Embedding Morphology into Transformers for Cross-Robot Policy Learning

Kei Suzuki, Jing Liu, Ye Wang, Chiori Hori, Matthew Brand, Diego Romeres, Toshiaki Koike-Akino
Mitsubishi Electric Research Laboratories (MERL), Cambridge, Massachusetts, USA
{kesuzuki, jiliu, yewang, chori, brand, koike}@merl.com

Abstract: Cross-robot policy learning—training a single policy to perform well across multiple embodiments—remains a central challenge in robot learning. Transformer-based policies, such as vision-language-action (VLA) models, are typically embodiment-agnostic and must infer kinematic structure purely from observations, which can reduce robustness across embodiments and even limit performance within a single embodiment. We propose an embodiment-aware transformer policy that injects morphology via three mechanisms: (1) kinematic tokens that factorize actions across joints and compress time through per-joint temporal chunking; (2) a topology-aware attention bias that encodes kinematic topology as an inductive bias in self-attention, encouraging message passing along kinematic edges; and (3) joint-attribute conditioning that augments topology with per-joint descriptors to capture semantics beyond connectivity. Across a range of embodiments, this structured integration consistently improves performance over a vanilla pi0.5 VLA baseline, indicating improved robustness both within an embodiment and across embodiments.

Keywords: vision-language-action model, imitation learning, multi-embodiment

1 Introduction

Transformer-based robot policies, especially vision-language-action (VLA) models [1, 2, 3], have advanced rapidly by scaling to large and diverse datasets, yielding increasingly generalist controllers that can execute a broad range of tasks. However, *cross-robot policy learning*—training a single policy to perform well across diverse robot embodiments—remains a central challenge. This includes robustness to embodiment changes in the real world, such as hardware variation or failures, design upgrades, or deployment on entirely different robot platforms. In practice, achieving strong performance across embodiments often requires additional data and training for each robot, e.g., via fine-tuning [4, 5] and/or replacing the action head [6, 7] to match the target action space. Such approaches reflect a structural limitation: many VLA policies are typically embodiment-agnostic and must learn kinematics and cross-joint coordination implicitly from observations, making cross-robot policy learning particularly challenging. This motivates injecting embodiment as an explicit inductive bias into the policy architecture. A common formulation represents robot morphology as a *kinematic graph*, where nodes correspond to actuated joints and edges capture physical connectivity. Prior work incorporates this structure through graph neural networks [8, 9, 10] or topology-aware attention in transformers [11, 12, 13], improving cross-robot policy learning. Nevertheless, existing embedding approaches face three challenges. (i) **Lack of a kinematic token interface:** VLA models such as $\pi_{0.5}$ [4] compress joint-space structure into a compact set of action tokens, making it unclear where to apply existing morphology-embedding methods. (ii) **Local–global trade-off in topology-aware attention:** Enforcing strong locality promotes kinematic message passing but can limit long-range coordination. (iii) **Missing joint semantics:** Existing methods do not capture per-joint semantics, even though joints with identical topology can play different functional roles.

10th Conference on Robot Learning (CoRL 2026), Austin TX, USA.

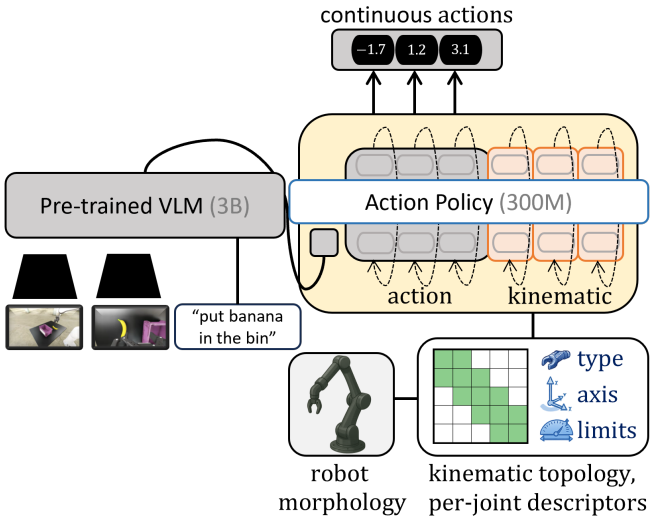


Figure 1: **Embedding robot morphology into a Transformer-based VLA policy**: We embed kinematic topology and per-joint semantics into the VLA to improves cross-robot policy learning.

Contributions We propose an embodiment-aware transformer policy that makes robot morphology an explicit input to the VLA action policy, rather than leaving kinematic structure to be inferred implicitly from observations. Our design provides a structured morphology interface through three components: (1) *kinematic tokens*, which provide a joint-wise action representation via temporal chunking, enabling direct morphology injection; (2) *topology-aware attention*, which injects local kinematic message passing while preserving global coordination through a local/global schedule; and (3) *joint-attribute conditioning*, which augments topology with per-joint semantics such as joint type, axis, and limits. Across DROID (Franka Panda), Unitree G1, and SO101, our method outperforms the vanilla $\pi_{0.5}$ baseline in single-embodiment, multi-embodiment, and real-world evaluations, demonstrating that our design improves robustness both within and across robot embodiments.

2 Related Work

Recent progress in robot policy learning has been driven by scaling vision-language-action (VLA) models—large transformer policies that couple a pretrained VLM backbone with an action policy head [1, 2, 3]. A common recipe is to pretrain on large-scale robot datasets [14, 15, 16], and then adapt to a target robot via fine-tuning or lightweight adaptation modules [6, 7]. Despite these advances, cross-robot policy learning—training a single policy to perform well across multiple embodiments—remains a central challenge, as these policies are typically embodiment-agnostic and must infer kinematic structure from observations alone. A common approach to improve cross-robot policy learning is to explicitly encode robot morphology as a *kinematic graph*, where nodes correspond to actuated joints and edges represent physical connectivity. This explicit representation provides a structured interface for multi-embodiment learning.

GNN policies with kinematic-graph message passing Graph neural network (GNN) policies encode morphology by running message passing on a kinematic graph, promoting body-wide coordination [8, 9, 10]. A key design question in this line of work is how information should propagate over the kinematic graph—balancing local interactions with global coordination. NerveNet [8] applies repeated local (one-hop) message passing so that information can gradually propagate across the body. SMP [9] further explores structured propagation by separating bottom-up and top-down information flow along the kinematic tree, aiming to capture hierarchical coordination patterns. Despite these advances, designing message-passing schemes that simultaneously support *local* and *global* information propagation remains challenging.

Transformer policies with topology-aware attention Transformer-based policies facilitate both local and global information exchange by self-attention, and inject kinematic priors by incorporating the kinematic graph directly into the attention mechanism [11, 12, 13]. Concretely, these methods use the kinematic graph to modulate attention, either by *Hard-Mask* attention or *Soft-Mask* attention. Hard-masking approaches [13, 17] inject topology by enforcing a binary attention constraint: each kinematic token can attend only to itself and its 1-hop kinematic neighbors in the kinematic graph, and all other joint pairs are disallowed (implemented by masking their attention logits before the softmax). To recover global coordination, some mixed designs alternate topology-masked (local) layers with fully connected (global) attention layers. Soft-Mask attention approaches [12, 18] inject topology by keeping full attention but adding a learnable, topology-conditioned bias term to the attention logits, so kinematically closer joints receive higher attention on average without strictly blocking any pairs. Although Soft-Mask approaches offer greater flexibility for embedding kinematics, prior works [13, 17] report optimization instability, whereas Hard-Mask designs tend to be more stable and can outperform Soft-Mask variants. In our method, we follow this taxonomy and consider both families of topology-aware attention within a unified framework (Full-/Mix-Mask for Hard-Mask; SPD-based bias for Soft-Mask). Despite these advances, three gaps remain.

