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@merl.com

Abstract

Transformer-based VLA policies have advanced rapidly as training data scales, yet cross-robot policy learning—training a single policy across multiple embodiments—remains challenging. Such policies are often embodiment-agnostic and must infer kinematics from observations, which can hurt robustness. We propose an embodiment-aware transformer that injects morphology via: (1) kinematic tokens with per-joint temporal chunking; (2) topology-aware attention bias to encourage message passing along kinematic edges; and (3) joint-attribute conditioning using per-joint descriptors. Across multiple embodiments, our method consistently outperforms the vanilla $\pi_{0.5}$ baseline.

1 Introduction

Transformer-based vision-language-action (VLA) policies Ahn et al. (2022); Brohan et al. (2022); Driess et al. (2023) have improved rapidly by scaling to large datasets, but *cross-robot policy learning*—a single policy that works well across diverse embodiments—remains challenging. In practice, strong performance often still requires per-robot adaptation, e.g., fine-tuning Intelligence et al. (2025); Kim et al. (2024) or modifying the action head Team et al. (2024); Doshi et al. (2024). A key reason is that many VLA policies are largely embodiment-agnostic, forcing them to learn kinematics and cross-joint coordination implicitly from observations. This motivates injecting robot morphology as an explicit inductive bias. Prior work represents morphology as a kinematic graph and incorporates it via GNNs Wang et al. (2018); Huang et al. (2020); Whitman et al. (2023) or topology-aware attention Veličković et al. (2017); Hong et al. (2021); Sferrazza et al. (2024); however, several limitations

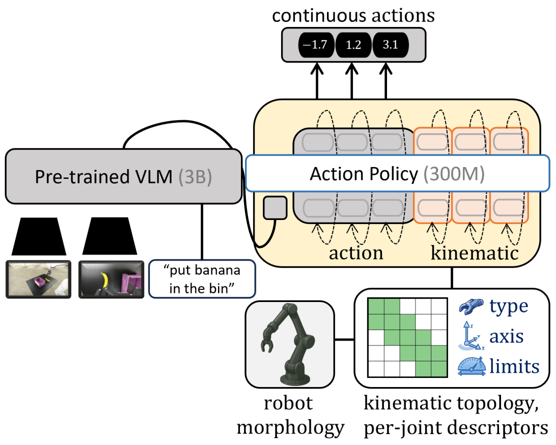
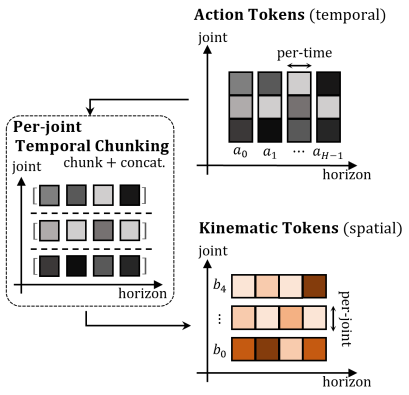
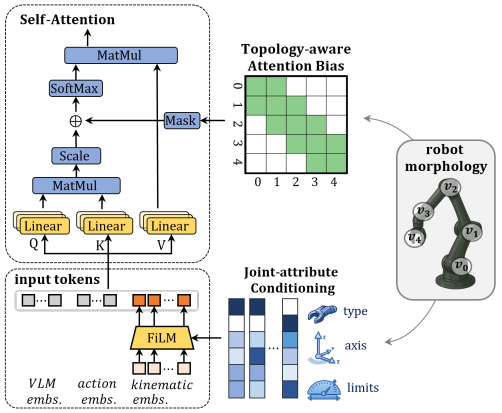


Figure 1: **Embedding robot morphology into a Transformer-based VLA policy**: We embed kinematic topology and per-joint semantics into the action policy. This design consistently improves cross-robot policy learning.



