

## A Details on RL environments

We adapt the IsaacGym humanoid environment for the three humanoid-related tasks, by modifying the observation space to include the vertical position of the torso, root coordinates and angular velocity, joint positions and velocities, and per-limb contact forces. We leave the reward for the Humanoid-Mod and Humanoid-Hill unchanged, while we adapt the reward for Humanoid-Bob by forcing the forward target velocity to zero, and appropriately adjusting the target and termination heights to take the balancing board into account. For the A1-Walk task, we adapt the codebase in Zhuang et al. [33] and train the policies using proprioception only for the actor, and additional simulation parameters for the critic. We define the task to maintain a target velocity of  $0.5 \text{ m/s}$  on an irregular terrain.

## B Real-World Deployment

We deployed the RL policy trained for A1-Walk task to a real-world Unitree A1 Robot. Computation was runned offboard on CPU and commands were sent via WiFi or Ethernet connection. We attach a supplementary video that demonstrates the real-world deployment. A frame overlay representing the robot motion are also shown in Figure 6.



Figure 6: **Real-World Deployment.** Frame overlay demonstrating the deployment of the BoT walking policy to a Unitree A1 quadruped robot.

## C Positional Encodings

For the reinforcement learning experiments presented in Section 5.2, we found that the use of positional encodings improves the performance of BoT architectures. Specifically, we compute positional encodings through an embedding layer that maps indices – up to  $n$  – to encoding vectors, which are then added to the tokenizers’ outputs. While this is beneficial for the reinforcement learning setting, we did not report a considerable improvement in the imitation learning setting, which we present without the use of positional encodings. In fact, these are not strictly necessary, as in the BoT architecture tokenizers do not share weights across body parts, and may in principle replace the role of positional encodings.

## D Additional Imitation Learning Ablations

	Return Normed		Length Normed	
	train	test	train	test
BoT-Hard (ours)	0.908 / <b>0.703</b>	<b>0.89</b> / <b>0.648</b>	<b>1.000</b> / <b>0.876</b>	<b>1.000</b> / <b>0.841</b>
BoT-Mix (ours)	<b>0.943</b> / 0.679	0.844 / 0.604	0.982 / 0.853	0.964 / 0.785
BoT-Soft	0.900 / 0.678	0.843 / 0.598	0.993 / 0.859	0.962 / 0.789
BoT-Hard/Random	0.850 / 0.661	0.835 / 0.600	0.995 / 0.845	0.962 / 0.782

Figure 7: **Additional Imitation Learning Ablations.** Statistics of the various architecture-criterion combinations are shown with two values, the leftside being the maximum mean value recorded throughout all evaluation epochs across three seeds, the rightside being the mean of all evaluation scores recorded in last 15 evaluation epochs across all three seeds.

In this section we provide several ablations in addition to those presented in Section 5.1. Specifically, we compare (i) BoT-Hard, (ii) BoT-Mix, (iii) BoT-Soft, which – similarly to [25] – learns a mapping between the graph shortest path matrix and the matrix  $B$  in (1), and (iv) BoT-Hard with a randomly sampled mask, i.e. having ones on its diagonal and the same sparsity as the mask  $M$  used for the correct implementation of BoT-Hard.

The table in Figure 7 shows the result of this comparison, with BoT-Hard outperforming all baselines on most of the metrics. The bottlenecks introduced by the masked attention result in better performance compared both to a mixed approach (BoT-Mix) and an approach that also accounts for structure but does not prevent long-range communication (BoT-Soft). As expected, simply sampling a random mask without properly accounting for the embodiment structure deteriorates performance.

## E Additional Reinforcement Learning Ablations

### E.1 Effect of Body-Induced Masking in BoT

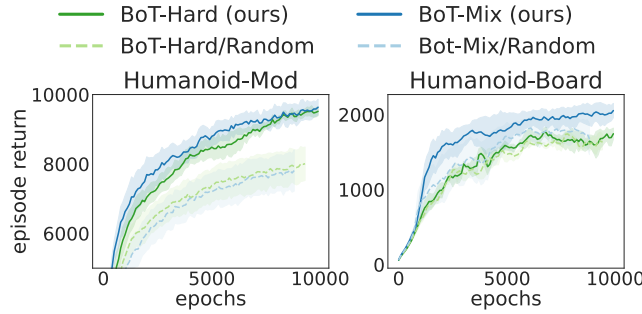


Figure 8: Additional RL Experimental Results on the Effect of Body-induced Masking.