- **VLA token-interface gap:** State-of-the-art VLA policies such as $\pi_{0.5}$ often compress joint-space structure into compact action tokens, making it difficult to inject morphology at the level of individual joints.
- **Local–global attention trade-off:** Strictly local topology masks promote kinematic message passing but can restrict long-range coordination, while fully connected attention weakens the topology prior.
- **Missing joint semantics:** Topology specifies how joints are connected, but not what each joint represents, such as joint type, axis, or limits.

These gaps motivate our structured morphology interface for VLA action policies.

3 Proposed Method

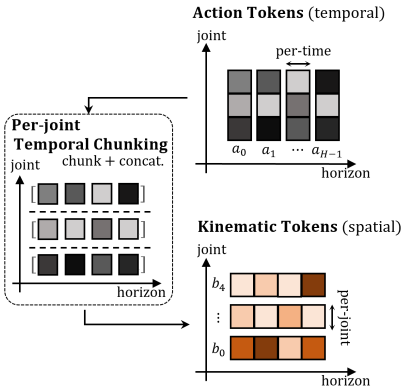
We propose an embodiment-aware transformer policy that injects robot morphology into the VLA action policy via three mechanisms: (1) kinematic tokens for factorized joint-action representation; (2) a topology-aware attention bias for kinematic message passing; and (3) joint-attribute conditioning for semantics beyond connectivity. An overview is depicted in Figure 2.

3.1 Kinematic tokens (KT)

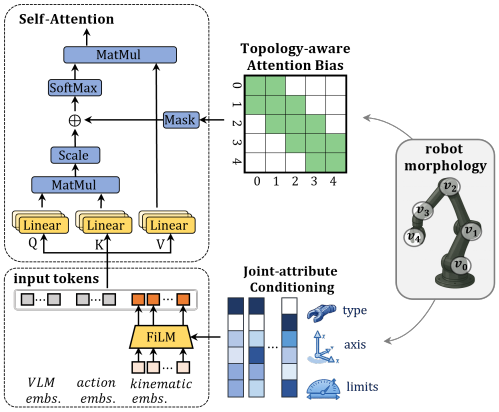
To factorize the action sequence across joints while compressing temporal information, we introduce *kinematic tokens* in addition to the standard action tokens used in VLA policies. The action tokens retain fine-grained temporal structure, while the kinematic tokens provide a compact, per-joint view that highlights cross-joint structure (Figure 2a). Let $a_{t,j}$ denote the scalar action for joint $j \in \{0, \dots, J-1\}$ at time $t \in \{0, \dots, H-1\}$, where J and H are the maximum number of joints and the maximum horizon length, respectively. The original $\pi_{0.5}$ VLA uses action tokens coupling all joints into one embedding per horizon, $[a_{t,j}]_{j=0}^{J-1}$, that prevents the use of topology-aware attention. To embed morphology, we decouple the action into the spatial domain. We then split the horizon into G non-overlapping temporal chunks of size $g = \frac{H}{G}$. For each joint j, we concatenate the g horizon actions within the chunk into a vector

$$b_{j,T_k} := [a_{t,j}]_{t \in T_k} \in \mathbb{R}^g, \tag{1}$$

where for each chunk $k \in \mathcal{K} := \{0, 1, \dots, G-1\}$, we define the index set $T_k = \{kg,\ kg+1,\ \dots,\ (k+1)g-1\}$. We refer to the vector b_{j,T_k} as the *kinematic token* for joint j and chunk k. This yields G kinematic tokens per joint (and JG kinematic tokens in total for J joints). We project each kinematic token b_{j,T_k} to a d-dimensional embedding $z_{j,T_k} = \mathrm{Enc}_0(b_{j,T_k}) \in \mathbb{R}^d$ with a lightweight multi-layer perceptron (MLP), and append them to the token sequence as additional



(a) **Kinematic Token (KT)**: Kinematic tokens provide a joint-wise interface for the VLA action policy. While the standard action tokens retain temporal structure, kinematic tokens compress the horizon into per-joint summaries, emphasizing cross-joint (spatial) structure and enabling topology/semantics embedding.



(b) **Topology bias and joint-attribute conditioning**: We embed kinematic topology and semantics in two ways: (i) a topology bias encourages kinematic message passing by restricting joint-to-joint self-attention to connected joints, and (ii) FiLM conditions kinematic-token embeddings on per-joint descriptors to disambiguate joint roles beyond connectivity.

Figure 2: **Embodiment-aware Transformer Policy**: Morphology is embedded via three mechanisms: (1) kinematic tokens; (2) topology-aware attention bias; and (3) joint-attribute conditioning.

context for the action expert of VLA. Following the VLA backbone, the policy predicts actions from the action tokens, which can attend to these *kinematic tokens* to leverage joint-wise structure. In our experiments, we find that using a single chunk (G=1) performs best.

Auxiliary kinematic tokens (AKT) Building on the kinematic token interface, we further increase the token capacity per joint by introducing *auxiliary kinematic tokens*. Specifically, for each kinematic token b_{j,T_k}, in addition to the standard embedding $z_{j,T_k} = \mathrm{Enc}_0(b_{j,T_k}) \in \mathbb{R}^d$, we generate M auxiliary embeddings using lightweight encoders with the same input but independent parameters. We then append these auxiliary tokens to the kinematic token sequence, so the transformer can attend to a richer set of per-joint representations.

$$z^{(m)}_{j,T_k} = \mathrm{Enc}_m(b_{j,T_k}) \in \mathbb{R}^d, \quad m = 1, \ldots, M, \tag{2}$$

3.2 Topology-aware attention

Vanilla self-attention treats the token sequence as a fully connected graph, allowing arbitrary interactions between joints. In contrast, robot embodiments admit a natural kinematic topology represented as a graph. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the kinematic graph, where each vertex $v_j \in \mathcal{V}$ corresponds to joint j, and each edge $(v_i, v_j) \in \mathcal{E}$ indicates physical connectivity. We encode this topology as an inductive bias in self-attention (Figure 2b).

Joint-to-joint self-attention modulation In the VLA token sequence, we inject kinematic structure only within the *joint-to-joint* attention block, while all other attention patterns remain identical to the original $\pi_{0.5}$ mask (Appendix 6). Let $\mathrm{logits}^{(\ell)}_{i,j}$ denote the scaled dot-product attention logits at layer ℓ between query joint i and key joint j. We modulate joint-to-joint attention by adding a topology-dependent term $B^{(\ell)}_{i,j}$:

$$\alpha^{(\ell)}_{i,j} = \mathrm{softmax}_j\left(\mathrm{logits}^{(\ell)}_{i,j} + B^{(\ell)}_{i,j}\right), \tag{3}$$

where $\alpha^{(\ell)}_{i,j}$ is the attention weight. We instantiate $B^{(\ell)}_{i,j}$ with three topology-bias variants: Full-Mask, Mix-Mask, and Soft-Mask. Full-Mask and Mix-Mask use a binary topology mask based on the 1-hop

(a) **DROID (Franka Panda):** "put the cube in the bowl"; "put the can in the mug"; "put banana in the bin"

(b) **Unitree G1 Dex1 (Unitree G1):** "Pick up the red block and place it inside the yellow box."

(c) **SO101:** "Grab orange and place into plate"

Figure 3: **Simulation environments**

neighborhood indicator

$$M_{i,j} = \begin{cases} 1, & (i = j) \text{ or } (i,j) \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases}$$

For masked layers, we set

$$B_{i,j}^{(\ell)} = \begin{cases} 0, & M_{i,j} = 1, \\ -\infty, & M_{i,j} = 0, \end{cases} \tag{4}$$

so that non-neighbor joints receive zero attention after the softmax.

