

# METAMORPH: LEARNING UNIVERSAL CONTROLLERS WITH TRANSFORMERS

**Agrim Gupta<sup>1</sup>, Linxi Fan<sup>1,3</sup>, Surya Ganguli<sup>1,2</sup>, Li Fei-Fei<sup>1,2</sup>**

<sup>1</sup>Stanford University, <sup>2</sup>Stanford Institute for Human-Centered Artificial Intelligence

<sup>3</sup>NVIDIA Corporation

{agrim, sganguli, feifeili}@stanford.edu, linxif@nvidia.com

## ABSTRACT

Multiple domains like vision, natural language, and audio are witnessing tremendous progress by leveraging Transformers for large scale pre-training followed by task specific fine tuning. In contrast, in robotics we primarily train a single robot for a single task. However, modular robot systems now allow for the flexible combination of general-purpose building blocks into task optimized morphologies. However, given the exponentially large number of possible robot morphologies, training a controller for each new design is impractical. In this work, we propose MetaMorph, a Transformer based approach to learn a universal controller over a modular robot design space. MetaMorph is based on the insight that robot morphology is just another modality on which we can condition the output of a Transformer. Through extensive experiments we demonstrate that large scale pre-training on a variety of robot morphologies results in policies with combinatorial generalization capabilities, including zero shot generalization to unseen robot morphologies. We further demonstrate that our pre-trained policy can be used for sample-efficient transfer to completely new robot morphologies and tasks.

## 1 INTRODUCTION

The field of embodied intelligence posits that intelligent behaviours can be rapidly learned by agents whose morphologies are well adapted to their environment (Brooks, 1991; Pfeifer & Scheier, 2001; Bongard, 2014; Gupta et al., 2021). Based on this insight, a robot designer is faced with a predicament: should the robot design be task specific or general? However, the sample inefficiency of *tabula rasa* deep reinforcement learning and the challenge of designing a single robot which can perform a wide variety of tasks has led to the current dominant paradigm of ‘one robot one task’. In stark contrast, domains like vision (Girshick et al., 2014; He et al., 2020) and language (Dai & Le, 2015; Radford et al., 2018), which are not plagued by the challenges of physical embodiment, have witnessed tremendous progress especially by leveraging large scale pre-training followed by transfer learning to many tasks through limited task-specific fine-tuning. Moreover, multiple domains are witnessing a confluence, with domain specific architectures being replaced by Transformers (Vaswani et al., 2017), a general-purpose architecture with no domain-specific inductive biases.

How can we bring to bear the advances in large-scale pre-training, transfer learning and general-purpose Transformer architectures, to the field of robotics? We believe that modular robot systems provide a natural opportunity by affording the flexibility of combining a small set of general-purpose building blocks into a task-optimized morphology. Indeed, modularity at the level of hardware is a motif which is extensively utilized by evolution in biological systems (Hartwell et al., 1999; Kashtan & Alon, 2005) and by humans in many modern engineered systems. However, prior works (Wang et al., 2018; Chen et al., 2018; Sanchez-Gonzalez et al., 2018) on learning policies that can generalize across different robot morphologies have been limited to: (1) manually constructed variations of a single or few base morphologies, i.e. little diversity in the kinematic structure; (2) low complexity of control ( $\leq 7$  degrees of freedom); (3) using Graph Neural Networks (Scarselli et al., 2008) based on the assumption that kinematic structure of the robot is the correct inductive bias.

In this work, we take a step towards a more challenging setting (Fig. 1) of learning a universal controller for a modular robot design space which has the following properties: (a) **generalization**

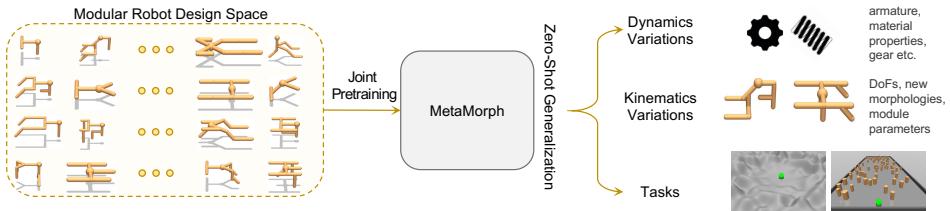


Figure 1: **Learning universal controllers.** Given a modular robot design space, our goal is to learn a controller policy, which can generalize to unseen variations in dynamics, kinematics, new morphologies and tasks. Video available at [this project page](#).

to unseen variations in dynamics (e.g. joint damping, armature, module mass) and kinematics (e.g. degree of freedom, morphology, module shape parameters) and (b) **sample-efficient** transfer to new morphologies and tasks. We instantiate the exploration of this general setting in the UNIMAL design space introduced by Gupta et al. (2021). We choose the UNIMAL design space as it contains a challenging (15 – 20 DoFs) distribution of robots that can learn locomotion and mobile manipulation in complex stochastic environments. Learning a single universal controller for a huge variety of robot morphologies is difficult due to: (1) differences in action space, sensory input, morphology, dynamics, etc. (2) given a modular design space, not all robots are equally adept at learning a task, e.g. some robots might inherently be less sample-efficient (Gupta et al., 2021).

To this end, we propose MetaMorph, a method to learn a universal controller for a modular robot design space. MetaMorph is based on the insight that robot morphology is just another modality on which we can condition the output of a Transformer. MetaMorph tackles the challenge of differences in embodiment by leveraging a Transformer based architecture which takes as input a sequence of tokens corresponding to the number of modules in the robot. Each input token is created by combining proprioceptive and morphology information at the level of constituent modules. The combination of proprioceptive and embodiment modalities and large scale joint pre-training leads to policies which exhibit zero-shot generalization to unseen variations in dynamics and kinematics parameters and sample-efficient transfer to new morphologies and tasks. Finally, to tackle the differences in learning speeds of different robots, we propose *dynamic replay buffer balancing* to dynamically balance the amount of experience collection for a robot based on its performance.

In sum, our key contributions are: (1) we introduce MetaMorph to learn a universal controller for a modular design space consisting of robots with high control complexity for challenging 3D locomotion tasks in stochastic environments; (2) we showcase that our learned policy is able to zero-shot generalize to unseen variations in dynamics, kinematics, new morphologies and tasks, which is particularly useful in real-world settings where controllers need to be robust to hardware failures; (3) we analyze the learned attention mask and discover the emergence of motor synergies (Bernstein, 1966), which partially explains how MetaMorph is able to control a large number of robots.

## 2 RELATED WORK

Prior works on learning control policies which can generalize across robot morphologies have primarily focused on parametric variations of a single (Chen et al., 2018) or few (2 – 3) robot types (Wang et al., 2018; Sanchez-Gonzalez et al., 2018; Huang et al., 2020; Kurin et al., 2021). For generalizing across parametric variations of a single morphology, various approaches have been proposed like using a learned hardware embedding (Chen et al., 2018), meta-learning for policy adaptation (Al-Shedivat et al., 2017; Ghadirzadeh et al., 2021), kinematics randomization (Exarchos et al., 2020), and dynamics randomization (Peng et al., 2018). In case of multiple different morphologies, one approach to tackle the challenge of differences in action and state spaces is to leverage Graph Neural Networks (Scarselli et al., 2008; Kipf & Welling, 2017; Battaglia et al., 2018). Wang et al. (2018); Huang et al. (2020) use GNNs to learn joint controllers for planar agents ( $\leq 7$  DoFs). Blake et al. (2021) propose freezing selected parts of networks to enable training GNNs for a single morphology but with higher control complexity. The usage of GNNs is based on the assumption that the robot morphology is a good inductive bias to incorporate into neural controllers, which can be naturally modelled by GNNs. Recently, Kurin et al. (2021) also proposed using Transformers for training planar agents. Our work differs from Kurin et al. (2021) in the diversity and scale of

training robots, complexity of the environments, conditioning the Transformer on morphological information, and showcasing strong generalization to unseen morphologies and tasks (see § B.1).