BoT relies on masked attention with its mask determined by the embodiment structure. We conduct an additional experiment in the RL setting to further demonstrate the effect of the body-induced masking in this setting. We compare with BoT-Hard/Random and BoT-Mix/Random, where the attention mask  $M$  is given by a randomly sampled symmetric binary matrix with the same degree of sparsity ( $\beta \approx 0.82$  for the IsaacGym humanoid). The results are presented in Figure 8. Overall, BoT with *random* masking (dotted lines) underperforms BoT with *body-induced* masking (solid lines) in both a simpler task (Humanoid-Mod) and a hard-exploration task (Humanoid-Board), which highlights that the use of body-induced masking is crucial for the performance of BoT.

### E.2 Effect of Per-Limb Tokenizer vs. Shared Tokenizer

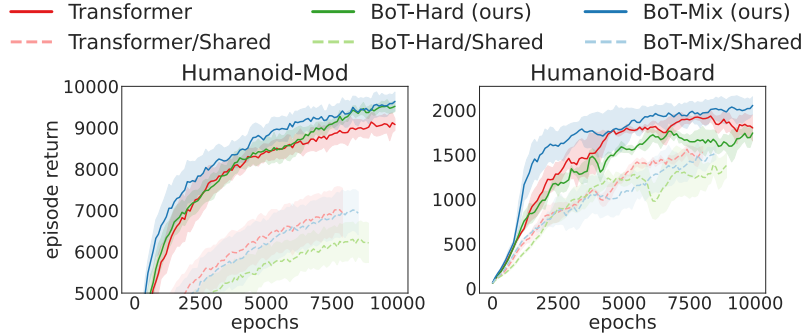


Figure 9: Additional RL Experimental Results on the Effect of Per-Node (De)Tokenizers.

The existing works using Transformer-based policies [23, 24, 25] for multi-task RL adopt *shared* linear projections for tokenizers and detokenizers to deal with the varying number of limbs, i.e.,

per-limb observation features are projected into embedding vectors by the single shared tokenizer network, and the per-limb hidden vectors are transformed to per-limb actions via the single shared detokenizer network. In contrast, our BoT is designed for tasks with a single morphology, thus we adopt *per-node* linear projections for tokenizer and detokenizer. We conduct an additional experiment to investigate the effect of this design choice, and the results are demonstrated in Figure 9.

In Figure 9, the solid lines denote the results of using per-node tokenizers/detokenizers, and the dotted lines present the results of using a shared tokenizer/detokenizer (which can be understood as representatives of the existing methods [23, 24, 25]). Overall, Transformer/BoT with per-node (de)tokenizers significantly outperform their shared (de)tokenizer counterparts in both a simpler task (Humanoid-Mod) and a hard-exploration task (Humanoid-Board). This shows that the use of tokenizers shared across different limbs for Transformer-based policies hinders efficient learning.

## F Training Details

Parameter	Values	
	MLP	Transformers
Batch Size	256	256
# Epochs	100	100
# Encoder Layers	3	16
Embedding Input Size	320	320
Feedforward Size	2500	1024
# Attention Heads	N/A	5
Learning Rate	1e-4	1e-4
# Parameters	16,696,656	17,544,120

(a) Training Parameters Used for Imitation Learning Experiments.

Parameter	Values	
	MLP	Transformers
Num Envs	2048	2048
Batch Size	8192	8192
# Encoder Layers	3	10
# Attention Heads	N/A	2
Embedding Input Size	N/A	64
Feedforward Size	150	128
# Parameters	699,467	688,762

(b) Training Parameters Used for Reinforcement Learning Experiments.

Figure 10: Training Parameters Used for Experiments in Section 5.

The training parameters of the experiments detailed in Section 5.1 and Section 5.2 are as summarized in Tables 10a and 10b.

