Per-Link Attention for Graph-Conditioned Diffusion Models

Oliver Yangluo Chen [1]

Abstract

A standard recipe in graph foundation models (GFMs) encodes a structured input as a graph, runs a graph neural network (GNN) over it, and *pools* the resulting per-node embeddings into a single fixed-size vector before a downstream task head. We show that this graph-level pooling step can collapse node-discriminative information into a low-dimensional default, leaving the downstream head with a near-constant code regardless of graph structure. We expose this failure concretely in graph-conditioned control, where a GNN encodes a kinematic graph and the pooled embedding conditions a diffusion policy: a seven-intervention conditioning audit, applied to two independently trained pipelines, yields sign-unanimous evidence (5/5 seeds, $p \approx 0.031$ each) that under this task the pooled graph embedding ceases to function as a structured graph code, collapsing instead to an actuation-magnitude default. We propose **Per-Link Attention Diffusion (PLA-D)**, which routes the per-node GNN embeddings directly into a Transformer denoiser's cross-attention as a variable-length key/value sequence with learned node-position embeddings and a key-padding mask. PLA-D preserves the $|V| \cdot d_h$ node-level dimensions that pooling discards, strictly subsumes the pooled baseline, and trains to comparable validation loss in less than half the optimization budget. The diagnosis and the remedy are not domain-specific: they apply to a broad class of GFM pipelines that pool node embeddings before a downstream head whenever the task requires node-level discrimination.

[1]University of California, Berkeley. Correspondence to: Oliver Yangluo Chen <oliver_chen@berkeley.edu>.

The Workshop on Graph Foundation Models at the 43rd International Conference on Machine Learning (ICML 2026), Seoul, South Korea.

1. Introduction

A common recipe across graph foundation models (GFMs) is to encode a structured input as a graph, run a graph neural network (GNN) over it, and *pool* the resulting node embeddings into a single fixed-size graph-level vector for downstream use. This pattern—graph encoder followed by readout pooling—underpins GFM pipelines in molecular property prediction (Gilmer et al., 2017), physics simulation (Sanchez-Gonzalez et al., 2020), and graph-conditioned control (Wang et al., 2018; Gupta et al., 2022; Kurin et al., 2021; Bohlinger et al., 2024). We ask a question that cuts across all of them: *does the pooled graph vector actually carry the node-level structure that the downstream head assumes it does?*

We answer this in a concrete and unforgiving instantiation: graph-conditioned control, where a GNN encodes a kinematic graph (nodes are links; edges are joints) and the pooled embedding z_m conditions a diffusion policy (Chi et al., 2023) that must behave differently for structurally different graphs. This setting is a stringent probe of a pooled graph code precisely because the downstream task fails visibly when node-level structure is lost. The implicit assumption is that z_m encodes graph-discriminative information—which nodes exist, their attributes, and the connectivity. **We audit this assumption and find it false for two of the three end-to-end encoder families (the GNN- and flat-vector encoders; the Transformer-token encoder passes).**

Our audit is a seven-intervention protocol that replaces the pooled z_m at inference with frozen training-set means, zero vectors, scaled variants, and frozen-random-encoder controls. Applied to two independently trained pipelines—an IsaacLab diffusion behavior-cloning (BC) pipeline and a Walker2d-v5 PPO (proximal policy optimization) pipeline—the audit yields three findings, all sign-unanimous across 5 seeds:

1. On GNN-pooled and flat-vector encoders, *zeroing* the graph embedding improves downstream task performance by 37–44 percentage points; the trained-distribution z_m is the *worst-case* input.

2. The mechanism is energy-rate gating: real-z_m roll-

outs use $\sim$1.8 J/step; zero-z_m rollouts use $\sim$23 J/step ($\sim$13$\times$).

3. A frozen-random-encoder control isolates contrastive graph pre-training—not freezing—as the variable that locks the policy out of recovery.

These findings motivate our architectural proposal: **Per-Link Attention Diffusion (PLA-D)** (in the kinematic graph the nodes are links, hence *per-link*; we use *per-node* and *per-link* interchangeably), which *does not pool*. Instead, it routes the GNN's per-node embeddings directly into the policy's cross-attention as a length-$|V|$ key/value sequence with a learned per-node position embedding and a key-padding mask for variable graph size. PLA-D preserves $|V|{\cdot}d_h$ node-level dimensions, strictly subsumes the pooled baseline, and trains stably to comparable validation loss in less than half the epochs.