(a) **Kinematic Tokens (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 (1) kinematic tokens, (2) topology-aware attention bias, and (3) joint-attribute conditioning.

remain: (i) VLA models such as $\pi_{0.5}$ Intelligence et al. (2025) compress joint structure into a small set of action tokens, leaving no clear interface for morphology embeddings; (ii) strict locality in topology-aware attention can hinder long-range coordination; and (iii) connectivity alone misses per-joint semantics (e.g., type, axis, limits). **Contributions:** We propose an embodiment-aware transformer action policy with (1) *kinematic tokens* for joint-wise action representation, (2) a topology-aware attention bias with a local/global schedule, and (3) *joint-attribute conditioning* for joint semantics. Across single- and multi-embodiment evaluations on DROID (Franka Panda), Unitree G1 Dex1, and SO101, our morphology encoding improves success rates over the vanilla $\pi_{0.5}$ baseline Intelligence et al. (2025).

2 Proposed Method

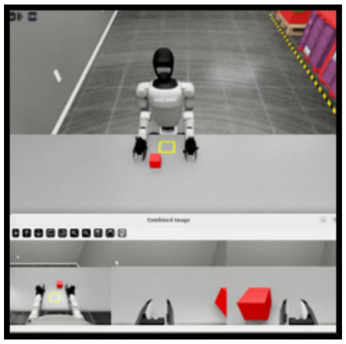
2.1 Kinematic Tokens (KT)

To factorize actions across joints, we introduce *kinematic tokens* that provide a compact per-joint view (Figure 2a). Let $a_{t,j}$ denote the scalar action for joint $j \in \{0, \dots, J-1\}$ at time $t \in \{0, \dots, H-1\}$. We split the horizon into G non-overlapping chunks of size $g = H/G$ and, for each joint j and chunk k, form $b_{j,T_k} := [a_{t,j}]_{t \in T_k} \in \mathbb{R}^g$ with $T_k = \{kg, \dots, (k+1)g - 1\}$. We refer to the vector b_{j,T_k} as the *kinematic token* for joint j and chunk k. We embed each token with a MLP, $z_{j,T_k} = \mathrm{Enc}_0(b_{j,T_k}) \in \mathbb{R}^d$, and append them to the token sequence as additional context. The policy predicts actions from the original action tokens, which can attend to kinematic tokens to leverage joint-wise structure.

2.2 Topology-aware attention

Robot embodiments admit a natural kinematic topology represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ over joints. We inject this topology only within the *joint-to-joint* attention block; all other attention patterns follow the original $\pi_{0.5}$ mask (Appendix 5). At layer ℓ, we add a topology-dependent bias $B^{(\ell)}_{i,j}$ to the attention logits: $\alpha^{(\ell)}_{i,j} = \mathrm{softmax}_j\left(\mathrm{logits}^{(\ell)}_{i,j} + B^{(\ell)}_{i,j}\right)$. We consider three variants. **Full-Mask** restricts attention to 1-hop neighbors in $\mathcal{G}$ (and self). **Mix-Mask** alternates such masked layers with full-attention layers to balance local message passing and

(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."

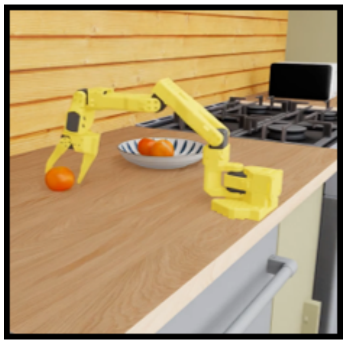

(c) **SO101:** "Grab orange and place into plate"

Figure 3: **Simulation environments for evaluation**.

Table 1: **Single-embodiment results on DROID**: Each component contributes to higher success rates, and the best performance is achieved by combining all (KT + Mix-Mask + FiLM). **Bold**: best SR in each column. <u>Underline</u>: second-best. 95% CI is shown with $\pm\Delta$.