Another closely related line of work is the design of modular robot design spaces and developing algorithms for co-optimizing morphology and control (Sims, 1994) within a design space to find task-optimized combinations of controller and robot morphology. When the control complexity is low, evolutionary strategies have been successfully applied to find diverse morphologies in expressive soft robot design spaces (Cheney et al., 2014; 2018). In the case of rigid bodies, Ha (2019); Schaff et al. (2019); Liao et al. (2019) have proposed using RL for finding optimal module parameters of fixed hand-designed morphology for rigid body robots. For more expressive design spaces, GNNs have been leveraged to share controller parameters (Wang et al., 2019) across generations or develop novel heuristic search methods for efficient exploration of the design space (Zhao et al., 2020). In contrast to task specific morphology optimization, III et al. (2021) propose evolving morphologies without any task or reward specification. Finally, for self reconfigurable modular robots (Fukuda & Nakagawa, 1988; Yim et al., 2007), modular control has been utilized in both real (Rubenstein et al., 2014; Mathews et al., 2017) and simulated (Pathak et al., 2019) systems.

### 3 LEARNING A UNIVERSAL CONTROLLER

We begin by reviewing the UNIMAL design space and formulating the problem of learning a universal controller for a modular robot design space as a multi-task reinforcement learning problem.

#### 3.1 THE UNIMAL DESIGN SPACE

An agent morphology can be naturally represented as a kinematic tree, or a directed acyclic graph, corresponding to a hierarchy of articulated 3D rigid parts connected via motor actuated hinge joints. The graph  $\mathcal{G} := (\mathcal{V}, \mathcal{E})$  consists of vertices  $\mathcal{V} = \{v_1, \dots, v_n\}$  corresponding to modules of the design space, and edges  $e_{ij} \in \mathcal{E}$  corresponding to joints between  $v_i$  and  $v_j$ . Concretely, in the UNIMAL (Gupta et al., 2021) design space, each node represents a component which can be one of two types: (1) a sphere parameterized by radius and density to represent the head of the agent and form the root of the tree; (2) cylinders parameterized by length, radius, and density to represent the limbs of the robot. Two nodes of the graph can be connected via at most two motor-actuated hinge joints (i.e.  $\mathcal{G}$  is a multi-graph), parameterized by joint axis, joint limits and a motor gear ratio.

#### 3.2 JOINT POLICY OPTIMIZATION

The problem of learning a universal controller for a set of  $K$  robots drawn from a modular robot design space is a multi-task RL problem. Specifically, the control problem for each robot is an infinite-horizon discounted Markov decision process (MDP) represented by a tuple  $(S, A, T, R, H, \gamma)$ , where  $S$  represents the set of states,  $A$  represents the set of available actions,  $T(s_{t+1}|s_t, a_t)$  represents the transition dynamics,  $R(s, a)$  is a reward function,  $H$  is the horizon and  $\gamma$  is the discount factor. At each time step, the robot  $k$  receives an observation  $s_t^k$ , takes an action  $a_t^k$ , and is given a reward  $r_t^k$ . A policy  $\pi_\theta(a_t^k|s_t^k)$  models the conditional distribution over action  $a_t^k \in A$  given state  $s_t^k \in S$ . The goal is to find policy parameters  $\theta$  which maximize the average expected return across all tasks:  $R = \frac{1}{K} \sum_{k=0}^K \sum_{t=0}^H \gamma^t r_t^k$ . We use Proximal Policy Optimization (PPO) (Schulman et al., 2017), a popular policy gradient (Williams, 1992) method for optimizing this objective.

### 4 METAMORPH

Progress in model-free reinforcement learning algorithms has made it possible to train locomotion policies for complex high-dimensional agents from scratch, albeit with tedious hyperparameter tuning. However, this approach is not suitable for modular design spaces containing exponentially many robot morphologies. Indeed, Gupta et al. (2021) estimates that the UNIMAL design space contains more than  $10^{18}$  robots. Hence, learning a separate policy for each robot is infeasible. However, the modular nature of the design space implies that while each robot morphology is unique, it is still constructed from the same set of modules and potentially shares subgraphs of the kinematic tree with other morphologies. We describe how MetaMorph exploits this structure to meet the challenge of learning a universal controller for different morphologies.

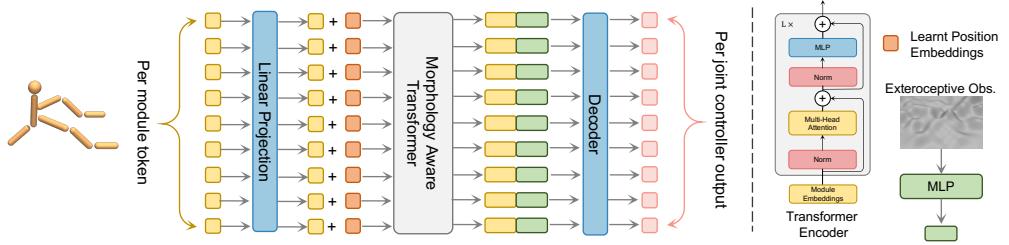


Figure 2: **MetaMorph.** We first *process* an arbitrary robot by creating a 1D sequence of tokens corresponding to depth first traversal of its kinematic tree. We then linearly embed each token which consists of proprioceptive and morphological information, add learned position embeddings and *encode* the resulting sequence of vectors via a Transformer encoder. The output of the Transformer is concatenated with a linear embedding of exteroceptive observation before passing them through a *decoder* to output per joint controller outputs.

#### 4.1 FUSING PROPRIOCEPTIVE STATES AND MORPHOLOGY REPRESENTATIONS

To learn policies that can generalize across morphologies, we must encode not only proprioceptive states essential for controlling a single robot, but also morphological information. From a multi-task RL perspective, this information can be viewed as a task identifier, where each task corresponds to a different robot morphology, all drawn from the same modular design space. Hence, instead of learning a policy which is *agnostic* to the robot morphology, we need to learn a policy *conditioned* on the robot morphology. Consequently, at each time step  $t$  (we drop the time subscript for brevity), the robot receives an observation  $s^k = (s_m^k, s_p^k, s_g^k)$  which is composed of the morphology representation ( $s_m^k$ ), the proprioceptive states ( $s_p^k$ ), and additional global sensory information ( $s_g^k$ ). See § A.1 for a detailed description of each observation type.

#### 4.2 MORPHOLOGY AWARE TRANSFORMER

The robot chooses its action via a stochastic policy  $\pi_\theta(a_t^k | s_t^k)$  where  $\theta$  are the parameters of a pair of deep neural networks: a policy network that produces an action distribution (Fig. 2), and a critic network that predicts discounted future returns. We use Transformers (Vaswani et al., 2017) to parametrize both policy and critic networks as described in detail below.

