

AnyMorph: Learning Transferable Policies By Inferring Agent Morphology

Brandon Trabucco¹ Mariano Phielipp^{*2} Glen Berseth^{*3}

Abstract

The prototypical approach to reinforcement learning involves training policies tailored to a particular agent from scratch for every new morphology. Recent work aims to eliminate the re-training of policies by investigating whether a morphology-agnostic policy, trained on a diverse set of agents with similar task objectives, can be transferred to new agents with unseen morphologies without re-training. This is a challenging problem that required previous approaches to use hand-designed descriptions of the new agent’s morphology. Instead of hand-designing this description, we propose a data-driven method that learns a representation of morphology directly from the reinforcement learning objective. Ours is the first reinforcement learning algorithm that can train a policy to generalize to new agent morphologies without requiring a description of the agent’s morphology in advance. We evaluate our approach on the standard benchmark for agent-agnostic control, and improve over the current state of the art in zero-shot generalization to new agents. Importantly, our method attains good performance *without* an explicit description of morphology.

1. Introduction

Agent-agnostic reinforcement learning (RL) is an emerging research challenge that involves training policies that are transferable to new agents with different morphology. Rather than training policies from scratch for every new agent, a pretrained agent-agnostic policy can provide an effective solution with potentially no additional training. The core philosophy motivating agent-agnostic RL is a notion that large multi-task models exhibit strong generalization behavior. This result is continually observed in domains outside robotics, where models like BERT (Devlin et al.,

^{*}Equal contribution ¹Machine Learning Department, Carnegie Mellon University, work done while at Intel AI ²Intel AI ³Mila. Correspondence to: Brandon Trabucco <btrabucco@cmu.edu>.

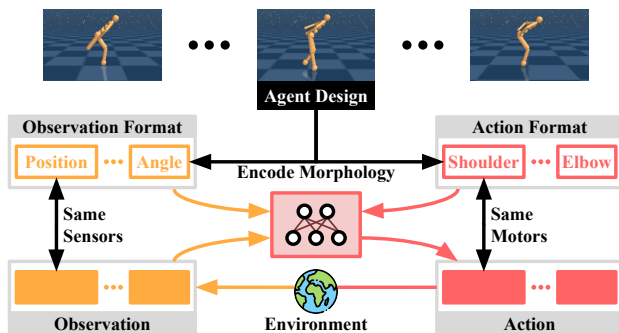


Figure 1. Overview of the architecture of our policy for an example reinforcement learning task with three agent morphologies. The agent’s morphology is represented by a sequence of tokens, and is processed by a sequence-to-sequence Transformer policy.

2019), GPT-3 (Brown et al., 2020), and CLIP (Radford et al., 2021) exemplify how large multi-task models zero-shot generalize in out-of-domain applications like CLIP in robotics (Khandelwal et al., 2021). Agent-agnostic RL has the potential to create policies with comparable generalization to these foundation models (Bommasani et al., 2021). However, agent-agnostic RL is hard because it requires an effective representation of the agent’s morphology (Huang et al., 2020), which remains a challenging research problem. The morphology representation is crucial in agent-agnostic reinforcement learning because it must contain all the information about the agent’s physics needed for control. In this work, we consider how to learn this representation instead of manually designing it, as is standard in current research.

Existing works have approached agent-agnostic RL by assuming the agent obeys strict design criteria. First, the agent has limbs. Second, each limb has similar proprioceptive sensors (Wang et al., 2018; Huang et al., 2020; Kurin et al., 2021). These assumptions restrict agent-agnostic RL to rigid-body agents in a MuJoCo-like (Todorov et al., 2012) simulator that exposes congruent limb-observations. Deviating from prior work, we consider a setting that is more flexible than prior work. Our method does not require the agent to have limbs, nor does it require observations to be annotated according to which limb they describe. Instead, our method automatically infers how the agent’s sensors and motors are connected from the RL objective, illustrated in Figure 1. Our approach performs 16% better (see Section 6) than existing methods in zero-shot generalization to

new morphologies, and requires *less* information about the morphology of the agent being controlled than prior art.

In this paper, we make the following contributions. First, we frame learning morphology as a sequence modelling problem, and represent an agent’s morphology as a sequence of tokens, with corresponding learnt embeddings. In this fashion, an agent’s morphology is differentiable, and is optimized from the reinforcement learning objective. Second, we propose an agent-agnostic neural network architecture that generalizes effectively and does not assume the agent has limbs, nor has per-limb observations. Third, we demonstrate that our approach generalizes up to 32% better than existing work on large tasks in a standard benchmark, and displays an emergent robustness to broken sensors.

2. Related Works

Generalization in reinforcement learning has a rich history, with early works demonstrating generalization to new tasks in robotics Sutton et al. (2011), and in video games (Schaul et al., 2015; Parisotto et al., 2016). Tasks were often defined via goal states Sutton et al. (2011), with a reinforcement learning objective defined as minimizing the distance to the goal throughout an episode (Andrychowicz et al., 2017; Nasiriany et al., 2019). These goal-based methods can be improved combining them with language (Jiang et al., 2019), and learnt latent spaces (Eysenbach et al., 2019; Rakelly et al., 2019). As the field matures, researchers are beginning to investigate larger-scale multi-task reinforcement learning (MTRL) (Vithayathil Varghese & Mahmoud, 2020), including generalization across all of Atari (Hafner et al., 2020), and generalization to new agents with different dynamics or morphology. The latter is an emerging topic called agent-agnostic reinforcement learning (Devin et al., 2017; III et al., 2020; Huang et al., 2020). This setting is challenging because different agents typically have incompatible (different cardinality) observations and actions, which precludes conventional deep reinforcement learning approaches.

Conventional deep reinforcement learning approaches expect fixed-size observations and actions, and devising effective alternatives is an open research challenge. Existing work has shown modularization can both improve multi-task generalization (Chang et al., 2021; Goyal et al., 2021), and allow processing of different-cardinality observations and actions (Huang et al., 2020). Many such approaches utilize Graph Neural Networks (Gori et al., 2005; Scarselli et al., 2005), conditioning the policy on a graph representation of the agent’s morphology. Graph-based approaches have demonstrated the ability to generalize to novel agent morphology (Huang et al., 2020), and learn complex gaits (Wang et al., 2018; Sanchez-Gonzalez et al., 2018). However, recent work has shown that generalization can be further improved using Transformers (Kurin et al., 2021), due to their success

in modelling dynamic structure via the attention mechanism (Tenney et al., 2019; Vig & Belinkov, 2019). Agent-agnostic reinforcement learning methods, including those with Transformers, currently rely on a manually-designed representation of morphology (see Section 3). Our approach differs from prior work by instead learning a representation of morphology with reinforcement learning that produces better generalization than a manually-designed representation.

