Low-Rank Modular Reinforcement Learning via Muscle Synergy

Abstract

Modular Reinforcement Learning (RL) decentralizes the control of multi-joint robots by learning policies for each actuator. Previous work on modular RL has proven its ability to control morphologically different agents with a shared actuator policy. However, with the increase in the Degree of Freedom (DoF) of robots, training a morphology-generalizable modular controller becomes exponentially difficult. Motivated by the way the human central nervous system controls numerous muscles, we propose a Synergy-Oriented LeARning (SOLAR) framework that exploits the redundant nature of DoF in robot control. Actuators are grouped into synergies by an unsupervised learning method, and a synergy action is learned to control multiple actuators in synchrony. In this way, we achieve a low-rank control at the synergy level. We extensively evaluate our method on a variety of robot morphologies, and the results show its superior efficiency and generalizability, especially on robots with a large DoF like `Humanoid++` and UNIMALs.

1 Introduction

Deep reinforcement learning (RL) has contributed significantly to the sensorimotor control of both simulated [Heess et al., 2017, Zhu et al., 2020] and real-world [Levine et al., 2016, Mahmood et al., 2018] robots. Monolithic learning is a popular paradigm for learning control policies. In this paradigm, a policy inferring a joint action for all limb actuators based on a global sensory state is learned. Although monolithic learning has made impressive progress [Chen et al., 2020, Kuznetsov et al., 2020], it has two major shortcomings. First, the input and output space is large. For robots with more joints, learning controlling policies puts a heavy burden on the representational capacity of neural networks. Second, the input and output dimensions are fixed, making it inflexible to transfer the learned control policies to robots with different morphologies.

Modular reinforcement learning provides an elegant solution to these problems. In this learning paradigm [Wang et al., 2018], the control policy is decentralized [Peng et al., 2021], and each limb actuator is controlled by a local policy. Recent research efforts show that the local policies can learn high-performance and transferable control strategies by sharing parameters [Huang et al., 2020], communicating to each other by message passing [Huang et al., 2020], and adaptively paying attention to other actuators via graph neural networks [Kurin et al., 2020]. By exploiting the flexibility and generalizability provided by modularity, a modular policy can now control robots of up to thousands of morphologies [Gupta et al., 2021a].

Despite the significant progress, modular reinforcement learning is still limited in terms of the complexity of morphological structures that can be controlled and struggles on robots with many joints like Humanoid [Kurin et al., 2020]. The large degree of control freedom presents a major challenge for learning control policies. A question is why humans can control hundreds of muscles with dexterity while the most advanced RL policy can only control less than fifteen actuators.

Studies on muscle synergies [d'Avella et al., 2003] may provide an answer. A human central nervous system decreases the control complexity by producing a small number of electrical signals and activating muscles in groups [Ting and McKay, 2007]. Muscle synergy is the coordination of muscles that are activated in synchrony. With muscle synergies, the human nervous system achieves

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

low-rank control over its actuators. In this paper, we aim to use the inspiration of muscle synergies to reduce the control complexity and improve the learning performance of modular RL.

The first challenge of incorporating muscle synergies into modular RL is to discover a synergy structure that can promote policy learning. Neuroscience researchers factorize electrical signals [Saito et al., 2018, Falaki et al., 2017, Kieliba et al., 2018] to analyze the synergy structure, but policy signals are sub-optimal or even absent during reinforcement learning. We thus exploit the functional similarity and morphological context of actuators and use a clustering algorithm to identify actuators in the same synergy. The intuition is that muscles in a synergy typically serve the same functional purpose and have similar morphological contexts. We quantify the functional similarity by the influence of an actuator's action on the global value function, and the morphological structure is encoded as a distance matrix. To use the two types of information simultaneously, we adopt the affinity propagation algorithm [Frey and Dueck, 2007]. The synergy structure is updated periodically during learning to promptly reflect changes in value functions.

To exploit the discovered synergy structure, we design a synergy-aware architecture for policy learning. The major novelty here is that the policy learns action selection for each synergy, and the synergy actions are transformed linearly to get actuator actions. Since the number of synergies is typically much smaller than actuators, we actually learn a low-rank control policy where the physical actions are a linear mapping from a low-dimensional action space. Moreover, for better processing state information, the synergy-aware policy adopts a two-level transformer structure, which first aggregates information within each synergy and then processes information across synergies.

We evaluate our Synergy-Oriented LeARning (SOLAR) framework on two MuJoCo [Todorov et al., 2012] locomotion benchmarks [Huang et al., 2020, Gupta et al., 2021b] and in multi-task to zero-shot learning, single-task settings. SOLAR significantly outperforms previous state-of-the-art algorithms in terms of both sample efficiency and final performance on all tested settings, especially on robots with a large DoF like `Humanoid++` [Huang et al., 2020] and UNIMALs [Gupta et al., 2021b]. Performance comparison and the visualization of learned synergy structures strongly support the effectiveness of our synergy discovery method and synergy-aware transformer-based policy learning approach. Our experimental results reveal the *low-rank* nature of multi-joint robot control signals.

2 Background

Modular RL. Modular Reinforcement Learning decentralizes the control of multi-joint robots by learning policies for each actuator. Each joint has its controlling policy and they coordinate with each other via various message passing schemes. Modular RL usually needs to deal with agents with different morphologies. To do so, Wang et al. [2018] and Pathak et al. [2019] represent the robot's morphology as a graph and use GNNs as policy and message passing networks. Huang et al. [2020] uses both bottom-up and top-down message passing scheme through the links between joints for coordinating. All of these GNN-like works show the benefits of modular policies over a monolithic policy in tasks tackling different morphologies. However, recently, Kurin et al. [2020] validated a hypothesis that any benefit GNNs can extract from morphological structures is outweighed by the difficulty of message passing across multiple hops. They further propose a transformer-based method, AMORPHEUS, that utilizes self-attention mechanisms as a message passing approach. AMORPHEUS outperforms prior works and our work is based on AMORPHEUS. Previous works mainly focused on effective message passing schemes, while our work aims at reducing learning complexities when the DoF of the robot is large.

Muscle Synergy. How the human central nervous system (CNS) coordinates the activation of a large number of muscles during movement is still an open question. According to numerous studies, the CNS activates muscles in groups to decrease the complexity required to control each individual muscle [d'Avella et al., 2003, Ting and McKay, 2007]. According to muscle synergy theory, the CNS produces a small number of signals. The combinations of these signals are distributed to the muscles [Wojtara et al., 2014]. Muscle synergy is the term for the coordination of muscles that activate at the same time [Ferrante et al., 2016]. A synergy can include multiple muscles, and a muscle can belong to multiple synergies. Synergies produce complicated activation patterns for a set of muscles during the performance of a task, which is commonly measured using electromyography (EMG) [Tresch et al., 2002, Singh et al., 2018]. EMG signals are typically recorded as a matrix with a column for activation signals for a moment and a row for activation of a muscle [Rabbi

et al., 2020]. Factorisation methods on the matrix are used to extract muscle synergies from muscle activation patterns. Four most commonly used factorization methods are non-negative matrix factorisation [Steele et al., 2015, Schwartz et al., 2016, Lee and Seung, 1999, Rozumalski et al., 2017, Shuman et al., 2016, Saito et al., 2018] , principal component analysis [Ting and Macpherson, 2005, Ting et al., 2015, Danion and Latash, 2010, Falaki et al., 2017], independent component analysis [Hyvärinen and Oja, 2000, Hart and Giszter, 2013], and factor analysis [Kieliba et al., 2018, Saito et al., 2015].

Affinity propagation [Frey and Dueck, 2007] is a clustering algorithm based on multi-round message passing between input data points. It does not need to pre-define the number of clusters and proceeds by finding each instance an exemplar. Data points that choose the same exemplar belongs to the same cluster.