Full-Mask applies Eq. (4) at every layer, enforcing strictly local joint-to-joint interactions. **Mix-Mask** alternates masked and unmasked layers: even-numbered layers apply Eq. (4), while odd-numbered layers use full attention ($B_{i,j}^{(\ell)} = 0$), balancing local message passing with global coordination. **Soft-Mask** biases attention by kinematic distance while retaining full attention paths, unlike binary masking which blocks non-neighbor joints. Let $d(i,j)$ denote the shortest-path distance between joints i and j on $\mathcal{G}$, with $d(i,i) = 0$. To encode topology without removing long-range attention paths, we use a learnable distance-indexed bias:

$$B_{i,j}^{(\ell)} = \theta_{d(i,j)}^{(\ell)}, \tag{5}$$

where $\theta_d^{(\ell)}$ is the bias for distance d at layer ℓ [19].

3.3 Joint-attribute conditioning

Topology-aware attention specifies *which* joints can exchange information based on kinematic connectivity, but connectivity alone does not capture the *semantics* of each joint (e.g., joints with similar local topology may play different functional roles). To complement topology, we condition kinematic-token embeddings on per-joint descriptors derived from robot morphology (Appendix B). For each joint j, we define a per-joint descriptor s_j using the features in Table 6, capturing the joint type, axis direction, motion limits, and contact-related properties. We use Feature-wise Linear Modulation (FiLM) [20] to map s_j to feature-wise scale and shift parameters.

$$\gamma_j, \beta_j = \mathrm{FiLM}(s_j), \tag{6}$$

where $\gamma_j, \beta_j \in \mathbb{R}^d$. Given the kinematic-token embedding before conditioning $z_j \in \mathbb{R}^d$, we apply feature-wise affine modulation. The conditioned embedding $\tilde{z}_j$ augments topology-based message passing with joint-specific semantics beyond connectivity.

$$\tilde{z}_j = (1 + \gamma_j) \odot z_j + \beta_j, \tag{7}$$

4 Experiments

We evaluate morphology-aware transformer policies in single-embodiment, multi-embodiment, and real-world settings to test whether explicit morphology improves performance across robot platforms.

Table 1: **Single-embodiment results on DROID**: We evaluate whether embedding robot morphology improves performance even under single-embodiment training on DROID (Franka Panda). Overall, each component improves performance, with the best result achieved by combining all three(KT+Mix+FiLM). **Bold**: best SR. Underline: second-best. $\pm\Delta$: 95% CI.

Kinematic Token (Chunk G=1)	Mask	FiLM	Success Rate (SR%) ↑ (95% CI)			
			Avg	Task 1	Task 2	Task 3
–	–	–	19.7 ± 4.5	18.3 ± 4.4	15.7 ± 4.1	25.0 ± 4.9
✓	–	–	36.0 ± 5.4	10.3 ± 3.6	67.7 ± 5.3	30.0 ± 5.2
✓	Full-Mask	–	30.1 ± 5.2	15.3 ± 4.1	34.0 ± 5.3	41.0 ± 5.5
✓	Mix-Mask	–	36.9 ± 5.4	27.7 ± 5.0	56.7 ± 5.6	26.3 ± 5.1
✓	Soft-Mask	–	26.1 ± 4.9	**29.7** ± 5.1	17.3 ± 4.3	31.3 ± 5.2
✓	–	✓	37.7 ± 5.5	6.0 ± 2.7	65.0 ± 5.4	42.0 ± 5.6
✓	Mix-Mask	✓	**47.4** ± 5.6	5.7 ± 2.7	**77.7** ± 4.7	**58.7** ± 5.5

Table 2: **Single-embodiment results on Unitree G1 Dex1**: Combining all three achieves the best SR. **Bold**: best SR. Underline: second-best.

KT (G=1)	Mask	FiLM	Avg SR%↑ (95% CI)
–	–	–	24.7 ± 4.9
✓	–	–	24.3 ± 4.9
✓	Full-Mask	–	23.8 ± 4.8
✓	Mix-Mask	–	27.3 ± 5.0
✓	–	✓	25.3 ± 4.9
✓	Mix-Mask	✓	**28.0** ± 5.0

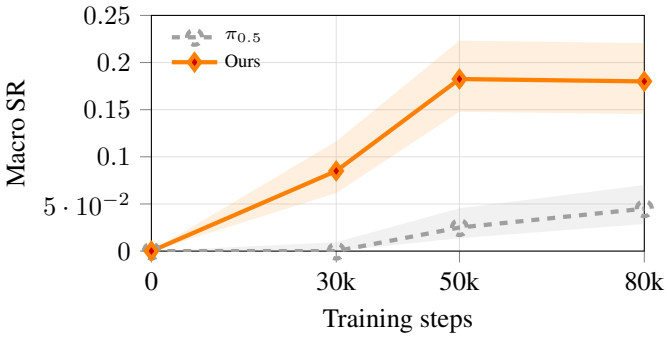


Figure 4: **Multi-embodiment results on Panda–SO101**: Our model improves performance throughout training.

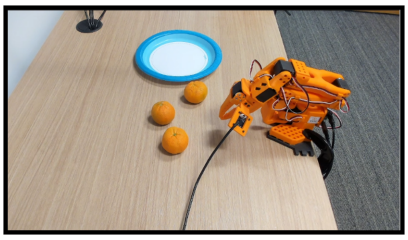
Benchmarks. For single-embodiment evaluation, we use DROID (Franka Panda) and Unitree G1 Dex1. For DROID, we train on a 1/8 subset and evaluate on three language-conditioned pick-and-place tasks [16, 21]. For Unitree G1 Dex1, we train on the 16-DoF joint-space dataset and evaluate in an IsaacLab-based simulator [22, 23]. For multi-embodiment evaluation, we jointly train on Panda and SO101 demonstrations, sampling Panda and SO101 batches with an 8:2 ratio. For SO101, we use the LeRobot dataset and LeIsaac simulator [24, 25]. We also evaluate DROID–SO101 joint-trained policies on a real SO101 robot over 30 trials, with no real-world SO101 fine-tuning.

Model variants and metrics. We evaluate $\pi_{0.5}$ variants with kinematic tokens, topology-aware attention, and joint-attribute conditioning. For topology-aware attention, we compare Full-Mask, Mix-Mask, and Soft-Mask. Kinematic tokens use a linear–SwiGLU–linear MLP, and joint-attribute conditioning uses a linear FiLM generator. The training configuration is summarized in Appendix D. As metrics, we report success rate (SR) over 300 simulation rollouts per condition, with 95% confidence intervals computed using the Wilson score interval [26]. For real-world SO101 evaluation, we report the total number of oranges successfully picked and placed over 10 trials per configuration.

5 Results

5.1 Single-embodiment results

Table 1 shows *average success rate* (Avg SR) on DROID. Overall, structured morphology encoding improves performance, and the best result is achieved by combining kinematic tokens with Mix-Mask and joint-attribute conditioning. Below, we isolate the contribution of (1) kinematic tokens, (2) topology-aware attention bias, and (3) joint-attribute conditioning. For completeness, Appendix E



(a) Real-world SO101 setup

(b) Real-world SO101 results

The table reports the total number of oranges successfully picked and placed over 10 trials per configuration.