**Encode.** We make a distinction between how we process local and global state information. Concretely, let  $s^k = (s_l^k, s_g^k)$  where  $s_l^k = (s_m^k, s_p^k)$ . Since the number of joints between two modules can vary, we zero pad  $s_{l_i}^k$  to ensure that input observation vectors are of the same size, i.e.  $s_l^k \in \mathbb{R}^{N \times M}$ . In order to provide an arbitrary robot morphology as input to the Transformer, we first create a 1D sequence of local observation vectors by traversing the kinematic tree in depth first order starting at the root node (torso in case of the UNIMAL design space). We then apply a single layer MLP independently to  $s_{l_i}^k$  to create a  $D$  dimensional module embedding. We also add learnable 1D position embeddings to the module embeddings to automatically learn positional information:

$$\mathbf{m}_0 = [\phi(s_{l_1}^k; \mathbf{W}_e); \dots; \phi(s_{l_N}^k; \mathbf{W}_e)] + \mathbf{W}_{\text{pos}}, \quad (1)$$

where  $\phi(\cdot)$  is the embedding function,  $\mathbf{W}_e \in \mathbb{R}^{M \times D}$  are the embedding weights, and  $\mathbf{W}_{\text{pos}} \in \mathbb{R}^{N \times D}$  are the learned positional embeddings. Note that in practice we zero pad the input sequence of local observation vectors for efficient batch processing of multiple morphologies.

**Process.** From the module embeddings described above, we obtain the output feature vectors as:

$$\mathbf{m}'_\ell = \text{MSA}(\text{LN}(\mathbf{m}_{\ell-1})) + \mathbf{m}_{\ell-1}, \quad \ell = 1 \dots L \quad (2)$$

$$\mathbf{m}_\ell = \text{MLP}(\text{LN}(\mathbf{m}'_\ell)) + \mathbf{m}'_\ell, \quad \ell = 1 \dots L \quad (3)$$

where MSA is multi-head attention (Vaswani et al., 2017) and LN is Layernorm (Lei Ba et al., 2016).

**Decode.** We integrate the global state information  $s_g^k$  consisting of high-dimensional sensory input from camera or depth sensors. Naively concatenating  $s_g^k$  and  $s_l^k$  in the encoder has two downsides: (1) it dilutes the importance of low-dimensional local sensory and morphological information; (2)

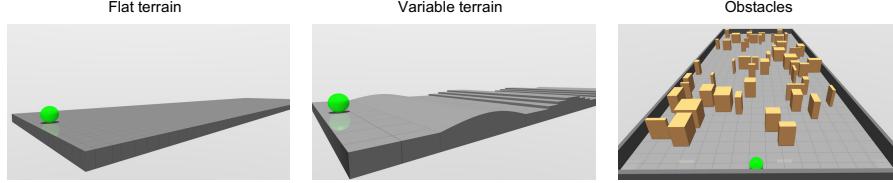


Figure 3: **Environments.** We evaluate our method on 3 locomotion tasks: Flat terrain, Variable terrain: consists of three stochastically generated obstacles: hills, steps and rubble, and Obstacles: cuboid shaped obstacles of varying sizes.

it increases the number of Transformer parameters due to an increase in the dimensionality of the input embedding ( $D$ ). Instead, we obtain the outputs of policy network as follows:

$$\mathbf{g} = \gamma(s_g^k; \mathbf{W}_g), \quad \mu(\mathbf{s}^k) = \phi(\mathbf{m}_{\mathbf{l}_i}, \mathbf{g}; \mathbf{W}_d), \quad \pi_\theta(a^k | s^k) = \mathcal{N}(\{\mu(s_i^k)\}_i^N, \Sigma), \quad (4)$$

where  $\gamma(\cdot)$  is a 2-layer MLP with parameters  $\mathbf{W}_g$ ,  $\phi(\cdot)$  is an embedding function with  $\mathbf{W}_d$  as the embedding weights. The action distribution is modeled as a Gaussian distribution with a state-dependent mean  $\mu(s_i^k)$  and a fixed diagonal covariance matrix  $\Sigma$ . Similarly, for the critic network, we estimate value for the whole morphology by averaging the value per limb.

#### 4.3 DYNAMIC REPLAY BUFFER BALANCING

Joint training of diverse morphologies is challenging as different morphologies are adept at performing different tasks. Consequently, some morphologies might be inherently better suited for the pre-training task. Let us consider two robots: (A) Robot A locomotes in a falling forward manner, i.e., robot A is not passively stable; (B) Robot B is passively stable. Especially with early termination, robot A will keep falling during the initial phases of training, which results in shorter episode lengths, whereas robot B will have longer episode lengths. Hence, more data will be collected for robot B in the earlier phases of training, and in turn will lead to robot B learning even faster, thereby resulting in a ‘rich gets richer’ training dynamic. However, our goal is to ensure that all morphologies have a similar level of performance at the end of training as we want to generalize across the entire distribution of morphologies.

To address this issue, we propose a simple *dynamic replay buffer balancing* scheme. On-policy algorithms like PPO (Schulman et al., 2017) proceed in iterations that alternate between experience collection and policy parameter update. Let  $\mathcal{E}_k$  be any performance metric of choice, e.g. normalized reward, episode length, success ratio, etc. Let  $\tau$  be the training iteration number. At each iteration, we sample the  $k^{\text{th}}$  robot with probability  $P_k$ , given by:

$$P_k = \frac{\mathcal{E}_k^\beta}{\sum \mathcal{E}_i^\beta} \quad \mathcal{E}_k^\tau = \alpha \mathcal{E}_k^\tau + (1 - \alpha) \mathcal{E}_k^{(\tau-1)}, \quad (5)$$

where  $\alpha \in [0, 1]$  is the discount factor and the exponent  $\beta$  determines the degree of dynamic prioritization, with  $\beta = 0$  corresponding to the uniform case. In practice, we use episode length as our performance metric. We determine  $P_k$  by replacing  $\mathcal{E}_i$  with  $\frac{1000}{\mathcal{E}_i}$ , where the numerator is the maximum episode length.

## 5 EXPERIMENTS

In this section, we evaluate our method MetaMorph in different environments, perform extensive ablation studies of different design choices, test zero-shot generalization to variations in dynamics and kinematics parameters, and demonstrate sample efficient transfer to new morphologies and tasks. For qualitative results, please refer to the video on our project website <sup>1</sup>.

<sup>1</sup><https://metamorph-iclr.github.io/site/>

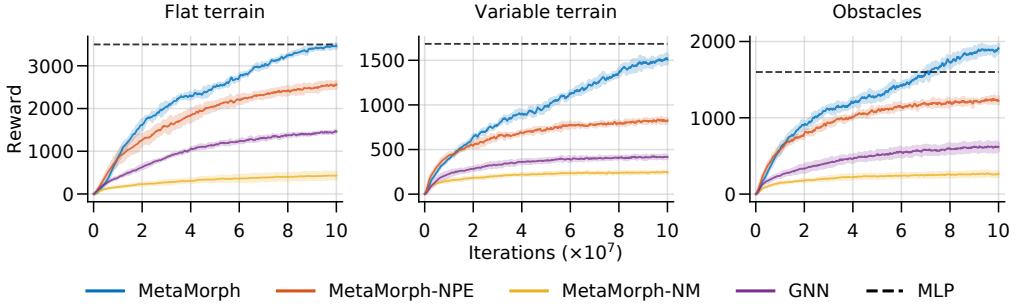


Figure 4: **Joint pre-training of modular robots.** Mean reward progression of 100 robots from the UNIMAL design space averaged over 3 runs in different environments for baselines and ablations described in § 5.2. Shaded regions denote standard deviation. Across all 3 environments, MetaMorph consistently outperforms GNN, and is able to match the average reward achieved by per morphology MLP baseline, an approximate upper bound of performance.

