One Policy to Control Them All: Shared Modular Policies for Agent-Agnostic Control

Wenlong Huang [1] **Igor Mordatch** [2] **Deepak Pathak** [3 4]

Abstract

Reinforcement learning is typically concerned with learning control policies tailored to a particular agent. We investigate whether there exists a single global policy that can generalize to control a wide variety of agent morphologies – ones in which even dimensionality of state and action spaces changes. We propose to express this global policy as a collection of *identical* modular neural networks, dubbed as Shared Modular Policies (SMP), that correspond to each of the agent's actuators. Every module is only responsible for controlling its corresponding actuator and receives information from only its local sensors. In addition, messages are passed between modules, propagating information between distant modules. We show that a single modular policy can successfully generate locomotion behaviors for several planar agents with different skeletal structures such as monopod hoppers, quadrupeds, bipeds, and generalize to variants not seen during training – a process that would normally require training and manual hyperparameter tuning for each morphology. We observe that a wide variety of drastically diverse locomotion styles across morphologies as well as centralized coordination emerges via message passing between decentralized modules purely from the reinforcement learning objective.

1. Introduction

Deep reinforcement learning (RL) has been instrumental to successful sensorimotor control, either in simulation (Heess et al., 2017) or on physical robots (Levine et al., 2016). However, state-of-the-art approaches today train a policy network from scratch that is specifically tailored to a particular robot morphology (i.e., kinematic shape) and characteristics. But if we are to ever create general, pre-trainable priors for robot control similar to those for image classification (Krizhevsky et al., 2012) or natural language (Devlin et al., 2018), it is imperative for policies to be applicable to agents with differing morphologies.

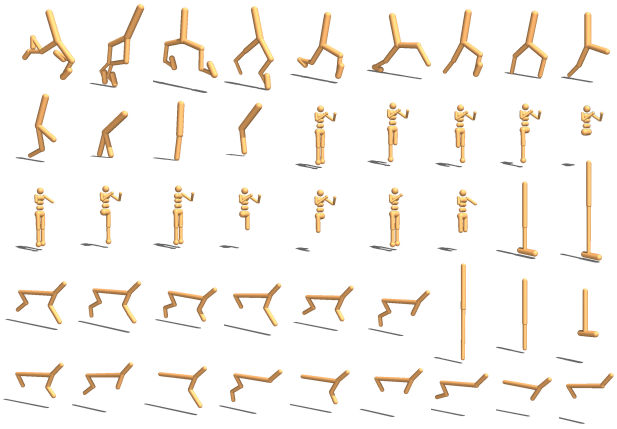

Figure 1. Our goal in this project is to train a single general-purpose policy controller that can perform well across many of these diverse agent morphologies. Our key idea is to represent policy as a collection of identical and locally communicating modular neural networks shared across all limbs of all agents. Video results at https://huangwl18.github.io/modular-rl/.

Can a general-purpose controller be pre-trained for multiple agents by simply reducing to a multi-task RL problem? This is not easy to manifest for several reasons. Although deep RL has been proven useful in making these agents learn from scratch without any priors, their success is limited to learning a separate controller for one agent at a time with tedious hyperparameter tuning (Henderson et al., 2017). Moreover, unlike pre-training of vision or language models, it is difficult to contemplate as to what does pre-training a controller mean for robots because each agent may have different number of limbs, sensory inputs, motor commands, morphology, and control behaviors, as shown in Figure 1.

Fortunately, our natural world is abound with examples of modularity and reuse in sensorimotor systems (d'Avella et al., 2015; Dickinson et al., 2000; Holmes et al., 2006). In

[1]UC Berkeley [2]Google [3]CMU [4]Facebook AI Research. Correspondence to: Deepak Pathak <dpathak@cs.cmu.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

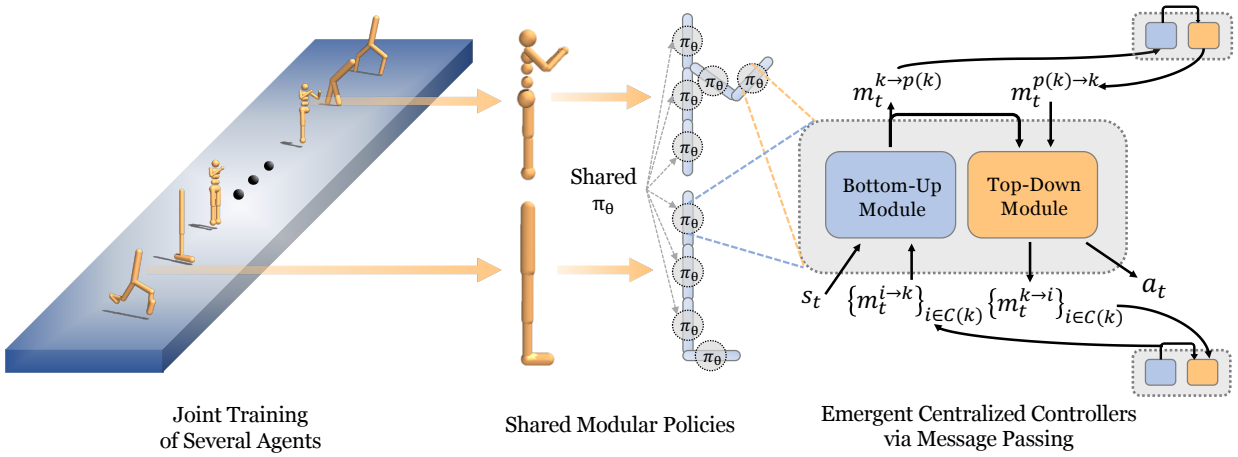

Figure 2. Overview of our approach: We investigate the possibility of learning general-purpose controllers by expressing agents as collections of modular components that use a shared control policy. Multiple agent controllers (left) are trained simultaneously with locally communicating modules with shared parameters (center). These modules learn to pass messages to local neighbors (right).

fact, an evidence for pre-trained controllers is seen in precocial and superprecocial animals that manage to fly or walk soon after birth, e.g. songbirds, horses, giraffe, etc. (Back & Clayton, 2013; Fox, 1964; Starck & Ricklefs, 1998). At the evolutionary level, modularity allows sensorimotor design motifs to only be developed once and reused across the organism's body and propagated efficiently to its descendants. At the level of organism's lifetime, modularity offers the tools of locality and parallel processing as a means to manage complexity. Sensorimotor units only sense and act locally, e.g., a motor neuron pool may only excite a particular muscle group and only receive information from sensors physically near that muscle group (Kandel et al., 2000).