Relevance to graph foundation models. The pooling bottleneck we diagnose is not specific to robotics. A broad class of GFM pipelines that pool node embeddings before a downstream head risks the same failure mode: when the pooled summary collapses cross-instance cosine similarity to ≈ 0.85–1.0, the downstream model receives a near-constant input regardless of graph structure. PLA-D's per-node routing is a general-purpose alternative to graph-level readout that preserves node-level structure through to the task head.

2. Background and Related Work

Graph-level readout in GNNs. The standard GNN pipeline (Xu et al., 2019; Gilmer et al., 2017; Battaglia et al., 2018) produces per-node embeddings $\{h_v^{(L)}\}_{v\in V}$ and collapses them via a readout function (mean, sum, or attention-weighted pooling) into a single graph-level vector $z_G = \mathrm{Readout}(\{h_v^{(L)}\})$. This is the dominant paradigm for graph classification and regression, and is inherited by downstream conditioning in cross-embodiment policies. A substantial literature improves the readout itself: degree-scaled multi-aggregator readouts (PNA (Corso et al., 2020)), hierarchical soft clustering (DiffPool (Ying et al., 2018)), and learned attention-based pooling (Set Transformer (Lee et al., 2019), Graph Multiset Transformer (Baek et al., 2021)). These methods are orthogonal to our point: they all still emit a *fixed-size* graph-level summary that the downstream head must consume, so any node-level structure they fail to encode in that summary is unrecoverable downstream. PLA-D instead removes the readout from the conditioning path entirely, exposing the full per-node set to an attention-based head. We show that the readout step can be a critical information bottleneck when the downstream task requires node-level discrimination—independent of which pooling operator is used.

Cross-embodiment policy learning. Open X-Embodiment (Open X-Embodiment Collaboration, 2023), Octo (Octo Model Team, 2024), CrossFormer (Doshi et al., 2024), HPT (Wang et al., 2024), and One-Policy-To-Run-Them-All (Bohlinger et al., 2024) all condition on a body representation derived from per-link tokens. NerveNet (Wang et al., 2018) runs GNN message passing over kinematic trees; MetaMorph (Gupta et al., 2022) and Amorpheus (Kurin et al., 2021) extend per-link Transformer encoders. A common pattern is to pool per-link representations before the policy head. None audit whether the pooled channel carries structured morphology information at inference.

Diffusion policies. Diffusion Policy (Chi et al., 2023) introduced denoising diffusion for robot control; DiT (Peebles & Xie, 2023) introduced adaptive layer-normalization (AdaLN) modulation. We extend diffusion policies to cross-embodiment conditioning with per-node graph-aware cross-attention routing.

3. Method

3.1. Morphology graph encoding

Given a robot's URDF, we extract a kinematic graph $G = (V, E)$ where nodes are links and edges are non-fixed joints. Node features $f_v \in \mathbb{R}^{10}$ encode mass, inertia, geometry, and link type; edge features $f_e \in \mathbb{R}^{12}$ encode joint type, limits, damping, and axis. A 6-layer edge-conditioned Graph Isomorphism Network (Xu et al., 2019) with hidden dimension $d_h = 128$ produces per-node embeddings:

$$h_v^{(\ell+1)} = \mathrm{MLP}\Big((1+\epsilon^{(\ell)})h_v^{(\ell)} + \sum_{u\in\mathcal{N}(v)} \sigma\big(\mathrm{MLP}_e(f_{vu})\big)\odot h_u^{(\ell)}\Big), \tag{1}$$

where σ is sigmoid (multiplicative edge gating), with LayerNorm and residual connections per layer.

3.2. Pooled baseline (audited)

The standard recipe collapses per-node embeddings to a graph-level vector:

$$z_m^{\mathrm{pool}} = W_{\mathrm{out}}\,\mathrm{MeanPool}\big(\{h_v^{(L)}\}\big) \in \mathbb{R}^{512}. \tag{2}$$

A 6-layer Transformer denoiser (action horizon $H = 50$, $d_{\mathrm{model}} = 256$, 8 heads) uses z_m^{pool} as a *single token* in cross-attention plus AdaLN modulation (Peebles & Xie, 2023). Training is DDPM (Ho et al., 2020) with cosine schedule; inference is DDIM (Song et al., 2021) with 10 steps.