## G FLOP Derivation for Custom Masked Attention Implementation

Below, we comparatively analyze an asymptotic bound for the amount of floating-point operations required in one scaled dot product (see Equation (1)) call between the vanilla and the masked approach. From hereon, let  $n$  denote the sequence length and  $d_k$  the input and output dimension of our attention mechanism.

**Computing  $\frac{QK^T}{\sqrt{d_k}}$ .** Considering  $Q \in \mathbb{R}^{n \times d_k}$  (and similarly for  $K$ ), the computation of  $QK^T$  will generally require  $d_k$  multiplications and  $d_k - 1$  additions for all of  $n^2$  pairs. Division by  $\sqrt{d_k}$  results in  $n^2$  divisions and one constant factor  $c_1$  of FLOPs for computing  $\sqrt{d_k}$ . The total amount of flops is  $2n^2d_k + c_1$ .

440 **Masked computation of  $\frac{QK^T}{\sqrt{d_k}}$ .** Exploiting sparsity, we ignore all inner product computations for  
 441 zero entries in  $M$ , computing only  $\beta n^2$  pairs of multiplications. This results in a reduction to  
 442  $2\beta n^2 d_k + c_1$  FLOPs.

443 **Computing Softmax( $S$ ).** A softmax for one vector of dimension  $n$  requires  $n$  exponentiations,  
 444  $n - 1$  additions, and  $n$  divisions, performed for  $n$  rows. Let exponentiations require  $c_2$  FLOPs per  
 445 element, then a total of  $(2 + c_2)n^2 - n$  FLOPs is performed.

446 **Masked computation of Softmax( $S$ ).** As a result of sparsity, there is instead a total of  $\beta n^2$   
 447 exponentiations  $\beta n^2$  divisions, and  $\beta n^2 - n$  additions to compute, reducing our demand to  $(2 +$   
 448  $c_2)\beta n^2 - n$  FLOPs.

449 **Computing the multiplication Softmax( $S$ ) $V$ .** A total of  $nd_k$  pairs are multiplied, where each  
 450 pair requires  $2n - 1$  operations to complete. The total amount of FLOPs is  $2n^2 d_k - nd_k$ . Following  
 451 a similar reasoning with previous writing, a total of  $2n^2 d_k - nd_k$  FLOPs are performed.

452 Assuming that our physical agent provides a graph-induced mask  $M \in \{0, 1\}^{n \times n}$  of sparsity  $\beta \in$   
 453  $[\frac{1}{n}, 1]$  (such that there are  $\beta n^2 > n$  nonzero entries), then the amount of FLOPs required by a vanilla  
 454 masked self-attention implementation is  $4n^2 d_k + (2 + c_2)n^2 - nd_k - n + c_1$ , while that of a custom  
 455 masked implementation is  $(2\beta + 2)n^2 d_k + (2 + c_2)\beta n^2 - nd_k - n + c_1$ . Therefore, the performance  
 456 gap between the vanilla and masked implementations is determined by the sparsity coefficient  $\beta$ , that  
 457 is, the number of FLOPs that a vanilla approach requires will be  $c(n)$  times the number of FLOPs a  
 458 custom masked approach requires:

$$\begin{aligned} & \lim_{n \rightarrow \infty} \frac{\# \text{ FLOPs in vanilla approach}}{\# \text{ FLOPs in masked implementation}} \\ &= \lim_{n \rightarrow \infty} \frac{4n^2 d_k + (2 + c_2)n^2 - nd_k - n + c_1}{(2\beta + 2)n^2 d_k + (2 + c_2)\beta n^2 - nd_k - n + c_1} \\ &= \frac{4d_{\text{model}} + 2 + c_2}{(2\beta + 2)d_{\text{model}} + 2\beta + \beta c_2} \geq 1 \end{aligned}$$

459 Therefore, even though these implementations share the same asymptotic bound  $\mathcal{O}(n^2 d_k)$ , the cus-  
 460 tom masked implementation's amount of FLOPs scales better than the vanilla implementation.

461 Note that it is possible to further optimize our implementation by sparsifying the multiplication  
 462 Softmax( $S$ ) $V$ ; this is left as a direction of future work, and requires the use of sparse array libraries,  
 463 which was not in the scope of this analysis.