3. How Do You Represent Morphology?

The goal of this section is to outline what information is needed to define a *morphology*. This terminology is used, for example, to denote an interpretation of the physical rigid-body of a MuJoCo-like (Todorov et al., 2012) agent as a graph, where nodes correspond to rigid limbs, and edges indicate two limbs are connected by a joint. This terminology is based on classical work on kinematic chains (Denavit & Hartenberg, 1955). In this section, we will explore a standard definition of morphology for decision-making agents used in existing work, and will discuss its benefits and limitations. Towards this goal, we first introduce the reader to notation that will be referred to throughout our paper.

Preliminaries. We take inspiration from and modify the notation presented by Huang et al. (2020, p. 3). Consider a set of N decision-making agents with different observation spaces and actions spaces—to avoid a recursive definition of *morphology*, we intentionally avoid using it here. For each agent $n \in \{1 \dots N\}$, there is a corresponding Markov Decision Process \mathcal{M}_n with observation space S_n , action space A_n , transition dynamics p_n , and reward function r_n . We will assume for notational convenience that each observation $\mathbf{s} \in S_n$ can be expressed as a vector with $N_S(n)$ elements, and that each action $\mathbf{a} \in A_n$ can be expressed as a vector with $N_A(n)$ elements. The goal of agent-agnostic reinforcement learning is to find a policy that maximizes the expected discounted return for the expected agent.

$$\max_{\pi} \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_n(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \right] \quad (1)$$

The optimal policy $\pi^*(\mathbf{a}_t|\mathbf{s}_t, n)$ for this objective depends on the morphology of the current agent n . How the morphology of the current agent is represented from the perspective of the policy is an important question in our investigation. Let’s first identify how prior work answers this question.

Using Graph-based Morphologies. The *morphology* of a decision-making agent is often represented as a graph. Consider the agent-conditioned undirected graph $\mathcal{G}_n = (V_n, E_n)$ whose vertices V_n each represent a limb the agent has, and whose set of edges E_n contains all pairs of limbs that are connected by a joint. For the typical MuJoCo-like (Todorov

Table 1. Comparison of the structural assumptions that prior works make about the decision-making agent. Our method is more flexible than prior work, requiring neither graph structure, nor explicitly aligning the agent’s sensors and actuators to its limbs in order to generalize effectively. Our approach requires less information about the agent’s morphology than prior work, and performs 16% better in zero-shot generalization to new morphologies than existing methods with stronger assumptions, which is discussed in Section 6.

Method \ Assumptions	Graph Structure	Alignment
Graph Networks (Sanchez-Gonzalez et al., 2018)	✗	✗
NerveNet (Wang et al., 2018)	✗	✗
Shared Modular Policies (SMP) (Huang et al., 2020)	✗	✗
Amorpheus (Kurin et al., 2021)		✗
AnyMorph (Ours)		

et al., 2012) agent from prior work, the topology of this graph closely resembles the topology of the agent’s body.

$$V_n = \{1 \dots N_L(n)\} \quad (2)$$

$$E_n = \{(i, j) : i \text{ and } j \text{ are connected limbs}\} \quad (3)$$

Graph-based morphologies are intuitive because they mirrors how the agent is physically connected, but \mathcal{G}_n is not a complete description of morphology. The number of limbs $N_L(n)$ is generally independent of the dimensionality of the action space $N_A(n)$ and observation space $N_S(n)$. The agent must also know which individual sensors and motors are connected to each limb. This knowledge is typically specified via an $N_L(n)$ -way partition P_A of the action space and an $N_L(n)$ -way partition P_S of the observation space. Figure 2 shows how sensors and motors are partitioned in prior work for an example humanoid morphology.

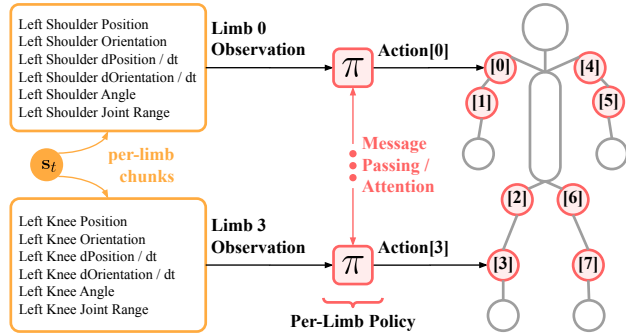


Figure 2. Illustration of how existing methods partition the action space and observation space of the `humanoid_2d_9_full` morphology. These works assume limbs are uniformly designed, and require additional annotations about how sensors and motors are connected to limbs (P_A and P_S) before learning can happen.

Consequences Of Graph-based Morphologies. One appealing quality of these graph-based morphologies are that they have an intuitive physical interpretation. Given only the morphology, one can invert the definition to recover the agent’s physical design, including the arrangement of its sensors. This appealing quality comes at the expense of strict

design criteria, however. In this section, we will discuss several consequences of the graph definition of morphology.

- First, this definition assumes the agent’s physical design can be expressed as a graph, which restricts methods to MuJoCo-like (Todorov et al., 2012) agents with rigid limbs, which is not true, for example, for a car with flexible rubber tires. We refer to this as the **graph structure assumption** in Table 1. The community is interested in agent-agnostic RL for real-world robotic systems, and a graph inductive bias may be insufficient to represent the complex physique of certain agents.
- Another common assumption is that sensors and actuators are categorizable according to a particular limb. Methods with this **alignment assumption** in Table 1 only maintain agnosticism to limbs, and not to individual sensors and actuators (Huang et al., 2020; Kurin et al., 2021). Some interesting reinforcement learning problems violate this property by not having per-limb sensors, or even well-defined limbs. Tackling these problems may require more flexible agent-agnostic methods that do not require aligning sensors to limbs.