In contrast, current RL policies are typically centralized and holistic objects that jointly output controls for all of the agent's actuators. A centralized and holistic artificial neural network policy misses an opportunity to exploit modularity and reuse advantages both at training and execution. Can we build artificial policies that simultaneously generalize to a wide variety of agents and exploit modularity and reuse?

To answer this question, we introduce *Shared Modular Policies* (SMP), a policy architecture built entirely out of a *single* reusable module that is re-instantiated at each of the agent's actuators. Each module instance only perceive information from the actuator's local sensors. What makes complex coordination between modules possible is a message passing procedure where each module receives and sends learned message vectors to its neighboring actuators – in our case neighboring limbs in the tree-structured morphology of the agent. The sensorimotor arrangement in SMP resembles a decentralized but communicating multi-agent population. Fascinatingly, such an arrangement makes it possible to orchestrate globally coherent, coordinated behaviors, such as locomotion for complex high-dimensional agents.

SMP is trained with standard policy gradient reinforcement learning as shown in Figure 2 and is able to generalize to control of variants not seen during training as we show in Section 5.3. This is a very hard task, as training a controller for only one of these morphologies is by itself a challenging task (Islam et al., 2017). The idea of sharing controllers across limbs has been investigated to control self-assembling agents in Pathak et al. (2019). Self-assemblies allow the flexibility to evolve agent morphologies to be easier to control, and hence, learn a *specialized*, yet generalizable policy. In contrast, our setup requires that the module must perform well across all morphologies to learn a *unified* policy. Such a requirement ensures we learn a policy that is truly appropriate to all agents. We find message passing – both top-down and bottom-up – to be crucial for successful operation and show that complex communication protocols emerge that transmit information across distant limbs despite only local connectivity as in show in Section 6.

Our contributions are as follows: firstly, we present a generalizable modular policy architecture appropriate for control of arbitrary agents. Secondly, we show that resulting policies can effectively control locomotion behaviors of several planar agents simultaneously and still match the performance of corresponding oracle, i.e., state-of-the-art method trained for the individual agents. Lastly, we analyze how centralized coordination can emerge from decentralized components in the context of sensorimotor control.

2. Learning General-Purpose Controllers

Consider N agents, each with a unique morphology. Each agent $n \in \{1 \dots N\}$ contains K_n different limb actuators which are connected together to constitute its overall morphological structure. Examples of such agents include half-

cheetah, 2D-humanoid, etc. agents with different physical structures (Figure 1), all geared towards a common goal of learning to walk. The objective is to learn a single, general-purpose controller that can simultaneously be trained on all these N agent morphologies via reinforcement learning and generalize to held-out morphologies in a zero-shot manner without any further finetuning. Our key insight is to employ modularity at the most fundamental level of sensorimotor learning loop, i.e., across limbs (i.e. actuators) of the agents.

2.1. Representing Agent Morphologies as Graphs

Consider a planar agent morphology that has K limbs, which we represent as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each node $v_i \in \mathcal{V}$ for $i \in \{1 \dots K\}$ represents a limb of the agent, and there is an undirected edge $(v_i, v_j) \in \mathcal{E}$ if v_i and v_j are connected by a joint. For the brevity of method description, let's assume that the graph is connected and acyclic (i.e., a tree) with the root node be one of the limbs, although it's easy to incorporate cycles as discussed in Section 3.3. Each node/limb thus contains an actuator that controls its movement relative to its parent node/limb.

2.2. Sensorimotor Modules

We develop a modular sensorimotor control policy $\pi_\theta(.)$ that is re-purposed to output the torques for each agent limb individually. The parameters θ of this module are *shared* across all the limbs $k \in \{1 \dots K_n\}$ of all agents $n \in \{1 \dots N\}$. At each discrete timestep t, the policy π_θ for the k^{th} limb of an agent n receives a local sensory state of the limb s_t^k as input and then outputs the torque values a_t^k for the corresponding actuator controlled by this limb. Upon executing the combined action $\{a_t^k\}_{k=1}^{K_n}$ for agent n at time t, the environment returns the next state $\{s_{t+1}^k\}_{k=1}^{K_n}$ corresponding to all individual limbs of the agent n and an overall reward for the whole morphology $r_t^n(\{s_{t+1}^k\}_{k=1}^{K_n}, \{a_t^k\}_{k=1}^{K_n})$. We now discuss the joint policy optimization and how the coordination emerges within each agent as a result of modularity.

2.3. Modular Policy Optimization

A straightforward way to implement a modular policy architecture is to train each limb's shared policy independently to optimize the joint reward function for whole morphology. Note that each limb has its own state-space containing positions, velocity, rotation etc., see Section 4 for details. The parameters θ of this policy π_θ are optimized to maximize the joint reward via deep reinforcement learning as follows:

$$\max_\theta \mathbb{E}_{\pi_\theta} \sum_{n=1}^{N} \sum_{t=0}^{\infty} \left[\gamma^t \, r_t^n \Big(\{s_{t+1}^k\}_{k=1}^{K_n}, \; \{a_t^k\}_{k=1}^{K_n} \Big) \right] \quad (1)$$

where $a_t^k = \pi_\theta(s_t^k)$ and γ is the discount factor.

In case of independent modular policies, this objective is optimized such that action is produced by a policy which is conditioned on the local state of the limb. This is similar in spirit to neural module networks used for visual question answering (Andreas et al., 2016) with the difference that our output is also modular and not just the input, i.e., each module directly outputs the limb actuation torques unlike in NLP where the output of all modules is aggregated to generate the answer.

We optimize this objective in Equation (1) via actor-critic setup of deterministic policy gradient algorithm which is standard practice for continuous control tasks (Lillicrap et al., 2016). In particular, we use the TD3 algorithm (Fujimoto et al., 2018). Note that we used an off-the-shelf implementation of TD3 without much change, and our method's ability to perform across diverse morphologies stems mostly from the modularity of proposed controllers.