Kinematic Token	Mask	FiLM	Success Rate (SR%) ↑ (95% CI)			
(Chunk G=1)			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	<u>67.7</u> ± 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	<u>27.7</u> ± 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
✓	–	✓	<u>37.7</u> ± 5.5	6.0 ± 2.7	65.0 ± 5.4	<u>42.0</u> ± 5.6
✓	Mix-Mask	✓	**47.4** ± 5.6	5.7 ± 2.7	**77.7** ± 4.7	**58.7** ± 5.5

global coordination. **Soft-Mask** keeps full attention but adds a learnable bias based on shortest-path distance on $\mathcal{G}$ (Graphormer-style Ying et al. (2021)). (Appendix B.1)

2.3 Joint-attribute conditioning

Topology-aware attention specifies *which* joints exchange information, but connectivity alone misses per-joint *semantics* (e.g., type, axis, limits). We therefore condition kinematic-token embeddings on per-joint descriptors derived from morphology (Figure 2b). For each joint j, we define a descriptor s_j (Appendix B.3) capturing joint properties, and use FiLM Perez et al. (2018) to predict scale and shift: $\gamma_j, \beta_j = \mathrm{FiLM}(s_j),\ \gamma_j, \beta_j \in \mathbb{R}^d$. Given the kinematic-token embedding $z_j \in \mathbb{R}^d$, we apply feature-wise affine modulation $z_j = (1 + \gamma_j) \odot z_j + \beta_j$. This augments topology-based message passing with per-joint semantics beyond connectivity.

3 Experiments

We evaluate morphology-aware transformer policies for *cross-robot policy learning*, where a single policy is trained and evaluated on the same set of robot embodiments. We compare model variants that progressively add kinematic tokens, topology-aware bias, and joint-attribute conditioning, and test whether the resulting gains hold under both single- and multi-embodiment training. We study single-embodiment imitation learning on **DROID (Franka Panda)** and **Unitree G1 Dex1 (Unitree G1)**, and multi-embodiment learning on a **mixture of Panda and SO101** (Panda: 8-DoF; SO101: 6-DoF). (Appendix C)

Table 2: **Single-embodiment results on Unitree G1 Dex1**: Our components remain effective in this setting. **Bold**: best SR. Underline: second-best.

Kinematic Token (Chunk G=1)	Mask	FiLM	Avg SR%↑ (95% CI)
–	–	–	24.7 ± 4.9
✓	–	–	24.3 ± 4.9
✓	Full	–	23.8 ± 4.8
✓	Mix	–	$\underline{27.3} \pm 5.0$
✓	–	✓	$\underline{25.3} \pm 4.9$
✓	Mix	✓	$\mathbf{28.0} \pm 5.0$

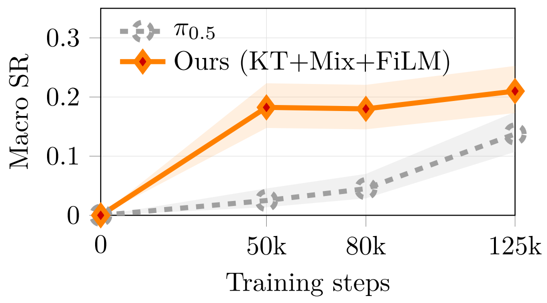


Figure 4: **Multi-embodiment learning curves on Panda–SO101**: Our model outperforms the vanilla $\pi_{0.5}$ throughout training.

Table 3: **Effect of temporal chunk size on DROID**:

Chunk (G)	Avg SR%↑
1	$\mathbf{36.0} \pm 5.4$
2	$\underline{35.8} \pm 5.4$
4	34.4 ± 5.3
8	30.5 ± 5.2
16	33.3 ± 5.3

Table 4: **Effect of auxiliary kinematic tokens (AKT) on DROID**:

Mask	AKT	Avg SR%↑
–	–	36.0 ± 5.4
–	✓	$\underline{37.0} \pm 5.4$
Full	–	$\underline{30.3} \pm 5.2$
Full	✓	33.0 ± 5.3
Mix	–	$\underline{37.0} \pm 5.4$
Mix	✓	$\mathbf{47.3} \pm 5.6$

Table 5: **Effect of bias initialization on DROID**:

Bias Init	Avg SR%↑
Zero	$\underline{26.1} \pm 4.9$
Hard	25.1 ± 4.9
Mix	$\mathbf{28.1} \pm 5.1$
Linear	20.4 ± 4.5