3.3. The pooling bottleneck

Mean-pooling collapses $|V| \cdot d_h$ per-node dimensions to d_h. For a 12-DoF (degree-of-freedom) quadruped ($|V| \approx$

17, $d_h = 128$), this is a 17× bandwidth reduction. The per-node dimensions that pooling discards—link-specific mass, inertia, joint range—are exactly the dimensions that distinguish one morphology from another at the node level. Our audit (Section 4) confirms the consequence: on two of three encoder families, the pooled z_m is indistinguishable across morphologies and the downstream policy ignores morphology identity.

3.4. Per-Link Attention Diffusion (PLA-D)

PLA-D replaces graph-level pooling with per-node cross-attention routing. Each node's GNN embedding is passed through the same output projection used by the pooled baseline, $g_v = W_{\text{out}}\, h_v^{(L)} \in \mathbb{R}^{512}$, then projected and augmented with a learned position embedding:

$$z_v^{\text{link}} = W_{\text{link}}\, g_v + \pi_v \in \mathbb{R}^{d_{\text{link}}}, \quad v \in V, \tag{3}$$

with $d_{\text{link}} = 256$ (so $W_{\text{link}} : 512 \to 256$), and where π_v is a learned per-slot embedding (disambiguating nodes even when their features are similar). Nodes occupy slots in a fixed canonical order—their index in the parsed URDF kinematic tree—so a given slot consistently denotes the same structural position. For variable graph size, the sequence is padded to $N_{\text{max}} = 48$ with a key-padding mask $K \in \{0,1\}^{N_{\text{max}}}$. Each denoiser layer's cross-attention uses per-node tokens as keys and values:

$$\begin{aligned}&\text{CA}(x; z^{\text{link}}, K) = \\ &\quad \text{MHA}(Q{=}x,\ K{=}z^{\text{link}},\ V{=}z^{\text{link}},\ \text{mask}{=}K).\end{aligned} \tag{4}$$

The pooled z_m^{pool} is retained for AdaLN modulation, so PLA-D *strictly subsumes* the pooled baseline: zeroing the per-node cross-attention path recovers pooled-only behavior exactly. The parameter overhead is ∼0.3M (+2%). Figure 1 contrasts the two architectures.

4. Experiments

4.1. Setup

Pipeline 1 (IsaacLab diffusion-BC). 120 robot morphologies from three families (quadruped, biped, hexapod) with parametric perturbations of link lengths (±30%), masses (±20%), joint limits (±20%), and damping (±30%). GPU-accelerated simulation in IsaacLab (Mittal et al., 2023). Train/test split: 70/50 morphologies. Per-morphology PPO experts (Schulman et al., 2017) collect 100 demonstrations each; diffusion policy trained for 300 epochs, 5 seeds.

Pipeline 2 (Walker2d-v5 PPO). 10 (or 50) procedurally perturbed Walker2d morphologies in Gymnasium (Towers et al., 2024) MuJoCo (Todorov et al., 2012). Distinct body, simulator, training algorithm (PPO, not diffusion-BC), and reward from Pipeline 1. Four encoder configurations (GNN, Flat, GNN-Pre, None) × 5 seeds.

Table 1. **Conditioning audit, Pipeline 1.** Biped survival rate (% of episodes avoiding the velocity-based termination; 5 seeds × 30 morphs × 5 episodes = 750 per cell). *real* = encoder's true z_m; *zero* = all-zero z_m. On GNN-pooled and flat-vector encoders, *zero* z_m *exceeds real* by 37–44 pp (5/5 seeds sign-unanimous).

Encoder	Survival (%) real	Survival (%) zero	Energy (J/step) real	Energy (J/step) zero
TT-E2E	**49.9**	39.9	28.4	27.0
GNN-E2E	0.0	**44.0**	1.8	**23.1**
Flat-E2E	0.0	**36.8**	2.0	**22.4**
GNN-Pre	0.0	0.1	3.4	1.6
Flat-Pre	0.0	**44.4**	1.9	**26.9**

Encoder families. We compare three graph/condition encoders: *TT* (per-node Transformer-token encoder), *GNN* (GIN with mean-pool readout), and *Flat* (a flat-vector MLP encoder). A suffix marks how the encoder is trained: *-E2E* (end-to-end with the policy) and *-Pre* (contrastively pre-trained, then used as a conditioning encoder); *None* omits the conditioning vector entirely.