3. Modular Communication

Learning a locomotion controller for walking across diverse agent morphologies, see Figure 2, is challenging for a pure reinforcement-based setup. First reason, of course, is the sheer complexity of hard joint optimization posed by this general setup. However, a bigger issue is the absence of a common gait that could perform locomotion with these agent morphologies. For instance, a bipedal walker can move efficiently with alternating walking gait while a walker with one leg (unipedal) will have to hop forward. A walker with one full leg and the other one short needs to lead with one leg and drag with the other and so on. Similar to the presence of different locomotion gaits across animals in the natural world, there exist many different locomotion gaits for our agents as different numbers of legs require different coordination with the torso and other non-locomotory limbs. Therefore, when the modular policies at each limb operate independently from other limbs, it is nearly impossible to represent different goal-directed behaviors (e.g. locomotion gaits) across different agent morphologies due to lack of ability to model coordination in absence of communication across limbs (actuators).

To facilitate limb coordination within an agent and represent different behaviors across agents, we propose to condition each limb's policy module π_θ on a message vector generated by its neighboring limbs in addition to conditioning on just the local state of the limb itself. Intuitively, we argue that the communication setup by these messages would enable the emergence of coherent full-body behaviors solely from identical local modules. Since our policies are fully shared, modular and communicate only with local neighbors via learned message vectors, we dub our approach as *Shared Modular Policies* (SMP).

3.1. Communication via Messages

As discussed in Section 2.1, we can represent each agent morphology as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The node $k \in K_n$ in the graph corresponds to the k^{th} limb and the edges denote the connectivity of limbs of agent n. Messages are passed along the edges between the neighboring nodes.

Let's define the torso as the root node for brevity, although any limb of the agent can act as root node of the graph and still achieve similar performance, as discussed in Section 5.4. Let $p(k)$ be the unique parent of the k^{th} node, $\mathcal{C}(k)$ be the set of its children nodes and $m_t^{i \to j}$ be the message from i^{th} node to j^{th} node. This message is a 32-dimensional learned vector generated by the limb policy.

The order in which these messages are passed governs prediction of each action a_t^k in Equation (1). We describe three message passing schemes: bottom-up, top-down, and both-way, in which the first two are decentralized while both-way can lead to the emergence of a centralized controller.

3.2. Decentralized Message Passing

The communication between our modules is naturally decentralized because we have only one type of module which gets shared across all limbs. In a decentralized setup, messages can be passed either from leaf nodes to the root node, or from root node to leaf nodes, discussed as follows.

Bottom-Up Message Passing Messages are passed from leaf nodes towards the root and the policy parameters θ are obtained by maximizing objective (1) under the following constraints for action generation:

$$a_t^k,\ m_t^{k \to p(k)} \ = \ \pi_\theta \Big(s_t^k,\ f\big(\{m_t^{i \to k}\}_{i \in \mathcal{C}(k)}\big) \Big) \tag{2}$$

where $f(.)$ is an aggregator function that collects all the messages from child nodes and combines them into a fixed dimension output. Examples of such functions include an element-wise sum, average or max operator, etc. We discuss alternatives to aggregation in Section 3.4.

Top-down Message Passing Messages are passed from the root node to leaves. For simplicity, let's assume that the parent nodes passes same message output to all of its children. The policy parameters θ are trained to optimize Equation (1) subject to following constraints on a_t^k:

$$a_t^k,\ m_t^{k \to c_k} \ = \ \pi_\theta \Big(s_t^k,\ m_t^{p(k) \to k} \Big) \tag{3}$$

where $m_t^{k \to c_k}$ is the message sent to all children nodes, i.e., $m_t^{k \to i} = m_t^{k \to c_k}\ \forall\ i \in \mathcal{C}(k)$. In many cases, passing a common message to all children may have issues for body-level coordination: for instance, if left and right legs have same state, then a common message won't be able to break the symmetry. To handle this, an alternative is to allow the parent to pass different messages to all its children via caching trick discussed in Section 3.4.

3.3. Both-way Message Passing: Emergence of Centralization

A purely decentralized controller should be sufficient when the diversity in morphologies is not huge and all the limb modules can converge to a *similar* whole-body strategy implicitly. However, in the presence of drastically different agents like humanoid and walker, some back-and-forth communication between modules is necessary to govern a consistent full-body behavior. Although, such centralization would have to emerge and can not be encoded because our reusable design does not permit any special module which can act as a 'master' node.

We allow centralization to emerge via **both-way message passing**: first from leaves to root, and then from root to leaves. The bottom-up pass generates only messages, and actions are predicted in the top-down pass. The root node can eventually learn to emerge as a centralized module that aggregates information from all other nodes and then pass on its information to others via messages. An intuitive way to understand this scheme is to draw an analogy with the central nervous system in animals where sensory neurons (upwards pass) carry information from end-effectors (leaves) to the brain (root) and then motor neurons (downward pass) carry instructions from the brain to generate output actions. To implement this, we divide our modular policy π_θ into two sub-policies with parameters θ_1 and θ_2 for the upwards pass and the downwards pass respectively. Parameters θ are trained to optimize Equation (1) subject to following constraints:

$$\begin{aligned} m_t^{k \to p(k)} \ &= \ \pi_{\theta_1} \Big(s_t^k,\ f\big(\{m_t^{i \to k}\}_{i \in \mathcal{C}(k)}\big) \Big) \\ a_t^k,\ m_t^{k \to c_k} \ &= \ \pi_{\theta_2} \Big(m_t^{k \to p(k)},\ m_t^{p(k) \to k} \Big) \\ m_t^{k \to i} \ &= \ m_t^{k \to c_k} \quad \forall\ i \in \mathcal{C}(k) \\ \theta \ &= \ \{\theta_1,\ \theta_2\} \end{aligned} \tag{4}$$

Note that the upwards pass occurs sequentially from leaf to root and only outputs a message passed to the parent. The downwards pass, in contrast, happens sequentially from root to leaf and generates the final action output and messages passed to children. The pseudo-code of the complete method is provided in the Appendix A.5.