Method	Config. 1	Config. 2	Config. 3	Total
$\pi_{0.5}$ baseline	2	3	2	7
Ours	**11**	**9**	**13**	**33**
Improvement	**+9**	**+6**	**+11**	**+26**

Figure 5: **Real-world evaluation on SO101**: We evaluate DROID–SO101 joint-trained policies on a real SO101 robot. Our method places 33 oranges in total, compared with 7 for the $\pi_{0.5}$ baseline.

reports *success time* on DROID as well as an ablation over encoder variants. **(1) kinematic tokens**: Compared to the baseline (Avg SR 19.7%), adding kinematic tokens improves Avg SR to 36.0%, showing that a joint-centric action representation is effective even without explicit topology or semantics. **(2) Topology-aware attention**: Starting from the kinematic token model (Avg SR 36.0%), Mix-Mask improves Avg SR to 36.9%, suggesting that alternating topology-constrained (local) and full (global) attention is beneficial. In contrast, Full-Mask reduces Avg SR to 30.1%, indicating that enforcing 1-hop locality at every layer can be overly restrictive. In addition, Soft-Mask achieves Avg SR 26.1%, which is lower than both masked variants. **(3) Joint-attribute conditioning**: Adding FiLM-based joint-attribute conditioning yields consistent gains. Without topology encoding, FiLM improves Avg SR from 36.0% to 37.7%, and with Mix-Mask Avg SR is improved from 36.9% to 47.4%, achieving the best overall performance. Notably, the full morphology encoding (KT+Mix+FiLM) yields significantly large gains over the baseline on Task 2 and Task 3, improving success rates by 5-fold and 2.3-fold, respectively. Table 2 shows Avg SR on Unitree G1 Dex1. Consistent with DROID, the full morphology encoding (KT, Mix-Mask and FiLM) achieves the best performance (Avg SR 28.0%). Topology-aware attention masking and joint-attribute conditioning provide additional gains on top of kinematic tokens: Mix-Mask improves Avg SR from 24.3% to 27.3%, and adding FiLM further improves to 28.0%.

5.2 Multi-embodiment results on DROID + SO101

Figure 4 shows learning curves for multi-embodiment training on the Panda-SO101 mixture. We report Macro SR, defined as $(\mathrm{SR}_{\mathrm{DROID}} + \mathrm{SR}_{\mathrm{SO101}})/2$. Overall, our embodiment-aware policy (KT, Mix-Mask, and FiLM) achieves higher Macro SR than $\pi_{0.5}$ across the evaluated training checkpoints. At 30k steps, our method already reaches a Macro SR of 8.5%, while $\pi_{0.5}$ remains at 0.0%. At 50k steps, our method further improves to 18.3%, compared with 1.8% for $\pi_{0.5}$. At 80k steps, our method reaches 17.0%, whereas $\pi_{0.5}$ achieves 3.0%. These results indicate that morphology-aware conditioning improves multi-embodiment policy learning on average across the two embodiments.

5.3 Real-world results on SO101

Figure 5 reports real-world SO101 performance. Our method improves performance in all configurations, placing 33 oranges in total compared with 7 for the $\pi_{0.5}$ baseline. Beyond the aggregate count, the baseline places at most one orange per trial, whereas our method often places multiple oranges in a trial, including one trial with all three oranges placed. These results show that our morphology-aware design retains its advantage on real hardware.

5.4 Ablation study

This section examines key design choices in our embodiment-aware Transformer and further probes a strong alternative for topology-aware attention. We study whether performance depends on (i) the kinematic token chunk size G, (ii) auxiliary kinematic tokens (AKT), and (iii) bias initialization for the Soft-Mask variant, which is potentially more expressive. **Effect of temporal chunk size:** To study

Table 3: **Effect of temporal chunk size**: Using a single chunk (G=1) achieves the best.

Chunk G	Avg SR%↑
1	**36.0** ±5.4
2	35.8 ±5.4
4	34.4 ±5.3
8	30.5 ±5.2
16	33.3 ±5.3

Table 4: **Effect of auxiliary kinematic tokens**: Adding AKT improves performance, especially with Mix-Mask.

Mask	AKT	Avg SR%↑
–	–	36.0 ±5.4
–	✓	37.0 ±5.4
Full-Mask	–	30.3 ±5.2
Full-Mask	✓	33.0 ±5.3
Mix-Mask	–	37.0 ±5.4
Mix-Mask	✓	**47.3** ±5.6

Table 5: **Effect of bias initialization**: Mix initialization performs best among the Soft-Mask variants.

Init.	Avg SR%↑
Zero	26.1 ±4.9
Hard	25.1 ±4.9
Mix	**28.1** ±5.1
Linear	20.4 ±4.5

how the temporal resolution of kinematic tokens affects performance, we evaluated five temporal chunk sizes ($G \in \{1, 2, 4, 8, 16\}$), where smaller G corresponds to more aggressive temporal compression. Table 3 shows that using a single chunk (G=1) achieves the best Avg SR, while performance generally degrades as G increases. **Effect of auxiliary kinematic tokens (AKT):** We test whether increasing per-kinematic token capacity improves performance via auxiliary kinematic tokens (AKT). Table 4 shows that AKT consistently improves Avg SR across configurations. Notably, under Mix-Mask, AKT increases Avg SR from 37.0% to 47.3%, indicating that scaling kinematic token capacity can substantially boost performance. **Effect of bias initialization:** Since Soft-Mask attention is potentially more expressive yet optimization-sensitive, we test whether bias initialization improves its performance. We study four initializations that impose different topology priors at the start of training: *Zero:* All bias parameters are initialized to zero, corresponding to no topology prior. *Hard:* biases are initialized to approximate a full-mask prior by assigning a negative bias (e.g., -3) to non-neighbor joint pairs. *Mix:* To mimic the alternating local/global schedule, even layers use the Hard initialization while odd layers are initialized to Zero. *Linear:* Biases are initialized as a distance-dependent prior, linearly interpolating from 0 for near pairs to -3 for far pairs. Table 5 shows the effect of bias initialization. We found that Mix initialization performs best. Nevertheless, across all initializations, the Soft-Mask variant does not surpass the Hard-Masked variants (Table 1). This is consistent to some related reports [13, 17]. In Appendix F, we further investigate Soft-Mask variants, but we do not observe consistent improvements.

6 Conclusion

We presented an embodiment-aware transformer policy that injects robot morphology into the VLA action policy through kinematic tokens, topology-aware attention, and joint-attribute conditioning. Across DROID, Unitree G1 Dex1, and SO101, our method consistently improves over the vanilla $\pi_{0.5}$ baseline in single-embodiment, multi-embodiment, and real-world evaluations. Ablations further show that a compact temporal chunking design (G=1) is most effective, and that increasing per-joint token capacity with auxiliary kinematic tokens further improves performance. These results show that our structured morphology interface improves the robustness across robot embodiments.

7 Limitations

Our evaluation covers three manipulation embodiments and a limited set of pick-and-place tasks, leaving broader generalization across robot morphologies and task families unverified. Our method uses static joint descriptors to condition kinematic-token embeddings; while such descriptors are available in simulation or from robot specifications, they may be unavailable or inaccurate for real hardware. This limits deployment to robots with reliable morphology metadata, and motivates estimating joint descriptors directly from data in future work.

Acknowledgments

References

[1] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[3] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[4] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, et al. π0. 5: a vision-language-action model with open-world generalization, 2025. *URL https://arxiv. org/abs/2504.16054*, 1(2):3, 2025.

[5] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[6] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[7] R. Doshi, H. Walke, O. Mees, S. Dasari, and S. Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. *arXiv preprint arXiv:2408.11812*, 2024.

[8] T. Wang, R. Liao, J. Ba, and S. Fidler. Nervenet: Learning structured policy with graph neural networks. In *International conference on learning representations*, 2018.

[9] W. Huang, I. Mordatch, and D. Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pages 4455–4464. PMLR, 2020.

[10] J. Whitman, M. Travers, and H. Choset. Learning modular robot control policies. *IEEE Transactions on Robotics*, 39(5):4095–4113, 2023.

[11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[12] S. Hong, D. Yoon, and K.-E. Kim. Structure-aware transformer policy for inhomogeneous multi-task reinforcement learning. In *International Conference on Learning Representations*, 2021.

[13] C. Sferrazza, D.-M. Huang, F. Liu, J. Lee, and P. Abbeel. Body transformer: Leveraging robot embodiment for policy learning. *arXiv preprint arXiv:2408.06316*, 2024.

[14] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

[15] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. 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.