4 Results

Table 1 reports average success rate (Avg SR) on DROID. Overall, embedding morphology consistently improves performance, with the full model (KT + Mix-Mask + FiLM) achieving the best Avg SR (47.4% vs. 19.7% baseline). Component-wise comparisons show that kinematic tokens provide the largest gain, Mix-Mask yields a further improvement, and FiLM substantially boosts performance. Notably, the largest gains occur on Task 2 and Task 3 (77.7% vs. 15.7% and 58.7% vs. 25.0%). (Appendix D) Table 2 shows results on Unitree G1 Dex1; consistent with DROID, the full morphology encoding achieves the best Avg SR. Figure 4 shows learning curves for multi-embodiment training on the Panda/SO101 mixture. Our method remains higher throughout training. (Appendix E) We further analyze design choices in Tables 3–5. Table 3 shows that a single temporal chunk (G=1) performs best, and larger G generally reduces Avg SR. We also ablate token capacity by adding auxiliary kinematic tokens (AKT), i.e., extra learned per-joint tokens; Table 4 shows that AKT consistently improve Avg SR across configurations. Finally, Table 5 shows that Soft-Mask performance is sensitive to bias initialization: Mix initialization performs best among tested settings, but still underperforms hard-masked variants (Table 1), consistent with prior reports Sferrazza et al. (2024); Buterez et al. (2024). (Appendix F)

Conclusion

We presented an embodiment-aware transformer policy that injects robot morphology through (1) kinematic tokens with per-joint temporal chunking, (2) a topology-aware attention mask, and (3) joint-attribute conditioning. Across single- and multi-embodiment evaluations, our structured morphology encoding improves success rates over the vanilla $\pi_{0.5}$ VLA baseline, indicating improved robustness both within an embodiment and across embodiments.

Ethics Statement

Our work aims to enable robot policies that perform well across diverse robot embodiments by explicitly encoding robot morphology. This fits within a broader research area working towards more capable and general robotics foundation models. The long-term vision of this field is to realize generalist robot policies that can readily adapt to new tasks, environments, and embodiments, in a manner more analogous to the open-ended flexibility of human intelligence, in contrast to traditional methods for robotics control. We believe that our work contributes some modicum of progress towards this goal, which may eventually have profound societal implications in enabling automation to drastically and rapidly replace more human labor. We feel unqualified to fully assess the potential societal and economic impacts, but think that perhaps, depending on social factors and context, these may range across the spectrum of good to bad, such as aging societies that require more automation to maintain productivity versus the challenges of a labor force made redundant.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

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.

Lawrence D Brown, T Tony Cai, and Anirban DasGupta. Interval estimation for a binomial proportion. *Statistical science*, 16(2):101–133, 2001.

David Buterez, Jon Paul Janet, Dino Oglic, and Pietro Lio. Masked attention is all you need for graphs. *arXiv preprint arXiv:2402.10793*, 2024.

Ria Doshi, Homer Walke, Oier Mees, Sudeep Dasari, and Sergey Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. *arXiv preprint arXiv:2408.11812*, 2024.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Sunghoon Hong, Deunsol Yoon, and Kee-Eung Kim. Structure-aware transformer policy for inhomogeneous multi-task reinforcement learning. In *International Conference on Learning Representations*, 2021.

Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pp. 4455–4464. PMLR, 2020.

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, 2025. *URL https://arxiv.org/abs/2504.16054*, 1(2):3, 2025.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

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.

Yingbo Luo, Meibao Yao, and Xueming Xiao. Gcnt: Graph-based transformer policies for morphology-agnostic reinforcement learning. *arXiv preprint arXiv:2505.15211*, 2025.

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)*, pp. 6892–6903. IEEE, 2024.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Carmelo Sferrazza, Dun-Ming Huang, Fangchen Liu, Jongmin Lee, and Pieter Abbeel. Body transformer: Leveraging robot embodiment for policy learning. *arXiv preprint arXiv:2408.06316*, 2024.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pp. 1723–1736. PMLR, 2023.

Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *International conference on learning representations*, 2018.

Julian Whitman, Matthew Travers, and Howie Choset. Learning modular robot control policies. *IEEE Transactions on Robotics*, 39(5):4095–4113, 2023.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021.