## References

- [1] H. Forssberg. Stumbling corrective reaction: a phase-dependent compensatory reaction during locomotion. *Journal of neurophysiology*, 42(4):936–953, 1979.
- [2] J. L. Secomb, O. R. Farley, L. Lundgren, T. T. Tran, A. King, S. Nimphius, and J. M. Sheppard. Associations between the performance of scoring manoeuvres and lower-body strength and power in elite surfers. *International Journal of Sports Science & Coaching*, 10(5):911–918, 2015.
- [3] L. Seminara, S. Dosen, F. Mastrogiovanni, M. Bianchi, S. Watt, P. Beckerle, T. Nanayakkara, K. Drewing, A. Moscatelli, R. L. Klatzky, et al. A hierarchical sensorimotor control framework for human-in-the-loop robotic hands. *Science Robotics*, 8(78):eadd5434, 2023.
- [4] S. H. Collins and A. D. Kuo. Recycling energy to restore impaired ankle function during human walking. *PLoS one*, 5(2):e9307, 2010.
- [5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [7] L. Dong, S. Xu, and B. Xu. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5884–5888. IEEE, 2018.
- [8] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.
- [9] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.
- [10] Q. Zheng, A. Zhang, and A. Grover. Online decision transformer. In *international conference on machine learning*, pages 27042–27059. PMLR, 2022.
- [11] I. Radosavovic, B. Zhang, B. Shi, J. Rajasegaran, S. Kamat, T. Darrell, K. Sreenath, and J. Malik. Humanoid locomotion as next token prediction. *arXiv preprint arXiv:2402.19469*, 2024.
- [12] H. Qi, B. Yi, S. Suresh, M. Lambeta, Y. Ma, R. Calandra, and J. Malik. General in-hand object rotation with vision and touch. In *Conference on Robot Learning*, pages 2549–2564. PMLR, 2023.
- [13] C. Sferrazza, Y. Seo, H. Liu, Y. Lee, and P. Abbeel. The power of the senses: Generalizable manipulation from vision and touch through masked multimodal learning, 2023.
- [14] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24, 2020.
- [15] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.
- [16] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

- [17] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.
- [18] 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.
- [19] D. Buterez, J. P. Janet, D. Oglic, and P. Lio. Masked attention is all you need for graphs, 2024.
- [20] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
- [21] T. Wang, R. Liao, J. Ba, and S. Fidler. Nervenet: Learning structured policy with graph neural networks. In *International conference on learning representations*, 2018.
- [22] 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.
- [23] V. Kurin, M. Igl, T. Rocktäschel, W. Boehmer, and S. Whiteson. My body is a cage: the role of morphology in graph-based incompatible control, 2021.
- [24] A. Gupta, L. Fan, S. Ganguli, and L. Fei-Fei. Metamorph: Learning universal controllers with transformers. *arXiv preprint arXiv:2203.11931*, 2022.
- [25] 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.
- [26] U. Alon and E. Yahav. On the bottleneck of graph neural networks and its practical implications, 2021.
- [27] W. Park, W. Chang, D. Lee, J. Kim, and S. won Hwang. Grpe: Relative positional encoding for graph transformer, 2022.
- [28] J. Yeom, T. Kim, R. Chang, and K. Song. Structural and positional ensembled encoding for graph transformer. *Pattern Recognition Letters*, 2024.
- [29] N. Wagener, A. Kolobov, F. V. Frujeri, R. Loynd, C.-A. Cheng, and M. Hausknecht. Mocapact: A multi-task dataset for simulated humanoid control, 2023.
- [30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [31] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021.
- [32] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.
- [33] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao. Robot parkour learning. *arXiv preprint arXiv:2309.05665*, 2023.
- [34] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg. Daydreamer: World models for physical robot learning. In *Conference on Robot Learning*, pages 2226–2240. PMLR, 2023.

- 352 [35] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust per-  
353 ceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822,  
354 2022.
- 355 [36] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré. Flashattention: Fast and memory-efficient  
356 exact attention with io-awareness, 2022.
- 357 [37] C. Sferrazza, D.-M. Huang, X. Lin, Y. Lee, and P. Abbeel. Humanoidbench: Simu-  
358 lated humanoid benchmark for whole-body locomotion and manipulation. *arXiv Preprint*  
359 *arxiv:2403.10506*, 2024.