Suppose $\{x_i\}_{i=1}^n$ is a set of data points. Define $\boldsymbol{S} \in \mathbb{R}^{n\times n}$ as a similarity matrix. When $i \neq j$, the element $s_{i,j}$ at ith row and jth column is the similarity between x_i and x_j, which can be measured as, for example, the negative squared distance of two data points. When $i = j$, the element $s_{i,j}$ represents how likely the corresponding instance is to become an exemplar. The vector of diagonal elements, $(s_{11}, s_{22}, \ldots, s_{nn})$, is called *preference*. Non-diagonal elements in $\boldsymbol{S}$ constitute the *affinity* matrix. The algorithm takes $\boldsymbol{S}$ as input and proceeds by updating two matrices: the responsibility matrix $\boldsymbol{R}$ whose values $r_{i,j}$ represent whether x_j is well-suited to be the exemplar for x_i; the availability matrix $\boldsymbol{A}$ whose values $a_{i,j}$ quantify the appropriateness for x_i picking x_j as its exemplar [Frey and Dueck, 2007]. These two matrices are initialized to be zeroes and can be regarded as log-probability tables. The algorithm then alternatives between two message-passing steps. First, the responsibility matrix is updated:

$$r_{i,j} \leftarrow s_{i,j} - \max_{j' \neq j} \left(a_{i,j'} + s_{i,j'}\right). \tag{1}$$

Then, the availability matrix is updated:

$$a_{i,j} \leftarrow \min\left(0, r_{j,j} + \sum_{i' \notin \{i,j\}} \max(0, r_{i',j})\right) \text{ for } i \neq j; \quad a_{j,j} \leftarrow \sum_{i' \neq j} \max(0, r_{i',j}). \tag{2}$$

Messages are passed until the clusters stabilize or the pre-determined number of iterations is reached. Then the exemplar of i is $\arg\max_j r_{i,j} + a_{i,j}$.

3 Method

In this section, we present our Synergy-Oriented LeARning (SOLAR) scheme that incorporates the muscle synergy mechanism into modular reinforcement learning to reduce its learning complexity.

Our method has two major components. The first one is an unsupervised learning module that utilizes the morphological structure and value information to discover the synergy hierarchy. The second is a novel attention-based policy architecture that supports synergy-aware learning. Both of the components are specially designed to enable the control of robots with different morphologies. We first introduce the problem settings and then describe the details of the two components.

Problem settings. We consider N robots, each with a unique morphology. Agent n contains K_n limb actuators that are connected together to constitute its overall morphological structure. Examples of such robots that are studied in this paper include `Humanoid++` and UNIMALs. At each discrete timestep t, actuator $k \in \{1, 2, \ldots, K_n\}$ of a robot $n \in \{1, 2, \ldots, N\}$ receives a local sensory state $s_t^{n,k}$ as input and outputs individual torque values $a_t^{n,k}$ for the corresponding actuator. Then the robot n executes the joint action $\boldsymbol{a}_t^n = \{a_t^{n,k}\}_{k=1}^{K_n}$ at time t, after which the environment returns the next state $\boldsymbol{s}_{t+1}^n = \{s_{t+1}^{n,k}\}_{k=1}^{K_n}$ corresponding to all limbs of the agent n and a collective reward for the whole morphology $r_t^n(\boldsymbol{s}_t^n, \boldsymbol{a}_t^n)$. We learn a policy π_θ to generate actions based on states. The learning objective of the policy is to maximize the expected return on all the tasks:

$$\mathcal{J}(\theta) = \mathbb{E}_\theta \sum_{n=1}^{N} \sum_{t=0}^{\infty} \left[\gamma^t r_t^n(\boldsymbol{s}_t^n, \boldsymbol{a}_t^n)\right], \tag{3}$$

where γ is a discount factor. We adopt an actor-critic framework for policy learning. The critic is shared among all tasks and estimates the Q-function for each robot n:

$$Q^{\pi_\theta}(\boldsymbol{s}^n, \boldsymbol{a}^n) = \mathbb{E}_\theta \sum_{t=0}^{\infty} \left[\gamma^t r_t^n(\boldsymbol{s}_t^n, \boldsymbol{a}_t^n) | \boldsymbol{s}_0^n = \boldsymbol{s}^n, \boldsymbol{a}_0^n = \boldsymbol{a}^n\right]. \tag{4}$$

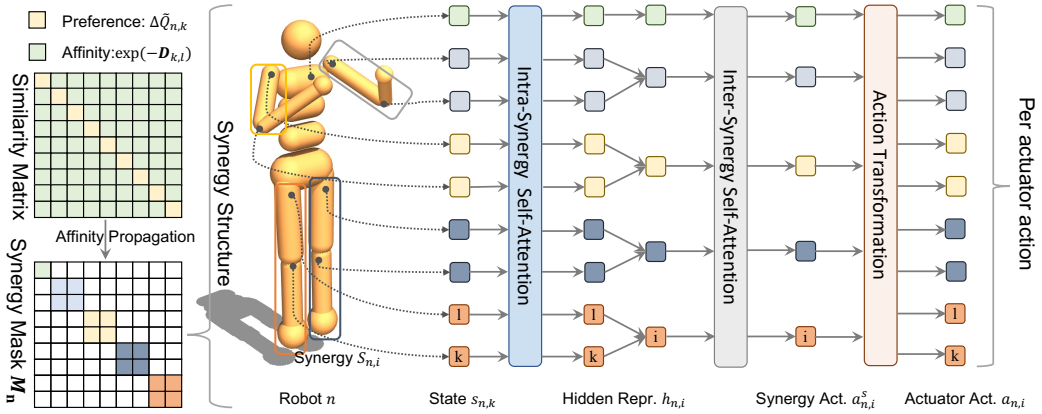

Figure 1: Synergy-aware policy learning. The intra-synergy attention module aggregates actuator information within each synergy. The inter-synergy attention module synthesizes information from all synergies to produce synergy actions. Synergy actions are then transformed linearly to obtain actuator actions. Actuator actions are of a lower rank, reducing the control complexity.

3.1 Discovering synergy structure

In Neuroscience, the muscle synergies are usually discovered by a factorization of electrical muscle signals during performing tasks [Todorov and Ghahramani, 2004, Rabbi et al., 2020]. Factorization is a method in hindsight – it statistically analyzes the optimal control policies of animals embodied in the electrical signals. By contrast, in reinforcement learning, we do not have the optimal control policies in advance. The synergy hierarchy learned from non-optimal policies is more likely incorrect, which will hamper policy learning. Therefore, we propose learning the synergy hierarchy by an unsupervised learning method that incorporates morphological information besides learning information. In this section, we describe our synergy hierarchy discovery methods.

Intuitively, actuators in the same synergy are activated simultaneously and they together finish a motion of an end effector. This gives us a hint that actuators with similar functions are supposed to be in a synergy. Formally, the function of an actuator (the kth actuator in robot n) can be modelled by its influence to the value function:

$$\Delta Q_{n,k} = \mathbb{E}_{\boldsymbol{s}^n, a^{n,k}, \boldsymbol{a}^{n,\text{-}k}} \left[Q^\pi(\boldsymbol{s}^n, [a^{n,k}, \boldsymbol{a}^{n,\text{-}k}]) - Q^\pi(\boldsymbol{s}^n, [b^{n,k}, \boldsymbol{a}^{n,\text{-}k}])\right], \tag{5}$$

where $[\cdot,\cdot]$ combines two terms, $a^{n,k}$ is the actual action of actuator k, $b^{n,k}$ is a default action of actuator k ($b^{n,k} = 0$ in MuJoCo environments) and $\boldsymbol{a}^{n,\text{-}k}$ is the actions of actuators on robot n except for the kth one. In practice, we use a SoftMax function to regularize $\Delta Q_{n,k}$: $\Delta\tilde{Q}_{n,k} = \exp(\Delta Q_{n,k}) / \sum_j \exp(\Delta Q_{n,j})$.