A Related Work

Data-driven generalist robot policies 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 Ahn et al. (2022); Brohan et al. (2022); Driess et al. (2023). A common recipe is to pretrain on large-scale robot datasets Walke et al. (2023); O'Neill et al. (2024); Khazatsky et al. (2024), and then adapt to a target robot via fine-tuning or lightweight adaptation modules Team et al. (2024); Doshi et al. (2024). 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.

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 Wang et al. (2018); Huang et al. (2020); Whitman et al. (2023). A key design question is how information should propagate over the kinematic graph—balancing local interactions with global coordination. NerveNet Wang et al. (2018) applies repeated local (one-hop) message passing so that information can gradually propagate across the body. SMP Huang et al. (2020) 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 Veličković et al. (2017); Hong et al. (2021); Sferrazza et al. (2024). Concretely, these methods use the kinematic graph to modulate attention, either by *Hard-Mask* attention or *Soft-Mask* attention. Hard-masking approaches Sferrazza et al. (2024); Buterez et al. (2024) 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 Hong et al. (2021); Luo et al. (2025) 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 Sferrazza et al. (2024); Buterez et al. (2024) 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. Despite these advances, two limitations remain:

- **Limited applicability to VLA token interfaces:** In state-of-the-art VLA models such as $\pi_{0.5}$ Intelligence et al. (2025), action generation often compresses joint-space structure into a small set of action tokens, making it difficult to apply existing kinematic biasing schemes at the level of individual joints.
- **Topology alone lacks joint semantics:** Connectivity specifies how information propagates, but not what each joint represents. Encoding per-joint semantics is important for disambiguating functional roles and enabling more structured action generation (e.g., actuation type, axis, or joint limits).

B Detailed Proposed Method

B.1 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 5). Let $\text{logits}_{i,j}^{(\ell)}$ 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 $B_{i,j}^{(\ell)}$:

$$\alpha_{i,j}^{(\ell)} = \text{softmax}_j \left(\text{logits}_{i,j}^{(\ell)} + B_{i,j}^{(\ell)} \right), \tag{1}$$

where $\alpha_{i,j}^{(\ell)}$ is the attention weight.

We consider three topology-bias variants in two families: Hard-Mask (Full-Mask and Mix-Mask) and Soft-Mask. Hard-Mask uses an adjacency-based *hard* bias that blocks attention to non-neighbor joints. Soft-Mask uses a *shortest-path distance* (SPD)-based *soft* bias that favors nearby joints while retaining full attention.

Hard masking family: We first define the 1-hop neighborhood indicator

$$M_{i,j} = \begin{cases} 1, & (i = j) \text{ or } (i,j) \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Hard masking is implemented by setting

$$B_{i,j}^{(\ell)} = \begin{cases} 0, & M_{i,j} = 1, \\ -\infty, & M_{i,j} = 0, \end{cases} \tag{3}$$

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

Full-Mask: Full-Mask applies hard masking Eq. equation 3 at every layer, enforcing strictly local (1-hop) joint-to-joint interactions.

Mix-Mask: Mix-Mask alternates masked and unmasked layers to balance local message passing with periodic global coordination. Even-numbered layers apply hard masking (Eq. equation 3), while odd-numbered layers use full attention ($B_{i,j}^{(\ell)} = 0$ for all i, j).

Soft-Mask: We define the shortest-path distance on $\mathcal{G}$ as

$$d(i,j) = \min_{p: i \to j} |p|, \tag{4}$$

where the minimum is taken over all paths p from joint i to joint j and $|p|$ is the number of edges in the path (with $d(i,i) = 0$). Following a Graphormer-style design Ying et al. (2021), we parameterize the topology term by a learnable bias table indexed by distance:

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

Unlike hard masking, Soft-Mask biases attention by kinematic distance while retaining full attention paths.