### 5.1 EXPERIMENTAL SETUP

We create a training set of 100 robots from the UNIMAL design space (Gupta et al., 2021) (see § A.2). We evaluate MetaMorph on three different environments (Fig. 3) using the MuJoCo simulator (Todorov et al., 2012). In all 3 environments the goal of the agent is to maximize forward displacement over the course of an episode which lasts 1000 timesteps. The 3 environments are: (1) Flat terrain (FT); (2) Variable terrain (VT): VT is an extremely challenging environment as during each episode a new terrain is generated by randomly sampling a sequence of terrain types and interleaving them with flat terrain. We consider 3 types of terrains: hills, steps, and rubble; (3) Obstacles: cuboid shaped obstacles of varying sizes on flat terrain.

We use a dense morphology *independent* reward function for all our tasks as it is not feasible to design a reward function tailored to each morphology. In all tasks, our reward function promotes forward movement using small joint torques (the latter obtained via a small energy usage penalty). In addition, as described in § 4.3, we use early termination across all environments when we detect a fall (i.e. if the torso height drops below 50% (FT, Obstacles) or 30% (VT) of its initial height).

### 5.2 BASELINES AND ABLATIONS

**Baselines:** We compare against the following baselines:

(1) **GNN:** We modify the NerveNet model proposed by Wang et al. (2018) to learn control policies for complex 3D morphologies with variable number of joints. Specifically, we replace how the NerveNet model receives input and produces output by our encode and decode steps respectively (§ 4.2). In addition, the model receives the same observation as MetaMorph i.e.  $s^k = (s_m^k, s_p^k, s_g^k)$  and is trained with dynamic replay buffer sampling. Thus, the only difference is in the **process** step. This helps test if the domain-specific inductive bias of the robot kinematic tree in the GNN is necessary.

(2) **MLP:** We train all 100 robots separately with a 2-layer MLP and report the average performance. This baseline serves as an approximate upper bound for our method.

**Ablations:** We also do an ablation study of different design choices involved in our method. We refer our full method as MetaMorph and consider the following ablations: (1) MetaMorph-NPE: no learned position embeddings; (2) MetaMorph-NM: we only provide

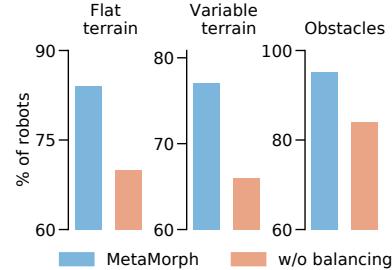
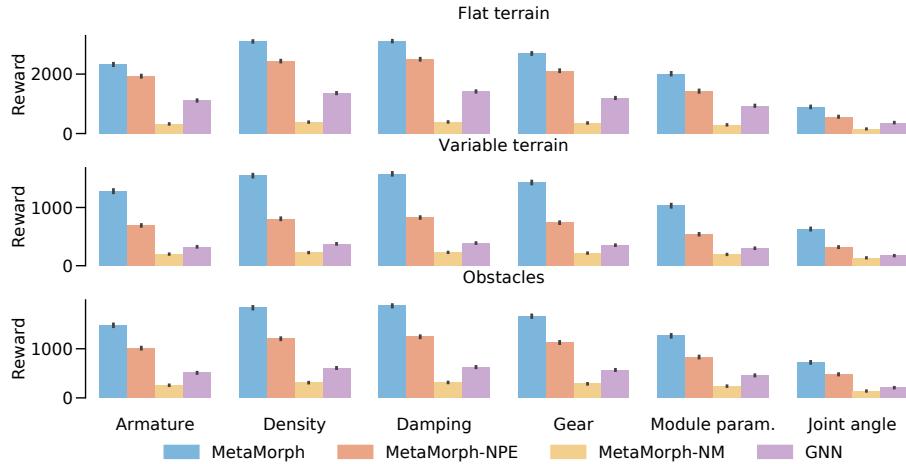


Figure 5: **Importance of dynamic replay buffer balancing.** We compare the percentage of robots with final performance greater than 75% of the MLP baseline performance when trained jointly. Across all 3 environments, on average for 10 – 15% robots, MetaMorph w/o balancing is unable to learn a good control policy.



**Figure 6: Zero-shot generalization.** We create 400 new robots for each type of variation in dynamics parameters (armature, density, damping, gear) and kinematics parameters (module shape, joint angle). Bars indicate average zero-shot reward over 10 trials for 400 robots and error bars denote 95% bootstrapped confidence interval. Across all types of variations, and environments we find that MetaMorph considerably outperforms GNN, and MetaMorph-NM.

$s^k = (s_p^k, s_g^k)$  as inputs, i.e. the model does not have access to information about the robot morphology.

Fig. 4 shows learning curves across 3 environments for training 100 morphologies. In all environments MetaMorph can successfully match the average reward achieved via the per morphology MLP baseline on both FT, and obstacle environment. While MetaMorph performance in VT is slightly below the MLP baseline, we note that it has not saturated and we stopped training at  $10^8$  iterations across all three environments. Moreover, MetaMorph is significantly more sample-efficient ( $5\times$ ) than training independent MLPs ( $5 \times 10^6$  iterations per robot). The GNN baseline saturates at a level 2 to 3 times below MetaMorph. In GNN based models, locality and neighborhood connectivity is explicitly baked into the model. Interestingly, just like ViT (Dosovitskiy et al., 2021) sparingly utilizes the 2D neighborhood structure of images at the beginning of the model by cutting the image into patches, MetaMorph uses the graph structure of the robot in the beginning by creating a 1D sequence corresponding to the kinematic tree by traversing the graph in depth first order. Moreover, the position embeddings carry no information about the graph structure and are learned from scratch. We highlight that the learned position embeddings significantly improve the performance of MetaMorph, just as they do in Transformer based image classification. Finally, without access to the morphological information, MetaMorph-NM fails to learn a policy that can control diverse robot morphologies. All of this substantiates our central claim that morphological state information is necessary to learn successful control policies, although the kinematic graph need not be explicitly baked into neural architectures to learn policies capable of controlling diverse robot morphologies. Finally, we test the importance of dynamic replay buffer balancing in Fig. 5, and find that balancing is necessary to learn a good control policy in 10 – 15% of robots across all 3 environments.

### 5.3 ZERO-SHOT GENERALIZATION

Our focus in this work is to learn policies that can generalize to unseen robots drawn from a modular robot design space. In this section, we demonstrate that MetaMorph shows favorable generalization properties across many different kinematic and dynamic variations.

