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 policy that generalizes to
controlling a wide variety of agent morphologies – ones in which even dimensionality of state and
action spaces changes. Such a policy would distill general and modular sensorimotor patterns
that can be applied to control arbitrary agents. We propose a policy expressed as a collection
of identical modular neural networks for each
of the agent’s actuators. Every module is only
responsible for controlling its own actuator and
receives information from its local sensors. In
addition, messages are passed between modules,
propagating information between distant modules.
A single modular policy can successfully generate
locomotion behaviors for over 20 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 a wide variety of drastically diverse
locomotion styles across morphologies as well
as centralized coordination emerging 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, the majority of these efforts have been concerned
with training a policy network specifically tailored to a par-
cular agent morphology and characteristics. But if we are
to ever create general, pre-trainable priors for movement
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.
Can a general-purpose controller be learned for multiple
agents by simply reducing to a multi-task RL problem? This
is not easy to manifest for several reasons. First, 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 time with
tedious hyperparameter tuning (Henderson et al., 2017).
Secondly, it is difficult to incorporate varying dimensionality
of output actions and input observations – such as those in
Figure 2 – in a standard deep RL setting.
Fortunately, the natural world is abound with examples of
modularity and reuse in sensorimotor systems (d’Avella
et al., 2015). At the evolutionary level, modularity allows
sensorimotor design motifs to only be developed once and
reused across the organism’s body and propagated efficiently

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 leverage modularity to learn locally communicating modules that
share the parameters across all limbs of all agents. Video results at https://huangwl18.github.io/modular-rl/.
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 (for example, 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 above 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 a policy architecture that is built entirely out of a single reusable module that is re-instantiated at each of the agent’s individual actuators. Each module instance only receives information from the actuator’s local sensors. What makes complex coordination between modules possible is a message passing procedure where each module sends and receives arbitrary message vectors to its neighboring actuators – in our case neighboring limbs in the tree-structured morphology of the agent. Our agent’s sensorimotor system arrangement 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.

Over 20 different morphologies shown in Figure 2 are able to perform effective locomotion with our reusable neural network module. The modular policy is trained with standard deterministic policy gradient reinforcement learning but it 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). We require that the module must perform well across all morphologies – we do not allow the flexibility to evolve morphologies to be easier to control such as in the recent work of Pathak et al. (2019). 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 be used to control locomotion behaviors of over 20 agents simultaneously, reaching the performance of methods tailored to training 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, acting in an environment $\mathcal{E}$. Each agent $n \in \{1 \ldots N\}$ contains $K_n$ different limb actuators which are connected together to constitute its overall morphological structure. Examples of such agents, as shown in Figure 2, include half-cheetah, humanoid etc. with different physical structures but 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 (RL) and generalize to held-out morphologies in a zero-shot manner without any further
We develop a modular sensorimotor control policy \( \pi_{\theta} \) where \( \gamma \) 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, i.e., \( a^k_t = \pi_{\theta}(s^k_t) \) for limb \( k \) of agent \( n \).

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. 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. Although independent modular policies can learn limb-specific control within an agent, as discussed in the previous section, 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 between limbs.

To facilitate limb coordination within an agent and represent different behaviors across agents, we propose to condition each limb’s policy module \( \pi_{\theta} \) on a message vector generated by its neighboring limbs in addition to conditioning on just the local state of the limb itself. Since our policies are modular and act on the learned message vectors received from local neighbors, we dub our approach as Shared Modular Policies (SMP). Intuitively, our hope is that the communication setup by these messages would enable the emergence of coherent full-body behaviors solely from identical local modules.

### 2.2. Sensorimotor Modules

We develop a modular sensorimotor control policy \( \pi_{\theta}(\cdot) \) that is repurposed to output the torques for each agent limb individually. The parameters \( \theta \) of this module are shared across every limb \( k \in \{1 \ldots K_n\} \) of each agent \( n \in \{1 \ldots N\} \). At each discrete timestep \( t \), the policy \( \pi_{\theta} \) for the \( k^{th} \) limb of an agent \( n \) receives a local sensory state of the limb \( s^k_t \) as input and then outputs the torque values \( a^k_t \) for the corresponding actuator controlled by this limb. Upon executing the combined action \( \{a^k_{t_i}\}_{i=1}^{K_n} \) for agent \( n \) at time \( t \), the environment returns the next state \( \{s^k_{t+1}\}_{i=1}^{K_n} \) corresponding to all individual limbs of the agent \( n \) and an overall reward for the whole morphology \( r^m_n(\{s^k_{t+1}\}_{i=1}^{K_n}, \{a^k_{t_i}\}_{i=1}^{K_n}) \). 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. Therefore, even in the absence of direct communication between limbs, an independent modular policy with shared parameters can still learn to output the needed torque because it is conditioned on the state of the limb as input. This policy \( \pi_{\theta} \) is represented by a neural network and parameters \( \theta \) are optimized to maximize the joint reward via deep reinforcement learning as follows:

\[
\max_{\theta} \mathbb{E}_{\pi_{\theta}} \sum_{n=1}^{N} \sum_{t=0}^{\infty} \gamma^t r^m_n(\{s^k_{t+1}\}_{i=1}^{K_n}, \{a^k_{t_i}\}_{i=1}^{K_n}) \tag{1}
\]

where \( \gamma \) is the discount factor. In case of independent modular policies, this objective is optimized such that action...
3.1. Communication via Messages