Ideally, $\Delta\tilde{Q}_{n,k}$ contains actuator clustering information, but the Q-values may be inaccurate during learning. We propose to treat $(\Delta\tilde{Q}_{n,1}, \Delta\tilde{Q}_{n,2}, \cdots, \Delta\tilde{Q}_{n,K_n})$ as the preference vector of the affinity propagation clustering algorithm (Sec. 2), i.e, the diagonal elements of similarity matrix, and in the next paragraph we further encode stable morphological information in the affinity matrix of affinity propagation, i.e, the non-diagonal elements of similarity matrix. The advantage of this design is that (1) we can simultaneously consider functional similarity and morphological context when using clustering to discover synergies and (2) static morphological information may help stabilize the clustering results.

We now discuss how to encode morphological information in the affinity matrix. A robot morphology is usually organized in a tree structure with its torso as the root node [Huang et al., 2020]. We

define the adjacency matrix $\boldsymbol{A}_d \in \{0,1\}^{K_n \times K_n}$ to represent robot's morphology, where $A_{i,j} = 1$ if actuators i, j are directly linked. To measure the connectiveness of every actuator pair, we use Floyd-Warshall algorithm [Cormen et al., 2022] to compute the shortest distances between actuators based on $\boldsymbol{A}_d$, resulting in a distance matrix $\boldsymbol{D} \in \mathbb{N}^{K_n \times K_n}$. The matrix $\exp(-\boldsymbol{D})$ will serve as the affinity matrix $\boldsymbol{A}$ of affinity propagation. After defining the preference and affinity for affinity propagation, the clustering algorithms run n_{max} rounds of message passing to determine actuator clusters. We treat each cluster as a synergy.

In practice, we update the synergy structure periodically during the learning process. While $\boldsymbol{D}$ remains unchanged, $\Delta\tilde{Q}_{n,k}$ changes as the Q-function is updated. Each time we run the affinity propagation with the new preference vector and then freeze the synergy structure for a while.

3.2 Learning synergy-aware low-rank policies

We now describe how we design a synergy-aware actor-critic learning framework that exploits the synergy hierarchy discovered in Sec. 3.1.

The core idea is that the policy generates control signals (actions) for synergies instead of actuators. As the number of synergies is much fewer than actuators, the learning complexity is reduced. The synergy actions will be combined linearly to generate physical actions for actuators. In this way, we learn a low-rank control policy – the physical actions are actually a linear combination of actions in a low-dimensional space.

Suppose that we find L_n synergies $\{S_{n,i}\}_{i=1}^{L_n}$ for the nth robot. We use a mask $\boldsymbol{M}_n \in \{0,1\}^{K_n \times K_n}$ to represent the synergy structure. The element at the ith row, jth column of $\boldsymbol{M}_n$ is 1 if actuator i and j are in the same synergy. The policy π_θ consists of three components: an intra-synergy attention module, an inter-synergy attention module, and an action transformation matrix as in Figure 1.

The first module aggregates information of actuators in a synergy. Input to this module is the joint state s_n, and this module outputs a hidden representation $h_{n,i} \in \mathbb{R}^d$ for each synergy $S_{n,i}$. In practice, we use a transformer with a two-head self-attention layer where the attentions between actuators belonging to different synergies are masked out. The output of the self-attention layer is of the size $K_n \times d$. We then use mean pooling to aggregate information of actuators in the same synergy. In this way, we get a hidden state $h_n \in \mathbb{R}^{L_n \times d}$, with each row being the aggregated information of a synergy.

The second module aggregates information from different synergies and outputs the synergy actions. It has two multi-head self-attention layers. We feed h_n into this these layers and get real-valued synergy actions $\boldsymbol{a}_n^s \in \mathbb{R}^{L_n}$. The synergy actions are then transformed by a learnable matrix $\boldsymbol{T} \in \mathbb{R}^{K_n \times L_n}$, and we obtain actions for actuators by $\boldsymbol{a}_n = \boldsymbol{T}\boldsymbol{a}_n^s$. One question about the transformation matrix $\boldsymbol{T}$ is that L_n may change during the learning process, requiring the dimension of $\boldsymbol{T}$ to be dynamic. In Appendix A, we discuss how to deal with this problem.

The critic uses the same intra- and inter-synergy attention structure as the policy for processing state information. The outputs of the inter-synergy attention module are fed into a fully-connected network to obtain the Q-value estimation. In practice, we use TD3 [Fujimoto et al., 2018] for the training of the policy and the critic. The intra- and inter-synergy transformer and the transformation matrix are updated in an end-to-end manner. We provide more details of our learning framework in Appendix A.

4 Experiments

In this section, we benchmark our method SOLAR on various MuJoCo [Todorov et al., 2012] locomotion tasks. we evaluate the effectiveness of SOLAR by asking the following questions: (1) Can our SOLAR outperforms other modular RL approaches when simultaneously trained on a large number of diverse agents with different morphologies? (Sec. 4.2) (2) Can the learned policy in multi-task environments generalize to new tasks with unseen morphologies? (Sec. 4.3) (3) How does SOLAR learn synergy clusters and how do synergy clusters facilitate learning? (Sec. 4.4) (4) Can SOLAR scale to single-tasks with numerous joints? (Sec. 4.5). For qualitative results, please refer to the videos on our project website[1].

[1] `https://sites.google.com/view/synergy-rl`

4.1 Experiment setup

We run experiments on two benchmarks, and report several results in this section. For additional results, please refer to Appendix B. We test all methods with 4 random seeds and show the mean performance as well as 95% confidence intervals.

Environments. For multi-task and zero-shot evaluation, we adopt the widely-used modular MTRL benchmarks [Huang et al., 2020, Kurin et al., 2020, Hong et al., 2021], which are created based on Gym MuJoCo locomotion tasks by Huang et al. [2020]. In this section, we report results on 3 in-domain settings: (1) 6 variants of walker [`Walker++`] (2) 8 variants of humanoid [`Humanoid++`] (3) 3 variants of hopper [`Hopper++`]; and 1 cross-domain settings: all 6 variants of walker, all 8 variants of humanoids, and all 3 variants of hopper [`Walker-Humanoid-Hopper++`]. For single-task evaluation, we sample 3 morphologies from UNIMALS [Gupta et al., 2021b,a], which are obtained from evolution and have numerous joints, making it suitable to evaluate the performance of modular RL policies in single-tasks. For a detailed description of the environments please refer to Gupta et al. [2021b] and Appendix A.

Baselines. In this section, we compare our method SOLAR against state-of-the-art modular RL methods SMP [Huang et al., 2020] and AMORPHEUS [Kurin et al., 2020]. SMP passes messages along the limbs using a bottom-up and top-down scheme. SOLAR and AMORPHEUS use transformer-based actors and critics, whose message passing schemes are realized by self-attention. We also compare SOLAR with standard TD3-based non-modular RL: Monolithic. Please refer to Appendix A for more details about baselines.

Ablations. There are two contributions that characterize our method. (1) An unsupervised learning module that utilizes the value information and the morphological contextual structure to discover the synergy hierarchy. (2) A novel attention-based synergy-aware policy architecture that supports synergy-aware learning. Our novelties are mainly about the discovery and utilization of synergies, and these two contributions closely rely on each other. If we totally ablate synergies, we will get AMORPHEUS. Therefore, the effectiveness of synergy-based learning can be demonstrated by comparing SOLAR against AMORPHEUS. We further design the following ablations: (1) AMORPHEUS *w/ synergy mask*. Incorporate synergy-aware inter-cluster masks into AMORPHEUS to test the contribution of synergy hierarchy. (2) SOLAR *w/o preference*. Remove value information (the preference vector) from affinity propagation of SOLAR. In this case, affinity propagation is based solely on morphological information, and we can test the effects of value information.

Implementations. We use TD3 [Fujimoto et al., 2018] as the underlying reinforcement learning algorithm for training the policy over all baselines, ablations and our method for fairness. We implement SOLAR in the AMORPHEUS codebase. And like AMORPHEUS, there is no weight sharing between actor and critic. SOLAR uses a traversal-based embedding for action transformation, which may incorporate structural information. For a fair comparison, we concatenate these embeddings into the original observation vectors of each environment which is applied to all tested algorithms.