Audit protocol. At inference, z_m is replaced with: (a) the encoder's true output (*real*); (b) the global training-set mean (*frozen_global*); (c) the within-family mean (*frozen_family*); (d) the all-zero vector (*zero*); (e) α-scaled variants $\alpha \cdot z_m^{\text{real}}$ for $\alpha \in \{0, 0.25, 0.5, 1, 2, 4\}$. Two controls: scaling z_s (the proprioceptive state-conditioning vector) instead of z_m (AdaLN-OOD negative control) and a frozen-random-encoder ablation.

4.2. Audit results: pooled graph embeddings can fail as a graph code

Table 1 shows that on GNN-E2E and Flat-E2E, removing the graph embedding entirely (zero z_m) *improves* survival by 37–44 pp. Per-seed differences are sign-unanimous: $\{+43, +30, +44, +56, +47\}$ pp for GNN-E2E (one-sided sign test $p \approx 0.031$). The pooled graph embedding regulates actuation magnitude (1.8 → 23 J/step, ∼13×), not morphology identity. The directionality is the key to the failure: at its trained value the pooled embedding *suppresses* actuation, driving the policy into a near-stationary, low-energy mode that trips the velocity-based termination (`vel_terminate`); zeroing it removes this suppression, restoring movement and survival. The pooled channel thus behaves as an energy gate rather than a morphology code.

α-scan dissociates encoders. Scaling $z_m \mapsto \alpha \cdot z_m^{\text{real}}$ for $\alpha \in [0, 4]$ reveals encoder-specific signatures (Table 2): TT-E2E peaks unimodally at $\alpha = 1$ (the trained value works); GNN-E2E and Flat-E2E exhibit U-shapes with $\alpha = 1$ as the *worst-case*. The graph-level pooled embedding is not merely uninformative—it is actively harmful on these encoders.

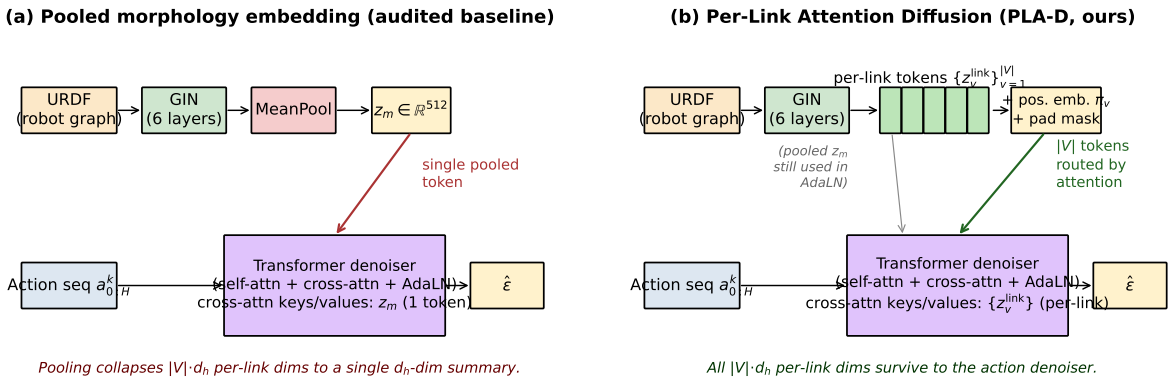

Figure 1. **Pooled graph readout vs. per-node routing (PLA-D).** (a) The audited baseline pools GNN node embeddings into a single graph-level token before the denoiser. (b) PLA-D routes per-node embeddings as a length-$|V|$ key/value sequence into cross-attention, preserving $|V| \cdot d_h$ node-level dimensions. The pooled z_m is retained for AdaLN compatibility.

Table 2. **α-scan.** Biped survival (%) on s42 checkpoints. GNN-E2E and Flat-E2E are U-shaped with the trained $\alpha = 1$ as the minimum; TT-E2E peaks at $\alpha = 1$.