4. Morphology As Sequence Modelling

Defining morphology in terms of limbs, which we discussed in Section 3, often requires making three unrealistic assumptions about the agent (see Table 1). In this section, we will present an alternate representation of morphology for general decision-making agents that *outperforms* existing methods while being applicable to a larger space of agent types. We approach this problem by considering two algorithmic desiderata: for a given agent n , (1) our method should be invariant to the dimensionality of the agent’s observations and actions, and (2) our method should only be given *minimal* information about the agent’s morphology. Our first insight is to interpret the reinforcement learning policy that accepts a state vector and outputs an action vector as sequence-to-sequence mappings, where the source

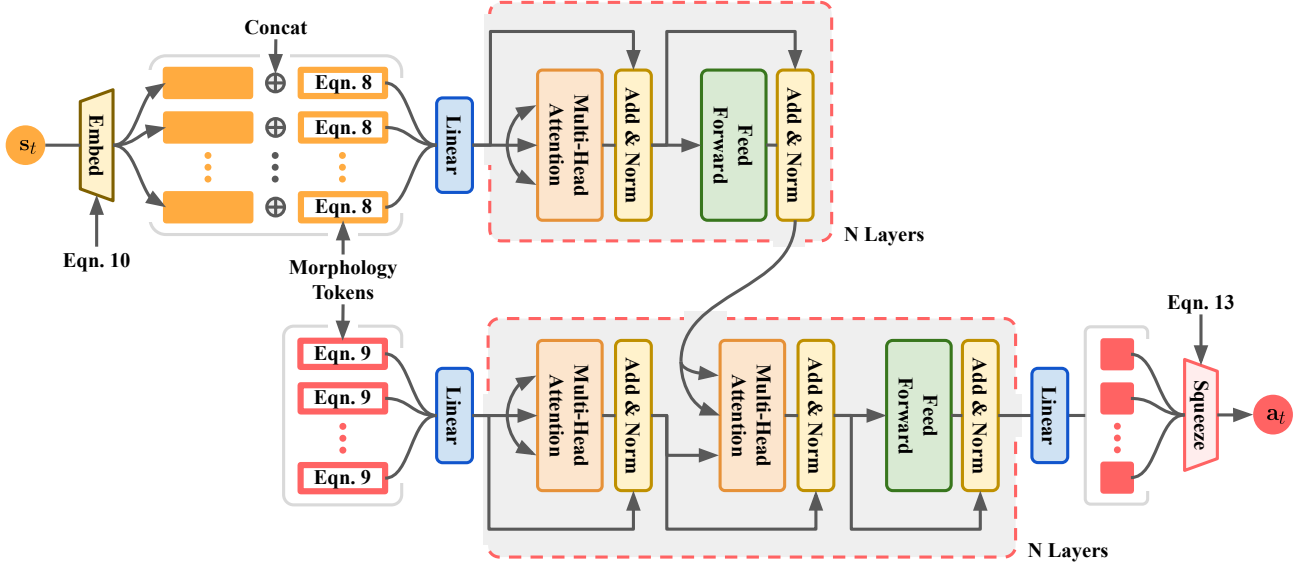


Figure 3. Visualization of the architecture of our policy. Our policy is a sequence-to-sequence deep neural network, consisting of a Transformer (Vaswani et al., 2017) encoder that processes the current state s_t and a sequence of embeddings of observation tokens (see Section 4), with a non-masked Transformer decoder that processes a sequence of embeddings of action tokens, given the encoder hidden state. Our policy is invariant to the dimensionality of observations and actions, and does not assume the agent has limbs.

sequence has $N_S(n)$ elements, and the target sequence has $N_A(n)$ elements. This interpretation bypasses all dependence on limbs by defining morphology in terms of which sensors and actuators comprise the agent. Inspired by the success of word embeddings (Mikolov et al., 2013) in natural language processing, and their role in large-scale models like BERT (Devlin et al., 2019), we propose to encode sensors and actuators as a sequence of tokens, and represent them with learnt embeddings. We learn two embeddings that represent observation and action tokens respectively.

$$H_n^{\text{obs}} \in \mathbb{R}^{N_S(n) \times D}, H_n^{\text{act}} \in \mathbb{R}^{N_A(n) \times D} \quad (4)$$

Each token encodes the identity of a single sensor or actuator. Embeddings representing these tokens are learned jointly with our policy directly from the reinforcement learning objective. Our policy is a sequence-to-sequence deep neural network that maps from a source sequence with $N_S(n)$ elements to a target sequence with $N_A(n)$ elements. We implement our policy as a variant of the Transformer (Vaswani et al., 2017) due to its success in multiple reinforcement learning settings (Kurin et al., 2021; Janner et al., 2021; Chen et al., 2021). In the following section, we describe our architecture (shown in Figure 3), and modifications that make it suitable for agent-agnostic reinforcement learning.

Morphology Tokens. Before detailing our policy architecture, we will explore how the necessary observation embeddings H_n^{obs} and action embeddings H_n^{act} can be obtained. In the previous section, we described a corresponding set of observation tokens I_n^{obs} and action tokens I_n^{act} that are

represented by these embeddings. These tokens are consecutive integers that uniquely index the sensors and motors comprising the agent’s action space and observation space. Since the action space has dimensionality $N_A(n)$, the action tokens I_n^{obs} can take at most $N_A(n)$ possible values. Similarly, the observation tokens I_n^{act} can take on at most $N_S(n)$ possible values. Emphasizing our inspiration from language modelling, the pair $(I_n^{\text{obs}}, I_n^{\text{act}})$ functions like word tokens (Mikolov et al., 2013). To honor this inspiration from language modelling, we label this pair *morphology tokens*.

$$I_n^{\text{obs}} \in \mathbb{N}^{N_S(n)}, I_n^{\text{act}} \in \mathbb{N}^{N_A(n)} \quad (5)$$

Just as word tokens specify an index into an associated set of word embeddings (Mikolov et al., 2013), morphology tokens specify an index into a corresponding pair of morphology embeddings W_e^{obs} , and W_e^{act} . These morphology embeddings are weight matrices with D columns. Each weight matrix learns an embedding for each sensor and motor in the observation and action spaces of every morphology, similar to how word embeddings learn an embedding for each word in a vocabulary. See Appendix D for a discussion of how our morphology tokens are selected in practice.

$$W_e^{\text{obs}} \in \mathbb{R}^{(\prod_{i=1}^N N_S(i)) \times D} \quad (6)$$

$$W_e^{\text{act}} \in \mathbb{R}^{(\prod_{i=1}^N N_A(i)) \times D} \quad (7)$$

Given the pair of morphology tokens and the pair of morphology embeddings, we use an embedding lookup operation in order to obtain the pair of observation and action