4.2 Multi-task with different morphologies

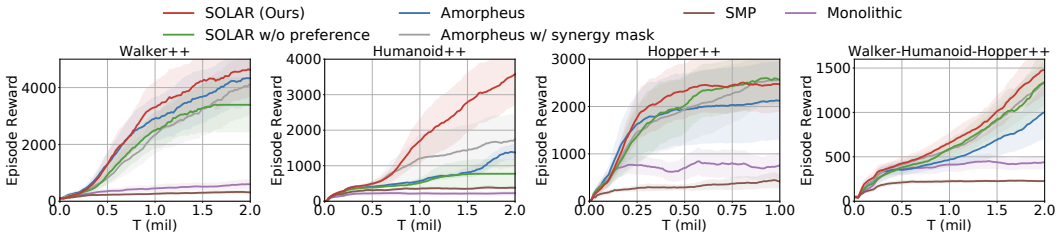

Figure 2: Multi-task performance of our method SOLAR compared to baselines and ablations.

We summarize the multi-task results in Figure 2. SOLAR outperforms the previous state-of-the-art algorithms AMORPHEUS and SMP in all settings. Furthermore, the performance gap between SOLAR and AMORPHEUS is notably larger in `Humanoid++` which has the maximum number of joints among the tested settings. We also plot the learning curves of six training variants (Figure 3) and two testing variants (Figure 4) of `Humanoid++`. These results consistently demonstrate the

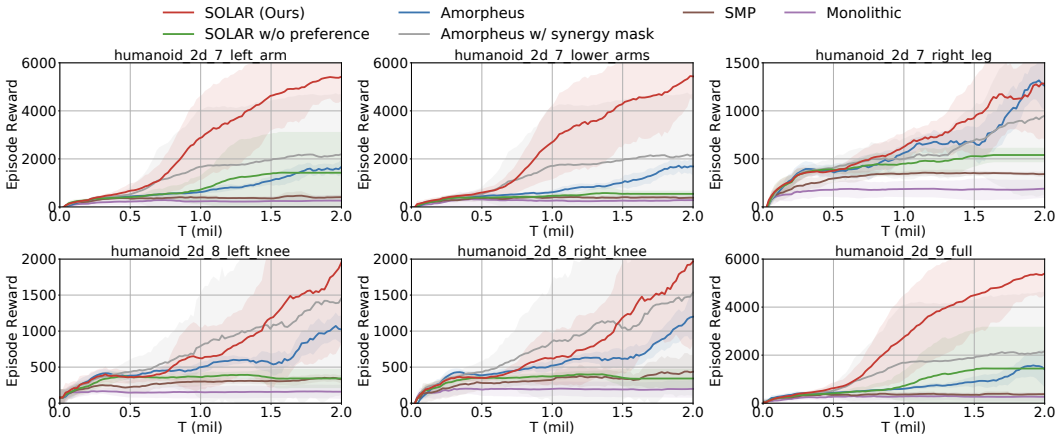

Figure 3: Multi-task performance of our method SOLAR compared to baseline and ablations on `Humanoid++`.

effectiveness of SOLAR in training robots with a large degree of freedom (DoF). We speculate that the reason of the effectiveness is that SOLAR controls robots in a lower-rank action space than the original space and the action spaces of robots with a large DoF are more likely to be reduced via the synergy mechanism. To further validate this speculation, we conduct extra evaluations on robots with much more joints in Sec. 4.5.

As for ablations, AMORPHEUS *w/ synergy mask* outperforms AMORPHEUS in most tasks in Figure 2 and Figure 3, which indicates the effectiveness of the unsupervised learning method that discovers synergy clusters and synergy-aware masks in the policy. However, SOLAR *w/o preference* performs comparable to or even worse than AMORPHEUS, which reflects the prominent usefulness of value information.

4.3 Zero-shot generalization

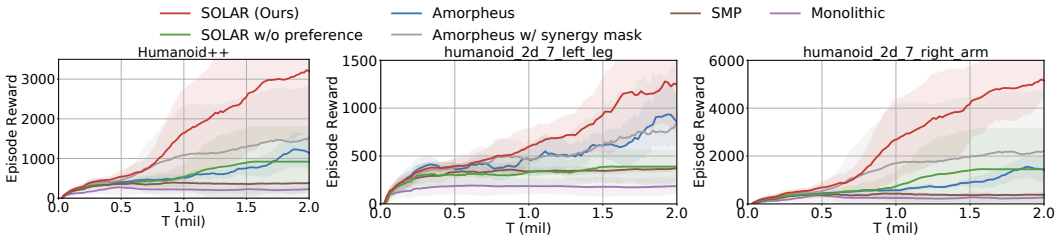

Figure 4: Zero-shot performance of our method SOLAR compared against baseline and ablations.

In this section, we benchmark SOLAR in a zero-shot learning setting where the policy is trained in multiple tasks and then generalized to unseen tasks. We compare SOLAR with AMORPHEUS, SMP, AMORPHEUS *w/ synergy mask* and SOLAR *w/o preference* in `Humanoid++`, which has the largest number of joints among the tested environments. `Humanoid++` has eight variants of humanoids, where six of them are used as training tasks (see Figure 3) and the other two are used as testing tasks (Figure 4). To obtain the results of testing tasks at each time step, we periodically load policy network parameters when training and evaluate the policy in these tasks. The trajectories on testing tasks will be discarded and will not be used for learning.

As shown in Figure 4, SOLAR outperforms all the baselines by a large margin in these unseen tasks. SOLAR learns faster than other baselines and shows higher sample efficiency in Figure 3, showing that the knowledge obtained from training tasks is effectively used in unseen tasks.

4.4 Analysis of synergies

To investigate why SOLAR performs better than baselines, in this section, we visualize the synergy clusters learned by SOLAR and the evolutionary process of synergy clusters in `Humanoid++`.

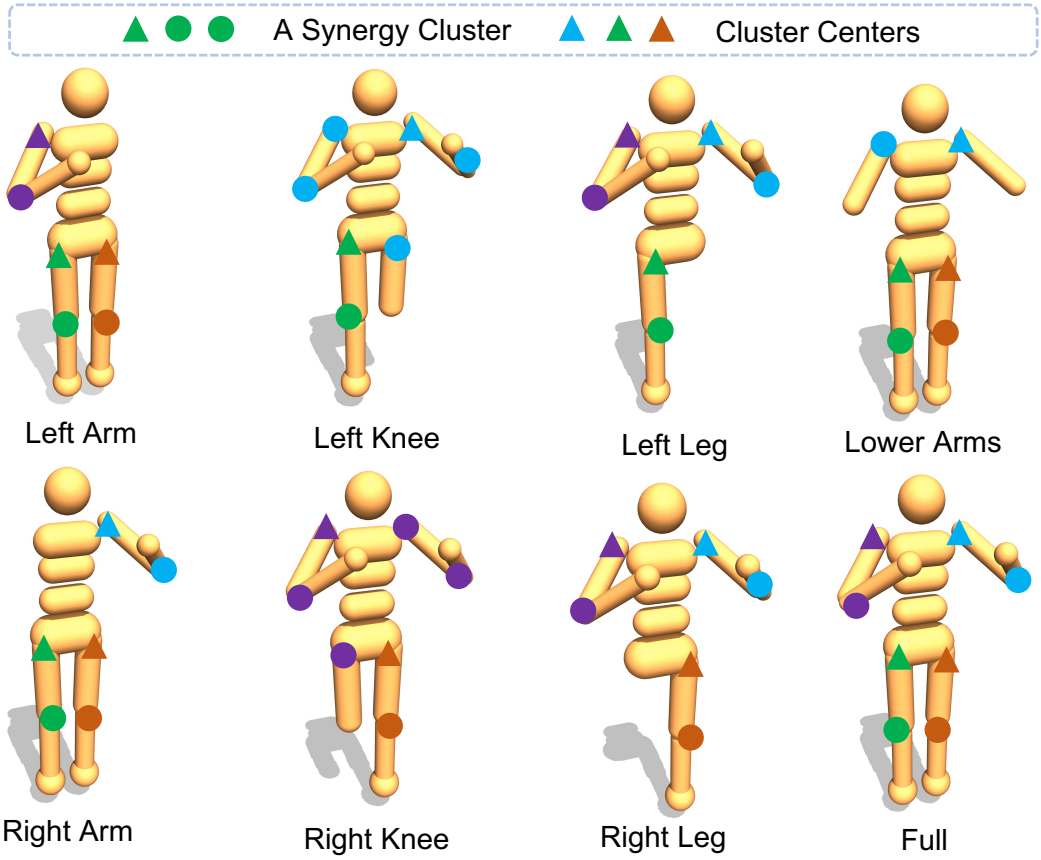

Figure 5: Synergy clustering results of SOLAR in `Humanoid++`. Different colors represent different synergy clusters, and joints marked with the same color are in the same cluster. Joints marked with triangles are the centers of their corresponding clusters.

