arXiv:2408.06316v1 [cs.RO] 12 Aug 2024

Body Transformer: Leveraging Robot Embodiment for Policy Learning

Carmelo Sferrazza **Dun-Ming Huang** **Fangchen Liu** **Jongmin Lee** **Pieter Abbeel**
UC Berkeley

Abstract: In recent years, the transformer architecture has become the de facto standard for machine learning algorithms applied to natural language processing and computer vision. Despite notable evidence of successful deployment of this architecture in the context of robot learning, we claim that vanilla transformers do not fully exploit the structure of the robot learning problem. Therefore, we propose Body Transformer (BoT), an architecture that leverages the robot embodiment by providing an inductive bias that guides the learning process. We represent the robot body as a graph of sensors and actuators, and rely on masked attention to pool information throughout the architecture. The resulting architecture outperforms the vanilla transformer, as well as the classical multilayer perceptron, in terms of task completion, scaling properties, and computational efficiency when representing either imitation or reinforcement learning policies. Additional material including the open-source code is available at https://sferrazza.cc/bot_site.

Keywords: Robot Learning, Graph Neural Networks, Imitation Learning, Reinforcement Learning

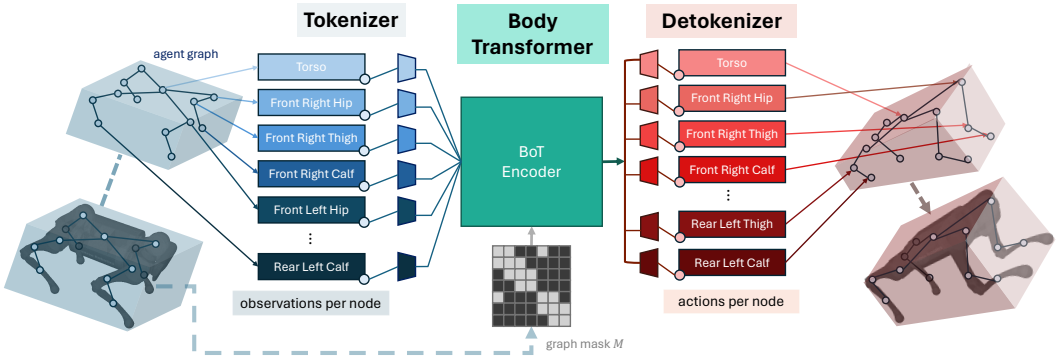


Figure 1: **Body Transformer** (BoT) is an architecture that considers physical agents as graphs of sensors and actuators as nodes, and edges reflecting the structure of the robot body. BoT leverages masked attention as a simple but flexible mechanism to provide a body-induced bias to the policy. The figure presents the overall schematic of our architecture, exemplified on a Unitree A1 robot.

1 Introduction

For most of their correcting and stabilizing actions, physical agents exhibit motor responses that are spatially correlated with the location of the external stimuli they perceive [1]. This is the case of a surfer, where the lower body, i.e. feet and ankles, is mostly responsible for counteracting the imbalance induced by the wave under the board [2]. In fact, humans present feedback loops at the level of the spinal cord's neural circuitry that are specifically responsible for the response of single actuators [3].

Corrective localized actuation is a main factor for efficient locomotion [4]. This is particularly important for robots too, where however, learning architectures do not typically exploit spatial interrelations between sensors and actuators. In fact, robot policies have mostly been exploiting the same architectures developed for natural language or computer vision, without effectively leveraging the structure of the robot body.

This work focuses on transformer policies, which show promise to effectively deal with long sequence dependencies and seamlessly absorb large amount of data. The transformer architecture [5] has been developed for unstructured natural language processing (NLP) tasks, e.g., language translations, where the input sequences often map to reshuffled output sequences. In contrast, we propose Body Transformer (BoT), an architecture that augments the attention mechanism of transformers by taking into account the spatial placement of sensors and actuators across the robot body.

BoT models the robot body as a graph with sensors and actuators at its nodes. Then, it applies a highly sparse mask at the attention layers, preventing each node from attending beyond its direct neighbors. Concatenating multiple BoT layers with the same structure leads to information being pooled throughout the graph, thus not compromising the representation power of the architecture.

Our contributions are listed below:

- We propose the BoT architecture, which augments the transformer architecture with a novel masking that leverages the morphology of the robot body.
- We incorporate this novel architecture in an imitation learning setting, showing how the inductive bias provided by BoT leads to better steady-state performance and generalization, as well as stronger scaling properties.
- We show how BoT improves online reinforcement learning (RL), outperforming MLP and vanilla transformer baselines.
- We analyze the computational advantages of BoT, by showing how reformulating the scaled dot product in the computation of the attention operation leads to near-200% runtime and floating point operations (FLOPs) reduction.

2 Related Work

Transformers in robotics. Originally developed for NLP applications [5], transformers have been successfully applied across domains, for example in computer vision [6] and audio processing [7]. Several works have shown applications of transformers as a means to represent robot policies [8, 9, 10], demonstrating its core advantages in this setting, i.e., variable context length, handling long sequences [11] and multiple modalities [12, 13]. However, these approaches use transformers as originally developed for unstructured or grid-like inputs, such as language or images, respectively. In this work, we leverage the robot embodiment by appropriately adapting the transformer attention mechanism.