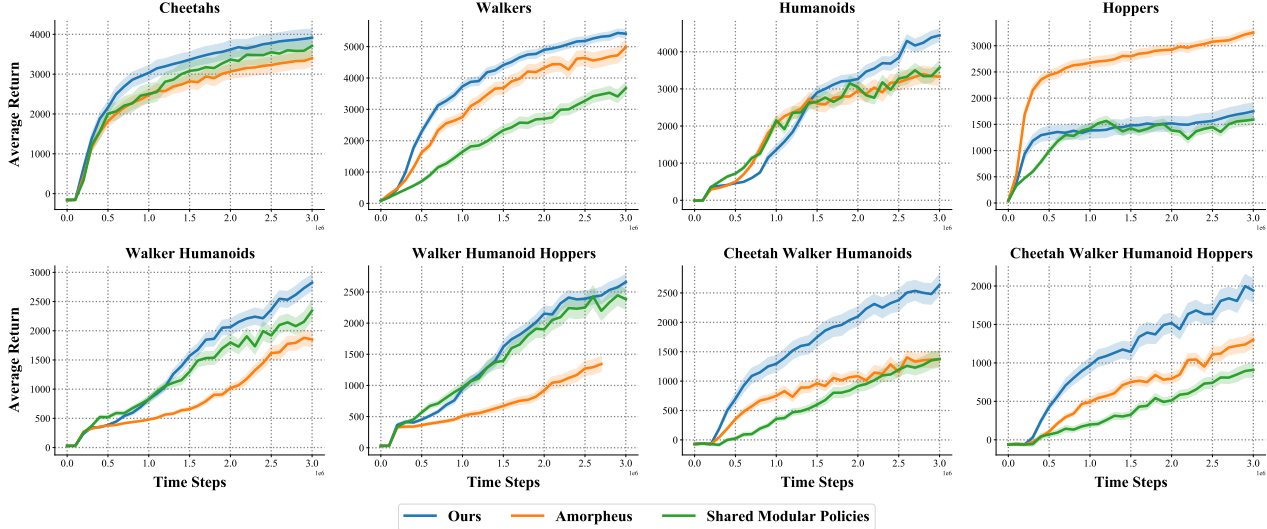


Figure 4. Average return of our method versus Amorpheus (Kurin et al., 2021) and Shared Modular Policies (Huang et al., 2020) across 4 random seeds. Performance on the y-axis is the average return with one episode per training morphology per seed. Dark colored lines indicate the average training performance, and a 95% confidence interval is shown with shading around each line. The x-axis indicates the total steps across all training morphologies. Our method frequently reaches and exceeds the performance of baselines, improving by 85% and 53% on Cheetah-Walker-Humanoids and Cheetah-Walker-Humanoid-Hoppers respectively, the tasks with the most morphologies.

token embeddings H_n^{obs} and H_n^{act} for each morphology. Recall from Equation 4 these have a fixed number of columns, and a variable number of rows, which depends on the dimensionality of the agent’s observations and actions.

$$H_n^{\text{obs}} = \text{embedding_lookup} (I_n^{\text{obs}}, W_e^{\text{obs}}) \quad (8)$$

$$H_n^{\text{act}} = \text{embedding_lookup} (I_n^{\text{act}}, W_e^{\text{act}}) \quad (9)$$

These observation and action token embeddings are shown in Figure 3 as a sequence of outlined rectangles on the left of the diagram. Each colored rectangle in each sequence represents a token embedding vector with cardinality D . Outlined yellow rectangles represent observation tokens, and outlined red rectangles represent action tokens. While this representation of morphology does not require the agent to have limbs, it requires instead a pair of morphology tokens that identify the agent’s sensors and actuators. Since this representation has the advantage of differentiability, one promising option to avoid manual annotation is to infer H_n^{obs} and H_n^{act} using an encoder for future work.

Embedding The Current Observation. How can we condition our policy, which is a sequence to sequence model, on the current state? We propose to view the current state as a sequence with $N_S(n)$ elements, embedding each element to a M -vector, and concatenate it with the matrix of observation token embeddings H_n^{obs} column-wise. We empirically find that passing the state through a sinusoidal embedding function before concatenation helps the model to perform well, which may be interpreted as increasing the

expressivity of the model class (Li & Pathak, 2021). Our embedding function passes the state through a series of M sinusoidal functions with geometrically increasing frequencies $k_1, k_2, \dots, k_{\lfloor M/2 \rfloor}$ from 1/10 to 1000. In Figure 3, we represent this operation with the yellow trapezoid labelled embed on the left, and denote the subsequent concatenation of the embedded state with H_n^{obs} via the plus symbol.

$$X = \left[H_n^{\text{obs}}; \underbrace{\cos(k_1 \mathbf{s}_t); \sin(k_1 \mathbf{s}_t); \dots}_{M \text{ columns}} \right] \quad (10)$$

Before processing X and H_n^{act} with our transformer, we apply a linear transformation that maps them to the cardinality of the Transformer hidden state. We learn two linear transformations W_p^{obs} and W_p^{act} for observation and action embeddings respectively, projecting to d_{model} components.

$$W_p^{\text{obs}} \in \mathbb{R}^{(D+M) \times d_{\text{model}}}, W_p^{\text{act}} \in \mathbb{R}^{D \times d_{\text{model}}} \quad (11)$$

After projection, we process these embeddings with a Transformer encoder-decoder model, following the architecture presented by Vaswani et al. (2017, p. 3). Note that we drop autoregressive masking, which is unnecessary in our setting. Details about the hyperparameters used with our model can be found in Appendix A. The output of our Transformer is a sequence of $N_A(n)$ hidden state vectors with d_{model} components. To parameterize an action distribution, we learn a final linear transformation W_p^{out} that projects from d_{model} -vectors to scalars. Our model outputs a vector y_t with

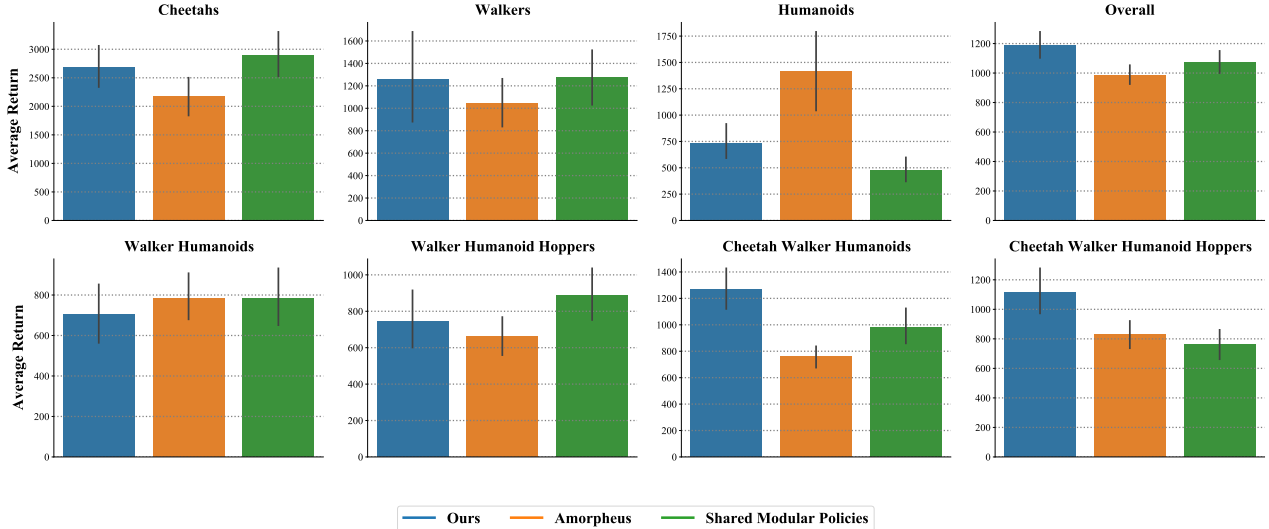


Figure 5. Average return of our method versus Amorpheus (Kurin et al., 2021) and Shared Modular Policies (Huang et al., 2020) for held-out morphologies with 4 random seeds. Performance on the y-axis is the average return with ten episodes per held-out morphology per seed. Colored bars represent average performance evaluated at 2.5 million environment steps, and a 95% confidence interval is shown with error bars. Overall performance is an aggregation of all episodes from each method. Our model improves by 16% overall, and by 28% and 32% on the Cheetah-Walker-Humanoids and Cheetah-Walker-Humanoid-Hoppers tasks respectively, the two hardest tasks.