Affinity propagation which we used as the method to discover synergy clusters can naturally outputs clustering results and the cluster centers in each cluster. We first visualize the synergy clusters of SOLAR in `Humanoid++` at the end of training in Figure 5. Here we use different colors and two shapes to mark each joint in robots. Different colors represent different synergy clusters, and joints marked with the same color are in the same cluster. Joints marked with triangles are the centers of their corresponding clusters. For example, the *Full humanoid* in the lowermost right corner has 4 synergy clusters, and the joints between the torso and two thighs and two arms are the cluster centers.

To further investigate how these clusters evolve, we visualize the learning process of *Left Knee Humanoid* in Figure 6. According to the change of synergy clusters, the learning process can be divided into three phases. In Phase 1, the episode reward is very low and $\Delta\tilde{Q}$ may only provide noisy information, which results in all joints being clustered in a same cluster. In Phase 2, SOLAR finds that the right shin is important and divides it into a separate cluster. Then in Phase 3, SOLAR finally obtains a suitable clustering solution and the episode reward increases correspondingly.

The visualization results reflect two facts: (1) close joints are more likely to be in the same synergy cluster, and (2) joints near the torso may be more influential than those who are far from the torso, and are thus selected as the cluster centers. These clusters can reduce learning and controlling complexities but are hard to discover through a standard learning mechanism. Instead, SOLAR makes use of early, less accurate learning information $\Delta\tilde{Q}$ and structural information via unsupervised learning to discover synergy clusters. The synergy clusters then further facilitate learning.

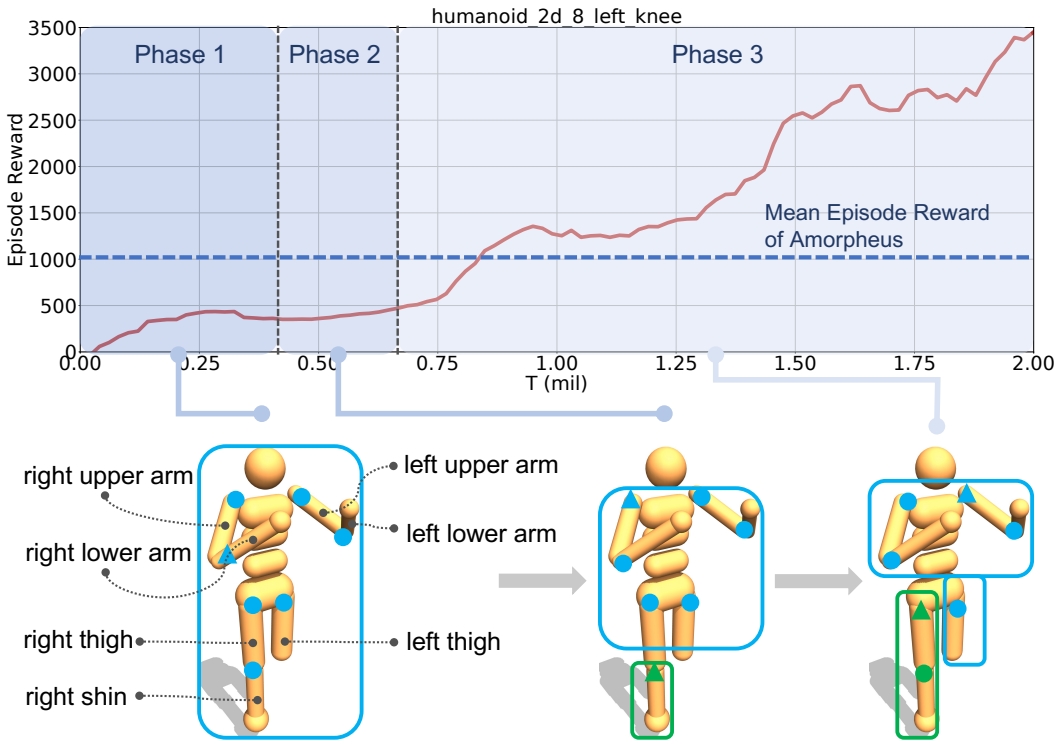

Figure 6: Synergy structure evolution of SOLAR in Humanoid. Phase are divided according to the change of synergy clusters and synergy clusters are masked with colored shapes.

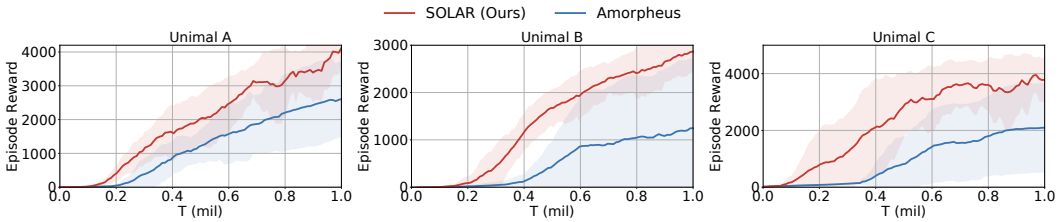

Figure 7: Multi-Task performance of our method SOLAR compared to AMORPHEUS

4.5 Single-task with numerous joints

To consolidate the speculation in Sec. 4.2 that SOLAR is suitable for tasks with numerous joints, we test SOLAR in three single-robot tasks sampled from UNIMALS [Gupta et al., 2021b]. The average number of joints is about twice as many as that in `Humanoid++`. For more details about these tasks, please refer to Appendix A. We showcase the results in Figure 7. SOLAR learns faster than state-of-the-art AMORPHEUS, showing much higher sample efficiency, and again outperforms it in average episode rewards.

5 Conclusions

In this paper, we use the inspiration of muscle synergies from neuroscience to reduce the control complexity of modular reinforcement learning in tasks with a large number of degree of freedom. We present SOLAR that leverages both structural information and policy learning information to learn synergy clusters. Moreover, a stacked-transformer architecture is proposed for learning synergy actions which are combined linearly to produce low-rank actuator actions. The evaluations of SOLAR on various tasks and settings showcase its effectiveness. The authors do not see obvious negative societal impacts of the proposed method.

References

Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

A Rupam Mahmood, Dmytro Korenkevych, Gautham Vasan, William Ma, and James Bergstra. Benchmarking reinforcement learning algorithms on real-world robots. In *Conference on robot learning*, pages 561–591. PMLR, 2018.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W Ross. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2020.

Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International Conference on Machine Learning*, pages 5556–5566. PMLR, 2020.

Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *International conference on learning representations*, 2018.

Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems*, 34, 2021.

Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pages 4455–4464. PMLR, 2020.

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 *International Conference on Learning Representations*, 2020.

Agrim Gupta, Linxi Fan, Surya Ganguli, and Li Fei-Fei. Metamorph: Learning universal controllers with transformers. In *International Conference on Learning Representations*, 2021a.

Andrea d'Avella, Philippe Saltiel, and Emilio Bizzi. Combinations of muscle synergies in the construction of a natural motor behavior. *Nature neuroscience*, 6(3):300–308, 2003.

Lena H Ting and J Lucas McKay. Neuromechanics of muscle synergies for posture and movement. *Current opinion in neurobiology*, 17(6):622–628, 2007.