Graph Neural Networks (GNNs). GNNs [14] are a class of learning architectures that can process inputs in the form of a graph [15]. While early versions of these architectures featured explicit message-passing schemes along the graph [16, 17], more recent architectures mostly feature attention-based approaches. In fact, the vanilla transformer, with its variable context length, inherently supports fully connected graphs. However, state-of-the-art performance on graph interpretation benchmarks is only achieved via modifications of the original transformer architecture, for example by means of learned graph encodings and attention biases [18]. A contemporaneous work, Buterez et al. [19], following a similar idea as in the work from Veličković et al. [20], utilizes masked attention layers, where each node only attends to its neighbors, and interleaves such layers with unmasked attention layers. In this work, we exploit masked attention in a policy learning setting, by additionally proposing an architecture that only comprises layers where each can attend to itself and its direct neighbors, resulting in naturally growing context over layers, i.e., the outputs of the first layers are computed using more local information compared to those of the last layers.

Exploiting body structure for policy learning. Graph neural networks have been explored by several works as a way to obtain multi-task RL policies that are effective across different robot morphologies. Earlier works focused on message passing algorithms [21, 22], and were later outperformed by vanilla transformers [23, 24] and transformer-based GNNs that make use of learned encoding and attention biases [25]. All these approaches were only demonstrated in simulated benchmarking scenarios and not applied to a real-world robotics setting. Compared to previous work, we additionally show that introducing bottlenecks in the attention mask fully exploits the embodiment structure and benefits policy learning also for tasks achieved by a single agent, leading to better performance and more favorable scaling.

3 Background

3.1 Attention Mechanisms in Transformers

Transformer, a foundational architecture in modern machine learning applications as well as in our work, is powered by the self-attention mechanism [5]. Self-attention weighs the values corresponding to each element of the sequence with a score that is computed from pairs of keys and queries extracted from the same sequence. Thus, it is able to identify relevant pairs of sequence elements in the model output.

Concretely, the self-attention output vector is computed through the following matrix operation:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

where Q, K, V (respectively query, key, and value matrices) are learnable linear projections of the sequence elements' embedding vectors, and d_k is the dimension of the embedding space. As embedding pairs with higher correspondence will have a higher score (from the computation of QK^T), the corresponding value vector of an embedding's associated key will receive a higher weight in the attention mechanism output.

3.2 Transformer-based GNNs

In this work, we model the agent embodiment as a graph whose nodes are sensors and actuators, and their connecting edges reflect the body morphology. While message-passing GNNs are suitable inductive biases for this formulation, they tend to suffer from oversmoothing and oversquashing of representations, preventing effective long-range interactions and discouraging network depth [26].

More recently, self-attention was proposed as an alternative to message-passing [18, 23]. While the standard self-attention mechanism amounts to modeling a fully connected graph, a popular transformer-based GNN, Graphormer [18], injects node-edges information through graph-based positional encodings [27, 28, 18, 25] and by biasing the scaled dot-product, i.e.,

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + B\right)V, \tag{1}$$

where B is a learnable matrix that depends on graph features, e.g., shortest path or adjacency matrix, and can effectively encode the graph structure in a flexible manner.

3.3 Masked Attention

The attention mechanism can be altered [5] with a binary mask $M \in \{0, 1\}^{n \times n}$ (where n is the sequence length), which is equivalent to replacing the elements of B in (1):

$$B_{i,j} = \begin{cases} 0 & M_{i,j} = 1 \\ -\infty & M_{i,j} = 0 \end{cases},$$

where i and j denote row and column indices. This operation effectively results in zeroing out the contribution of the pairs indicated with zeros in the mask M to the computation of the attention.

4 Body Transformer

Robot learning policies that employ the vanilla transformer architecture as a backbone typically neglect the useful information provided by the embodiment structure. In contrast, here we leverage this structure to provide a stronger inductive bias to transformers, while retaining the representation power of the original architecture. We propose Body Transformer (BoT), which is based on masked attention, where at each layer in the resulting architecture, a node can only attend to information from itself and its direct neighbors. As a result, information flows according to the graph structure, with the upstream layers reasoning according to local information and the downstream layers pooling more global information from the farther nodes.

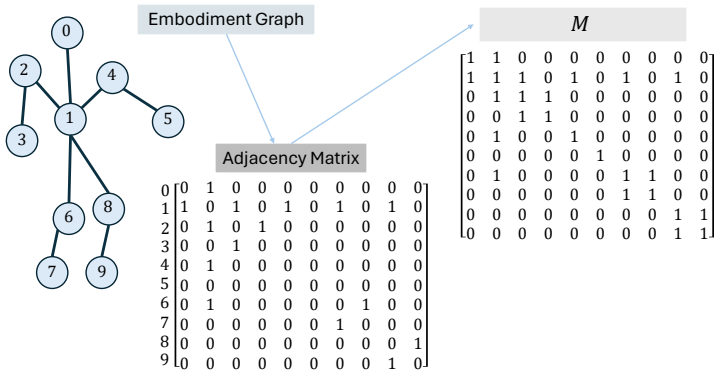


Figure 2: **Formulation of Embodiment Mask.** The mask M is constructed by adding a diagonal of 1s to the embodiment graph's adjacency matrices. Here, we visualize a simple example of a mask M for an arbitrary agent's embodiment where $n = 10$.