For the brevity of method description, let’s assume that the morphological structure of all the agents constitutes a connected acyclic graph (i.e., a tree), although it’s easy to incorporate cycles as discussed later. For each agent \( n \), there is a corresponding graph \( G_n \) with \( K_n \) nodes. Although, this morphological graph is undirected but let’s consider any one particular (out of many) topological ordering of nodes in the tree to identify leaf nodes and a root. 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 \( n \). Let \( p(k) \) be the unique parent of the \( k^{th} \) node, \( C(k) \) be the set of its children nodes and \( x^{i \rightarrow j}_t \) be the message from \( i^{th} \) node to \( j^{th} \) node. This message is a 32-dimensional learned vector generated by the limb policy. Since the same modular policy network is shared across parent as well as children nodes, the dimension of the message passed as input to policy network is the same as the dimension of message outputted by the policy. The message passing schemes primarily govern how each action \( a^k_t \) in Equation (1) is generated.

We now describe three types of message passing schemes: bottom-up, top-down and both-ways, in which first two are decentralized while both-ways can lead to emergence of 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 \( \theta \) are obtained by maximizing objective (1) under the following constraints for action generation:

\[
a^k_t, x^{k \rightarrow p(k)}_t = \pi_\theta \left( s^k_t, f \left( \{ x^{i \rightarrow k}_t \}_{i \in C(k)} \right) \right) \tag{2}
\]

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 \( \theta \) are trained to optimize Equation (1) subject to following constraints on \( a^k_t \):

\[
a^k_t, x^{k \rightarrow c_k}_t = \pi_\theta \left( s^k_t, x^{p(k) \rightarrow k}_t \right) \tag{3}
\]

where \( x^{k \rightarrow c_k}_t \) is the message sent to all children nodes, i.e., \( x^{k \rightarrow i}_t = x^{k \rightarrow c_k}_t \ \forall \ i \in C(k) \). 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. Emergence of Centralized Message Passing

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. 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 analogy with the central nervous system in animals where sensory neurons (upwards pass) carry information from end-effectors (leaves) to brain (root) and then motor neurons (downward pass) carry instructions from brain to generate output actions. To implement this, we divide our modular policy \( \pi_\theta \) into two subpolicies with parameters \( \theta_1 \) and \( \theta_2 \) for upwards and downwards pass respectively. Parameters \( \theta \) are trained to optimize Equation (1) subject to following constraints:

\[
x^{k \rightarrow p(k)}_t = \pi_{\theta_1} \left( s^k_t, f \left( \{ x^{i \rightarrow k}_t \}_{i \in C(k)} \right) \right) \tag{4}
\]

\[
a^k_t, x^{k \rightarrow c_k}_t = \pi_{\theta_2} \left( x^{p(k)}_t, x^{p(k) \rightarrow k}_t \right)
\]

\[
x^{k \rightarrow i}_t = x^{k \rightarrow c_k}_t \ \forall \ i \in C(k)
\]

\[
\theta = \{ \theta_1, \theta_2 \}
\]

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. If the morphological graph contains cycles, which however is rarely seen in 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 messages converge, as is the case of...
loopy-belief-propagation (Murphy et al., 1999) in 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 same or different messages to each child node. Section 3.2 described the scenarios when 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 number 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 output as many messages as the max number of child nodes across joints of all agents, i.e., $\max_n K_n$. If certain joint has fewer children, the first few distinct messages are used by each child and the remaining ones are simply ignored. Similar idea is employed in 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 same message to all child nodes. A generic alternative to handle different messages across child nodes is to implement the aggregator function $f(\cdot)$ as a recurrent neural network.

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, DGNs implemented only no-message and bottom-up message passing as it was 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.

#### 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 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 on the hopper environment because there are only three variants possible. And we do not perform cross-category training with cheetah environment because it uses a different integrator, making it unstable when jointly trained with other environments.
To create the variants for each agent, consider each agent as a tree with 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 \( n \), \( \{ s_k \}_{k=1}^{K_n} \), is a collection of local limb states. Each of these limb states, \( s_k \), 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 \( \{ \text{position}_t, \text{low}, \text{high} \} \) normalized to \([0, 1]\), where \( \text{position}_t \) is the joint position at time \( t \), and \([\text{low}, \text{high}]\) is the allowed joint range. To handle different number 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.

In addition, we also examine whether Shared Modular Policies are robust to the choice of root node while constructing kinematic graph for message passing by choosing non-torso limbs as 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 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 outperform 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 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 of our experiments, 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 torso-root setup presented in the main paper, 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Training Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours (both-ways) + root is left foot</td>
<td>3709.87 ± 580.87</td>
</tr>
<tr>
<td>Ours (both-ways) + root is torso</td>
<td>3215.04 ± 447.82</td>
</tr>
<tr>
<td>Monolithic Baseline</td>
<td>3592.70 ± 111.13</td>
</tr>
</tbody>
</table>

¹https://huangwl18.github.io/modular-rl/
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 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 a 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 either 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 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 have presented a policy 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.
References