If the morphological graph contains cycles, which however is rarely seen in the animal kingdom, the message passing can be generalized to perform multiple both-way (i.e., bottom-up and then top-down) message passing until the

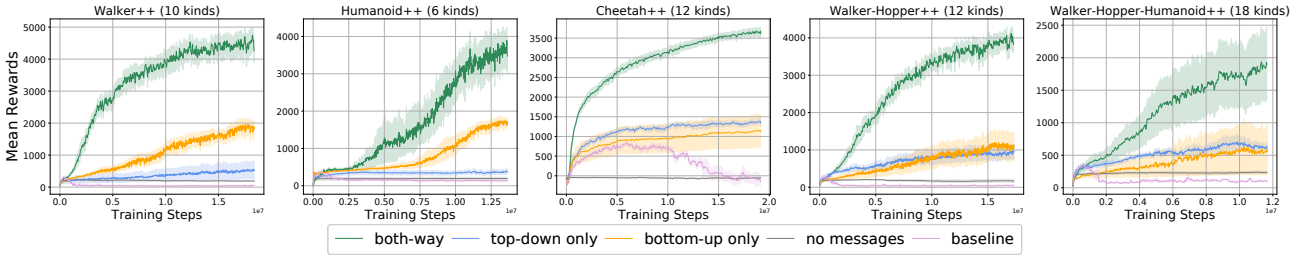

Figure 3. Average rewards across all agents in each of the five environments. For each environment discussed in Section 4, a single policy is simultaneously trained on multiple variants. Note that only 80% of the variants are used for training and the rest 20% are used for held-out testing. The figure shows that the multi-task baseline performs poorly on the environments containing difficult agents such as walker and humanoid, and it is also unstable in learning simpler agents like the cheetahs. In addition, the figure shows that different message passing has a pivotal impact on performance. Both-way message passing has a clear advantage in modeling diverse gaits across different agents compared to other decentralized schemes (e.g. top-down only and bottom-up only).

messages converge, as in loopy-belief-propagation (Murphy et al., 1999) for Bayesian networks with cycles.

3.4. Handling Different Number of Children Nodes

A parent node can have multiple children in an acyclic graph which poses a choice whether to pass the same or different messages to each child node. Section 3.2 described the scenarios when the same message is transmitted to all children nodes, which is not always optimal. For instance, when left and right legs are not symmetric and have different numbers of limbs, the torso would want to pass different latent 'instruction' to each leg. In our implementation, we allow different messages via a simple caching trick where the parent node in top-down pass always outputs as many messages as the max number of child nodes across joints of all agents, i.e., $max_n K_n$. If a certain joint has fewer children, the first few distinct messages are used by each child and the remaining ones are simply ignored. A similar idea is employed in the bottom-up pass to prevent loss of information in sum or average operation over messages from different children nodes. The bottom-up policy takes $max_n K_n$ number of messages, and if the number of actual children is fewer at some node, zero vector is appended to compensate. We found that, in practice, allowing different messages between each parent-child pair in this manner works better than passing the same message to all child nodes. A generic alternative to handle different messages across child nodes is to implement the aggregator function $f(.)$ as a recurrent neural network.

The emergence of complex coordination within agent limbs by local communication between shared modules has also been explored in dynamic graph networks (DGN) (Pathak et al., 2019). However, there are two key differences: (a) The agent shapes in our setup are static and not dynamic, thus, we do not allow the flexibility to dynamically adapt the physical morphology to make the controller learning easier. (b) Furthermore, our emphasis is on learning diverse control behaviors or gaits across these different static morphologies via different message passing mechanisms as discussed above. In contrast, in DGNs, no-message and bottom-up message passing is good enough for agents that are allowed to adjust their shape.

4. Experiment Setup

We investigate our proposed general-purpose controllers on the standard Gym MuJoCo locomotion tasks. We run all environments in parallel with the shared controller across limbs. Each experiment is run with four seeds to report the mean and the standard error. The reward for each environment is calculated as the sum of instant rewards across an episode of $1,000$ time-steps.

Environment and Agents We choose the following environments from Gym MuJoCo to evaluate our methods: 'Walker2D-v2', 'Humanoid-v2', 'Hopper-v2', HalfCheetah-v2'. To facilitate the study of general-purpose locomotion principles across these agents, we modify the standard 3D humanoid to constrain it to a 2D plane similar to walker, hopper, and cheetah.

To systematically investigate the proposed method when applied to multi-task training, we construct several variants of each of the above agents, as shown in Figure 2. We create the following collections of environments using these variants: (1) 12 variants of walker [walker++], (2) 8 variants of humanoid [humanoid++], (3) 15 variants of cheetah [cheetah++], (4) all 12 variants of walker and 3 variants of hopper [walker-hopper++], and (5) all 12 variants of walker, 3 variants of hopper, and all 8 variants of humanoid [walker-hopper-humanoid++]. We keep 20% of the variants as the held-out set and use the rest for training. Note we do not solely evaluate the methods on the hopper environment because there are only three variants possible. And we do not perform cross-category training with the cheetah environment because it uses a different integrator, making it

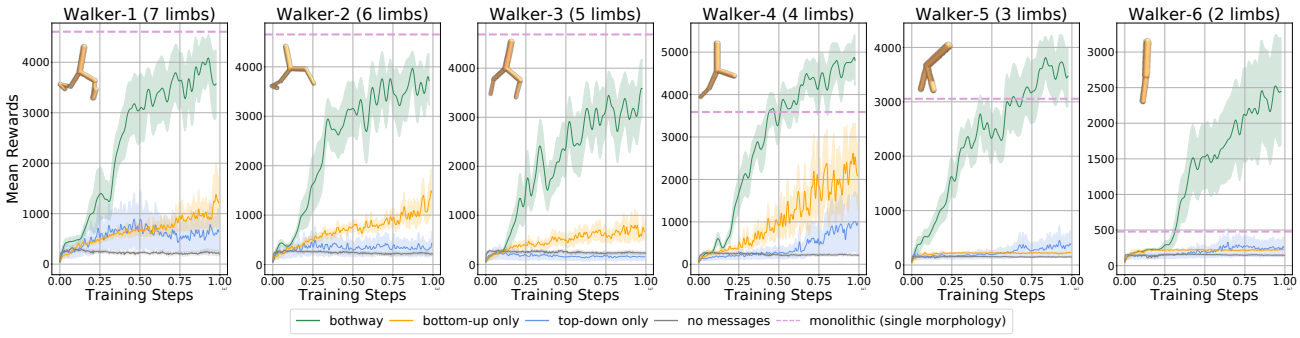

Figure 4. Comparisons between different message passing schemes on walker morphologies. The policy is jointly trained on ten walker variants and the figure shows a subset of six. Decentralized message passing schemes (e.g. top-down only or bottom-up only) can learn locomotion task for walkers with four to seven limbs to some extent, but fail to learn anything meaningful for three-limb and two-limb variants. In contrast, both-way message passing can model multiple gaits and demonstrates a clear advantage. Video results at https://huangwl18.github.io/modular-rl/.

unstable when jointly trained with other environments.

To create the variants for each agent, consider each agent as a tree with the root being the torso. We create all possible subsets (the power set) of all the nodes in the tree and keeping only those that contain the torso and form connected graphs. This can also be done by procedurally removing one leaf node at a time and enumerating all possible combinations. Note that we leave out those variants that are structurally infeasible for locomotion (e.g. humanoid without legs) in training and testing.