$N_A(n)$ components, representing pre-activation actions.

$$W_p^{\text{out}} \in \mathbb{R}^{d_{\text{model}}} \quad (12)$$

$$y_t = \text{Transformer}(XW_p^{\text{obs}}, H_n^{\text{act}}W_p^{\text{act}})W_p^{\text{out}} \quad (13)$$

With a final hyperbolic tangent activation, we generate the mean action $\mathbf{a}_t = \tanh(y_t)$ for the current timestep. This conversion from Transformer outputs to actions is shown in Figure 3 by the Linear and Squeeze boxes on the right. By framing learning morphology as sequence modelling, our model is applicable to a larger space of agent types, given appropriate morphology tokens and sufficient training. Furthermore, our model does not assume the agent conforms to strict design criteria seen in prior work and detailed in Section 3. In the remaining sections, we will benchmark our model on a standard set of agent-agnostic reinforcement learning tasks, and visualize what our model learns.

Policy Optimization. We choose to employ TD3 (Fujiwara et al., 2018) for optimizing our policy because it is highly efficient and consistently performs well for MuJoCo-like (Todorov et al., 2012) agents. Efficiency is an important consideration in agent-agnostic reinforcement learning because training with up to 32 morphologies can require large amounts of experience otherwise. Following the TD3 algorithm, we sample exploratory actions from a normal distribution $\mathcal{N}(\mathbf{a}_t, \sigma^2)$ where \mathbf{a}_t is the mean action, and the variance σ^2 is a hyperparameter that controls the degree of randomness in the exploration policy of TD3. Following prior work (Huang et al., 2020) we store a separate replay

buffer of transitions for each training morphology, and alternate between collecting environment steps and training for each morphology in lockstep. We train each agent for up to 3 million environments steps total across all training morphologies, and run 4 random seeds per method. Our model fits on a single Nvidia 2080ti GPU, and requires seven days of training to reach 3 million environments steps. We provide a table of hyperparameters in Appendix A for our policy and reinforcement learning optimizer. Additionally, we have released the source code for our method and summarized how the model works at the following [site](#).

5. Training Performance

We have constructed a method that is agnostic the morphology of an agent. This model is trained across a collection of different agents to develop skills that generalize to new morphologies. To understand how well our method succeeds we evaluate three aspects of the model. First, how quickly can we train the morphology agnostic policy. We should receive gains in sample efficiency as the model learns how to share experience across morphologies. Second, how well does the method generalize to novel morphologies not seen during the training process. Last, how robust is our method to sensor issues. To answer these questions, we leverage a benchmark for agent-agnostic reinforcement learning developed by Huang et al. (2020, p. 1). This benchmark contains a set of eight reinforcement learning tasks, where the goal is to maximize the average return over a

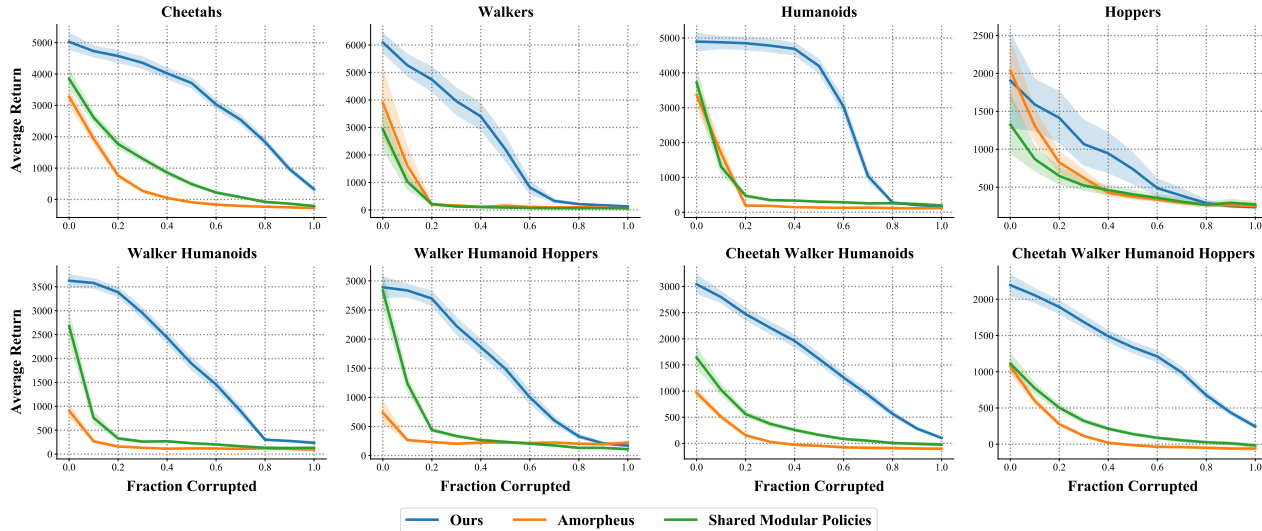


Figure 6. Average return of our method versus Amorpheus (Kurin et al., 2021) and Shared Modular Policies (Huang et al., 2020) as a function of how many sensors are corrupted by random noise. Performance on the y-axis is the average return of 4 random seeds with one episode per morphology for each seed. Dark colored lines indicate the average performance for each method given a fraction of sensors that are corrupted by noise sampled from a standard normal distribution, and a 95% confidence interval is shown with shading around each line. The x-axis indicates the ratio of how many sensors in the agent’s observation space are corrupted with noise, divided by the total number of sensors the agent has. Our method consistently improves robustness, illustrated by a greater area under the curve.

set of N agents with different morphologies. The agents present in this benchmark and inspired by and derived from standard OpenAI Gym tasks: HalfCheetah-v2, Walker2d-2, Hopper-v2, and Humanoid-v2 (Brockman et al., 2016). For the Cheetahs, Walkers, Humanoids, and Hoppers tasks, there are a total of 15, 6, 8, and 3 different morphologies used in prior work (Kurin et al., 2021; Huang et al., 2020). The benchmark consists of four tasks containing morphologies of a single *kind* of agent (see Cheetahs, Walkers, Humanoids, and Hoppers in Figure 4), and four tasks that mix together multiple kinds of agents (see Walker-Humanoids, Walker-Humanoid-Hoppers, Cheetah-Walker-Humanoids, and Cheetah-Walker-Humanoid-Hoppers in Figure 4). Solving the latter mixed tasks requires the policy to acquire gaits that generalize to multiple kinds of agents. The two hardest tasks in the benchmark are Cheetah-Walker-Humanoids, and Cheetah-Walker-Humanoid-Hoppers, which involve controlling 29 and 32 different agent morphologies respectively.

