ across suites (Tab. 9).

Table 9: LIBERO inference configuration. All suites share the same trained checkpoint (training H=20); only the inference action horizon and replan step differ.

Suite	Inference H	Replan step
Spatial	14	5
Object	14	10
Goal	18	7
Long	18	12

C Task Difficulty Definitions in MetaWorld MT50

We follow [39] and partition the MetaWorld MT50 tasks into four difficulty tiers (Tab. 10).

Table 10: Task difficulty partitioning for MetaWorld MT50.

Difficulty	Task Names
Easy (28 tasks)	`button-press`, `button-press-topdown`, `button-press-topdown-wall`, `button-press-wall`, `coffee-button`, `dial-turn`, `door-close`, `door-lock`, `door-open`, `door-unlock`, `drawer-close`, `drawer-open`, `faucet-close`, `faucet-open`, `handle-press`, `handle-press-side`, `handle-pull`, `handle-pull-side`, `lever-pull`, `plate-slide`, `plate-slide-back`, `plate-slide-back-side`, `plate-slide-side`, `reach`, `reach-wall`, `window-close`, `window-open`, `peg-unplug-side`
Medium (11 tasks)	`basketball`, `bin-picking`, `box-close`, `coffee-pull`, `coffee-push`, `hammer`, `peg-insert-side`, `push-wall`, `soccer`, `sweep`, `sweep-into`
Hard (6 tasks)	`assembly`, `hand-insert`, `pick-out-of-hole`, `pick-place`, `push`, `push-back`
Very Hard (5 tasks)	`shelf-place`, `disassemble`, `stick-pull`, `stick-push`, `pick-place-wall`

D Real-World Tasks: Setup and Protocol



Figure 5: **Key frames of real-world rollouts from ALAM.**

All real-world experiments use an Agile-X Piper 6-DoF arm with a parallel-jaw gripper. The robot is observed by two synchronized RGB cameras: a third-person Intel RealSense D435 on a tripod and a wrist-mounted RealSense D405. Both streams are resized to 224×224 and synchronized at the control rate. We collect 50 teleoperated demonstrations per task (200 in total), used only for downstream finetuning; no real-world data is used during ALAM pretraining.

We evaluate on four tasks. **Insert Cylinder.** The robot grasps a cylinder and inserts it into a matching socket base, with one object and its base on the tabletop per trial; we run 20 trials, and a trial counts as successful if the cylinder is correctly inserted and remains stably seated after release. The success rate is reported on a 0–100 scale, equivalent to 5 points per successful trial. **Insert Cube.** Same setup as *Insert Cylinder*, but with a cube; the geometry requires more accurate orientation alignment, making the task more sensitive to execution precision. **Stack Cup.** Three cups are placed upside down on the table, and the robot stacks the left and right cups onto the center cup in sequence. We run 10 trials; in each trial, stacking one cup gives 5 points and both cups give 10, so the total over 10 trials is on a 0–100 scale. **Fold Towel.** Starting from a flat towel, the robot folds it in half twice. We run 10 trials; each successful fold gives 5 points, so completing both folds gives 10 per trial, and the sum across trials is on a 0–100 scale.

E More Transfer Results



Figure 6: **Cross-domain additivity (ground-truth target reference).** Each panel has two rows. **Source** is a trajectory in the source domain; **Target** is the reconstruction in a different target domain obtained by transferring the inferred latent transitions. Columns: anchor o_a; forward state $\hat{o}_a^b$; forward state $\hat{o}_a^c$; and additive composition $\hat{o}_a^b + \hat{o}_b^c$, which should align with $\hat{o}_a^c$ if the latent transitions are additive. Hatted symbols denote reconstructions; un-hatted denote ground truth.

We pick two domains from the pretraining mixture and use one as source and the other as target. Fig. 6 uses ground-truth target-domain frames as the rendering reference, which gives an upper bound on visual fidelity that depends only on the decoder. Fig. 7 uses transferred latent transitions as the only source of target-domain dynamics, with no ground-truth target frames as reference. In both settings, the composed reconstruction $\hat{o}_a^b + \hat{o}_b^c$ is close to the direct $\hat{o}_a^c$ rollout across the four source/target pairs, including pairs that differ in viewpoint, embodiment, and scene appearance. We do not claim domain-invariance in general; the qualitative match supports using the latent transitions as an auxiliary signal for downstream policies (Sec. 3.3).