States and Actions The total state space of agent n, $\{s_t^k\}_{k=1}^{K_n}$, is a collection of local limb states. Each of these limb states, s_t^k, contain global positions, positional velocities, rotational velocities, 3D rotations, and range of movement of the limb body. We represent these 3D rotations via an exponential map representation (Grassia, 1998). The range of movement is represented as three scalar numbers $(position_t, low, high)$ normalized to $[0, 1]$, where $position_t$ is the joint position at time t, and $[low, high]$ is the allowed joint range. To handle different numbers of children nodes, we implement the simple caching trick discussed in Section 3.4. Note that the torso limb has no actuator in any of these environments, so we still keep a sensorimotor module for torso for message passing but ignore its predicted torque values.

We use TD3 (Fujimoto et al., 2018) as the underlying reinforcement learning method. The internal modules which are shared across all limbs of over 20 agents are just two 4-layered fully-connected neural networks with ReLU and tanh non-linearity, one for bottom-up message passing and the other one for top-down message passing. The dimension of message vectors is 32. Other details of training and a sanity check section that compares the Shared Modular Policies to a standard monolithic policy trained on single-agent environments can be found in the appendix.

5. Results and Ablations

We evaluate the effectiveness of our approach by asking three questions: Can our Shared Modular Policies (SMP) outperform the standard multi-task RL approach when simultaneously trained on many diverse agents? How do different message passing schemes compare and does centralized control emerge? Can it generalize to unseen morphologies in a zero-shot manner, a task that has been considered infeasible for the standard RL approach? We examine these questions in three steps:

- We first compare against the standard multi-task baseline and see how well our proposed method compares to such a monolithic policy simultaneously trained on multiple agents.
- Next we examine the role of message passing, specifically the performance resulted from different message passing schemes.
- Finally we test our learned modular policy on unseen agent morphologies in a zero-shot manner.

Also, we examine whether Shared Modular Policies are robust to the choice of root node while constructing the kinematic graph for message passing by choosing non-torso limbs as the root.

5.1. Multi-Task RL Baseline

Following the setup by Chen et al. (2018), the baseline that we compare to is a standard monolithic RL policy trained on all environments with TD3. The state space for each environment consists of the state of the agent in joint-coordinate (as in most existing methods) plus a task descriptor containing the number of limbs present and a one-hot environment ID. The policy is a four-layered fully-connected neural network. For each agent, it takes the state of the entire agent as input and outputs the continuous torque values for all the

Figure 5. Zero-Shot Generalization: Average rewards across held-out morphologies for different message passing schemes. The policy in each plot is trained jointly on 80% of variants from that environment and tested on the 20% unseen variants in a zero-shot manner. We take the trained SMP policies from each timestep during training (x-axis) and test on the held-out set. SMP with both-way message passing generalizes without troubles to unseen environments, demonstrating it has learned important priors for locomotion.

actuators. Note that the dimensions of the state space and the action space differ across different environment, so we zero-pad the states and actions to the maximum dimension across all environments.

As shown in Figure 3, the multi-task baseline fails to perform well in any environment, possibly due to the diversity of the agents and hence the difficulty of learning a single controller for all the agents. In contrast, SMP with both-way message passing can model many different gaits across these drastically different agents.

5.2. Role of Message Passing

As shown in Figure 3, different schemes of message passing have a significant impact on the performance of the morphologies. Not only does the both-way message passing scheme outperforms the multi-task RL baseline, but it performs significantly better than the decentralized message passing schemes (e.g. top-down only and bottom-up only).

Figure 3 shows the superiority of both-way message passing in obtaining higher average rewards across a number of agents, yet it does not show in what ways both-way message passing is superior than decentralized message passing schemes. To investigate this, we plot one figure for each morphology in Walker++, where all the agent morphologies are trained with a single policy. As shown in Figure 4, although decentralized message passing schemes seem to work in few morphologies, they fail to model different types of motion as these morphologies exhibit drastically different gaits (e.g. a two-limb walker can only hop forward). Both-way message passing, on the other hand, learns these gaits simultaneously, a task that is even infeasible by the formulation of most RL methods.

5.3. Zero-Shot Generalization

There are several examples in animal kingdom where locomotion abilities are present at birth (i.e. almost 'zero-shot'), for instance, foals start to walk soon after they are born (Back & Clayton, 2013; Fox, 1964). Similarly, our goal of learning a general-purpose controller is not limited to training morphologies but also to generalize to new ones in a zero-shot manner without any further training.

During test time, the modular policies can potentially adapt to many morphological structures, and in this section, we test the trained policy on a set of held-out agent morphologies. As shown in Figure 5, both-way message passing has a definitive advantage in generalization, achieving high rewards even in a zero-shot manner. This demonstrates that it can generalize to a wide variety of different morphologies with no fine-tuning, showing it has learned important priors for locomotion – a key step towards learning general-purpose controllers. Please look at the success as well as the failure videos on the project website [1].

5.4. Training with a Non-Torso Limb as Root

As our method operates on a per-actuator level, it relies on the graphical representation of an agent's morphological structure, which is often a tree. Most MuJoCo environments come with such morphological structures by defining agents as an acyclic graph where torso is the root. In all the experiments in the previous sections, we simply adopt the built-in structure for each environment. However, we note that our method is agnostic to where the root is defined. To verify this, we construct another walker environment where the root is the left foot instead of the torso. We run four different seeds for the same walker morphology with this foot-root setup, the default torso-root setup, and the monolithic baseline. We report the mean and standard deviation of training rewards at 1M timesteps. Note that treating left foot as root even performs slightly better.