Our method excels at controlling many morphologies, and sees greater improvements as the amount of morphologies increases. Shown in Figure 4, on all tasks but Hoppers, our policy meets and exceeds the performance of Amorpheus (Kurin et al., 2021) and Shared Modular Policies (Huang et al., 2020), and avoids making strong assumptions about the agent’s design. Furthermore, on the hardest tasks, Cheetah-Walker-Humanoids, and Cheetah-Walker-Humanoid-Hoppers, we see an improvement of 85% and

53% respectively. This improvement suggests that our method is more scalable than prior work, as it trains across up to 32 different morphologies, while maintaining strong performance. Our method performs consistently well across all tasks except Hoppers. The Hoppers task has only three morphologies, the fewest in the benchmark, and we suspect this relates to our diminished performance.

6. Zero-Shot Generalization

In the previous section, we evaluated the training performance of our model, and demonstrated a significant improvement when controlling many morphologies. Now we ask: how well does our model generalize to new morphologies it was not trained on? To answer this question, we follow Kurin et al. (2021) and hold out 3 Cheetahs, 2 Walkers, and 2 Humanoids respectively. See Appendix C for which specific morphologies are used for testing. We then evaluate the policies learned by our method, Amorpheus, and Shared Modular policies at 2.5 million environment steps on each morphology that are held out. For tasks that involve multiple kinds of agents, we evaluate using all held out morphologies for each kind. We report the average return for each method over 4 random seeds, and ten episodes per seed, with a 95% confidence interval in Figure 5.

Despite not explicitly conditioning on morphology via graph structure or sensor-to-limb alignment, our method



Figure 7. Visualization of the cross attention weights in the final layer of our policy model on the Humanoids task and Humanoid 2d 9 Full morphology. Each cell in the attention matrix corresponds to the maximum attention value for that cell over an entire episode. Cells with dark shading indicate that our model ignores those sensors over an entire episode by zeroing their attention weights.

improves by 16% in overall zero-shot generalization. On tasks with fewer morphologies, our methods performs on par with existing methods, with an exception on the Humanoids tasks. We obtain an improvement of 28% on Cheetah-Walker-Humanoids, and 32% on Cheetah-Walker-Humanoid-Hoppers, the two hardest tasks in the benchmark. The disparity between training and testing performance on certain tasks (such as the Humanoids task) suggests observing many morphologies during training is key to regularization that enables our model to generalize effectively.

7. Visualizing Robustness

In the two previous experiments, we evaluated the training performance and zero-shot generalization ability of our model, and displayed compelling gains. One remaining question, however, is how resilient our model is to *minor* changes in the agent’s morphology, caused by one or more sensors breaking. We investigate this question by designing an experiment where a fraction of the agent’s sensors are replaced with noise sampled from the standard normal distribution. The goal of this experiment is to evaluate the robustness of our model, measured by average return, as a function of how many of the agent’s sensor readings (state features) are replaced with random noise. Our methodology for selecting the order to corrupt sensors is described in Appendix B. An evaluation of the robustness of our method compared to prior works is presented in Figure 6, where the average return for each method given a certain fraction of corrupted sensors is calculated from one episode per morphology per task using deterministic actions, with 4 random seeds per method on each task. Training average return is plotted on the y-axis, with a 95% confidence interval.

This experiment demonstrates that our policy architecture is more robust to broken sensors than prior works. Our policy achieves a higher area under the curve on all eight tasks. Though our method initially performs worse than Amorpheus (illustrated in Figure 4) on the Hoppers task, broken sensors quickly cripple existing methods. One promising

hypothesis for this improved robustness is that our model is actively ignoring certain sensors. To evaluate this hypothesis, we visualize the attention weights throughout an episode in the final cross attention mechanism in our model (that is, actuator-to-sensor attention) in Figure 7. For each cell in the visualization, we take the maximum attention weight over an entire episode (rows no longer sum to one) for one trial on the Humanoids task and Humanoid 2d Full morphology. This visualization shows that our model learns sparse attention weights that ignore the majority of sensors, shown by the majority of dark cells in the visualization. This quality in the attention weights suggests our model improves robustness by sparsely attending to sensors.

8. Conclusion

We have presented a method for learning transferable policies between agents of different morphology, by inferring the agent’s morphology via a learned embedding. Our approach is more scalable than existing methods, and is able to learn composable policies for up to 32 different morphologies at once while maintaining a performance lead of 82% in training, and 32% in zero-shot generalization on our benchmarking task with the most morphologies. Our method operates by framing learning morphology as a sequence modelling problem, and learns a Transformer-based policy that is invariant to the dimensionality of the agent’s observations and actions. In addition to improving performance, we demonstrate that our policy is more resilient to broken sensors than existing methods. Importantly, our method attains these improvements while also relaxing the amount of information required about the agent’s design compared to prior work. Our method does not require the agent to have limbs, graph structure, or aligned sensors and actuators. By relaxing these assumptions, our approach helps to improve the applicability of agent-agnostic learning in a more general reinforcement learning context.

Our research is a step towards broadly applicable agent-agnostic reinforcement learning methods, and there are several opportunities to expand our work. Firstly, we observed in Section 6 that our methods benefits from observing many morphologies during training. Further scaling of our method to agent-agnostic reinforcement learning tasks with significantly more morphologies may further improve zero-shot generalization. Secondly, we evaluated our method on MuJoCo-like agents with rigid limbs in this work. Applying our method to reinforcement learning tasks *without* an underlying graph structure or aligned sensors and limbs (such as Atari) poses an interesting challenge. Finally, our method requires morphology tokens (see Section 3) for each agent. Inferring these tokens from trajectories the agent has collected may enable our method to generalize out-of-the-box without requiring any manual annotation of the task.