B.2 Attention Mask

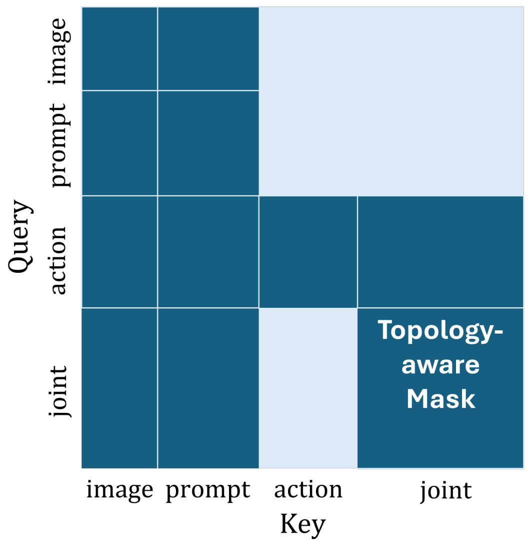


Figure 5: **Attention mask**: Attention mask used in our VLA action policy with kinematic tokens. Dark cells indicate unmasked attention and light cells indicate masked attention. Tokens are grouped by type for visualization (image/prompt/action/kinematic; each group may contain multiple tokens). We 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.3 Per-joint descriptors for joint-attribute conditioning

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 Detailed experimental setting

C.1 Single-embodiment evaluations

We first study single-embodiment imitation learning on two benchmarks to test whether explicit morphology encoding improves performance even outside cross-embodiment settings. **DROID (Franka Panda):** We use the public DROID dataset Khazatsky et al. (2024), which contains demonstrations collected on an 8-DoF (including the gripper) Franka Panda arm. We train on a 1/8 subset of DROID and evaluate in a simulation evaluation suite consisting of three language-conditioned pick-and-place tasks[1] (Figure 3a). **Unitree G1 Dex1 (Unitree G1):** We also evaluate on the Unitree Dex1 benchmark in simulation, which focuses on manipulation with the Unitree G1 platform while the lower body remains stationary. We train on the public Unitree Dex1 dataset with 16-DoF joint-space actions[2] and evaluate in an IsaacLab-based simulator[3] (Figure 3b).

C.2 Multi-embodiment evaluation

We evaluate multi-embodiment imitation learning by jointly training a single policy on a mixture of Panda and SO101 demonstrations, where the two robots differ in joint-space action dimensionality (Panda: 8-DoF; SO101: 6-DoF). This setting is challenging because the policy must reconcile embodiment-specific kinematics and coordination patterns under mismatched action spaces, and naive joint training can lead to interference between embodiments. For the Panda portion, we use the same 1/8 subset of DROID and evaluation suite as in the single-embodiment setting. For SO101, we use the LeRobot dataset[4] and evaluate in the corresponding LeIsaac simulator[5] (Figure 3c). During training, each mini-batch contains trajectories from a single embodiment, and we sample Panda and SO101 batches with an 8:2 ratio (i.e., 80% Panda and 20% SO101). We choose this ratio to keep the same amount of DROID (Panda) data as in our single-embodiment setting, while SO101 can be trained adequately with a smaller dataset in this regime.

C.3 Metrics

We evaluate policies using task *success rate* (SR), computed over 300 rollout trials per experimental condition. Success is determined by an axis-aligned bounding box (AABB) criterion: an episode is successful if the object remains inside the target region for a minimum dwell time and is approximately stationary. To quantify statistical uncertainty, we also report a 95% confidence interval (CI) for each SR using the Wilson score interval Brown et al. (2001) for a binomial proportion (n=300). In tables, we present results in the format SR $\pm$ Δ, where Δ denotes the CI half-width, i.e., $\Delta = (\mathrm{CI}_{\mathrm{upper}} - \mathrm{CI}_{\mathrm{lower}})/2$. Across trials, we vary the random seed to change initial conditions (e.g., object placement).

C.4 Model variants and Training Protocol

We evaluate variants built on the $\pi_{0.5}$ backbone by adding kinematic tokens, topology-aware attention, and joint-attribute conditioning. For topology-aware attention, we compare Full-Mask, Mix-Mask and Soft-Mask variants. Kinematic tokens are embedded with a linear–SwiGLU–linear MLP, and joint-attribute conditioning uses a linear FiLM generator. In addition, we ablate the temporal chunk size ($G \in \{1, 2, 4, 8, 16\}$) and the kinematic token capacity by introducing auxiliary kinematic tokens.