**Experimental Setup.** For each of the 100 training robots, we create a dedicated test set to test zero-shot transfer performance across two types of variations: dynamics (armature, density, damping, and motor gear) and kinematics (module shape parameters like radius, and length of cylindrical limbs, and joint angles). For each training robot, we randomly create 4 different variants for each property, i.e. 400 robots with armature variations, and so on. While creating a new variant, we change the

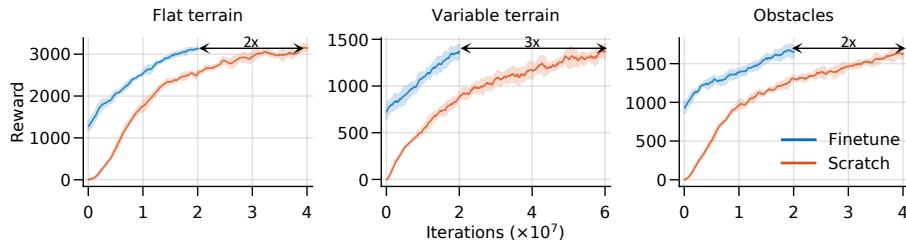


Figure 7: **Fine tuning: New robot morphologies.** Comparison of reward progression of 100 new robot morphologies averaged over 3 runs for pre-trained MetaMorph in the same environment vs from scratch. Shaded regions denotes standard deviation. Across all environments pre-training leads to strong zero-shot performance and  $2 - 3 \times$  more sample efficiency.

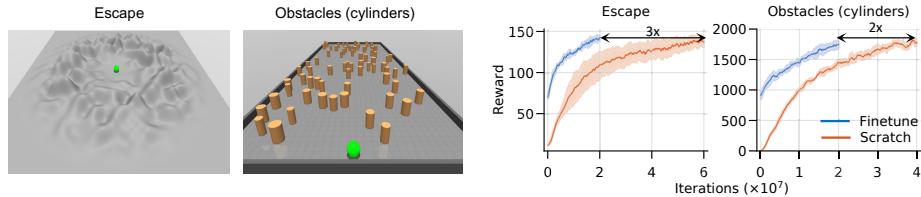


Figure 8: **Fine tuning: New robot morphologies and tasks.** *Left:* Test environments. *Right:* Comparison of reward progression of 100 test robots averaged over 3 runs for pre-trained MetaMorph (VT  $\rightarrow$  Escape, Obstacles  $\rightarrow$  Obstacles (cylinders)) vs from scratch. Shaded regions denotes standard deviation. Across all environments pre-training leads to strong zero-shot performance and  $2 - 3 \times$  savings in training iterations to achieve the same level of average reward.

relevant property of all modules or joints. See Table 2 for sampling ranges. We then compare zero-shot performance averaged over 10 trials.

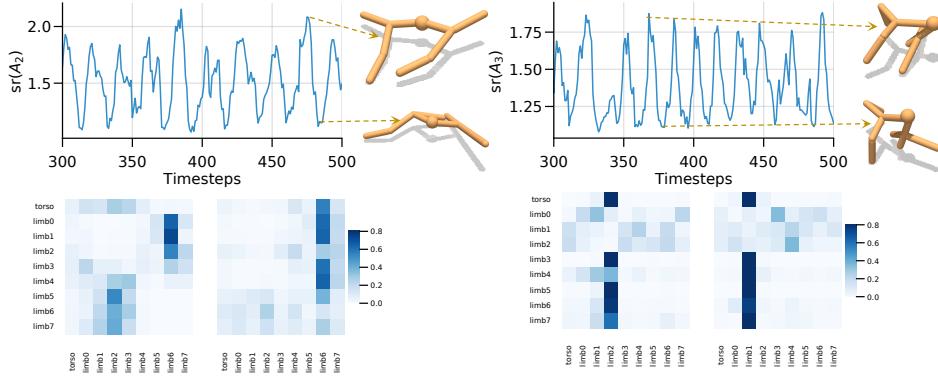
**Generalization: Dynamics.** First, we consider generalization to different dynamics (Fig. 6). We find consistently that MetaMorph performs significantly better than MetaMorph-NM and GNN across all types of dynamic variations and all environments. In fact, the difference is more pronounced for harder tasks like VT and Obstacles. We note that this result is surprising as we do not do dynamics randomization during training, i.e., all robots in the training set have the same armature and damping parameters. Despite this, we see strong generalization performance.

**Generalization: Kinematics.** Next, we consider generalization to different kinematics parameters (Fig. 6). This is a significantly more challenging setting as the model has to generalize to unseen variations in module shape parameters and changes to joint angle ranges. In fact, changes to joint angles can significantly alter the range of possible motion and might necessitate a different gait for successful locomotion. Consequently, we find that even though MetaMorph exhibits strong generalization performance compared to MetaMorph-NM and GNN in all 3 environments, there is indeed a performance drop for the challenging setting of variations in joint angles. However, the zero-shot performance is encouraging and motivates our next set of experiments on transfer learning.

#### 5.4 SAMPLE EFFICIENT TRANSFER LEARNING

**Experimental Setup.** Here we create 100 robots from the UNIMAL design space which were not part of the training set, and we demonstrate that MetaMorph shows favorable sample efficient transfer to unseen morphologies, and even unseen morphologies performing novel tasks.

**Different morphologies.** We first consider sample efficient learning of controllers for new morphologies on the same task by taking a pre-trained MetaMorph model on an environment, and then fine tuning it to learn to control morphologies in the test set. In Fig. 7 we compare the number of training iterations required to achieve a particular performance level when we fine tune MetaMorph



**Figure 9: Emergent motor synergies.** We plot the stable rank of the attention matrix ( $\text{sr}(A_l)$ ) for two agents performing locomotion on a flat terrain.  $\text{sr}(A_l)$  is small and oscillates between two values which correspond to attention maps where groups of limbs are activated simultaneously (denoted by dark columns), a characteristic signature of motor synergies.

vs training from scratch on the test set. Across all 3 environments we not only find strong zero-shot performance , but also 2 to 3 times higher sample efficiency compared to training from scratch.

**Different morphologies and novel tasks.** Finally, we consider the most realistic and general setting mimicking the promise of modular robots, where we are faced with a novel task and want to use a new robot morphology which may be suited for this task. We consider the same set of 100 test robots on two new tasks (Fig. 8): (1) Escape: The agent starts at the center of a bowl shaped terrain surrounded by small hills (bumps), and has to maximize the geodesic distance from the start location (escape the hilly region). (2) Obstacles (cylinders): cylinder shaped obstacles of varying sizes (the size distribution is also different from the train task i.e. cuboid shapes). We transfer the learned policy from VT and Obstacles to Escape and Obstacles-Cylinders respectively. Again, we find that there is a strong zero-shot performance across all 3 environments and fine-tuning is 2 to 3 times more sample efficient than training from scratch.

## 5.5 EMERGENT MOTOR SYNERGIES

We next search for a potential explanation for how MetaMorph can coordinate the large number of DoFs ( $\sim 16 \times 100$ ) across several agents. Our hypothesis is inspired by [Bernstein \(1966\)](#), which proposed the existence of muscle synergies as a neural strategy to simplify the control of multiple DoFs by the central nervous system. A muscle synergy corresponds to the constrained movements of sets of joints (or muscles) through co-activation by a single neural signal. Such synergies obviate the need to control all joints independently, by coupling sets of joints into adaptive functional groups.

Although the definition of synergies ([Bruton & O'Dwyer, 2018](#)) vary in the literature, dimensionality reduction is generally accepted as a signature of synergistic control ([Todorov & Ghahramani, 2004; Todorov, 2004](#)). Consequently, to test this hypothesis, in Fig. 9 we plot the stable rank of the attention matrix. For attention matrix  $A_l \in \mathbb{R}^{m \times m}$  for layer  $l$ , the stable rank is defined as:  $\text{sr}(A_l) = \frac{\|A_l\|_F^2}{\|A_l\|_2} = \frac{\sum \sigma_i^2}{\sigma_{max}^2}$ , where  $\sigma_i$  are the singular values of  $A_l$ . We note that  $\text{sr}(A_l)$  is small and oscillates between two values which correspond to attention maps where groups of limbs are activated simultaneously (denoted by dark columns), a characteristic signature of motor synergies. Hence, MetaMorph simplifies the control problem by learning to activate different motor synergies depending on both  $s_m^k$  and  $s_p^k$ .