[16] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[17] D. Buterez, J. P. Janet, D. Oglic, and P. Lio. Masked attention is all you need for graphs. *arXiv preprint arXiv:2402.10793*, 2024.

[18] Y. Luo, M. Yao, and X. Xiao. Gcnt: Graph-based transformer policies for morphology-agnostic reinforcement learning. *arXiv preprint arXiv:2505.15211*, 2025.

[19] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021.

[20] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[21] A. Jain. Simulation evaluations for robot policies. https://github.com/arhanjain/sim-evals, 2024. Accessed: 2025-10-25.

[22] Unitree Robotics. G1 dex1 pickplaceredblock dataset sim. https://huggingface.co/datasets/unitreerobotics/G1_Dex1_PickPlaceRedBlock_Dataset_Sim, 2025. Accessed: 2025-10-25.

[23] Unitree Robotics. Unitree simulation isaaclab. https://github.com/unitreerobotics/unitree_sim_isaaclab, 2025. Accessed: 2025-11-15.

[24] Lightwheel AI. Leisaac pick orange dataset. https://huggingface.co/datasets/LightwheelAI/leisaac-pick-orange, 2025. Accessed: 2025-12-10.

[25] Lightwheel AI. Leisaac simulator. https://github.com/LightwheelAI/leisaac, 2025. Accessed: 2025-12-10.

[26] L. D. Brown, T. T. Cai, and A. DasGupta. Interval estimation for a binomial proportion. *Statistical science*, 16(2):101–133, 2001.

A Attention Mask

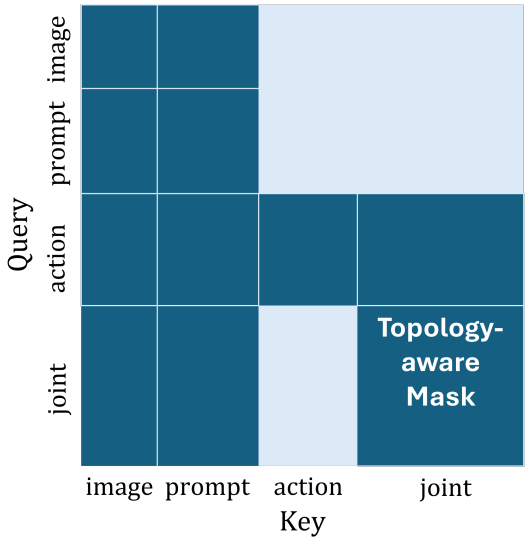


Figure 6: **Attention mask**: Attention mask used in our VLA action policy with kinematic tokens. Dark cells indicate unmasked attention and light cells indicate masked attention. Tokens are grouped by type for visualization (image/prompt/action/kinematic; each group may contain multiple tokens). We append kinematic tokens and apply a topology-aware mask only in the joint-to-joint block to encode kinematic connectivity, while all other attention patterns follow the base $\pi_{0.5}$ mask.

B Per-Joint Descriptor

Table 6: **Per-joint descriptors for joint-attribute conditioning**: Each joint j is represented by a descriptor s_j, which is used for FiLM-based conditioning of kinematic-token embeddings. The descriptor includes joint type indicators, axis direction, motion limits, and contact-related properties (log-transformed where indicated); the *Example* column shows a representative instance from the DROID (Franka Panda arm) embodiment.

Feature	**Description**	**Example**
`type_pris`	1 if prismatic joint, else 0	0
`type_rev`	1 if revolute joint, else 0	1
`ax`	Joint axis X component (unit vector)	0
`ay`	Joint axis Y component (unit vector)	0
`az`	Joint axis Z component (unit vector)	1
`hard_lower`	Hard lower limit [rad or m]	-2.9671
`hard_upper`	Hard upper limit [rad or m]	2.9671
`damping_log`	$\log$(contact damping)	6.90776
`friction_anchor`	Friction anchor flag (0/1)	1
`lateral_friction`	Lateral friction coefficient	1
`spinning_friction`	Spinning friction coefficient	0.1
`stiffness_log`	$\log$(contact stiffness)	10.30895

C Task Distribution in the DROID 1/8 Subset

We summarize the task distribution of the DROID 1/8 training subset used in our experiments. We compute frequency statistics of verbs (Figure 7) and objects (Figure 8) appearing in the task instructions.

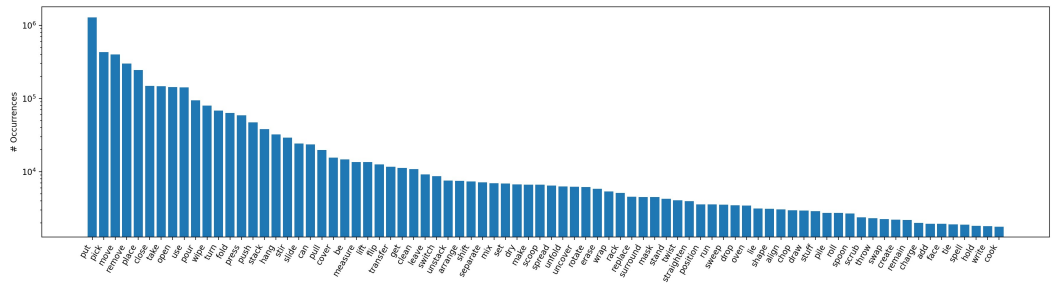


Figure 7: **Verb distribution in the DROID 1/8 subset**: Verb frequencies in task instructions.

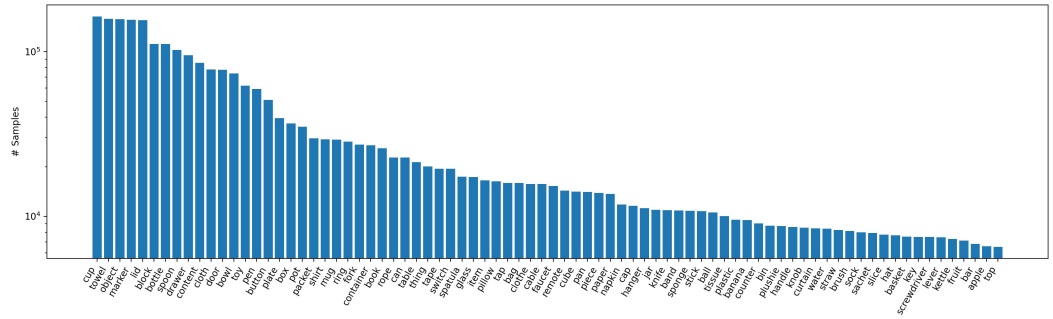


Figure 8: **Object distribution in the DROID 1/8 subset**: Object frequencies in task instructions.

D Training Protocol

All variants are initialized from the $\pi_{0.5}$ checkpoint `pi05-base`[1]. For architectures that add morphology modules, we zero-initialize the final layer of each module so that it starts near an identity mapping, stabilizing optimization. For each benchmark, our **baseline** is vanilla $\pi_{0.5}$, fine-tuned from `pi05-base` on that benchmark. Depending on computational budget, we use either action policy fine-tuning (**AP-FT**) which updates only the diffusion action expert, or full fine-tuning (**Full-FT**) which updates the entire model.

Table 7: **Training protocol**: Fine-tuning configurations and key hyperparameters for each benchmark. All runs start from `pi05-base` with a cosine learning-rate schedule. **AP-FT** updates only the action-policy components, whereas **Full-FT** updates the entire model.

Setting	Fine-tuning	Batch	Horizon	Steps
DROID (Panda)	AP-FT (∼450M)	32	16	100k
Unitree G1 Dex1	Full-FT (∼3.5B)	8	32	60k
DROID+SO101	AP-FT (∼450M)	32	16	80k

Table 8: **Trainable vs. frozen parameters in AP-FT**: VLM parameters are frozen, while the action-policy parameters are optimized.