Akira Saito, Aya Tomita, Ryosuke Ando, Kohei Watanabe, and Hiroshi Akima. Muscle synergies are consistent across level and uphill treadmill running. *Scientific reports*, 8(1):1–10, 2018.

Ali Falaki, Xuemei Huang, Mechelle M Lewis, and Mark L Latash. Motor equivalence and structure of variance: Multi-muscle postural synergies in parkinsons disease. *Experimental brain research*, 235(7):2243–2258, 2017.

Paulina Kieliba, Peppino Tropea, Elvira Pirondini, Martina Coscia, Silvestro Micera, and Fiorenzo Artoni. How are muscle synergies affected by electromyography pre-processing? *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 26(4):882–893, 2018.

Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

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*, pages 5026–5033. IEEE, 2012.

Agrim Gupta, Silvio Savarese, Surya Ganguli, and Li Fei-Fei. Embodied intelligence via learning and evolution. *Nature communications*, 12(1):1–12, 2021b.

Deepak Pathak, Christopher Lu, Trevor Darrell, Phillip Isola, and Alexei A Efros. Learning to control self-assembling morphologies: a study of generalization via modularity. *Advances in Neural Information Processing Systems*, 32, 2019.

Tytus Wojtara, Fady Alnajjar, Shingo Shimoda, and Hidenori Kimura. Muscle synergy stability and human balance maintenance. *Journal of neuroengineering and rehabilitation*, 11(1):1–9, 2014.

Simona Ferrante, Noelia Chia Bejarano, Emilia Ambrosini, Antonio Nardone, Anna M Turcato, Marco Monticone, Giancarlo Ferrigno, and Alessandra Pedrocchi. A personalized multi-channel fes controller based on muscle synergies to support gait rehabilitation after stroke. *Frontiers in neuroscience*, 10:425, 2016.

Matthew C Tresch, Philippe Saltiel, Andrea dAvella, and Emilio Bizzi. Coordination and localization in spinal motor systems. *Brain Research Reviews*, 40(1-3):66–79, 2002.

Rajat Emanuel Singh, Kamran Iqbal, Gannon White, and Tarun Edgar Hutchinson. A systematic review on muscle synergies: from building blocks of motor behavior to a neurorehabilitation tool. *Applied bionics and biomechanics*, 2018, 2018.

Mohammad Fazle Rabbi, Claudio Pizzolato, David G Lloyd, Chris P Carty, Daniel Devaprakash, and Laura E Diamond. Non-negative matrix factorisation is the most appropriate method for extraction of muscle synergies in walking and running. *Scientific reports*, 10(1):1–11, 2020.

Katherine M Steele, Adam Rozumalski, and Michael H Schwartz. Muscle synergies and complexity of neuromuscular control during gait in cerebral palsy. *Developmental Medicine & Child Neurology*, 57(12):1176–1182, 2015.

Michael H Schwartz, Adam Rozumalski, and Katherine M Steele. Dynamic motor control is associated with treatment outcomes for children with cerebral palsy. *Developmental Medicine & Child Neurology*, 58(11):1139–1145, 2016.

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

Adam Rozumalski, Katherine M Steele, and Michael H Schwartz. Muscle synergies are similar when typically developing children walk on a treadmill at different speeds and slopes. *Journal of biomechanics*, 64:112–119, 2017.

Benjamin Shuman, Marije Goudriaan, Lynn Bar-On, Michael H Schwartz, Kaat Desloovere, and Katherine M Steele. Repeatability of muscle synergies within and between days for typically developing children and children with cerebral palsy. *Gait & Posture*, 45:127–132, 2016.

Lena H Ting and Jane M Macpherson. A limited set of muscle synergies for force control during a postural task. *Journal of neurophysiology*, 93(1):609–613, 2005.

Lena H Ting, Hillel J Chiel, Randy D Trumbower, Jessica L Allen, J Lucas McKay, Madeleine E Hackney, and Trisha M Kesar. Neuromechanical principles underlying movement modularity and their implications for rehabilitation. *Neuron*, 86(1):38–54, 2015.

Frédéric Danion and Mark Latash. *Motor control: theories, experiments, and applications*. Oxford University Press, 2010.

Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

Corey B Hart and Simon Giszter. Distinguishing synchronous and time-varying synergies using point process interval statistics: motor primitives in frog and rat. *Frontiers in computational neuroscience*, 7:52, 2013.

Akira Saito, Kohei Watanabe, and Hiroshi Akima. Coordination among thigh muscles including the vastus intermedius and adductor magnus at different cycling intensities. *Human Movement Science*, 40:14–23, 2015.

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, pages 4637–4640. IEEE, 2004.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596, 2018.

Sunghoon Hong, Deunsol Yoon, and Kee-Eung Kim. Structure-aware transformer policy for inhomogeneous multi-task reinforcement learning. In *International Conference on Learning Representations*, 2021.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Sequence-to-sequence modeling with nn.transformer and torchtext. `https://pytorch.org/tutorials/beginner/transformer_tutorial.html`, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In Appendices.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] In Appendices and supplemental materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] In Appendices.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All results have confidence intervals.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In Appendices.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] In Sec. 4.1.
   (b) Did you mention the license of the assets? [Yes] In Appendices.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Website for qualitative results.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Reproducibility

A.1 Environment description

We run experiments on two benchmarks as described in Table 1. All these benchmarks are implemented in MuJoCo [Todorov et al., 2012]. Huang et al. [2020] created `Walker++`, `Humanoid++`, `Hopper++` and `Walker-Humanoid-Hopper++`, by removing some limbs and joints from the original intact morphology of Gym MuJoCo robots [Todorov et al., 2012, Brockman et al., 2016]. In this benchmark, every robot has the ability to hop, walk, or run, *i.e.*, only the robots that can move forward are left. Following Huang et al. [2020], Kurin et al. [2020], Hong et al. [2021], we use this benchmark for the evaluation of multi-task learning (Sec. 4.2) and zero-shot generalization ability (Sec. 4.3). As for the single-task learning benchmark, we sampled some morphologies from UNIMALS [Gupta et al., 2021b,a]. Gupta et al. [2021b] created UNIMALS by evolution on robots' morphologies, kinematics, and dynamics. At the end of the evolution, they had 100 task optimized robots. We choose a subset of these robots that are optimized for locomotion in flat terrain environment for single-task evaluation in Sec. 4.5. We ensure that the selected robots have different morphologies from each other. For multi-task evaluation, we use the whole 100 robots for training in Appendix B.1. And for zero-shot evaluation in robots' dynamic and kinematic properties in Appendix B.1, we use the test set created by Gupta et al. [2021a], which contains 2400 robots.

The reward of these tasks is given by the speed of the robot moving forward, and is penalized by the action norm. An episode terminates when the robot's height is low or the episode length exceeds 1000 time steps. For all the environments, we can extract the morphology of the robot as a graph from the corresponding MuJoCo XML file. Then the morphological information is concatenated in observations in baselines or is used to generate embeddings for action transformation in SOLAR. The visualization of these robots can be found in synergy clustering results: `Walker++` as in Figure 11, `Humanoid++` as in Figure 5, `Hopper++` as in Figure 10 and UNIMALS as in Figure 12.

Table 1: Full list of environments.

Environment	Train set	Test set
`Walker++`	walker_2_main walker_4_main walker_5_main walker_7_main	
`Humanoid++`	humanoid_2d_7_left_arm humanoid_2d_lower_arms humanoid_2d_7_right_leg humanoid_2d_8_left_knee humanoid_2d_8_right_knee humanoid_2d_9_full	humanoid_2d_7_left_leg humanoid_2d_7_right_arm
`Hopper++`	hopper_3 hopper_4 hopper_5	
`Walker-Humanoid-Hopper++`. Union of `Walker++`, `Humanoid++` and `Hopper++`		
UNIMALS	unimalA, unimalB, unimalC	

A.2 Implementation details

We implement SOLAR based on AMORPHEUS [Kurin et al., 2020], which is built on Official PyTorch Tutorial [Seq, 2020]. AMORPHEUS also shares the codebase with SMP [Huang et al., 2020]. Table 2 provides the hyperparameters needed to replicate our experiments.