9. Acknowledgements

This work was supported by Intel AI and CIFAR. We thank Benjamin Eysenbach for providing feedback on an early draft of the paper and for fruitful discussions about prior work. We thank Sergey Levine, Ruslan Salakhutdinov, and Deepak Pathak for providing feedback on the paper and for helpful discussions about the approach and how to evaluate it. We thank the authors Huang et al. (2020) for providing an open source benchmark and an implementation of SMP. We thank the authors Kurin et al. (2021) for providing an implementation of Amorpheus, which we used as a starting point. We thank Wenlong Huang for help debugging the SMP baseline. We thank Vitaly Kurin for pointing us to new papers in agent-agnostic RL that helped our literature review. We thank the anonymous reviewers at ICML for providing detailed feedback on a draft of the paper.

References

- Andrychowicz, M., Crow, D., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. Hindsight experience replay. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5048–5058, 2017.
- Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N. S., Chen, A. S., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel, K., Goodman, N. D., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D. E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Koh, P. W., Krass, M. S., Krishna, R., Kudithipudi, R., and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021.
- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *CoRR*, abs/1606.01540, 2016.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- Chang, M., Kaushik, S., Levine, S., and Griffiths, T. Modularity in reinforcement learning via algorithmic independence in credit assignment. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1452–1462. PMLR, 2021.
- Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021.
- Denavit, J. and Hartenberg, R. S. A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices. *Journal of Applied Mechanics*, 22(2):215–221, 06 1955. ISSN 0021-8936. doi: 10.1115/1.4011045.
- Devin, C., Gupta, A., Darrell, T., Abbeel, P., and Levine, S. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, pp. 2169–2176. IEEE, 2017. doi: 10.1109/ICRA.2017.7989250.
- Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T. (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423.
- Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.
- Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1582–1591. PMLR, 2018.

- Gori, M., Monfardini, G., and Scarselli, F. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pp. 729–734 vol. 2, 2005. doi: 10.1109/IJCNN.2005.1555942.
- Goyal, A., Lamb, A., Hoffmann, J., Sodhani, S., Levine, S., Bengio, Y., and Schölkopf, B. Recurrent independent mechanisms. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.
- Hafner, D., Lillicrap, T. P., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.
- Huang, W., Mordatch, I., and Pathak, D. One policy to control them all: Shared modular policies for agent-agnostic control. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4455–4464. PMLR, 2020.
- III, D. J. H., Pinto, L., and Abbeel, P. Hierarchically decoupled imitation for morphological transfer. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4159–4171. PMLR, 2020.
- Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021.
- Jiang, Y., Gu, S., Murphy, K., and Finn, C. Language as an abstraction for hierarchical deep reinforcement learning. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d’Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 9414–9426, 2019.
- Khandelwal, A., Weihs, L., Mottaghi, R., and Kembhavi, A. Simple but effective: CLIP embeddings for embodied AI. *CoRR*, abs/2111.09888, 2021.
- 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 *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.
- Li, A. C. and Pathak, D. Functional regularization for reinforcement learning via learned fourier features. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021.
- Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. In Bengio, Y. and LeCun, Y. (eds.), *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.
- Nasiriany, S., Pong, V., Lin, S., and Levine, S. Planning with goal-conditioned policies. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d’Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 14814–14825, 2019.
- Parisotto, E., Ba, L. J., and Salakhutdinov, R. Actor-mimic: Deep multitask and transfer reinforcement learning. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.
- Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021.
- Rakelly, K., Zhou, A., Finn, C., Levine, S., and Quillen, D. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5331–5340. PMLR, 2019.
- Sanchez-Gonzalez, A., Heess, N., Springenberg, J. T., Merel, J., Riedmiller, M. A., Hadsell, R., and Battaglia, P. W. Graph networks as learnable physics engines for inference and control. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4467–4476. PMLR, 2018.

- Scarselli, F., Yong, S. L., Gori, M., Hagenbuchner, M., Tsoi, A. C., and Maggini, M. Graph neural networks for ranking web pages. In *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, pp. 666–672, 2005. doi: 10.1109/WI.2005.67.
- Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320, Lille, France, 07–09 Jul 2015. PMLR.
- Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2, AAMAS '11*, pp. 761–768, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 0982657161.
- Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., Kim, N., Durme, B. V., Bowman, S., Das, D., and Pavlick, E. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*, 2019.
- Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017.
- Vig, J. and Belinkov, Y. Analyzing the structure of attention in a transformer language model. In Linzen, T., Chrupala, G., Belinkov, Y., and Hupkes, D. (eds.), *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@ACL 2019, Florence, Italy, August 1, 2019*, pp. 63–76. Association for Computational Linguistics, 2019. doi: 10.18653/v1/W19-4808.
- Vithayathil Varghese, N. and Mahmoud, Q. H. A survey of multi-task deep reinforcement learning. *Electronics*, 9(9), 2020. ISSN 2079-9292. doi: 10.3390/electronics9091363.
- Wang, T., Liao, R., Ba, J., and Fidler, S. Nervenet: Learning structured policy with graph neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

A. Hyperparameters

In this section, we describe the hyperparameters used with our model. These include hyperparameters in our model architecture, as well as hyperparameters for the TD3 optimizer that we use to optimize our policy. The hyperparameters for our model architecture can be found in the below table, while hyperparameters for TD3 can be found two tables below.

Table 2. Hyperparameters for our model architecture.

Hyperparameter Name	Hyperparameter Value
D (Token Embedding Size)	32
M (Sinusoidal Embedding Size)	96
Transformer Hidden Size	128
Transformer Feedforward Size	256
Attention Heads	2
Transformer Encoder Layers	3
Transformer Decoder Layers	3
Transformer Activation	relu
Dropout Rate	0.0

In the below table, we report the standard hyperparameters that are typically exposed to the user in TD3. Note that we employ the same TD3 hyperparameters across every task, which demonstrates our model does not require per-task tuning.

Table 3. Hyperparameters for our TD3 implementation.

Hyperparameter Name	Hyperparameter Value
Num Random Seeds	4
Batch Size	100
Max Episode Length	1000
Max Replay Size Total	10000000
Max Environment Steps	3000000
Policy Update Interval	2
Initial Exploration Steps	10000
Policy Noise	0.2
Policy Noise Clip	0.5
τ	0.046
σ	0.126
Discount Factor	0.99
Gradient Clipping	0.1
Learning Rate	0.00005

We use identical hyperparameters for the Amorpheus (Kurin et al., 2021) and Shared Modular Policies (Huang et al., 2020) baselines, except we update the learning rate to 0.0001, which we found to result in the best performance for both methods. We use the same reinforcement learning framework as Kurin et al. (2021) and Huang et al. (2020) for our experiments.