We present below the various components of the BoT architecture (see also Figure 1): (1) a tokenizer that projects the sensory inputs into the corresponding node embedding, (2) a transformer encoder that processes the input embeddings and generates output features of the same dimension, and (3) a detokenizer that decodes the features to actions (or values, for RL critic's training).

Tokenizer. We map the observation vector to a graph of local observations. In practice, we assign global quantities to the root element of the body, and local quantities to the nodes representing the corresponding limbs, similarly to previous GNN approaches [22, 23, 24, 25]. Then, a linear layer projects the local state vector into an embedding vector. Each node's state is fed into its *node-specific* learnable linear projection, resulting in a sequence of n embeddings, where n represents the number of nodes (or sequence length). This is in contrast to the existing works [23, 24, 25] that use a single *shared* learnable linear projection to deal with varying number of nodes in multi-task RL.

BoT Encoder. We use a standard transformer encoder [5] with several layers as a backbone, and present two variants of our architecture:

- *BoT-Hard* masks every layer with a binary mask M that reflects the structure of the graph. Specifically, we construct the mask as $M = I_n + A$, where I_n is the identity matrix of dimension n, and A is the adjacency matrix corresponding to the graph (see Figure 2 for an example). Concretely, this allows each node to attend to itself and its direct neighbors, and introduces considerable sparsity in the problem, which is particularly appealing from a computational perspective as highlighted in Section 5.4.
- *BoT-Mix* interleaves layers with masked attention (constructed as in BoT-Hard) with layers with unmasked attention. This is similar to the concurrent work in Buterez et al. [19], with the distinctions that, I) we find it more effective in our experimental setting to have a masked attention layer as the first layer, II) our mask M is not equivalent to the adjacency matrix, allowing a node to additionally attend to itself at every layer of the architecture.

Detokenizer. The output features from the transformer encoder are fed into linear layers that project them to the actions associated with the node's limb, which are assigned based on the proximity of the corresponding actuator with the limb. Once again, these learnable linear projection layers are separate for each of the nodes. When BoT is employed as a critic architecture in the RL setting, as in the experiments presented in Section 5.2, the detokenizers output values rather than actions, which are then averaged across body parts.

5 Experiments

We assess the performance of BoT across imitation learning and reinforcement learning settings. We keep the same structure as in Figure 1 and only replace the BoT encoder with the various baseline architectures to single out the effect of the encoder. Particularly, across the various experiments listed in this section, we present the following baselines and variations: (i) an MLP that stacks all embedding vectors as its input, (ii) a vanilla unmasked transformer encoder, (iii) BoT-Hard that only uses masked self-attention layers, (iv) BoT-Mix that alternates between masked and unmasked self-attention layers. All comparisons are made across models with a similar number of trainable parameters.

With the following experiments, we aim to answer the following questions:

- Does masked attention benefit imitation learning in terms of performance and generalization?
- Does BoT exhibit a positive scaling trend compared to a vanilla transformer architecture?
- Is BoT compatible with the RL framework, and what are sensible design choices to maximize performance?
- Can BoT policies be applied to a real-world robotics task?
- What are the computational advantages of masked attention?

5.1 Imitation Learning Experiments

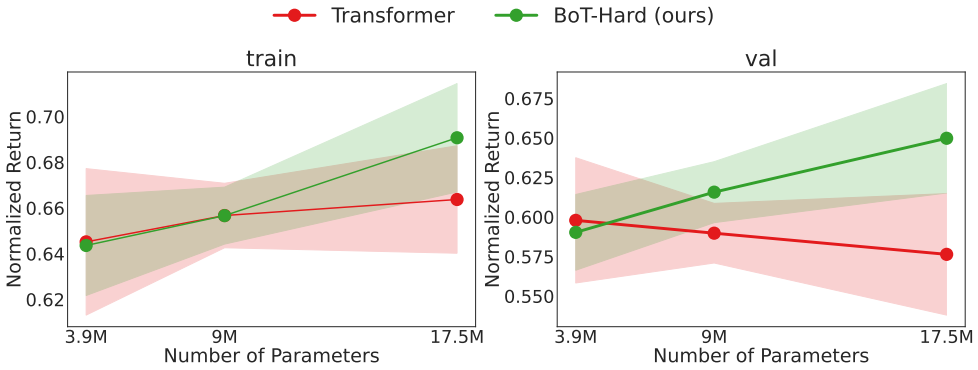
We evaluate the imitation learning performance of the BoT architecture in a body-tracking task defined through the MoCapAct dataset [29], which comprises action-labeled humanoid state trajectories with over 5M transitions, spanning a total of 835 tracking clips. For each architecture, we train a deterministic behavioral cloning (BC) policy. We evaluate mean returns normalized by the length of a clip, in addition to the normalized length of an episode, which terminates when the tracking error goes beyond a threshold. We run the evaluations both on the training and the (unseen) validation clips.

We report results in the table shown in Figure 3a, where BoT consistently outperforms the MLP and transformer baselines. Remarkably, the gap with these architectures further increases on the unseen validation clips, demonstrating the generalization capabilities provided by the embodiment-aware inductive bias. We also report the performance on the training set obtained by a tailored multi-clip policy presented in Wagener et al. [29] with the MoCapAct dataset. While the multi-clip policy is competitive with the vanilla transformer baseline, it is strongly outperformed by our architecture. This is a particularly remarkable result, as the comparison presents conditions more favorable to the baseline, which features a more flexible stochastic policy, was optimized in a recurrent fashion tailored to the tracking task, and was trained on a larger set of rollouts.