As in Figure 1, there are three main components in our implementation, *i.e.*, *Intra-Synergy Self-Attention*, *Inter-Synergy Self-Attention* and *Action Transformation*. To achieve *Intra-Synergy Self-Attention* efficiently, we pass a synergy-based mask into PyTorch's *TranformerEncoderLayer*

Table 2: Hyperparameters of our SOLAR.

Hyperparameter	Value
Learning rate	0.0001
Gradient clipping	0.1
Normalization	LayerNorm
Total attention layers	3
Intra-synergy attention layers	1
Inter-synergy attention layers	2
Attention heads	2
Attention hidden size	256
Encoder output size	128
Mini-batch size	100
Replay buffer size	10M
Attention embedding size	128

through *src_mask* function argument. The synergy-based mask is discussed in detail in Sec. 3.2. *Inter-Synergy Self-Attention* is implemented by the normal *TranformerEncoderLayer* without masks.

As for *Action Transformation*, the main challenge is how to obtain a transformation matrix $\mathbf{T} \in \mathbb{R}^{K_n \times L_n}$, where K_n and L_n are the number of actuators and the number of synergies of robot $n \in \{1, 2, \cdots, N\}$. L_n may change during the learning process. Intuitively, the matrix $\mathbf{T}$ is dependent on the robot's morphology. Here we use a traversal-based positional embedding [Hong et al., 2021] technique, and these embeddings together represent the robot's morphological information. To obtain the traversal-based positional embedding, we first apply left-child-right-sibling representation to represent a general tree as a binary tree. Then, we traverse the binary tree in pre-order, in-order and post-order, which forms a triple for each actuator consisting of its orders. And these triples together are sufficient to reconstruct the original tree, which indicates that each triple contains the structural information of the corresponding actuator. The triple of each joint serves as indices into a embedding pool $\{\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_p\}$ where $p = \max_n K_n$ is the maximum number of actuators in a robot, and each $\boldsymbol{r}_i \in \mathbb{R}^s$ is a learnable vector. The selected embeddings are processed by a network, and we concatenate the outputs to obtain a representation of the corresponding actuator. Using dot product between representations of two actuators, we get a matrix $\mathbf{H} \in \mathbb{R}^{K_n \times K_n}$ where $\mathbf{H}_{i,j}$ indicates the relation between actuator i and actuator j. Finally, we average the columns corresponding to actuators in the same synergy cluster and obtain the transformation matrix $\mathbf{T} \in \mathbb{R}^{K_n \times L_n}$. The embedding pool and network are updated by backpropagating the RL loss.

Experiments are carried out on NVIDIA GTX 2080 Ti GPUs. For `Humanoid++`, our method requires approximately 10G of RAM and 5G of video memory, and takes about 23 hours to finish 2M timesteps of training.

The code for our method is included in the supplementary materials with sufficient setup and running instructions. The code follows the MIT license.

A.3 Baselines and ablations

We compare our method against various baselines and ablations. Here we explain the implementations of these baselines and ablations.

AMORPHEUS. We use the original implementation of AMORPHEUS released by Kurin et al. [2020]. For fair comparison, SOLAR uses the same value for those hyperparameters that are shared with AMORPHEUS (Table 2).

SMP. We use the implementation of SMP in the AMORPHEUS codebase, which is the same as the original implementation of SMP provided by Huang et al. [2020].

Monolithic. Following the setup by Huang et al. [2020], we choose TD3 as the standard monolithic RL baseline. The actor and critics of TD3 are implemented by fully-connected neural networks, which take the concatenation of observations of all actuators as input. Since number of actuators

varies in different robots, the dimension of observations is incompatible. To overcome this issue, we zero-pad the observations and actions to the maximum dimension across all robots.

AMORPHEUS *w/ synergy mask*. We passed a synergy-based mask to the first layer of self-attention in AMORPHEUS, and the mask is learned in the same way as in our method. The other two layers of AMORPHEUS are not modified.

SOLAR *w/o preference*. We simply set the preference vector of the affinity propagation clustering algorithm to None, and the algorithm will use the median of the input affinity matrix as a default preference vector.

B Additional experiments and discussions

B.1 UNIMALS Locomotion task and Manipulation task

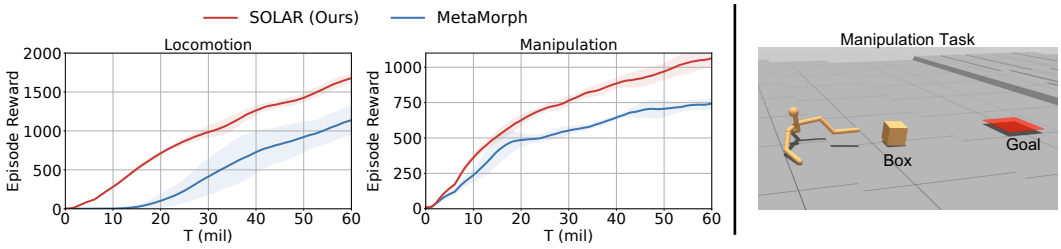

Figure 8: Left: Multi-Task performance in UNIMALS of our method SOLAR compared to MetaMorph. Right: Manipulation task.

To further analyze the scalability of our SOLAR when the number of robots and the number of actuators are large, we use the original UNIMALS [Gupta et al., 2021b] Locomotion tasks in flat terrain and Manipulation tasks in variable terrain. In Locomotion tasks, the robot needs to go forward as fast as possible. In Manipulation tasks, the robot needs to first reach a box, and then push this box to a randomly generated goal. The UNIMALS has 100 robots with different morphologies, kinematics, and dynamics for multi-task training. We compared our method SOLAR against MetaMorph [Gupta et al., 2021a]. In this subsection, SOLAR is built upon MetaMorph. Like MetaMorph, SOLAR incorporates structural information into AMORPHEUS and uses a dynamic replay buffer balancing technique to deal with the large number of robots. SOLAR can be simply combined with MetaMorph by modifying its self-attention structure. The training results are shown in Figure 8. SOLAR outperforms MetaMorph by a large margin, which suggests that SOLAR is also effective with numerous different robots.

Table 3: Zero-shot generalization performance of SOLAR compared against MetaMorph in UNIMALS Locomotion task.

	Variants	SOLAR	MetaMorph
Dynamics	Armature	**1253.39** $\pm$ 28.25	843.31 $\pm$ 14.80
	Density	**1654.43** $\pm$ 24.55	1089.64 $\pm$ 22.64
	Damping	**1663.64** $\pm$ 26.21	1113.04 $\pm$ 10.18
	Gear	**1480.45** $\pm$ 30.49	987.29 $\pm$ 8.05
Kinematics	Module param.	**1084.41** $\pm$ 13.93	700.43 $\pm$ 17.02
	Joint angle	**477.50** $\pm$ 6.83	274.37 $\pm$ 6.55

In Sec. 4.3, we test the zero-shot performance of SOLAR on robots with different morphologies. However, in reality, there are cases where the morphologies are not changed, but the dynamics (armature, density, damping, and motor gear) and kinematics (module shape parameters and joint angles) of robots are new in unseen tasks. Thus we also benchmark the zero-shot performance of SOLAR in these cases. Gupta et al. [2021a] created a test set to test zero-shot performance based on the 100 training robots of UNIMALS. For each robot in the training set, they create 4 different variants for each property (regarding the dynamics and kinematics), leading to a total number of

Table 4: Zero-shot generalization performance of SOLAR compared against MetaMorph in UNIMALS Manipulation task.

	Variants	SOLAR	MetaMorph
Dynamics	Armature	**792.69** $\pm$ 32.35	685.36 $\pm$ 9.03
	Density	**943.07** $\pm$ 57.46	823.20 $\pm$ 18.95
	Damping	**984.19** $\pm$ 24.14	860.70 $\pm$ 11.24
	Gear	**862.04** $\pm$ 36.90	784.54 $\pm$ 12.01
Kinematics	Module param.	**663.18** $\pm$ 12.47	611.43 $\pm$ 5.80
	Joint angle	314.16 $\pm$ 4.95	**455.49** $\pm$ 11.28

Table 5: Dynamics and kinematics variations to generate the test robot set, reproduced from Gupta et al. [2021a].