Encoder	0	0.25	0.5	1.0	2.0	4.0
TT-E2E	42.7	52.7	62.7	**65.3**	56.7	36.0
GNN-E2E	45.3	29.3	2.0	**0.0**	28.7	47.3
Flat-E2E	15.3	6.7	0.0	**0.0**	15.3	38.7

External-pipeline validation. On Walker2d-v5 (Pipeline 2), all encoder configurations produce real $\gg$ zero: GNN-E2E 312.3 ± 6.5 vs. 207.3 ($\Delta = +105$), Flat-E2E 317.6 vs. 209.1 ($\Delta = +109$), GNN-Pre 304.7 vs. 165.8 ($\Delta = +139$), all 5/5 seeds sign-unanimous. The α-scan is unimodal on this pipeline (Figure 2). The audit thus returns qualitatively different and interpretable signatures across pipelines: the graph-level readout's failure mode depends on the downstream task structure, not the graph encoder alone.

4.3. PLA-D architectural validation

We train PLA-D on the Pipeline 1 demonstration set under matched hyperparameters (same DDPM schedule, AdamW lr=10^{-4}, batch 256). Figure 3 reports the validation loss trajectory.

PLA-D reaches val_loss $= 0.197$ at epoch 120 versus the pooled GNN-Pre baseline's 0.173 at epoch 265: less than half the optimization budget reaches the same loss neighborhood, with the trajectory still trending downward at termination.

5. Discussion: Implications for Graph Foundation Models

5.1. The pooling bottleneck is not robotics-specific

The failure mode we diagnose—graph-level readout collapsing cross-instance node-level discrimination—is a generic property of mean/sum pooling over GNN node embeddings. In our setting, pooling collapses cross-morphology cosine similarity to ≈ 0.85–1.0 (Figure 4), rendering the graph-level embedding nearly constant across structurally diverse inputs. A GFM pipeline that pools node embeddings before a downstream task head is susceptible to the same collapse when the downstream task requires node-level structural information; whether the collapse actually degrades performance is task-dependent (Section 5.3).

5.2. Per-node routing as a GFM design principle

PLA-D's per-node cross-attention is a general alternative to graph-level readout: rather than collapsing $|V|$ node embeddings to a single vector, it presents them as a variable-length token sequence to an attention-based downstream head. This is directly analogous to how vision Transformers present patch tokens (not a pooled CLS token) to cross-attention in vision-language models (Alayrac et al., 2022; Li et al., 2023). The key-padding mask handles variable graph size, and the learned per-slot position embedding provides canonical node ordering. The pattern generalizes to a broad class of GFM pipelines whose downstream head is attention-based:

- **Molecular property prediction:** atom-level tokens from a molecular GNN routed into a Transformer predictor, rather than pooled into a single molecular fingerprint.
- **Physics simulation:** per-particle GNN embeddings

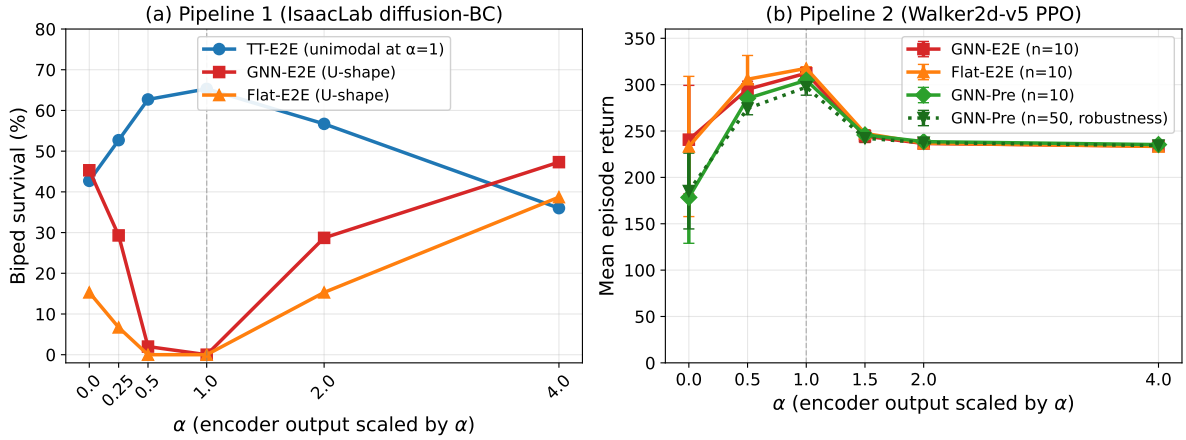

Figure 2. **α-scan across pipelines.** *Left:* Pipeline 1 (IsaacLab diffusion-BC). GNN-E2E and Flat-E2E are U-shaped with $\alpha = 1$ as the minimum. *Right:* Pipeline 2 (Walker2d-v5 PPO). All encoders peak unimodally at $\alpha = 1$. The same graph-level readout produces different failure modes under different downstream tasks.