## 6 CONCLUSION

In this work, we explored how we can learn a universal controller for a large modular robot design space. To this end, we proposed MetaMorph, a Transformer approach based on the insight that robot morphology is just another modality on which we can condition the output of a Transformer. We showcased that pre-training on a large collection of diverse morphologies leads to policies which can generalize to unseen variations in kinematics, dynamics, new morphologies, and tasks. We hope that our work serves as a step towards realizing the potential of large-scale pre-training and fine-tuning in the field of robotics, a paradigm that has seen tremendous success in vision and language.

## 7 ACKNOWLEDGEMENT

We gratefully acknowledge the support by Department of Navy awards (N00014-16-1-2127, N00014-19-1-2477) issued by the Office of Naval Research.

## 8 REPRODUCIBILITY STATEMENT

We have released a PyTorch (Paszke et al., 2019) implementation of MetaMorph on GitHub (<https://github.com/agrimgupta92/metamorph>). In addition, the repository also includes all the robots and environments used for benchmarking.

## REFERENCES

- Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. *arXiv preprint arXiv:1710.03641*, 2017.
- Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.
- Nikolai Bernstein. The co-ordination and regulation of movements. *The co-ordination and regulation of movements*, 1966.
- Charlie Blake, Vitaly Kurin, Maximilian Igl, and Shimon Whiteson. Snowflake: Scaling gnns to high-dimensional continuous control via parameter freezing. *arXiv preprint arXiv:2103.01009*, 2021.
- Josh Bongard. Why morphology matters. *The horizons of evolutionary robotics*, 6:125–152, 2014.
- Rodney A Brooks. New approaches to robotics. *Science*, 253(5025):1227–1232, 1991.
- Michaela Bruton and Nicholas O’Dwyer. Synergies in coordination: A comprehensive overview of neural, computational, and behavioral approaches. *Journal of Neurophysiology*, 120(6):2761–2774, 2018.
- Tao Chen, Adithyavairavan Murali, and Abhinav Gupta. Hardware conditioned policies for multi-robot transfer learning. In *NIPS*, 2018.
- Nick Cheney, Robert MacCurdy, Jeff Clune, and Hod Lipson. Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding. *SIGEVOlution*, 7(1):11–23, August 2014. doi: 10.1145/2661735.2661737. URL <https://doi.org/10.1145/2661735.2661737>.
- Nick Cheney, Josh Bongard, Vytas SunSpiral, and Hod Lipson. Scalable co-optimization of morphology and control in embodied machines. *Journal of The Royal Society Interface*, 15(143):20170937, 2018.
- Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. *Advances in neural information processing systems*, 28:3079–3087, 2015.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.
- Ioannis Exarchos, Yifeng Jiang, Wenhao Yu, and C Karen Liu. Policy transfer via kinematic domain randomization and adaptation. *arXiv preprint arXiv:2011.01891*, 2020.
- Toshio Fukuda and Seiya Nakagawa. Dynamically reconfigurable robotic system. In *ICRA*, pp. 1581–1586. IEEE, 1988.

- Ali Ghadirzadeh, Xi Chen, Petra Poklukar, Chelsea Finn, Mårten Björkman, and Danica Kragic. Bayesian meta-learning for few-shot policy adaptation across robotic platforms. *arXiv preprint arXiv:2103.03697*, 2021.
- Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.
- Agrim Gupta, Silvio Savarese, Surya Ganguli, and Li Fei-Fei. Embodied intelligence via learning and evolution. *Nature communications*, 12(1):5721, 2021.
- David Ha. Reinforcement learning for improving agent design. *Artificial life*, 25(4):352–365, 2019.
- Leland H Hartwell, John J Hopfield, Stanislas Leibler, and Andrew W Murray. From molecular to modular cell biology. *Nature*, 402(6761):C47–C52, 1999.
- Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pp. 9729–9738, 2020.
- Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *ICML*, pp. 4455–4464. PMLR, 2020.
- Donald Joseph Hejna III, Pieter Abbeel, and Lerrel Pinto. Task-agnostic morphology evolution. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=CGQ6ENUMX6>.
- Nadav Kashtan and Uri Alon. Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences*, 102(39):13773–13778, 2005.
- Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907, 2017.
- Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.
- Vitaly Kurin, Maximilian Igl, Tim Rocktäschel, Wendelin Boehmer, and Shimon Whiteson. My body is a cage: the role of morphology in graph-based incompatible control. In *ICLR*, 2021.
- Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47), 2020.
- Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *arXiv e-prints*, art. arXiv:1607.06450, July 2016.
- T. Liao, G. Wang, B. Yang, R. Lee, K. Pister, S. Levine, and R. Calandra. Data-efficient learning of morphology and controller for a microrobot. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2488–2494, 2019. doi: 10.1109/ICRA.2019.8793802.
- Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *5th Annual Conference on Robot Learning*, 2021. URL <https://openreview.net/forum?id=JrsfBJtDFdI>.
- Nithin Mathews, Anders Lyhne Christensen, Rehan O’Grady, Francesco Mondada, and Marco Dorigo. Mergeable nervous systems for robots. *Nature communications*, 8(1):1–7, 2017.

- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.
- Deepak Pathak, Christopher Lu, Trevor Darrell, Phillip Isola, and Alexei A Efros. Learning to control self-assembling morphologies: A study of generalization via modularity. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 3803–3810. IEEE, 2018.
- Rolf Pfeifer and Christian Scheier. *Understanding intelligence*. MIT press, 2001.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training (2018), 2018.
- Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.
- Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *ICML*, pp. 4470–4479. PMLR, 2018.
- Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.
- Charles Schaff, David Yunis, Ayan Chakrabarti, and Matthew R Walter. Jointly learning to construct and control agents using deep reinforcement learning. In *ICRA*, pp. 9798–9805. IEEE, 2019.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv e-prints*, art. arXiv:1707.06347, July 2017.
- Karl Sims. Evolving 3d morphology and behavior by competition. *Artificial life*, 1(4):353–372, 1994.
- Emanuel Todorov. Optimality principles in sensorimotor control. *Nature neuroscience*, 7(9):907–915, 2004.
- Emanuel Todorov and Zoubin Ghahramani. Analysis of the synergies underlying complex hand manipulation. In *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 2, pp. 4637–4640. IEEE, 2004.
- Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.
- Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *ICLR*, 2018.
- Tingwu Wang, Yuhao Zhou, Sanja Fidler, and Jimmy Ba. Neural graph evolution: Automatic robot design. In *ICLR*, 2019.
- Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Mark Yim, Wei-Min Shen, Behnam Salemi, Daniela Rus, Mark Moll, Hod Lipson, Eric Klavins, and Gregory S Chirikjian. Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1):43–52, 2007.

Allan Zhao, Jie Xu, Mina Konaković-Luković, Josephine Hughes, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Robogrammar: graph grammar for terrain-optimized robot design. *ACM Transactions on Graphics (TOG)*, 39(6):1–16, 2020.

## A IMPLEMENTATION DETAILS

In this section, we provide additional implementation details for our method.