As shown in Figure 3b, we also find that BoT-Hard exhibits strong scaling capabilities, as its performance keeps improving with the number of trainable parameters compared to the transformer baseline, both on the training and validation clips. This further indicates a tendency for BoT-Hard to not overfit to the training data, which is induced by the embodiment bias. Additional comparisons are reported in Appendix E, including experiments on a dexterous manipulation benchmark, see Figure 4.

	normalized episode return		normalized episode length	
	train	val	train	val
MLP	0.623 / 0.572 ± 0.022	0.568 / 0.534 ± 0.025	0.808 / 0.762 ± 0.018	0.777 / 0.741 ± 0.022
Transformer	0.713 / 0.664 ± 0.024	0.656 / 0.576 ± 0.038	0.875 / 0.836 ± 0.022	0.834 / 0.779 ± 0.026
BoT-Hard (ours)	**0.751** / **0.691** ± 0.024	**0.698** / **0.650** ± 0.035	**0.908** / **0.865** ± 0.018	**0.879** / **0.835** ± 0.025
Multi-Clip [29]	- / 0.654	- / -	- / 0.855	- / -

(a) **Training and Validation Performance Across Architectures.** Statistics of the various architecture-criterion combinations are shown with two values, the leftside being the maximum value recorded during training, and the rightside being the mean evaluation scores with standard deviation. Results are averaged across five seeds.



(b) **Training and Validation Performance on MoCapAct Across Number of Trainable Parameters.** Each datapoint represents performance averaged across five seeds. We use the 17.5M models for the other imitation learning experiments in this paper.

Figure 3: **BoT Performance on Imitation Learning.**

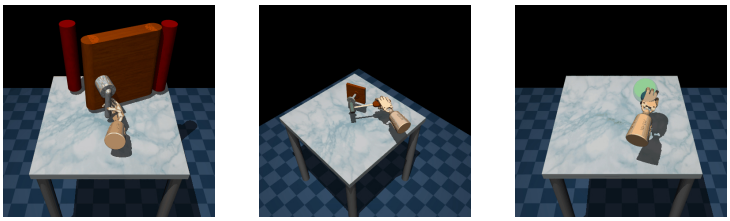


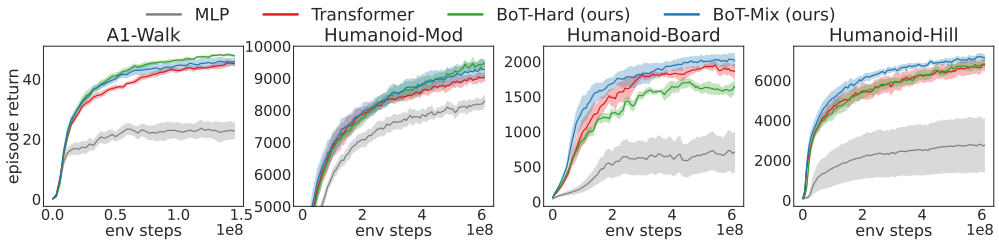
Figure 4: **Adroit Hand Door, Hammer, and Relocate Tasks (See Results in the Appendix).**

5.2 Reinforcement Learning Experiments

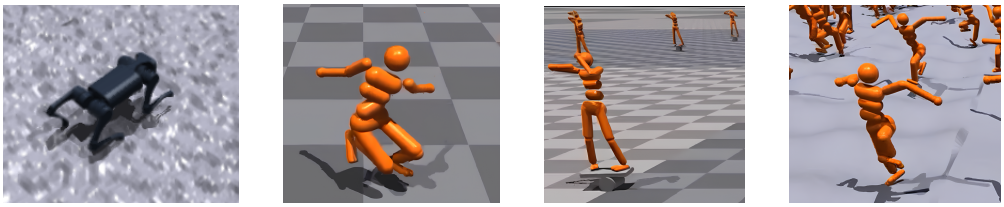
We evaluate the RL performance of BoT and baselines using PPO [30] on 4 robotic control tasks in Isaac Gym [31]: Humanoid-Mod, Humanoid-Board, Humanoid-Hill, and A1-Walk.

All humanoid environments build on top of the classical Humanoid environment in Isaac Gym, where we modify the observation space to increase the number of distributed sensory information (see details in Appendix A) and include contact forces at all limbs. Humanoid-Mod features the classical running task on flat ground, while in Humanoid-Hill we replaced the flat ground with an irregular hilly terrain. Humanoid-Board is a newly designed task, where the task is for the humanoid to keep balancing on a board placed on top of a cylinder. Finally, we adapt the A1-Walk environment, which is part of the Legged Gym repository [32], where the task is for a Unitree A1 quadruped robot to follow a fixed velocity command.

Figure 5 presents the average episode return of evaluation rollouts during training for MLP, Transformer, and BoT (Hard and Mix). The solid curve corresponds to the mean, and the shaded area to the standard error over five seeds. The result shows that BoT-Mix consistently outperforms both the MLP and vanilla transformer baselines in terms of sample efficiency and asymptotic performance, highlighting the efficacy of integrating body-induced biases into the policy network architecture.



(a) Performance on various tasks across evaluated architectures.



(b) Snapshots of successful rollouts of BoT policies.