Status	Part	Parameter name
Frozen: VLM part (2,923,335,408 parameters)		
Frozen	VLM	`PaliGemma/img/Transformer/encoder_norm/bias`
Frozen	VLM	`PaliGemma/img/Transformer/encoder_norm/scale`
Frozen	VLM	`PaliGemma/img/Transformer/encoderblock/LayerNorm_0/bias`
Frozen	VLM	`PaliGemma/img/Transformer/encoderblock/LayerNorm_0/scale`
Frozen	VLM	`PaliGemma/img/Transformer/encoderblock/LayerNorm_1/bias`
Frozen	VLM	`PaliGemma/img/Transformer/encoderblock/LayerNorm_1/scale`
Frozen	VLM	`PaliGemma/img/Transformer/encoderblock/MlpBlock_0/Dense_0/bias`
Frozen	VLM	`PaliGemma/img/Transformer/encoderblock/MlpBlock_0/Dense_0/kernel`
Frozen	VLM	`PaliGemma/img/Transformer/encoderblock/MlpBlock_0/Dense_1/bias`
Frozen	VLM	`PaliGemma/img/Transformer/encoderblock/MlpBlock_0/Dense_1/kernel`
Frozen	VLM	`PaliGemma/img/Transformer/encoderblock/MultiHeadDotProductAttention_0/key/bias`
Frozen	VLM	`PaliGemma/img/Transformer/encoderblock/MultiHeadDotProductAttention_0/key/kernel`
Frozen	VLM	`PaliGemma/img/Transformer/encoderblock/MultiHeadDotProductAttention_0/out/bias`
Frozen	VLM	`PaliGemma/img/Transformer/encoderblock/MultiHeadDotProductAttention_0/out/kernel`
Frozen	VLM	`PaliGemma/img/Transformer/encoderblock/MultiHeadDotProductAttention_0/query/bias`
Frozen	VLM	`PaliGemma/img/Transformer/encoderblock/MultiHeadDotProductAttention_0/query/kernel`
Frozen	VLM	`PaliGemma/img/Transformer/encoderblock/MultiHeadDotProductAttention_0/value/bias`
Frozen	VLM	`PaliGemma/img/Transformer/encoderblock/MultiHeadDotProductAttention_0/value/kernel`
Frozen	VLM	`PaliGemma/img/embedding/bias`
Frozen	VLM	`PaliGemma/img/embedding/kernel`
Frozen	VLM	`PaliGemma/img/head/bias`
Frozen	VLM	`PaliGemma/img/head/kernel`
Frozen	VLM	`PaliGemma/img/pos_embedding`
Frozen	VLM	`PaliGemma/llm/embedder/input_embedding`
Frozen	VLM	`PaliGemma/llm/final_norm/scale`
Frozen	VLM	`PaliGemma/llm/layers/attn/attn_vec_einsum/w`
Frozen	VLM	`PaliGemma/llm/layers/attn/kv_einsum/w`
Frozen	VLM	`PaliGemma/llm/layers/attn/q_einsum/w`
Frozen	VLM	`PaliGemma/llm/layers/mlp/gating_einsum`
Frozen	VLM	`PaliGemma/llm/layers/mlp/linear`
Frozen	VLM	`PaliGemma/llm/layers/pre_attention_norm/scale`
Frozen	VLM	`PaliGemma/llm/layers/pre_ffw_norm/scale`
Trainable: Action policy part (430,098,464 parameters)		
Trainable	Action	`PaliGemma/llm/final_norm_1/Dense_0/bias`
Trainable	Action	`PaliGemma/llm/final_norm_1/Dense_0/kernel`
Trainable	Action	`PaliGemma/llm/layers/attn/attn_vec_einsum_1/w`
Trainable	Action	`PaliGemma/llm/layers/attn/kv_einsum_1/w`
Trainable	Action	`PaliGemma/llm/layers/attn/q_einsum_1/w`
Trainable	Action	`PaliGemma/llm/layers/mlp_1/gating_einsum`
Trainable	Action	`PaliGemma/llm/layers/mlp_1/linear`

Continued on next page

[1] `gs://openpi-assets/checkpoints/pi05_base`

Status	Part	Parameter name
Trainable	Action	PaliGemma/llm/layers/pre_attention_norm_1/Dense_0/bias
Trainable	Action	PaliGemma/llm/layers/pre_attention_norm_1/Dense_0/kernel
Trainable	Action	PaliGemma/llm/layers/pre_ffw_norm_1/Dense_0/bias
Trainable	Action	PaliGemma/llm/layers/pre_ffw_norm_1/Dense_0/kernel
Trainable	Action	action_in_proj/bias
Trainable	Action	action_in_proj/kernel
Trainable	Action	action_out_proj/bias
Trainable	Action	action_out_proj/kernel
Trainable	Action	time_mlp_in/bias
Trainable	Action	time_mlp_in/kernel
Trainable	Action	time_mlp_out/bias
Trainable	Action	time_mlp_out/kernel

E Detailed DROID experiments

In addition to success rate (SR), this table reports time-to-success (task completion time) and includes additional ablation variants. Time-to-success is an auxiliary metric and may trade off with SR.

E.1 Full-/Mix-Mask variants without joint-attribute conditioning (FiLM)

Table 9: DROID Evaluation on Simulation

Training	Kinematic Token		Encoder	Mask	Batch	Train Data	Avg		Task 1		Task 2		Task 3	
	Chunk	AKT					SR(%)↑	Time[s]↓	SR(%)↑	Time[s]↓	SR(%)↑	Time[s]↓	SR(%)↑	Time[s]↓
Full-FT	–	–	–	–	256	Full Dataset	**66.3**	3.215	**65.7**	**2.526**	67.7	**4.178**	**65.7**	2.939
AP-FT	–	–	–	–	32	1/8 subset	19.7	4.849	18.3	4.731	15.7	6.616	25.0	3.201
AP-FT	1	–	linear	–	32	1/8 subset	26.7	4.606	14.7	5.746	34.3	4.984	31.0	3.088
AP-FT	1	✓	linear	–	32	1/8 subset	35.2	4.571	15.3	4.824	57.0	5.175	33.3	3.714
AP-FT	1	–	lin_swiGlu_lin	–	32	1/8 subset	36.0	4.925	10.3	5.450	67.7	5.254	30.0	4.072
AP-FT	1	✓	lin_swiGlu_lin	–	32	1/8 subset	37.0	4.361	11.0	4.618	60.3	5.110	39.7	3.355
AP-FT	16	–	linear	–	32	1/8 subset	16.9	5.026	14.3	5.565	16.7	6.090	19.7	3.422
AP-FT	16	✓	linear	–	32	1/8 subset	18.6	5.190	8.7	5.277	27.3	5.999	19.7	4.294
AP-FT	16	–	lin_swiGlu_lin	–	32	1/8 subset	33.3	3.724	27.3	3.489	55.7	4.500	17.0	3.182
AP-FT	16	✓	lin_swiGlu_lin	–	32	1/8 subset	31.0	4.876	16.7	5.664	53.7	4.748	22.7	4.216
AP-FT	1	–	linear	Full	32	1/8 subset	30.5	5.317	16.0	5.378	37.0	6.541	38.0	4.033
AP-FT	1	✓	linear	Full	32	1/8 subset	28.0	4.824	11.0	4.556	51.0	6.018	22.0	3.899
AP-FT	1	–	lin_swiGlu_lin	Full	32	1/8 subset	30.1	4.902	15.3	4.572	34.0	6.501	41.0	3.632
AP-FT	1	✓	lin_swiGlu_lin	Full	32	1/8 subset	33.0	4.520	16.0	4.296	61.0	5.120	22.0	4.144
AP-FT	16	–	linear	Full	32	1/8 subset	11.9	5.568	3.3	5.812	19.7	6.603	12.7	4.289
AP-FT	16	✓	linear	Full	32	1/8 subset	20.5	4.471	8.0	4.843	4.7	5.659	48.7	**2.912**
AP-FT	16	–	lin_swiGlu_lin	Full	32	1/8 subset	11.8	5.372	13.0	5.303	2.0	6.950	20.3	3.856
AP-FT	16	✓	lin_swiGlu_lin	Full	32	1/8 subset	16.9	5.203	13.7	6.026	3.0	6.258	34.0	3.326
AP-FT	1	–	linear	Mix	32	1/8 subset	27.7	4.835	13.3	5.777	33.0	5.370	36.7	3.358
AP-FT	1	✓	linear	Mix	32	1/8 subset	35.7	4.538	22.3	4.934	44.0	5.502	40.7	3.178
AP-FT	1	–	lin_swiGlu_lin	Mix	32	1/8 subset	36.9	4.979	27.7	4.848	56.7	5.555	26.3	4.534
AP-FT	1	✓	lin_swiGlu_lin	Mix	32	1/8 subset	47.1	**3.117**	21.0	5.010	**79.3**	4.589	41.0	3.548
AP-FT	16	–	linear	Mix	32	1/8 subset	21.1	5.272	0.0	–	21.7	6.326	41.7	4.218
AP-FT	16	✓	linear	Mix	32	1/8 subset	27.2	5.105	6.3	5.807	51.7	5.916	23.7	3.591
AP-FT	16	–	lin_swiGlu_lin	Mix	32	1/8 subset	19.0	5.284	0.0	6.240	25.0	5.949	30.3	3.662
AP-FT	16	✓	lin_swiGlu_lin	Mix	32	1/8 subset	29.6	4.849	5.3	4.510	55.3	5.689	28.0	4.348