All variants are initialized from the $\pi_{0.5}$ checkpoint `pi05-base`[6]. For architectures that add morphology modules, we zero-initialize the final layer of each module so that it starts near an

[1] `https://github.com/arhanjain/sim-evals`

[2] `https://huggingface.co/datasets/unitreerobotics/G1_Dex1_PickPlaceRedBlock_Dataset_Sim`

[3] `https://github.com/unitreerobotics/unitree_sim_isaaclab`

[4] `https://huggingface.co/datasets/LightwheelAI/leisaac-pick-orange`

[5] `https://github.com/LightwheelAI/leisaac`

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

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. Appendix 7 summarizes the training configuration for each setting.

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	125k

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`
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`

Continued on next page

Status	**Part**	**Parameter name**
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`

C.5 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 6) and objects (Figure 7) appearing in the task instructions.

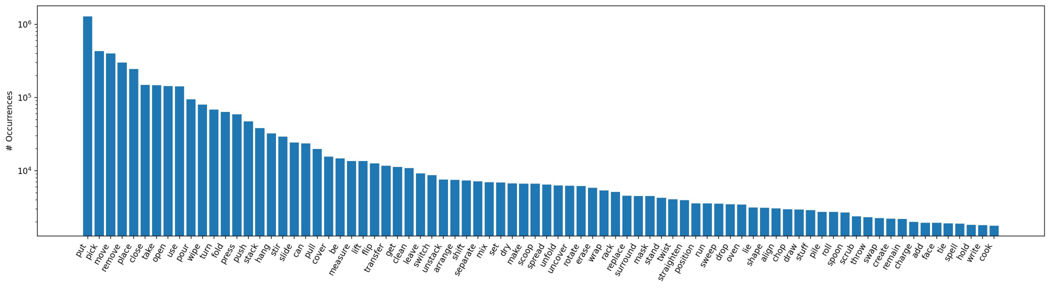


Figure 6: **Verb distribution in the DROID 1/8 subset**: Verb frequencies in task instructions.

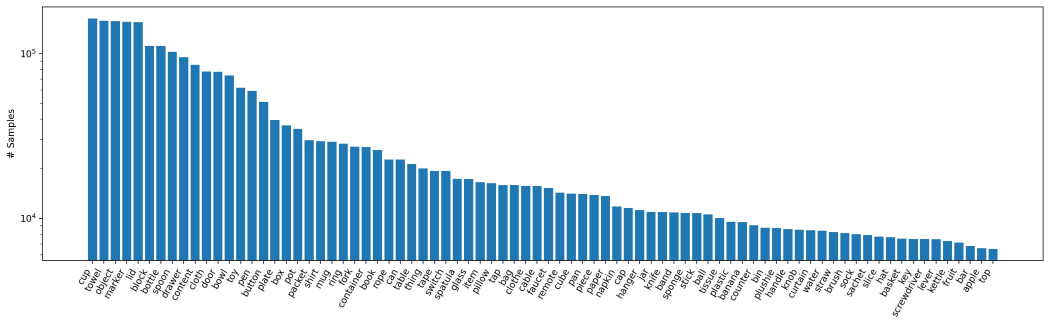


Figure 7: **Object distribution in the DROID 1/8 subset**: Object frequencies in task instructions.

D 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.

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

Table 9: **DROID simulation evaluation summary**: Best and second-best are indicated by **bold** and underline, respectively.

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

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

Table 10: **DROID simulation evaluation summary**: Best and second-best are indicated by **bold** and underline, respectively.

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

Effect of bias initialization: Since Soft-Mask attention is potentially more expressive yet optimization-sensitive, we test whether bias initialization improves its performance on DROID. All models use kinematic tokens (G=1) and no FiLM; only the initialization of the learnable bias table is changed. 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 (SR 28.1%). Nevertheless, across all initializations, the Soft-Mask variant does not surpass the Hard-Masked variants (e.g., Mix-Mask in Table 1). This is consistent with some related reports Sferrazza et al. (2024); Buterez et al. (2024). In Appendix G, we further investigate Soft-Mask with more detailed variants, but we do not observe consistent improvements.

E Multi-embodiment Results

Figure 8 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 on DROID throughout training, while achieving comparable performance on SO101.