Kinematics	
Variation type	**Value**
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)]
Dynamics	
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

$100 \times 6 \times 4 = 2400$ test variants. Here we reproduce from their paper the sampling ranges of test variants in Table 5 . Please refer to Gupta et al. [2021a] for more details. Due to the limitation of computing resources, we reduce the batch size of SOLAR and MetaMorph from 5120 to 128, and other hyperparameters follow Gupta et al. [2021a].

To evaluate the zero-shot generalization performance, we load our models trained on multi-task UNIMALS and show the averaged performance (and variance) over 4 random seeds in Table 3 and Table 4 for Locomotion task and Manipulation task, respectively. We find that SOLAR exhibits strong generalization performance compared to the baseline, which indicates that the synergy clusters and action transformations learned on the training set are robust to dynamic and kinematic variations.

B.2 Compared with other baselines in single-task

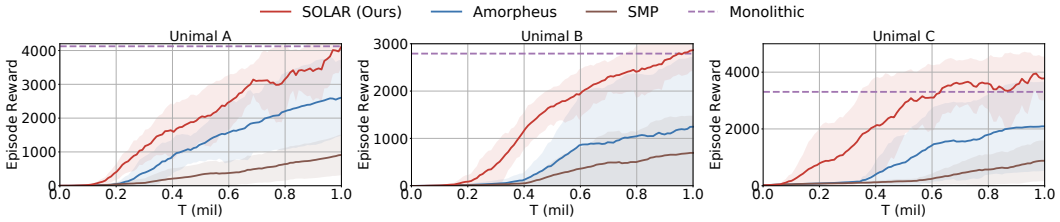

Figure 9: Multi-Task performance of our method SOLAR compared to AMORPHEUS, SMP and the final performance of Monolithic.

In Sec. 4.5, we compare SOLAR with transformer-based baseline AMORPHEUS, here we also compare our SOLAR with the GNN-style baseline SMP in single-task evaluations in Figure 9 as additional results.

B.3 Synergy clustering results of more morphologies

In Figure 5, we visualize the synergy clustering results of `Humanoid++` to analyze the effect of synergy. In this section, we provide visualizations of synergy clustering results for `Hopper++` (Figure 10), `Walker++` (Figure 11), and UNIMALS (Figure 12) as additional results.

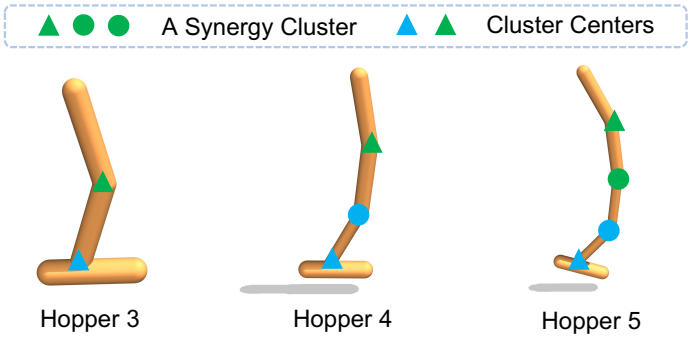

Figure 10: Synergy clustering results of SOLAR in `Hopper++`. Different colors represent different synergy clusters, and joints marked with the same color are in the same cluster. Joints marked with triangles are the centers of their corresponding clusters.

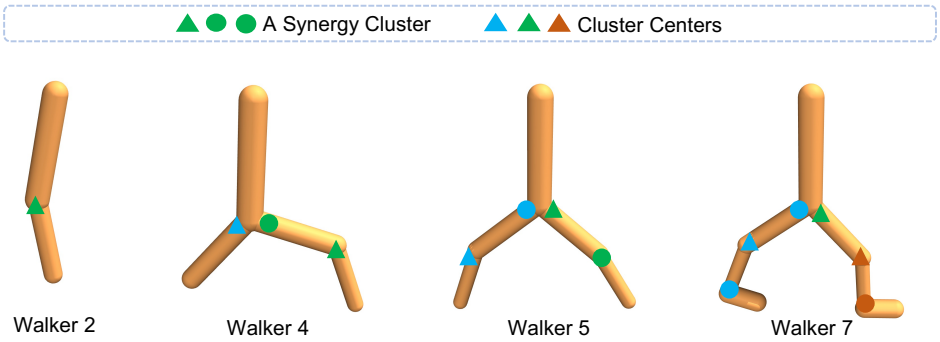

Figure 11: Synergy clustering results of SOLAR in `Walker++`.

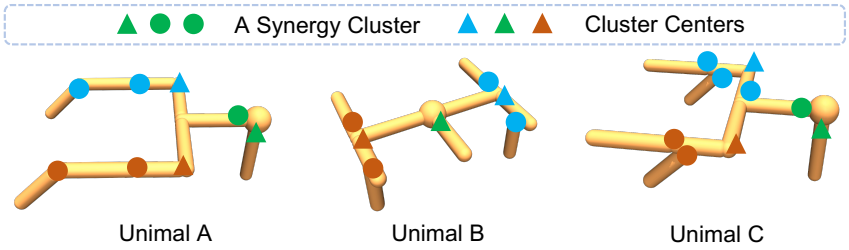

Figure 12: Synergy clustering results of SOLAR in UNIMALS.

These additional results further consolidate our analysis in Sec. 4.4: (1) close joints are more likely to be in the same synergy cluster, and (2) joints near the torso may be more influential than those who are far from the torso, and are thus selected as the cluster centers.

B.4 Connection to conventional notion of muscle synergy

Mathematically, conventional muscle synergy studies typically use non-negative matrix factorization (Rabbi et al, 2020) and aim to solve the following optimization problem:

$$\min_{H \in \mathbb{R}_+^{N \times M}, A \in \mathbb{R}_+^{M \times T}} \|U - HA\|_F,$$

where $U \in \mathbb{R}^{N \times T}$ is a given matrix of observed electrical control signals. The element at ith row and tth column of U is the control signal to muscle i at timestep t. In this optimization problem, the num-

ber of synergies, M (where $M < N$), is typically pre-defined or chosen according to a pre-defined reconstruction error threshold. By solving this problem, one can discover the synergy structure by observing matrix H. Conventional muscle synergy studies that use other factorization methods (PCA, ICA, and FA) share the similar optimization problem with different matrix constraints.

We study a similar optimization problem but with additional constraints:

$$\max_{U,H,A} \sum_t \gamma^t R(s_t, U_t) - \|U - HA\|_F.$$

Here s_t is the environment state at timestep t, and U_t is the column t of matrix U, i.e., actions at timestep t. And $R(s_t, U_t)$ is the reward of choosing action U_t at state s_t.

The differences are:

1. We additionally maximize the expected return of muscle actions.
2. In our formulation, the number of synergies is not pre-defined but is learned in an unsupervised manner.
3. The matrix U is also not given, but is generate by an attention-based policy. This policy are optimized to maximize return as well as to minimize the decomposition loss (term 2).

In summary, we share a common optimization object with conventional muscle synergy studies, which is a decomposition loss. However, we need to additionally learn the number of synergies and control signals which are inputs in the conventional studies. Structurally, our framework gives an attention function class that covers a synergy decomposition solution which can minimize the decomposition loss while enable an efficient control policy.

B.5 Limitations and future works

When generalizing to unseen robots with larger numbers of actuators than training robots, the embeddings of testing robots are not learned, which will hamper the zero-shot performance. One possible future direction will involve designing a more scalable actuator embedding method. Moreover, SOLAR is more suitable for robots with large number of actuators. Our approach is able to reduce a great degree of control complexity for these robots. But for robots with few actuators, SOLAR may only have a little positive impact and the learning of synergy-aware policy may even damage the performance.