Figure 5: **Reinforcement Learning Performance on Robotic Control Tasks.**

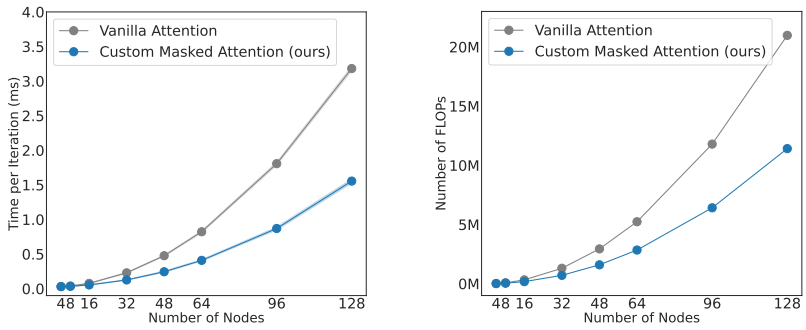
Meanwhile, BoT-Hard performs better than the vanilla transformer on simpler tasks (A1-Walk and Humanoid-Mod), but shows relatively inferior results in hard-exploration tasks (Humanoid-Board and Humanoid-Hill). Given that the masked attention bottlenecks information propagation from distant body parts, BoT-Hard's strong constraints on information communication may hinder efficient RL exploration: In Humanoid-Board and Humanoid-Hill, it may be useful for information about sudden changes in ground conditions to be transmitted from the toes to the fingertips in the upstream layers. For such tasks, BoT-Mix strikes a good balance between funneling information through the embodiment graph and enabling global pooling at intermediate layers to ensure efficient exploration. In contrast, in A1-Walk or Humanoid-Mod, the environment's state changes more regularly, thus the strong body-induced bias can effectively reduce the search space, enabling faster learning with BoT-Hard.

5.3 Real World Experiments

The Isaac Gym simulated locomotion environments are widely popular for sim-to-real transfer of RL policies without requiring adaptation in the real-world [32]. To verify that our architecture is suitable for real-world applications, e.g., running in real time, we deploy one of the BoT policies trained above to a real-world Unitree A1 robot, adapting the codebases in Zhuang et al. [33] and Wu et al. [34]. This is showcased in the supplementary video, demonstrating feasibility of our architecture for real-world deployment. We note that for simplicity we did not make use of teacher-student training or memory mechanisms [35] as common in the locomotion literature, which are known to further improve the transfer by resulting in more natural gaits.

5.4 Computational Analysis

Connections between body parts of a physical agent are often sparse, and so are the pre-computable masks M for embodiment graphs in BoT. Masked attention mechanisms can benefit from this sparsity, as their computational cost can be reduced by ignoring unnecessary computations of those matrix elements that will eventually be masked out. Large-purpose deep learning libraries such as `PyTorch` feature largely optimized matrix multiplication and attention routines (e.g., FlashAttention [36]), but do not leverage possible sparsity in the masked attention mechanism, ascribed by Buterez et al. [19] to missing use cases so far. For a fair computational comparison, we re-implement the scaled dot product in Equation (1) using CPU-based `NumPy` and evaluate on a



(a) Runtime (b) FLOPs

Figure 6: **Computational Analysis of the Custom Masked Attention Implementation.** Across 10,000 randomly sampled masks, we found that our custom implementation provides a 200% speed-up in runtime at sequence lengths up to 128 nodes and scales better in number of FLOPs.

single sample and attention head, being their batched and multi-head versions further parallelizable on GPUs.

Our custom implementation comprises two major changes in the computation of the masked attention mechanism from its vanilla implementation in `PyTorch`. First of all, we only perform the dot product computation for elements that will not be masked, resulting in fewer matrix multiplication-induced FLOPs from the computation of $\frac{1}{\sqrt{d_k}}QK^T$. Secondly, we only use unmasked values to compute the softmax in Equation (1), also resulting in reduced FLOPs.

We measure the average runtime of each implementation of the attention mechanism across 10,000 set of randomly generated Q, K, V, and M. For each randomization, the generated masks M have a diagonal of 1s and sparsity equal to that used in the MoCapAct experiments ($\beta = 0.908$) [1].

In Figure 6, we present scaling results of our implementation against vanilla attention across sequence lengths (number of nodes). We observe potential speed-ups of up to 206% for 128 nodes (e.g., in the order of humanoids with dexterous hands [37]). Overall, this shows that the body-induced bias in our BoT architecture not only enhances the overall performance of physical agents but also benefits from the naturally sparse masks that the architecture imposes. With adequate parallelization, this approach may significantly reduce the training time of learning algorithms as shown above.

Further details and derivations about these experiments can be found in Appendix H.

6 Conclusion

In this work, we presented a novel graph-based policy architecture, Body Transformer, which exploits the robot body structure as an inductive bias. Our experiments show how masked attention, which is at the core of BoT, benefits both imitation and reinforcement learning algorithms. Additionally, the architecture exhibits favorable scaling and computational properties, making it appealing for applications on high-dimensional systems.

Here, we used transformers to process sequences of distributed sensory information from the same timestep. However, transformers have been shown to excel at processing information across time too. We leave the extension of BoT to the temporal dimension as future work, as it promises to further improve real world deployment of robot policies, such as the one demonstrated on the Unitree A1 robot.