DROID (Panda): Our method improves earlier and remains higher: at 50k steps it reaches Avg SR 0.210 ($\pi_{0.5}$: 0.000), and at 125k steps it remains higher at 0.213 ($\pi_{0.5}$: 0.100).

SO101: Performance is comparable across training: at 50k steps both reach Avg SR 0.100, and at 125k steps our method achieves 0.200 while $\pi_{0.5}$ achieves 0.250 (difference: 0.050).

We hypothesize that the larger gains on DROID are due to the training setup and the mixture ratio being skewed toward Panda, which can favor improvements on DROID relative to SO101. Under AP-FT, single-embodiment SO101 training with $\pi_{0.5}$ reaches only about Avg SR 0.2, suggesting that SO101 performance is generally more constrained in this fine-tuning regime.

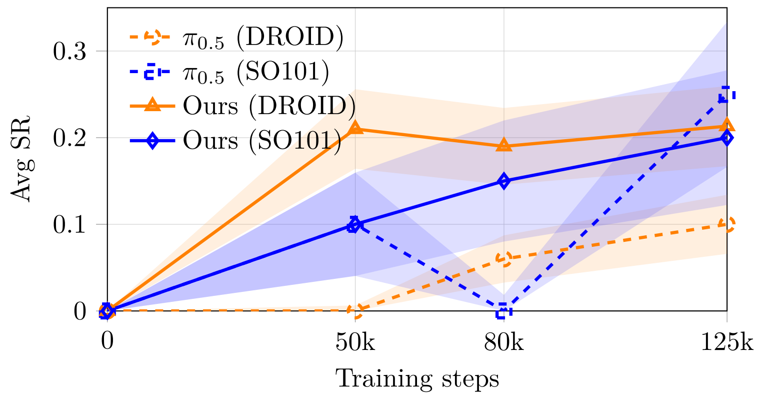


Figure 8: **Multi-embodiment learning curves on Panda–SO101**: Shaded regions indicate 95% CI (per-checkpoint).

F 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 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 of 36.0%, while performance generally degrades as G increases (33.3% at G=16).

Effect of 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

$$z_{j,T_k}^{(m)} = \mathrm{Enc}_m(b_{j,T_k}) \in \mathbb{R}^d, \quad m = 1, \dots, M, \tag{6}$$

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. 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 on DROID. All models use kinematic tokens (G=1) and no FiLM; only the initialization of the learnable bias table is changed. 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 (Avg SR 28.1%). Nevertheless, across all initializations, the Soft-Mask variant does not surpass the Hard-Masked variants (e.g., Mix-Mask in Table 1). This is consistent with some related reports Sferrazza et al. (2024); Buterez et al. (2024). In Appendix G, we further investigate Soft-Mask with more detailed variants, but we do not observe consistent improvements.

G 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

$$\alpha_{ij}^{(\ell)} = \mathrm{softmax}_j\left(\mathrm{logits}_{ij}^{(\ell)} + B_{ij}^{(\ell)}\right). \tag{7}$$

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

G.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

$$B_{ij}^{(\ell)} = (M_{ij} - 1)\, s_{ij}^{(\ell)}, \qquad s_{ij}^{(\ell)} \in \mathbb{R}, \tag{8}$$

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. equation 8 with an edge-wise, layer-wise strength

$$s_{ij}^{(\ell)} = \exp\left(\min\left(\theta_{ij}^{(\ell)}, \theta_{\max}\right)\right), \qquad \theta_{ij}^{(\ell)} \in \mathbb{R}. \tag{9}$$

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 $\theta^{(\ell)} \in \mathbb{R}$:

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

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:

$$s_{ij}^{(\ell)} = \theta^{(\ell)}, \qquad \theta^{(\ell)} \in \mathbb{R}. \tag{11}$$

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).

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.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:

$$B_{ij}^{(\ell)} = \theta_{d(i,j)}^{(\ell)}, \qquad \theta_d^{(\ell)} \in \mathbb{R},\ d \in \{0, 1, \dots, D_{\max}\}. \tag{12}$$

The resulting joint-to-joint attention follows Eq. equation 7.

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.

G.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.