F Inference-time intervention on the latent stream

As an additional check, we apply three test-time interventions to the frozen π_0+ALAM model (no retraining or finetuning). *Freeze-at-noise* keeps the latent input fixed at the initial Gaussian noise sample throughout the flow-matching trajectory. *Block attention* masks the action-to-latent cross-attention. *Temporal shuffle* permutes the temporal order of the generated latents within each rollout while keeping their values. On MetaWorld MT50, all three interventions lower the average success rate from 85.0% to between 75 and 77 points (Tab. 11); we include this as a sanity check that the



Figure 7: **Cross-domain additivity (transferred target).** Same layout as Fig. 6, but the target rows are produced solely from transferred latent transitions, with no ground-truth target frames as reference.

latent stream is used at inference, and do not draw further conclusions from the relative ordering of the three interventions.

Table 11: Test-time interventions on the latent stream (MetaWorld MT50, per-tier success rate %). All rows share the same trained weights.

Variant	Easy	Med.	Hard	V.Hard	Avg.
π_0 (no-latent baseline)	71.8	48.2	41.7	30.0	47.9
ALAM (default)	**89.3**	**83.6**	**85.0**	**82.0**	**85.0**
Freeze latent	86.4	62.7	65.0	90.0	76.0
Block A→L attention	83.9	68.2	63.3	90.0	76.4
Shuffle latent	84.3	65.5	63.3	88.0	75.3

G Broader Impacts

This work may help reduce reliance on expensive action-labeled robot data and improve the data efficiency of robot policy learning, which can make robot learning research more accessible. As with other learned policies, real-world deployment still requires task-specific evaluation and system-level safety measures.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main claims of the paper: ALAM learns algebraically structured latent transitions from action-free video, improves algebraic consistency and long-horizon reconstruction, and improves downstream VLA policy performance on MetaWorld, LIBERO, and real-world tasks. These claims are supported by the experimental results in Secs. 4.1 and 4.2, Tabs. 1, 2, 3, and 4, Fig. 3, and Appendix F.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper explicitly discusses limitations in Sec. 5, including non-zero algebraic residuals, bounded temporal supervision, limited real-world evaluation on four tasks and one embodiment, and the added inference cost from latent-token generation.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present formal theoretical results such as theorems, lemmas, or proofs. The algebraic relations are used as modeling principles and training regularizers rather than as formal claims requiring proof.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides substantial details needed to reproduce the main results, including model architectures, loss definitions, training objectives, dataset mixture, sampling strategy, optimization hyperparameters, benchmark protocols, and evaluation settings; see Secs. 3.2, 3.3, and 4, and Appendices A, A.2, B, and D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Our code is available at: *https://github.com/Mark-zjtang/ALAM/tree/main*

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the main training and evaluation details, including architectures, data splits, hyperparameters, optimizers, action horizons, pretraining data mixture, benchmark protocols, and real-world evaluation setup; see Sec. 4 and Appendices A, A.2, B, and D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The paper reports averaged success rates and evaluation counts, and notes statistical noise for a small task tier in Appendix F, but it does not report confidence intervals, standard deviations across random seeds, or formal statistical significance tests for the main results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports the main compute resources used for pretraining and downstream finetuning, including GPU type, GPU count, memory, and approximate runtime; see Sec. 4 and Appendices A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors have reviewed the NeurIPS Code of Ethics and, to the best of our knowledge, the research conforms to it. The work uses standard robot-learning datasets and benchmarks and does not involve deceptive practices or human-subject experimentation.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses broader impacts in Appendix G, including the potential to reduce reliance on action-labeled robot data and the need for appropriate evaluation and safety measures in real-world deployment.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a high-risk generative model or scraped dataset requiring special safeguards. The method is evaluated as a robotics policy-learning component under standard research benchmarks and controlled robot settings.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [No]

Justification: The paper credits the original sources of the datasets, backbones, and benchmarks used throughout the experiments via citations, but it does not explicitly enumerate the licenses and terms of use of each external asset in the manuscript or appendix.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper does not currently release a new public dataset, codebase, or model asset as part of the submission.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing experiments or research with human subjects. The demonstrations used for real-world robot training are teleoperated robot demonstrations rather than a human-subject study.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject research requiring IRB review. No personal or sensitive human data is collected or analyzed.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core methodological contribution of this work is ALAM and its transfer mechanism, not a new LLM-based method. Existing PaliGemma/Gemma backbones are used only as standard downstream VLA components, as described in Sec. 4 and Appendix A.2.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.