Method	Training Reward
Ours (both-way) + root is left foot	**3709.87** $\pm$ 580.87
Ours (both-way) + root is torso	3215.04 $\pm$ 447.82
Monolithic Baseline	3592.70 $\pm$ 111.13

[1] https://huangwl18.github.io/modular-rl/

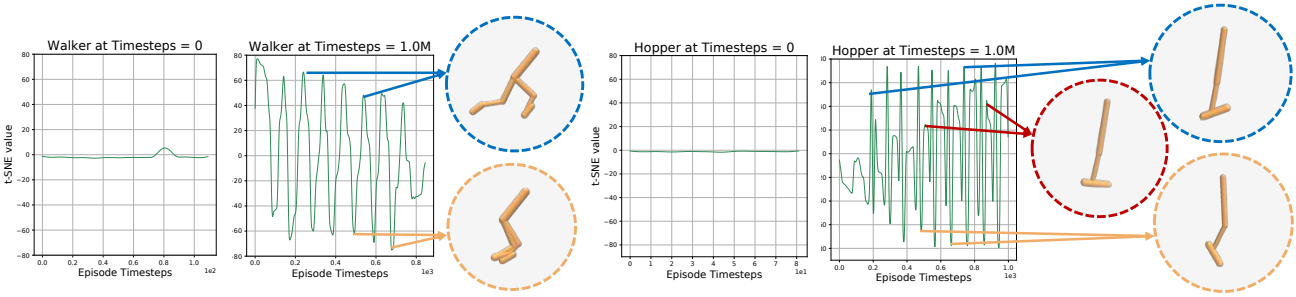

Figure 6. We investigate whether messages capture the alternating gait corresponding to locomotion behaviors. For both walker and hopper environment, we run t-SNE to plot the top-down root node messages against timesteps in an episode. The cyclic pattern show that the root node message not only captures such alternating gait but also plays an important role in governing the overall agent pose.

6. Analysis of Message Passing

Illustrated by Section 5.2, message passing plays a crucial role for agents to orchestrate globally coherent behaviors. However, does message passing convey contextual information essential for learning general-purpose controllers or is it purely an empowering technique for modeling high-complexity tasks? We answer this question by examining the role of message passing in this section.

Consistency over Time In many of the locomotion tasks, we repeatedly observe alternating behaviors, a result of global coordination, e.g. walker moves by alternating its two legs and hopper hops by contracting and relaxing its leg. Do our learned messages capture this essence of locomotion? We investigate this question by plotting one-dimensional t-SNE (Maaten & Hinton, 2008) of the torso message, which has aggregated global information after bottom-up message passing, over the time of an episode. As shown in Figure 6, a clear message pattern emerges over the course of training. Furthermore, we visualize the agent across the episode time-steps and found that the agent's pose is also highly consistent with the torso message, again proving that a centralized controller can emerge from training decentralized controllers via message passing.

7. Related Works

Modular approaches to control that are similar to ours have been explored by robotics and virtual evolution communities. To control customizable and reconfigurable robot platforms, Chen et al. (2018); Schaff et al. (2018) condition the control policy on an encoding of the robot's morphology. Ha et al. (2017) avoid learning a parametric control policy altogether and instead use trajectory optimization to control the robots. When the morphology of the robot is fixed but some pre-determined parameters vary, meta-learning can be used to adapt the policy online (Al-Shedivat et al., 2017; Nagabandi et al., 2018) or to train with variability over parameters to make the control policy insensitive to their precise value (Akkaya et al., 2019). Virtual evolution similarly requires co-adaptation of the morphology and the control mechanism. Advantages of modular control have been observed in this context by Cheney et al. (2014); Pathak et al. (2019); Sims (1994); Wang et al. (2019).

Another recent line of work exploiting modularity and reuse in deep learning are graph-structured neural networks (Scarselli et al., 2009) – see (Battaglia et al., 2018) for a comprehensive review. Global coordination in such graph networks is either implemented via global aggregation or decentralized message passing, as in (Gilmer et al., 2017; Zhang et al., 2019). In deep reinforcement learning, graph structure has typically been used to efficiently encode agent's observations (i.e. world entities and interactions) as in (Baker et al., 2019; Sanchez-Gonzalez et al., 2018). Exceptions are works of Pathak et al. (2019); Wang et al. (2018), which similarly to our work exploit graph structure present in the agent's morphology.

Our message passing modules also bear resemblance to a communicating multi-agent system. Global coordination emerging from decentralized agents was observed from deep reinforcement learning agents in (Foerster et al., 2016; Mordatch & Abbeel, 2018; Sukhbaatar et al., 2016). Modularity has also been observed to an important component of a biological sensorimotor organization – see (d'Avella et al., 2015) for a review. Examples of global coordination observed in this area include central pattern generators for control of rhythmic behaviors (Marder & Bucher, 2001) and muscle synergies (d'Avella et al., 2003).

8. Conclusion

In this work, we presented *Shared Modular Policies*, an architecture built entirely out of a single reusable module – that while acting and sensing only locally creates globally-coordinated complex movement behaviors. Such an architecture can produce locomotion for a wide variety of agents simultaneously, even those not seen during training. Overall, we hope that our work provides the foundation for general-purpose pre-trained priors of sensorimotor control.

Acknowledgments

We would like to thank Alyosha Efros, Yann LeCun, Jitendra Malik, Pieter Abbeel, Hang Gao and the members of BAIR community for fruitful discussions. This work was supported in part by Google faculty research award.

References

Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019. 8

Al-Shedivat, M., Bansal, T., Burda, Y., Sutskever, I., Mordatch, I., and Abbeel, P. Continuous adaptation via meta-learning in nonstationary and competitive environments. *arXiv preprint arXiv:1710.03641*, 2017. 8

Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. Neural module networks. In *CVPR*, 2016. 3

Back, W. and Clayton, H. M. *Equine Locomotion-E-Book*. Elsevier Health Sciences, 2013. 2, 7

Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., and Mordatch, I. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019. 8

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018. 8

Chen, T., Murali, A., and Gupta, A. Hardware conditioned policies for multi-robot transfer learning. In *NIPS*, 2018. 6, 8

Cheney, N., MacCurdy, R., Clune, J., and Lipson, H. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. *ACM SIGEVOlution*, 2014. 8

d'Avella, A., Saltiel, P., and Bizzi, E. Combinations of muscle synergies in the construction of a natural motor behavior. *Nature neuroscience*, 2003. 8

d'Avella, A., Giese, M., Ivanenko, Y. P., Schack, T., and Flash, T. Modularity in motor control: from muscle synergies to cognitive action representation. *Frontiers in computational neuroscience*, 2015. 1, 8

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P. Openai baselines. `https://github.com/openai/baselines`, 2017. 11

Dickinson, M. H., Farley, C. T., Full, R. J., Koehl, M., Kram, R., and Lehman, S. How animals move: an integrative view. *Science*, 2000. 1

Foerster, J., Assael, I. A., De Freitas, N., and Whiteson, S. Learning to communicate with deep multi-agent reinforcement learning. In *NIPS*, 2016. 8

Fox, M. A phylogenetic analysis of behavioral neuroontogeny in precocial and nonprecocial mammals. *Canadian journal of comparative medicine and veterinary science*, 1964. 2, 7