### A.1 INPUT OBSERVATIONS

In our setup, at each time step  $t$  (we drop the time subscript for brevity) the robot receives an observation  $s^k = (s_m^k, s_p^k, s_g^k)$  which is composed of the morphology representation ( $s_m^k$ ), the proprioceptive states ( $s_p^k$ ) and additional global sensory information ( $s_g^k$ ). We note that prior work (Kurin et al., 2021; Huang et al., 2020) has defined morphology as the connectivity of the limbs. However, connectivity is only one aspect of morphology, and in general a substantial amount of additional information may be required to adequately describe the morphology. Consider a modular robot composed of  $n \in 1 \dots N_k$  modules. For each module  $s_{m_i}^k$  consists of: (1) *Module Parameters*: Module shape parameters (e.g. radius, height), material information (e.g. density), and local geometric orientation and position of the child module with respect to the parent module. (2) *Joint Parameters*: This consists of information about joint type (e.g. hinge) and its properties (e.g. joint range and axis), and actuator type (e.g. motor) and its properties (e.g. gear). All this information can be found for example, in the Universal Robot Description Format (URDF) or in simulator specific kinematic trees (e.g. MuJoCo XML (Todorov et al., 2012)).

Similarly for each module  $s_{p_i}^k$  consists of the instantaneous state of the system: (1) *Module Proprioception*: 3D Cartesian position, 4D quaternion orientation, 3D linear velocity, and 3D angular velocity. (2) *Joint Proprioception*: Position and velocity in the generalized coordinates. Except the root node, each module will be connected to its parent module via a set of joints. We adopt the convention of providing joint parameters and proprioception information to the child node. We note that the importance of proprioceptive observations has also been observed in learning good locomotion (Lee et al., 2020; Kumar et al., 2021) and manipulation policies (Mandlekar et al., 2021).

In general, additional global sensory information ( $s_g^k$ ) can be camera or depth sensor images. To save computation, we provide the information about the terrain as 2D heightmap sampled on a non-uniform grid. The grid is created by decreasing the sampling density as the distance from the root of the body increases. All heights are expressed relative to the height of the ground immediately under the root of the agent. The sampling points range from 1m behind the agent to 4m ahead of it along the direction of motion, as well as 4m to the left and right.

### A.2 ENVIRONMENTS

All our training and testing environments are implemented in MuJoCo (Todorov et al., 2012). For a detailed description about the environments and reward functions please refer Gupta et al. (2021). Gupta et al. (2021) evolved UNIMALS in 3 different environments: flat terrain, variable terrain and manipulation in variable terrain. For each environment at the end of evolution, they had 100 task optimized robots. Out of these 300, we choose a subset of 100 UNIMALS as our train set. We ensure that no robot has the exact same kinematic tree or kinematic parameters. Similarly, we created a test set of 100 UNIMALS. Finally, for zero shot evaluation experiments we created the variants as described in § 5.3. For creating kinematic variations for joint angles, we randomly selected a joint range for each joint from Table 2 which had atleast 50% overlap with the original joint angle range. This helped in preventing robot variants which could not be trained.

### A.3 TRAINING HYPERPARAMETERS

We use Transformers (Vaswani et al., 2017) to parametrize both policy and critic networks. Global sensory information is encoded using a 2-layer MLP with hidden dimensions [64, 64]. We use Proximal Policy Optimization (Schulman et al., 2017) for joint training of agents in all environments. All hyperparameters for Transformer and PPO are listed in Table 1. In addition, we use dynamic replay buffer sampling with  $\alpha = 0.1$  and  $\beta = 1.0$ . For all our ablations and baselines, we use the same hyperparameters. For MLP baseline we use a 2-layer MLP with hidden dimensions [64, 64] and for GNN baseline we use a 5-layer MLP with hidden dimension 512. The details of the joint training of modular robots are shown in Algorithm 1. We ensure that all baselines and ablations have approximately the same number of parameters ( $\sim 3.3$  Million).

#### A.4 EVALUATION METHODOLOGY

In general, the performance of RL algorithms is known to be strongly dependent on the choice of seed for random number generators (Henderson et al., 2018). Hence, we run all our baseline and ablation experiments for 3 random seeds. We found that the training curves were robust to the choice of seeds as indicated by small standard deviation (Fig. 4, 10). Consequently, we evaluate the zero shot performance using pre-trained model corresponding to a single seed (Fig. 6). For our transfer learning experiments (Fig. 7, 8), we fine tune the model with the random seed corresponding to the one used for pre-training (for all 3 seeds).

## B ADDITIONAL EXPERIMENTS

### B.1 BASELINES AND ABLATIONS

**Baselines:** We compare against the following baselines:

(1) *MetaMorph-AO*: Direct comparison to Amorpheus (Kurin et al., 2021) is not feasible due to the following reasons: (1) does not work with 3D morphologies with variable number of joints between limbs; (2) does not incorporate exteroceptive observations; (3) Amorpheus ensures a balanced collection of experience by sequentially collecting data on each morphology and maintaining a separate replay buffer per morphology. Further, updates to policy parameters are also performed sequentially. This sequential nature of the algorithm is not amenable to scaling, and would require  $\sim 30$  GPU days to train for 100 million iterations on Nvidia RTX 2080, while our method only needs 1.5 GPU days ( $\sim 20x$  training speedup).

Hence, we compare with MetaMorph-AO (Amorpheus Observation) as the closest variant to Amorpheus, where we provide the same input observations as described in Kurin et al. (2021). Specifically, we provide the following input observations: one hot encoding of unique limb ID, 3D Cartesian position, 4D quaternion orientation, 3D linear velocity, and 3D angular velocity, position in generalized coordinates, and normalized joint angle ranges. Note that although both MetaMorph and Amorpheus don't explicitly incorporate the graph structure as input to the policy, our input observations include additional morphology information (see § B.1). In contrast, except for joint angle ranges, Amorpheus does not incorporate morphology information.

(2) *Multi-Task MLP*: We train a 6-layer MLP with a hidden dimension of 512. The MLP receives the same observations as MetaMorph, and has the same number of parameters.

**Ablations:** We perform additional ablation studies to understand the importance of learnt position embeddings and morphology information in input observation. We refer our full method as MetaMorph and consider the following additional ablations: (1) MetaMorph-NMT (No Morphology information + Task encoding): we only provide  $s^k = (s_p^k, s_g^k)$  as inputs, i.e. the model does not have access to information about the robot morphology. In addition, we provide a context token with binary encoding to identify the morphology; (2) MetaMorph-HPE (Hand-designed Position Encoding): we provide  $s^k = (s_m^k, s_p^k, s_g^k)$  as inputs, with  $s_m^k$  containing a per limb unique ID.

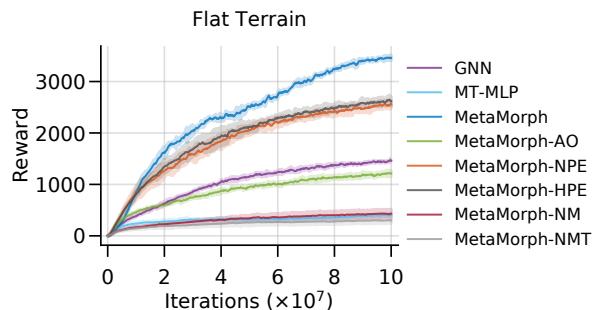


Figure 10: **Baselines and ablations.** Mean reward progression of 100 robots from the UNIMAL design space averaged over 3 runs in flat terrain for baselines and ablations described in § 5.2 and § B.1. Shaded regions denote standard deviation.