E.2 Full-/Mix-Mask variants with joint-attribute conditioning (FiLM)

Table 10: DROID Evaluation on Simulation

Training	Kinematic Token		Encoder	Mask	FiLM	Batch	Train Data	Avg		Task 1		Task 2		Task 3	
	Chunk	AKT						SR(%)↑	Time[s]↓	SR(%)↑	Time[s]↓	SR(%)↑	Time[s]↓	SR(%)↑	Time[s]↓
AP-FT	–	–	–	–	–	32	1/8 subset	19.7	4.849	18.3	4.731	15.7	6.616	25.0	3.201
AP-FT	–	–	–	–	–	32	1/8 subset	19.7	4.849	18.3	4.731	15.7	6.616	25.0	3.201
AP-FT	1	–	lin_swiGlu_lin	–	–	32	1/8 subset	36.7	4.747	11.3	5.802	67.3	4.597	31.3	3.843
AP-FT	1	–	lin_swiGlu_lin	–	✓	32	1/8 subset	37.7	3.794	6.0	**3.727**	65.0	**4.488**	42.0	3.168
AP-FT	1	✓	lin_swiGlu_lin	–	–	32	1/8 subset	37.0	4.361	11.0	4.618	60.3	5.110	39.7	3.355
AP-FT	1	✓	lin_swiGlu_lin	–	✓	32	1/8 subset	39.4	4.245	14.6	4.502	72.7	4.606	31.0	3.626
AP-FT	1	–	lin_swiGlu_lin	Mix	–	32	1/8 subset	36.3	4.881	**27.0**	4.881	54.0	5.220	28.0	4.543
AP-FT	1	–	lin_swiGlu_lin	Mix	✓	32	1/8 subset	**47.4**	4.641	5.7	6.019	77.7	4.877	**58.7**	**3.028**
AP-FT	1	✓	lin_swiGlu_lin	Mix	–	32	1/8 subset	47.1	**3.117**	21.0	5.010	**79.3**	4.589	41.0	3.548

F Soft-Mask variants and stability exploration

Soft-Mask attention is potentially more expressive than Hard-Mask designs because it retains fully connected attention while injecting topology as an additive bias on the joint-to-joint attention logits. In practice, however, Soft-Mask can be optimization-sensitive. Motivated by the bias-initialization ablation in the main text, here we provide a more detailed exploration of Soft-Mask designs, focusing on parameterizations and initialization choices that may improve training stability. We organize Soft-Mask variants into two families: (i) **Adj-SoftMask**, which uses a binary adjacency indicator (connected vs. disconnected) and learns a suppression strength; and (ii) **SPD-SoftMask**, which uses a shortest-path-distance (SPD) index (Graphormer-style) and learns a distance-indexed bias table (the main-text choice).

Soft-Mask: general form: We inject topology into the joint-to-joint attention through an additive bias term $B_{ij}^{(\ell)}$ applied to the attention logits. Following the main text, we write

$$\alpha_{ij}^{(\ell)} = \mathrm{softmax}_j\left(\mathrm{logits}_{ij}^{(\ell)} + B_{ij}^{(\ell)}\right). \tag{8}$$

Different Soft-Mask variants correspond to different parameterizations of $B_{ij}^{(\ell)}$.

F.1 Adj-SoftMask (adjacency + learnable strength)

Adj-SoftMask parameterizes $B_{ij}^{(\ell)}$ using the binary adjacency indicator $M_{ij} \in \{0, 1\}$ defined in the main text ($M_{ij} = 1$ for $i = j$ or $(i, j) \in \mathcal{E}$, and $M_{ij} = 0$ otherwise). We instantiate the bias as

$$B_{ij}^{(\ell)} = (M_{ij} - 1)\, s_{ij}^{(\ell)}, \qquad s_{ij}^{(\ell)} \in \mathbb{R}, \tag{9}$$

so that $B_{ij}^{(\ell)} = 0$ when $M_{ij} = 1$ and $B_{ij}^{(\ell)} = -s_{ij}^{(\ell)}$ when $M_{ij} = 0$. Thus, the effect on non-adjacent pairs depends on the sign of $s_{ij}^{(\ell)}$.

Adj-SoftMask (v1.0): We instantiate Eq. (9) with an edge-wise, layer-wise strength

$$s_{ij}^{(\ell)} = \exp\left(\min\left(\theta_{ij}^{(\ell)}, \theta_{\max}\right)\right), \qquad \theta_{ij}^{(\ell)} \in \mathbb{R}. \tag{10}$$

Since $s_{ij}^{(\ell)} > 0$, the resulting bias on disconnected pairs ($M_{ij} = 0$) is always negative.

Adj-SoftMask (v1.1): We share the strength across edges within each layer by using a single scalar $\theta^{(\ell)} \in \mathbb{R}$:

$$s_{ij}^{(\ell)} = \exp\left(\min\left(\theta^{(\ell)}, \theta_{\max}\right)\right). \tag{11}$$

Again, $s_{ij}^{(\ell)} > 0$ implies a strictly negative bias on disconnected pairs.

Adj-SoftMask (v2.0): We use a layer-wise scalar without exponential mapping:

$$s_{ij}^{(\ell)} = \theta^{(\ell)}, \qquad \theta^{(\ell)} \in \mathbb{R}. \tag{12}$$

Compared to v1.x, this formulation avoids the sharp scaling introduced by $\exp(\cdot)$ and is intended to improve optimization stability.

Initialization: We consider two initializations, matching the main-text bias-initialization taxonomy:

- **Zero**: initialize the effect to be weak (close to NO-MASK).
- **Hard**: initialize the effect to be strong (close to HARD-MASK).

F.2 SPD-SoftMask

Adj-SoftMask uses a binary adjacency signal (M_{ij}) and cannot distinguish how far two joints are on the kinematic graph. SPD-SoftMask (the main-text SOFT-MASK) instead indexes $B_{ij}^{(\ell)}$ by the shortest-path distance (SPD), enabling distance-dependent inductive bias while preserving fully connected attention.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the kinematic graph and let $d(i, j)$ denote the SPD between joints i and j as defined in the main text. We parameterize the topology term using a learnable bias table:

$$B_{ij}^{(\ell)} = \theta_{d(i,j)}^{(\ell)}, \qquad \theta_d^{(\ell)} \in \mathbb{R},\ d \in \{0, 1, \ldots, D_{\max}\}. \tag{13}$$

The resulting joint-to-joint attention follows Eq. (8).