Fujimoto, S., Van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. *ICML*, 2018. 3, 6, 11

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017. 8

Grassia, F. S. Practical parameterization of rotations using the exponential map. *Journal of graphics tools*, 1998. 6

Ha, S., Coros, S., Alspach, A., Kim, J., and Yamane, K. Joint optimization of robot design and motion parameters using the implicit function theorem. In *RSS*, 2017. 8

Heess, N., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, A., Riedmiller, M., et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017. 1

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017. 1

Holmes, P., Full, R. J., Koditschek, D., and Guckenheimer, J. The dynamics of legged locomotion: Models, analyses, and challenges. *SIAM review*, 2006. 1

Islam, R., Henderson, P., Gomrokchi, M., and Precup, D. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133*, 2017. 2

Kandel, E. R., Schwartz, J. H., Jessell, T. M., of Biochemistry, D., Jessell, M. B. T., Siegelbaum, S., and Hudspeth, A. *Principles of neural science*. McGraw-hill New York, 2000. 2

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *ICLR*, 2015. 11

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 2016. 1

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *ICLR*, 2016. 3

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 2008. 8

Marder, E. and Bucher, D. Central pattern generators and the control of rhythmic movements. *Current biology*, 2001. 8

Mordatch, I. and Abbeel, P. Emergence of grounded compositional language in multi-agent populations. In *AAAI*, 2018. 8

Murphy, K. P., Weiss, Y., and Jordan, M. I. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, 1999. 5

Nagabandi, A., Finn, C., and Levine, S. Deep online learning via meta-learning: Continual adaptation for model-based rl. *arXiv preprint arXiv:1812.07671*, 2018. 8

Pathak, D., Lu, C., Darrell, T., Isola, P., and Efros, A. Learning to control self-assembling morphologies: A study of generalization via modularity. In *NeurIPS*, 2019. 2, 5, 8

Polosukhin, I. and Zavershynskyi, M. nearai/torchfold: v0.1.0, Jun 2018. URL https://doi.org/10.5281/zenodo.1299387. 11

Sanchez-Gonzalez, A., Heess, N., Springenberg, J. T., Merel, J., Riedmiller, M., Hadsell, R., and Battaglia, P. Graph networks as learnable physics engines for inference and control. *arXiv preprint arXiv:1806.01242*, 2018. 8

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Network*, 2009. 8

Schaff, C., Yunis, D., Chakrabarti, A., and Walter, M. R. Jointly learning to construct and control agents using deep reinforcement learning. *arXiv preprint arXiv:1801.01432*, 2018. 8

Sims, K. Evolving virtual creatures. In *Computer graphics and interactive techniques*, 1994. 8

Starck, J. M. and Ricklefs, R. E. Patterns of development: the altricial-precocial spectrum. *Oxford Ornithology Series*, 1998. 2

Sukhbaatar, S., Fergus, R., et al. Learning multiagent communication with backpropagation. In *NIPS*, 2016. 8

Wang, T., Liao, R., Ba, J., and Fidler, S. Nervenet: Learning structured policy with graph neural networks. *ICLR*, 2018. 8

Wang, T., Zhou, Y., Fidler, S., and Ba, J. Neural graph evolution: Towards efficient automatic robot design. *arXiv preprint arXiv:1906.05370*, 2019. 8

Zhang, L., Xu, D., Arnab, A., and Torr, P. H. Dynamic graph message passing networks. *arXiv preprint arXiv:1908.06955*, 2019. 8

A. Appendix

A.1. Result Videos

We show videos for all variants trained with a single policy on the project website: https://huangwl18.github.io/modular-rl/. We recommended referring to videos to observe how our single 4-layer network policy can represent different gate behaviors across different agent morphologies. One way message passing is sometimes able to learn for more than one morphology, but cannot represent multiple gates. However, both-way message passing is able to represent multiple gates due to *emergence of centralization from decentralized modules*.

A.2. Implementation and Training

We use TD3 (Fujimoto et al., 2018) as the underlying reinforcement learning method. The internal optimizer for TD3 is Adam (Kingma & Ba, 2015). The initial positions and velocities of each agent are randomized at the beginning of each episode. For the first $10,000$ time-steps during training, actions are uniformly sampled from the action space. The policy is trained with a learning rate of $4\mathrm{e}{-4}$, a tau of 0.046, and a exploration noise of 0.13. All internal modules are 4-layered fully-connected neural networks with ReLU and tanh non-linearity. The dimension of message vectors are 32. Messages are normalized before passed to the children and the parent.

For multi-morphology training, each morphology has its own environment and an independent replay buffer of size $1e6$. The maximum size of all the replay buffers is capped at $1e7$ and whenever there are $n > 10$ environments, each environment has a replay buffer of size of $1e7/n$. We run all environments in parrallel using vectorized environment from OpenAI Baselines (Dhariwal et al., 2017). To speed up training and inference, we also use the dynamic batching package (Polosukhin & Zavershynskyi, 2018) when there is no dependency between modules, e.g. when the limbs are neither an ancestor nor descendent of each other.

For each single-category training (walker++, humanoid++, hopper++, and cheetah++), we use its default reward function from Gym. For multi-category training (walker-hopper++ and walker-hopper-humanoid++), we use the reward function from walker++ which consists of x-axis displacement (distance covered by agent), alive reward, and sum of squared actions (for penalizing large action values).

A.3. Analysis of Message Propagation Range

In both-way message passing, messages are initially passed from the leaves to the root and then from the root back to the leaves. We here investigate whether the messages convey global information by showing the correlation between the states, actions, and the messages passed at different levels. Specifically, we test in the walker environment and we investigate how much the states and the actions are correlated with the furthest messages to the closest messages. Due to the different dimensionality, we first reduce the dimensions of the data to one by running Principle Component Analysis (PCA), and the final results are averaged over an episode. As shown in Figure 7, both the leaves' states and the actions are more correlated with the closer messages passed to them, demonstrating that messages are indeed conveying meaningful contextual information for locomotion.