Fig. 10 shows the learning curves for joint training in the flat terrain environment for 100 training morphologies. Multi-task MLP baseline struggles to learn a good policy, which we attribute to the difficulty of the task due to the rich diversity of morphologies. We next investigate the role of learnt position embeddings and morphological information. MetaMorph-HPE performs slightly better than MetaMorph-NPE, indicating that adding unique limb ids improves performance. However, there is

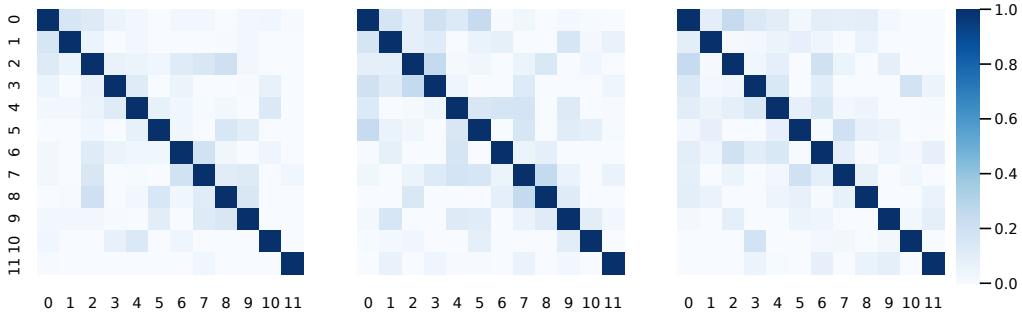


Figure 11: **Position embeddings.** Cosine similarity of position embeddings of MetaMorph for 3 different seeds in flat terrain environment.

still a large gap between MetaMorph and MetaMorph-HPE, suggesting that learnt position embeddings capture additional information. Although MetaMorph-AO performs better than MetaMorph-NM due to the addition of unique limb ids and joint angle ranges, it is still significantly worse than MetaMorph as it does not incorporate morphological information. Finally, MetaMorph-NMT also performs poorly due to the lack of morphological information.

## B.2 POSITION EMBEDDING

To better understand the information captured by position embeddings we visualize their cosine similarity. We found that across all seeds and in all environments (we only show flat terrain for brevity) a diagonal pattern emerged (Fig 11). Although, this might suggest that learnt position embeddings are capturing a unique position identifier, we found that manually specifying a unique limb identifier to perform much worse (see MetaMorph-HPE in Fig. 10). Thus, indicating that these embeddings are capturing additional useful information. We believe that further investigation is needed to better understand the information captured by position embeddings and defer it to future work.

## B.3 KINEMATIC TREE TRAVERSAL ORDER

We first *process* an arbitrary robot by creating a 1D sequence of tokens corresponding to the depth first traversal of its kinematic tree. We note that the DFS ordering is not unique, and for nodes with similar depth, we adopted the convention of visiting the nodes which come first in the MuJoCo XML. This convention is also adopted by MuJoCo when parsing the XML. We found that zero-shot transfer of the learnt policy was not robust to an opposite ordering of nodes and the performance dropped by  $\sim 75\%$ . However, we found that the policy could be made robust to variations in the DFS ordering by training the model with a simple data augmentation strategy of randomizing the visitation order of nodes at the same depth.

## C LIMITATIONS AND FUTURE WORK

In this work, we explored how we can learn a universal controller for a large modular robot design space. We make a key assumption that we already had access to a large collection of robots which were optimized for the task of locomotion. Hence, currently we require a two stage pipeline where we first create robots and then pre-train them jointly. An important direction of future work would be to combine these two phases in an efficient manner. Moreover, although we found that MetaMorph has strong zero-shot generalization to unseen variations in kinematics and dynamics, there is indeed a performance drop on zero-shot generalization to new morphologies. Hence, an important avenue of exploration is creating algorithms for sample efficient transfer learning. Finally, our current suite of tasks focuses primarily on locomotion skills. An important line of future work will involve designing general purpose controllers which could perform multiple skills.

---

**Algorithm 1** MetaMorph: Joint Training of Modular Robots

---

```

1: Input:
 $\pi_\theta$ : policy function
 $V_\phi$ : value function
 $R$ : replay buffer
 $K$ : training pool of robots
 $P_k$ : probability of sampling robot  $k$ 

2: Initialize: Learnable parameters  $\theta$  for  $\pi_\theta$ ,  $\phi$  for  $V_\phi$ ,  $P$  uniform distribution
3: for  $i=1,2,\dots N_{\text{iter}}$  do
4:   # Collect robot experience in parallel
5:   for  $j=1,2,\dots N_{\text{workers}}$  do
6:     Sample a robot  $k \in K$ , according to Equation 5
7:     # Collect one episode of experience
8:      $\tau_j \equiv \{s_t, a_t, s_{t+1}, r_{t+1}\}_j \sim \pi_\theta$ 
9:     # Add data to the buffer
10:     $R \leftarrow R \cup \tau_j$ 
11:  end for
12:  # Update policy and value functions
13:  for  $j=1,2,\dots N_{\text{epochs}}$  do
14:    Sample a minibatch of data  $r$  from  $R$ 
15:     $\pi_\theta \leftarrow \text{PPOUpdate}(\pi_\theta, r)$ 
16:     $V_\phi \leftarrow \text{PPOUpdate}(V_\phi, r)$ 
17:  end for
18:  # Sample all robots uniformly at random initially
19:  if  $i >= N_{\text{warmup}}$  then
20:     $P \leftarrow \text{samplingProbUpdate}(R)$ , according to Equation 5
21:  end if
22: end for

```

---

	<b>Hyperparameter</b>	<b>Value</b>
PPO	Discount $\gamma$	.99
	GAE parameter $\lambda$	0.95
	PPO clipping parameter $\epsilon$	0.2
	Policy epochs	8
	Batch size	5120
	Entropy coefficient	0.01
	Reward normalization	Yes
	Reward clipping	[−10, 10]
	Observation normalization	Yes
	Observation clipping	[−10, 10]
	Timesteps per rollout	2560
	# Workers	16
	# Environments	32
	Total timesteps	$1 \times 10^8$
	Optimizer	Adam
	Initial learning rate	0.0003
	Learning rate schedule	Linear warmup and cosine decay
Transformer	Warmup Iterations	5
	Gradient clipping ( $l_2$ norm)	0.5
	Clipped value function	Yes
	Value loss coefficient	0.5
	Number of layers	5
	Number of attention heads	1
	Embedding dimension	128
	Feedforward dimension	1024
	Non linearity function	ReLU
	Dropout	0.1

Table 1: **Training hyperparameters.**

<b>Kinematics</b>	
<b>Variation type</b>	<b>Value</b>
Limb radius	[0.03, 0.05]
Limb height	[0.15, 0.45]
Joint angles	[ $(-30, 0), (0, 30), (-30, 30), (-45, 45), (-45, 0), (0, 45), (-60, 0), (0, 60), (-60, 60), (-90, 0), (0, 90), (-60, 30), (-30, 60)$ ]

<b>Dynamics</b>	
Armature	[0.1, 2]
Density	[0.8, 1.2] $\times$ limb density
Damping	[0.01, 5.0]
Gear	[0.8, 1.2] $\times$ motor gear

Table 2: **Dynamics and kinematics variation parameters.**