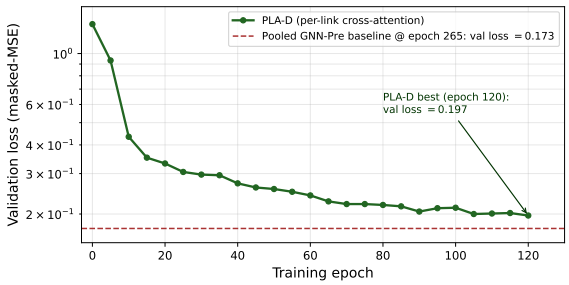

Figure 3. **PLA-D training trajectory.** Per-node cross-attention converges to val_loss $= 0.197$ at epoch 120 versus the pooled baseline's 0.173 at epoch 265. The per-node routing architecture reaches baseline-comparable loss in less than half the epochs.

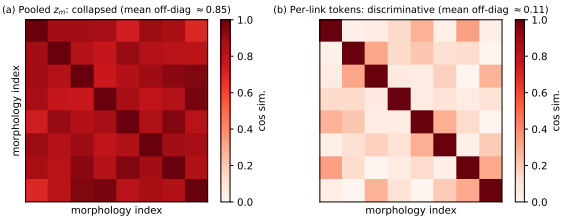

Figure 4. **Pooling collapse.** (a) Pooled z_m: mean off-diagonal cosine ≈ 0.85 across morphologies (near-constant embedding). (b) Per-node tokens: mean off-diagonal cosine ≈ 0.11 (morphology-discriminative). Graph-level readout destroys the node-level signal that the GNN learned to encode.

fed as cross-attention context to a dynamics model, preserving spatial structure.

- **Program analysis:** per-node AST embeddings routed into a code-generation Transformer.

In each case, the argument is the same: if the downstream task requires node-level discrimination, graph-level pooling is an unnecessary bottleneck.

5.3. When does pooling suffice?

Our audit shows that pooling is not universally broken. The Transformer-token (TT) encoder on Pipeline 1 passes the audit: z_m^{real} produces the best performance, and the α-scan peaks unimodally at the trained value. Pipeline 2 (Walker2d-v5 PPO) shows that all encoders produce functional graph-level embeddings under a different reward structure. The failure mode is *task-dependent*: pooling fails when the downstream task structure permits a degenerate strategy (here, stationarity under `vel_terminate`) that the pooled embedding's amplitude knob enables. The audit protocol we introduce is a practical tool for detecting whether a given GFM pipeline suffers from this failure mode.

6. Conclusion

We audit graph-level pooling as a conditioning channel and find, in a graph-conditioned control testbed, a sign-unanimous failure: the pooled GNN embedding collapses node-level structure and functions as an actuation gate rather than a structured graph code. Per-Link Attention Diffusion (PLA-D) fixes this by routing per-node GNN embeddings directly through cross-attention, preserving $|V| \cdot d_h$ node-level dimensions. The diagnosis and remedy apply broadly to graph foundation model pipelines where graph-level readout precedes a downstream attention-based head: *if the task needs node-level information, do not pool it away.*

An anonymized release of the codebase, the audit harness, the PLA-D reference implementation, and every per-seed raw audit JSON behind the paper's tables is hosted at `https://anonymous.4open.science/`

r/morpho-diffusion-B5EF/.

References

Alayrac, J.-B. et al. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022.

Baek, J., Kang, M., and Hwang, S. J. Accurate learning of graph representations with graph multiset pooling. In *ICLR*, 2021.

Battaglia, P. W. et al. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261*, 2018.

Bohlinger, N., Czechmanowski, G., Krupka, M., Kicki, P., Walas, K., Peters, J., and Tateo, D. One policy to run them all: An end-to-end learning approach to multi-embodiment locomotion. In *CoRL*, 2024.