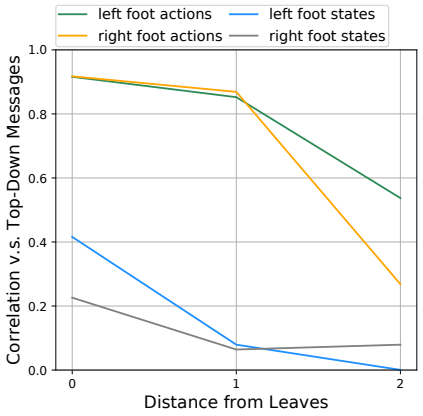

Figure 7. Message Range Analysis. We examine how far the messages reach and what relationship it has with the leaves' local states and actions by doing correlation analysis between leaves' local states/actions and the messages from its predecessors. The figure shows the closer a limb is to a leaf, the more its message contains the 'instructions' for the leaf.

A.4. Sanity Check Experiment

We perform a sanity check on the single-agent environment to see if a modular policy can learn as well as a monolithic one based off the message passing formulation discussed in Section 3. It can be seen from Figure 3 that a good coordination between limbs is of the utmost importance for control because under the no message setting, all limbs act as independent agents and cannot coordinate with each other and thus cannot learn anything meaningful. This shows that a modular policy is certainly at a disadvantage when only trained and tested on a single morphology against a monolithic policy, since the latter is a much easier optimization problem and does not need to learn to generate messages just to coordinate limbs. In contrast a modular policy has to learn message passing as well as the controller to generate meaningful behaviors. Despite the challenges, we observe that, as shown in Figure 8, the Shared Modular Policies with both-way message-passing can achieve comparable performance to a monolithic baseline in all experiments.

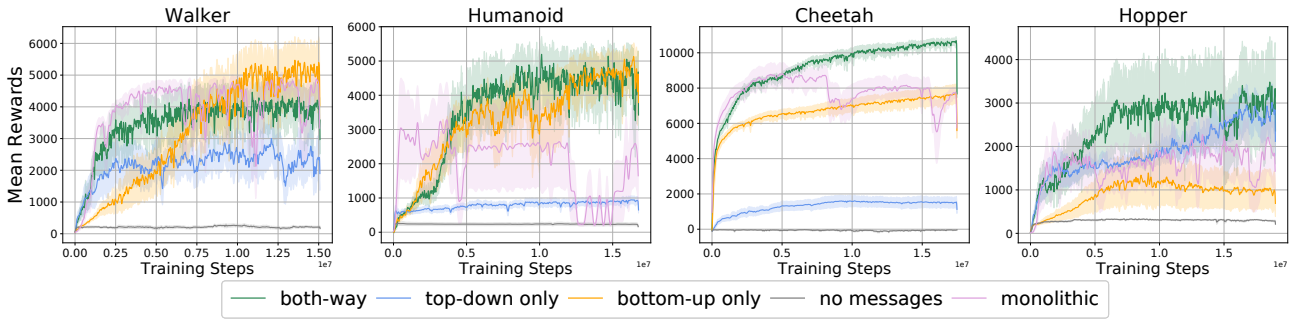

Figure 8. A sanity check that compares the average rewards on standard environments by the Shared Modular Policies with different message-passing schemes and the monolithic baseline. All the policies are only trained with one agent morphology at a time. The figure shows that the Shared Modular Policies with both-way message-passing can model multiple simple locomotion tasks just as well as a monolithic baseline.

A.5. Pseudo-Code for Shared Modular Policies

We provide the pseudo-code for training shared modular policies with both-way message passing. Algorithm 1 discusses the joint setup of sharing policies across motors of all agents, and Algorithm 2 discusses the end-to-end training.

Algorithm 1 Joint Training of All Agents

Notation Summary:
NN_{bu}: bottom-up module parameterized by θ_1
NN_{td}: top-down module parameterized by θ_2
$s_t^{(e)}$: all limbs' states of environment e at time t
$a_t^{(e)}$: all limbs' actions of environment e at time t
$r_t^{(e)}$: rewards of environment e at time t
$done^{(e)}$: whether environment e is done
$rb^{(e)}$: replay buffer for environment e

init: SMP = (NN_{bu}, NN_{td}) from scratch.
    empty replay buffer $rb^{(e)}$ for each environment e.
while not converged **do**
  // collect one episode of data for all environments
  for all environment e **do**
    while e is not done **do**
      $a_t^{(e)} \leftarrow \text{SMP}(s_t^{(e)})$
      $s_{t+1}^{(e)}, r_t^{(e)}, done^{(e)} \leftarrow \text{simulate}(e, a_t^{(e)})$
      Add $(s_t^{(e)}, s_{t+1}^{(e)}, a_t^{(e)}, r_t^{(e)}, done^{(e)})$ to $rb^{(e)}$
    end while
  end for
  // train SMP for each environment one by one
  for all environment e **do**
    SMP $\leftarrow$ trainWithTD3(SMP, $rb^{(e)}$)
  end for
end while

Algorithm 2 Both-way Shared Modular Policies (SMP)

Notation Summary:
NN_{bu}: bottom-up module parameterized by θ_1
NN_{td}: top-down module parameterized by θ_2
s_i: local states of limb i
a_i: local action of limb i
$m^{i \to p(i)}$: message passed from limb i to the parent of i
$m^{p \to i}$: message passed from the parent of limb i to i
$\{m^{c \to i}\}_{c \in \mathcal{C}(i)}$: the set of all messages passed from the children of limb i to i
$\{m^{i \to c}\}_{c \in \mathcal{C}(i)}$: the set of different messages passed from limb i to each of its children
Input: environment e and all limbs' states $\{s_i\}_{i=1}^k$
Output: actions for all limbs of that agent $\{a_i\}_{i=1}^k$

// get agent limbs in topological order (root to leaf)
nodeList $\leftarrow$ topologicalOrdering(e)
// dynamically change the policy's graph structure to match that of the agent
SMP $\leftarrow$ changeGraph(SMP, nodeList)
// bottom-up message passing (leaf to root)
for node i in reversed(nodeList) **do**
  if i is leaf **then**
    $m^{i \to p(i)} \leftarrow NN_{bu}(s_i, \vec{0})$
  else
    $m^{i \to p(i)} \leftarrow NN_{bu}(s_i, \{m^{c \to i}\}_{c \in \mathcal{C}(i)})$
  end if
end for
// top-down message passing (root to leaf)
for node i in nodeList **do**
  if i is root **then**
    $a_i, \{m^{i \to c}\}_{c \in \mathcal{C}(i)} \leftarrow NN_{td}(m^{i \to p(i)}, \vec{0})$
  else
    $a_i, \{m^{i \to c}\}_{c \in \mathcal{C}(i)} \leftarrow NN_{td}(m^{i \to p(i)}, m^{p(i) \to i})$
  end if
end for
return $\{a_i\}_{i=1}^k$