[1] A mask with sparsity β has βn^2 zero elements. When $\beta = 1 - \frac{1}{n}$, the mask reduces to the identity I_n. In practice, the maximum degree of a vertex (or node) in robots will be approximately constant, making the computational complexity of masked attention grow linearly with the number of nodes.

A limitation of our approach is the fact that its computational advantages are currently not fully exploited by modern deep learning libraries, and we hope that this work may stimulate future developments in this direction. In addition, our architecture requires a minimum amount of transformer layers to make sure that the architecture does not lose representation power in modeling long-range relations, which generally increases the amount of required trainable parameters.

Acknowledgments

We would like to thank Huang Huang and Antonio Loquercio for their help setting up the A1 experiments. This work was supported in part by the SNSF Postdoc Mobility Fellowship 211086, ONR MURI N00014-22-1-2773, BAIR Industrial Consortium, NSF AI4OPT AI Centre, NSF ACTION AI Centre Award IIS-2229876, an ONR DURIP grant. We also thank NVIDIA for providing compute resources through the NVIDIA Academic DGX Grant.

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.

[35] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022.

[36] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022.

[37] C. Sferrazza, D.-M. Huang, X. Lin, Y. Lee, and P. Abbeel. Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation. *arXiv Preprint arxiv:2403.10506*, 2024.

[38] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[39] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

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 mantain a target velocity of 0.5 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. Three main components – which are standard for sim-to-real locomotion [33] – were required for successful transfer, namely I) terrain randomization during training, similar to Zhuang et al. [33], II) higher stiffness in the joint controllers, and III) a low-pass action filter. The experiments were run on flat ground, with offboard computation on CPU. Commands were sent via WiFi or Ethernet connection.

The policy was evaluated through 5 different rollouts, which were considered successful as long as the robot walked for 10 seconds (based on the available experimental space) without falling. All five rollouts were successful.

We attach a supplementary video that demonstrates the real-world deployment. A frame overlay representing the robot motion is also shown in Figure 7.

Figure 7: **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 Allocating Observations and Actions to Nodes

We follow simple rules to allocate observations and actions to the different nodes of the graph, each of which represents a robot limb and is mapped to a row (or column) of the mask M. We distinguish between two types of nodes:

- A root node, to which we assign global observations (e.g., robot position and orientation) and environment observations (e.g., door handle angle, etc.).
- Non-root nodes, to which we assign local observations (e.g., joint angles) and local actions (e.g., joi

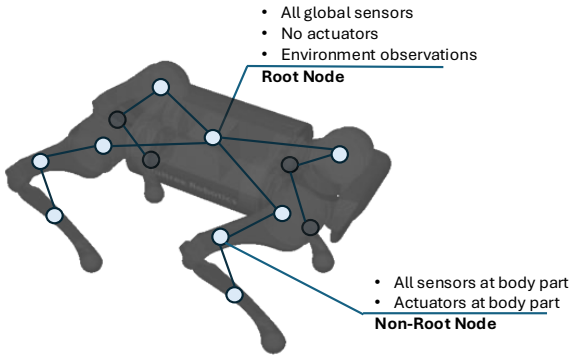


Figure 8: **Rules for Allocating Quantities to Nodes.**

As a rule of thumb, observations or actions that spanned multiple nodes were assigned to the closest node in the graph, or to either of the nodes involved.

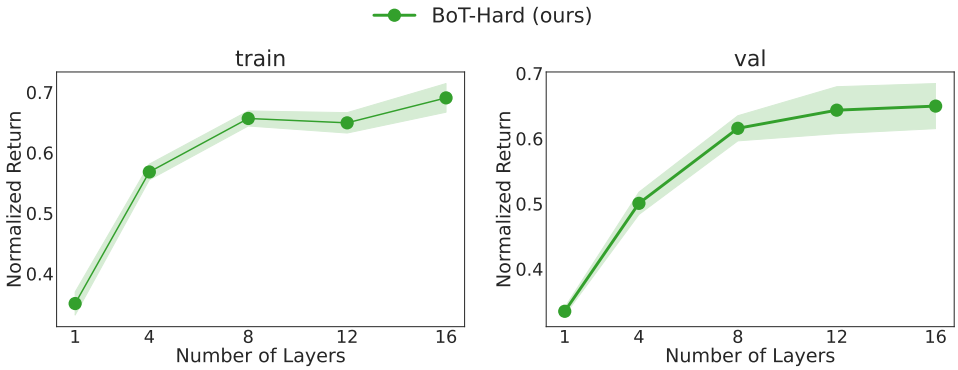
We present an example allocation for the A1-Walk task below:

- **base** – *observations:* orientation and angular velocity, *actions:* none.
- **front-left-hip** – *observations:* corresponding joint angle, joint velocity, and previous joint command, *actions:* corresponding joint command.
- **front-left-thigh** *observations:* corresponding joint angle, joint velocity, and previous joint command, *actions:* corresponding joint command.
- **front-left-calf** *observations:* corresponding joint angle, joint velocity, and previous joint command, *actions:* corresponding joint command.
- **front-right-hip** *observations:* corresponding joint angle, joint velocity, and previous joint command, *actions:* corresponding joint command.
- **front-right-thigh** *observations:* corresponding joint angle, joint velocity, and previous joint command, *actions:* corresponding joint command.
- **front-right-calf** *observations:* corresponding joint angle, joint velocity, and previous joint command, *actions:* corresponding joint command.
- **rear-left-hip** *observations:* corresponding joint angle, joint velocity, and previous joint command, *actions:* corresponding joint command.
- **rear-left-thigh** *observations:* corresponding joint angle, joint velocity, and previous joint command, *actions:* corresponding joint command.
- **rear-left-calf** *observations:* corresponding joint angle, joint velocity, and previous joint command, *actions:* corresponding joint command.
- **rear-right-hip** *observations:* corresponding joint angle, joint velocity, and previous joint command, *actions:* corresponding joint command.
- **rear-right-thigh** *observations:* corresponding joint angle, joint velocity, and previous joint command, *actions:* corresponding joint command.
- **rear-right-calf** *observations:* corresponding joint angle, joint velocity, and previous joint command, *actions:* corresponding joint command.

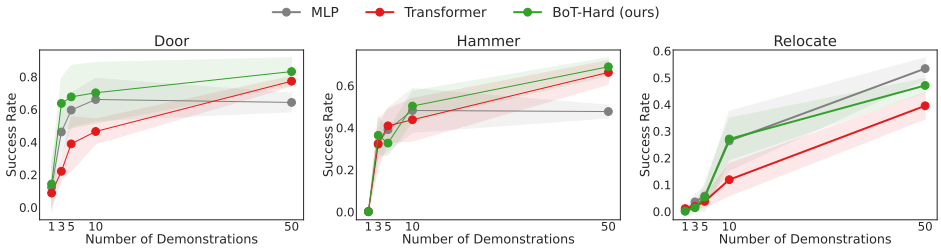
E Additional Imitation Learning Experiments

	normalized episode return		normalized episode length	
	train	val	train	val
BoT-Hard (ours)	**0.75 / 0.69** $\pm$ 0.02	**0.69 / 0.65** $\pm$ 0.04	**0.91 / 0.87** $\pm$ 0.02	**0.88 / 0.84** $\pm$ 0.03
BoT-Mix (ours)	0.72 / 0.68 $\pm$ 0.03	0.66 / 0.63 $\pm$ 0.01	0.89 / 0.86 $\pm$ 0.03	0.85 / 0.82 $\pm$ 0.01
BoT-Soft	0.71 / 0.68 $\pm$ 0.01	0.64 / 0.58 $\pm$ 0.03	0.88 / 0.87 $\pm$ 0.01	0.83 / 0.78 $\pm$ 0.03
BoT-Hard/Random	0.71 / 0.68 $\pm$ 0.01	0.65 / 0.60 $\pm$ 0.03	0.89 / 0.85 $\pm$ 0.02	0.83 / 0.80 $\pm$ 0.02
BoT-Hard-Stochastic (ours)	0.76 / 0.70 $\pm$ 0.02	0.70 / 0.66 $\pm$ 0.04	0.91 / 0.88 $\pm$ 0.01	0.88 / 0.84 $\pm$ 0.02

(a) **Additional Imitation Learning Ablations on MoCapAct Tracking Task.** Statistics of the various architecture-criterion combinations are shown with two values, the leftside being the maximum value recorded during training, and the rightside being the mean evaluation scores with standard deviation. Results are averaged across five seeds.



(b) **BoT-Hard vs Number of Layers.** Normalized return as a function of the number of masked attention layers. Note that the diameter of the MoCapAct humanoid graph is 14. Results are averaged across seeds.



(c) **Adroit Tasks.** We report success rates for a varying number of training demonstrations. Statistics are averaged across the last ten training epochs and five seeds, with the shaded areas indicating the standard deviation.

Figure 9: **Additional Imitation Learning Experiments.**

In this section we provide several ablations on the MoCapAct dataset, in addition to those presented in Section 5.1, as well as results on different tasks in the Adroit Hand benchmark [38, 39].

MoCapAct. We compare (i) BoT-Hard, (ii) BoT-Mix, (iii) BoT-Soft, which – similarly to [25] – learns the matrix B in (1) as a function of the graph's shortest path matrix, and (iv) BoT-Hard/Random 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 9a shows the result of this comparison, with BoT-Hard outperforming all baselines on most of 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. While this work focused on deterministic policies, we also implemented a stochastic version of BoT-Hard, which uses a zero-

mean Gaussian policy with fixed variance of 0.01. This variant shows that explicitly accounting for stochasticity can marginally improve performance further.

In Figure 9b, we show BoT-Hard performance as a function of the number of masked attention layers. Despite requiring 14 layers (equal to the embodiment graph diameter) for full information propagation across the graph, the plots show how our architecture may already exhibit good performance for a smaller number of layers.

Adroit Hand. We compare our BoT-Hard architecture with baselines on three dexterous manipulation tasks, which are part of the Adroit Hand benchmark. Specifically, we restrict the experiments to a low-data regime, where transformers notably struggle and tend to overfit [6]. The results in Figure 9c show how BoT-Hard consistently outperforms the vanilla Transformer, while being competitive or outperforming the MLP baseline across all three tasks.