Chi, C., Xu, Z., Feng, S., Cousineau, E., Du, Y., Burchfiel, B., Tedrake, R., and Song, S. Diffusion policy: Visuomotor policy learning via action diffusion. In *RSS*, 2023.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Veličković, P. Principal neighbourhood aggregation for graph nets. In *NeurIPS*, 2020.

Doshi, R., Walke, H., Mees, O., Dasari, S., and Levine, S. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. In *CoRL*, 2024.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *ICML*, 2017.

Gupta, A., Fan, L., Ganguli, S., and Fei-Fei, L. Metamorph: Learning universal controllers with transformers. In *ICLR*, 2022.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Kurin, V., Igl, M., Rocktäschel, T., Boehmer, W., and Whiteson, S. My body is a cage: the role of morphology in graph-based incompatible control. In *ICLR*, 2021.

Lee, J., Lee, Y., Kim, J., Kosiorek, A. R., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, 2019.

Li, J., Li, D., Savarese, S., and Hoi, S. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, 2023.

Mittal, M., Yu, C., Yu, Q., Liu, J., Rudin, N., Hoeller, D., Yuan, J. L., Singh, R., Guo, Y., Mazhar, H., et al. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 2023.

Octo Model Team. Octo: An open-source generalist robot policy. In *RSS*, 2024.

Open X-Embodiment Collaboration. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv:2310.08864*, 2023.

Peebles, W. and Xie, S. Scalable diffusion models with transformers. In *ICCV*, 2023.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. Stable-baselines3: Reliable reinforcement learning implementations. *JMLR*, 2021.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *ICML*, 2020.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. In *ICLR*, 2021.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *IROS*, 2012.

Towers, M. et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv:2407.17032*, 2024.

Wang, L., Chen, X., Zhao, J., and He, K. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. In *NeurIPS*, 2024.

Wang, T., Liao, R., Ba, J., and Fidler, S. Nervenet: Learning structured policy with graph neural networks. In *ICLR*, 2018.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *ICLR*, 2019.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*, 2018.

A. Hyperparameters

GIN morphology encoder. 6 layers, $d_h = 128$, output projection to $d_{\text{morph}} = 512$. Edge MLP: $12 \to 256 \to 128$ with sigmoid gate. Node MLP: $128 \to 256 \to 128$, LayerNorm, dropout 0.1. Per-layer learnable ϵ and residual.

Diffusion policy. Transformer denoiser: 6 layers, $d_{\text{model}} = 256$, 8 heads, dropout 0.1. DDPM cosine schedule, $T = 100$. DDIM 10-step inference. Action horizon $H = 50$, $D_{\text{max}} = 32$. AdamW, lr=10^{-4}, weight decay 10^{-5}, gradient clip norm 1.0. Cosine LR schedule, 1000-step warmup. Batch 256, 300 epochs (Pipeline 1).

PLA-D additions. $W_{\text{link}} : 512 \to 256$. Position embedding: $\text{Embedding}(48, 256)$. Parameter overhead: $\sim$0.3M ($+2\%$).

Walker2d-v5 PPO (Pipeline 2). SB3 PPO (Schulman et al., 2017; Raffin et al., 2021), MlpPolicy [512,256,128]. Forward velocity reward. lr=3×10^{-4}, n_steps 4096, batch 256, $\gamma = 0.99$, GAE $\lambda = 0.95$, clip 0.2, ent_coef 0.01. 3M env steps, 5 seeds. Contrastive pre-training (GNN-Pre): InfoNCE + 4 property-prediction heads, 2000 epochs, batch 16.

B. Audit Protocol Details

For Pipeline 1, each audit cell runs 5 seeds $\times$ 30 test morphologies $\times$ 5 episodes of 1000 steps (20s at 50Hz). Metric: survival rate $= 1 -$ `vel_terminate` trigger rate (trigger: $\|v_{xy}\| < 0.05$ m/s for 50 consecutive steps after 50-step grace window). Energy: total mechanical work $\sum_t \|\tau_t \cdot \dot{q}_t\| \Delta t$ / episode length.

Pipeline 2 uses 30 episodes per (seed, mode) cell. Metric: mean episode return under Walker2d-v5 forward-velocity-tracking reward.