B. Details Of Robustness Experiment

In the main paper, we performed an experiment testing the robustness of our approach versus Amorpheus (Kurin et al., 2021) and Shared Modular Policies (Huang et al., 2020) baselines. In order to determine the order in which sensors "break," simulated by replacing sensor readings with random noise sampled from the standard normal distribution, we visualized the cross attention weights in the final decoder layer of our Transformer policy, and sorted the sensors in increasing order according to how frequently our policy attends to them throughout an episode. Specifically, we take the average value of the

final cross attention mask throughout an episode, and average again over the queries axis, in order to obtain a vector with $N_S(n)$ elements, representing the average attention weight applied to a given sensor by *any* actuator through an episode. In order to corrupt a particular fraction c of the agent’s sensors, we replace the first $\lceil c \cdot N_S(n) \rceil$ sensors with random noise, according to their sorted order as previously described. This methodology ensures that sensor corruptions are cumulative. That is, for two given fractions $c_1 > c_0$, the sensors corrupted by c_0 are also corrupted by c_1 .

C. Held Out Morphologies

In this section, we present a table that shows which morphologies are used for training and which morphologies are used for testing for each kind of agent. For tasks that mix multiple kinds of agents, the training morphologies for each kind are mixed, and the testing morphologies are mixed, but no training morphology becomes a testing morphology and vice versa.

Table 4. Morphologies used for training and testing.

Task	Training Morphologies	Testing Morphologies
Cheetahs	cheetah_2_back cheetah_2_front cheetah_3_back cheetah_3_front cheetah_4_allback cheetah_4_allfront cheetah_4_back cheetah_4_front cheetah_5_balanced cheetah_5_front cheetah_6_back cheetah_7_full	cheetah_3_balanced cheetah_5_back cheetah_6_front
Walkers	walker_2_main walker_4_main walker_5_main walker_7_main	walker_3_main walker_6_main
Humanoids	humanoid_2d_7_left_arm humanoid_2d_7_lower_arms humanoid_2d_7_right_arm humanoid_2d_7_right_leg humanoid_2d_8_left_knee humanoid_2d_9_full	humanoid_2d_7_left_leg humanoid_2d_8_right_knee
Hoppers	hopper_3 hopper_4 hopper_5	

D. Morphology Token Selection

We document the integer values selected for the morphology tokens used in our paper for every full morphology in the benchmark provided by (Huang et al., 2020). These full morphologies are `cheetah_7_full`, `walker_7_main`, `humanoid_2d_9_full` and `hopper_5`. Documenting morphology tokens for these is sufficient to reproduce the paper because, for any sub-morphology that has one or more limbs missing, such as `cheetah_6_back` which is missing the

limb titled *ffoot*, the morphology tokens corresponding to sensors and motors attached to the missing limb(s) are withheld.

Table 5. Morphology tokens for the observation spaces of every task. The variable i indicates that each of the agent’s limbs has all of the above sensors, and each limb’s sensor has a unique consequently increasing integer token value assigned to it. The range of token values provided illustrates that certain sensors produce vector observations, such as the position sensor, which produces three values.

Sensor Name	Token Integer Range
Limb[i] Position	$[18i + 0, 18i + 3)$
Limb[i] Orientation	$[18i + 3, 18i + 6)$
Limb[i] dPosition / dt	$[18i + 6, 18i + 9)$
Limb[i] dOrientation / dt	$[18i + 9, 18i + 12)$
Limb[i] Type	$[18i + 12, 18i + 16)$
Limb[i] Angle	$[18i + 16, 18i + 17)$
Limb[i] Joint Range	$[18i + 17, 18i + 18)$

Table 5 shows how observation tokens are selected for a single kind of morphology. In particular, the table shows how tokens are assigned for the *full* morphology in each family. For each sub-morphology that has one or more limbs missing, the sensor observations connected to that limb are missing from the agent’s total observation, and token values for that limb are similarly withheld. When selecting observation token values for sensors with multiple kinds of morphologies present, such as Humanoids and Walkers, we follow an identical process with an additional step of adding a predetermined constant to the observation token values of each morphology kind to ensure no collisions occur. For example, suppose the *humanoid_2d_9_full* morphology has m_0 consecutive observation tokens values assigned to it, starting from zero. We then add m_0 to every observation token value for the *walker_7_main* morphology. If the *walker_7_main* morphology has m_1 consecutive observation tokens values assigned to it, then combining these two kinds of morphologies results in $m_0 + m_1$ unique token values. This ensures no sensors share the same token value between kinds of morphologies.

Table 6. Morphology tokens for the action spaces of every task.

Motor Name	Token Integer Value	Motor Name	Token Integer Value
Cheetah FFoot	0	Walker Left1	0
Cheetah FShin	1	Walker Left2	1
Cheetah FThigh	2	Walker Left3	2
Cheetah BFoot	3	Walker Right1	3
Cheetah BShin	4	Walker Right2	4
Cheetah BThigh	5	Walker Right3	5
Motor Name	Token Integer Value	Motor Name	Token Integer Value
Humanoid Left Shoulder	0	Hopper Thigh	0
Humanoid Left Elbow	1	Hopper Leg	1
Humanoid Left Hip	2	Hopper Lower Leg	2
Humanoid Left Knee	3	Hopper Foot	3
Humanoid Right Shoulder	4		
Humanoid Right Elbow	5		
Humanoid Right Hip	6		
Humanoid Right Knee	7		

Table 6 shows how action tokens are selected for a single kind of morphology. We follow a similar procedure for selecting action tokens and observations tokens, which assigns consecutive integers token values to each motor in the *full* morphology for each family. For each sub-morphology that has one or more limbs missing, the motor actions connected to that limb are missing from the agent’s total action, and action token values for that limb are similarly withheld. When selecting action token values for motors with multiple kinds of morphologies present, such as Humanoids and Walkers, we follow

the previously described procedure for selecting observation tokens, where we add a predetermined constant (a different constant than for observation tokens) to the action token values of each morphology kind to ensure no collisions occur.

Our results show this procedure is effective for MuJoCo-like (Todorov et al., 2012) agents, but this procedure, in its simplicity, has several limitations. First, since no token collisions occur between kinds of morphologies, the token selection method described above does not support *out-of-domain* zero-shot generalization without modification. This is the kind of zero-shot generalization necessary when training our method on the Humanoid family of morphologies, and testing it on the Walker family of morphologies, without training our method on a single walker morphology. Notably, prior works in agent-agnostic RL, including (Huang et al., 2020) and (Kurin et al., 2021) only consider *in-domain* zero-shot generalization, such as training on non-empty subsets of both Humanoid morphologies and Walker morphologies, and testing on different subsets, allowing the policy to experience at least one example each of a humanoid and walker during training.

Evaluating our method on *out-of-domain* zero-shot generalization requires modifying how observation and action tokens are selected, to ensure the learned morphology representation can be transferred to novel morphology families. Effective token selection methods for *out-of-domain* zero-shot generalization likely need some degree of token collision to occur between morphology families. Further research into how to select the morphology tokens to enable *out-of-domain* zero-shot generalization is a promising research direction, and is a necessary step to apply our method to more realistic RL problems.