Bias initialization: To probe sensitivity to initialization, we consider four initializations of the SPD bias table $\theta_d^{(\ell)}$:

- **Zero**: initialize $\theta_d^{(\ell)} = 0$ (no topology prior).
- **Hard**: initialize $\theta_d^{(\ell)}$ to strongly suppress larger distances (strong locality prior).
- **Mix**: use **Hard** on even layers and **Zero** on odd layers.
- **Linear**: initialize $\theta_d^{(\ell)}$ with a distance-dependent prior, interpolating from 0 to a negative value (e.g., -3) as d increases.

Warm-start transfer: In addition to enabling SPD-SoftMask from `pi05-base`, we also study a warm-start setting where SPD-SoftMask is switched on starting from stronger checkpoints: (i) a `pi05-base+JT` model (trained with kinematic tokens but without SPD bias), and (ii) a `pi05-base+JT+Mix` model (trained with kinematic tokens and Mix-Mask). We then continue training under the same protocol to evaluate whether SPD-SoftMask can serve as a refinement step on top of strong masked baselines.

F.3 Results

Across these experiments (Table 11), we did not find a Soft-Mask configuration that surpasses the Hard-Mask **Mix-Mask** variant under our training protocol. Within Adj-SoftMask, v2.0 achieves the strongest results among the tested parameterizations, suggesting that learning the bias magnitude directly can be preferable to exponential mappings. For SPD-SoftMask, performance is sensitive to the bias initialization and warm-start choice, and we do not observe consistent gains over strong masked baselines.

Table 11: **Soft-Mask results (SR%)**: We report per-task success rates (Task1–Task3) in % along with macro averages, shown as SR% ± 95% CI.

Base Model	**Variant**	**Init.**	**Success Rate (SR%) (95% CI)↑**			
			Avg	**Task1**	**Task2**	**Task3**
pi05-base	Soft-Mask (v1.0)	Zero	25.9 ± 4.9	26.7 ± 5.0	20.3 ± 4.5	30.7 ± 5.2
pi05-base	Soft-Mask (v1.0)	Hard	25.4 ± 4.9	17.7 ± 4.3	15.7 ± 4.1	<u>42.7</u> ± 5.6
pi05-base	Soft-Mask (v1.1)	Zero	26.4 ± 5.0	21.3 ± 4.6	16.7 ± 4.2	41.3 ± 5.5
pi05-base	Soft-Mask (v1.1)	Hard	<u>29.5</u> ± 5.1	16.7 ± 4.2	<u>28.7</u> ± 5.1	**43.0** ± 5.6
pi05-base	Soft-Mask (v2.0)	Zero	**34.4** ± 5.3	27.6 ± 5.0	**37.0** ± 5.4	38.6 ± 5.5
pi05-base	Soft-Mask (v2.0)	Hard	23.0 ± 4.7	26.3 ± 5.0	11.6 ± 3.6	31.0 ± 5.2
pi05-base	Soft-Mask (v3.0)	Zero	26.1 ± 4.9	<u>29.7</u> ± 5.1	17.3 ± 4.3	31.3 ± 5.2
pi05-base	Soft-Mask (v3.0)	Hard	25.1 ± 4.9	18.0 ± 4.3	20.7 ± 4.6	36.7 ± 5.4
pi05-base	Soft-Mask (v3.0)	Mix	28.1 ± 5.1	17.0 ± 4.2	24.3 ± 4.8	**43.0** ± 5.6
pi05-base	Soft-Mask (v3.0)	Linear	20.4 ± 4.5	11.7 ± 3.6	16.7 ± 4.2	32.7 ± 5.3
pretrained $\pi_{0.5}$ w/ KT	Soft-Mask (v3.0)	–	25.8 ± 4.9	23.0 ± 4.7	28.3 ± 5.1	26.0 ± 4.9
pretrained $\pi_{0.5}$ w/ KT+Mix-Mask	Soft-Mask (v3.0)	–	28.9 ± 5.1	**45.3** ± 5.6	17.7 ± 4.3	23.7 ± 4.8

G Multi-embodiment Results

Figure 9 reports learning curves for multi-embodiment joint training on the Panda–SO101 mixture, evaluated separately on DROID (Panda) and SO101 in terms of average success rate (Avg SR). For our method, we use full morphology encoding: kinematic tokens (chunk G=1), Mix-Mask, and FiLM-based joint-attribute conditioning. Overall, our embodiment-aware policy improves performance over the baseline on both embodiments by 80k training steps.

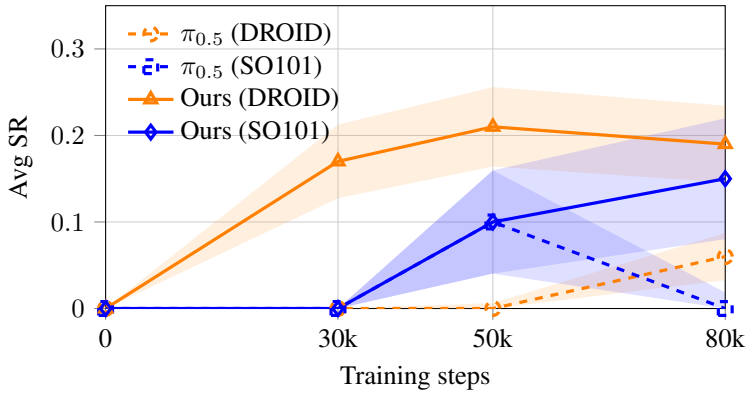


Figure 9: **Multi-embodiment learning curves on Panda–SO101**: Shaded regions indicate 95% CI (per-checkpoint).

H Detailed Real-World SO101 Evaluation

Figure 10 shows the three object configurations used in the real-world SO101 evaluation. Each configuration consists of 10 trials with three oranges per trial.

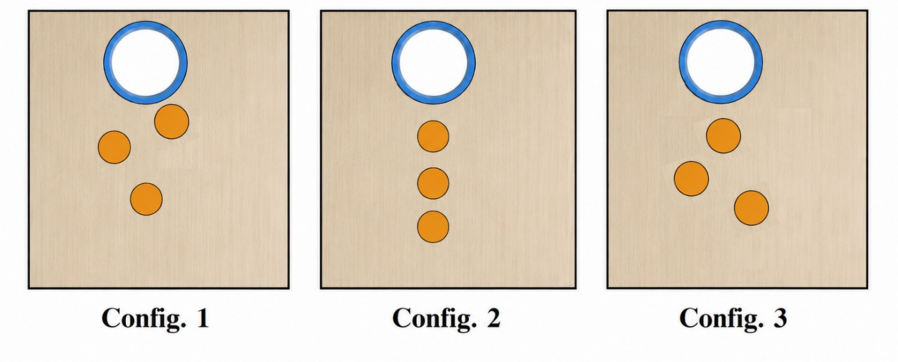


Figure 10: **Real-world SO101 object configurations**: Three initial orange-placement configurations used in the real-world evaluation.

Table 12: **Detailed real-world SO101 results**: We report the per-trial placement distribution over 10 trials per configuration. Each entry in the first three rows denotes the number of trials in which 1, 2, or 3 oranges were successfully picked and placed. The last row reports the total number of placed oranges.

Number of oranges	$\pi_{0.5}$ **baseline**			**Ours**		
placed per trial	**Config. 1**	**Config. 2**	**Config. 3**	**Config. 1**	**Config. 2**	**Config. 3**
1 orange	2	3	2	3	3	2
2 oranges	0	0	0	4	3	4
3 oranges	0	0	0	0	0	1
Total oranges	2	3	2	**11**	**9**	**13**