F Additional Reinforcement Learning Ablations

F.1 Effect of Body-Induced Masking in BoT

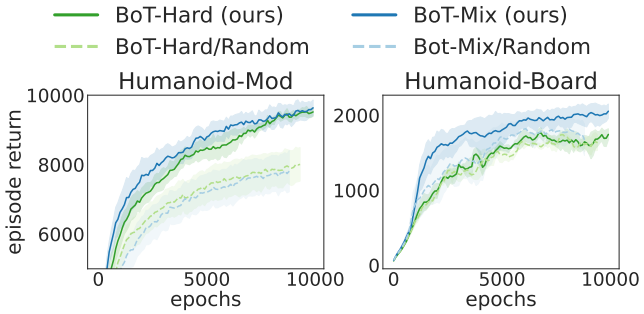


Figure 10: **Additional RL Experimental Results on the Effect of Body-induced Masking.**

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 10. 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.

F.2 Effect of Per-Limb Tokenizer vs. Shared Tokenizer

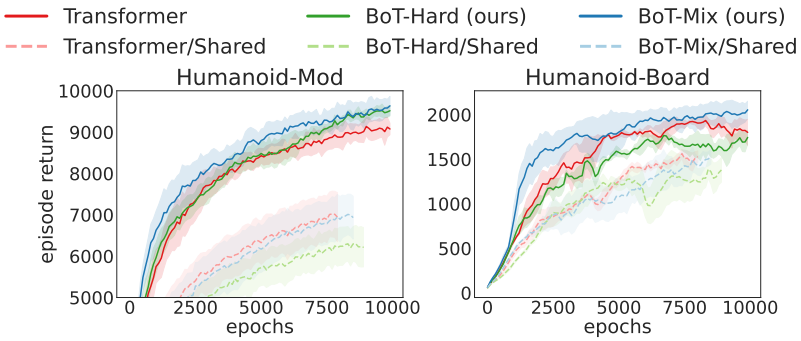


Figure 11: **Additional RL Experimental Results on the Effect of Per-Node (De)Tokenizers.**

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 11.

In Figure 11, 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.

G Training Details

The training parameters of the experiments detailed in Section 5.1 and Section 5.2 are as summarized in Tables 12a, 12b, and 12c.

H 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 $\frac{\boldsymbol{QK^T}}{\sqrt{\boldsymbol{d_k}}}$. Considering $Q \in \mathbb{R}^{n \times d_\text{k}}$ (and similarly for K), the computation of QK^T will generally require d_k multiplications and $d_\text{k} - 1$ 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 $2n^2 d_\text{k} + c_1$.

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

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

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

Computing the multiplication $\mathbf{Softmax}(\boldsymbol{S})\boldsymbol{V}$. A total of nd_k pairs are multiplied, where each pair requires $2n - 1$ operations to complete. The total amount of FLOPs is $2n^2 d_\text{k} - nd_\text{k}$. Following a similar reasoning with previous writing, a total of $2n^2 d_\text{k} - nd_\text{k}$ FLOPs are performed.

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

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
Eval Rollouts	100	100
# Parameters	16,696,656	17,544,120

(a) **Training Parameters Used for MoCapAct Experiments.**

Parameter	Values	
	MLP	Transformers
Batch Size	256	256
# Epochs	200	200
# Encoder Layers	3	16
Embedding Input Size	320	320
Feedforward Size	2500	1024
# Attention Heads	N/A	5
Learning Rate	1e-4	1e-4
Eval Rollouts	20	20
# Parameters	14,305,524	17,131,292

(b) **Training Parameters Used for Adroit Hand 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

(c) **Training Parameters Used for the Humanoid Reinforcement Learning Experiments.**

Figure 12: **Training Parameters Used for Experiments in Section 5.**

custom masked approach requires:

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

Therefore, even though these implementations share the same asymptotic bound $\mathcal{O}(n^2 d_{\mathrm{k}})$, the custom masked implementation's amount of FLOPs scales better than the vanilla implementation.

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

I Computation Analysis vs Sparsity

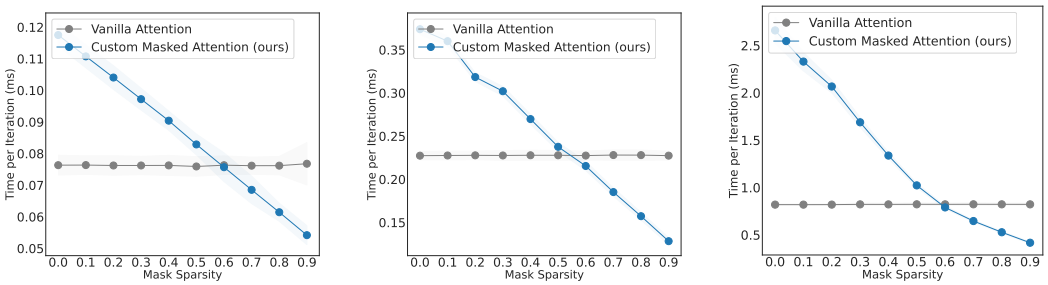


Figure 13: **Computational Analysis vs Mask Sparsity.**
Across 50,000 randomly sampled masks for different number of nodes (16, 32, 64, from left to right), runtime decreases with mask sparsity. BoT-Hard remains more efficient than the vanilla Transformer for mask sparsities lower than ≈ 0.6, which are much lower than common mask sparsities encountered in a robotics setting (including those in this paper).

Figure 13 shows a comparison in terms of runtime between masked and unmasked attention for varying degrees of sparsity. The plots highlight how BoT-Hard retains its computational advantages across a wide range of sparsity degrees, covering the most common scenarios encountered in a robotics setting.