

# Arnold: a generalist muscle transformer policy

Alberto Silvio Chiappa<sup>1,2</sup>, Boshi An<sup>1,2</sup>, Merkourios Simos<sup>1</sup>,

Chengkun Li<sup>1</sup>, Alexander Mathis<sup>✉,1</sup>

<sup>1</sup>Brain Mind Institute, School of Life Sciences, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

<sup>2</sup>These authors contributed equally.

✉Correspondence: alexander.mathis@epfl.ch

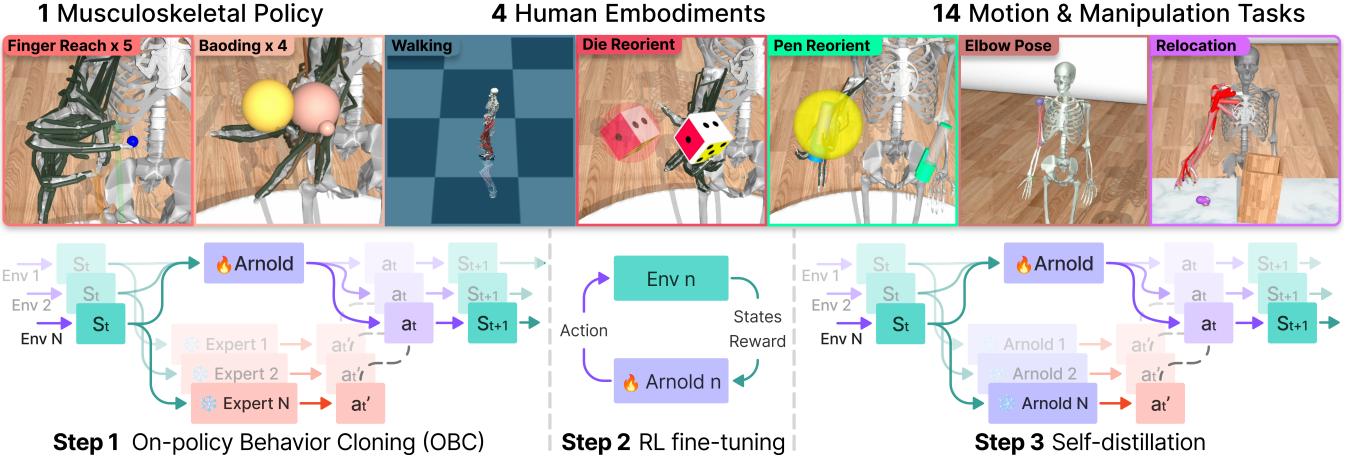
## Abstract

Controlling high-dimensional and nonlinear musculoskeletal models of the human body is a foundational scientific challenge. Recent machine learning breakthroughs have heralded policies that master individual skills like reaching, object manipulation and locomotion in musculoskeletal systems with many degrees of freedom. However, these agents are merely "specialists", achieving high performance for a single skill. In this work, we develop Arnold, a generalist policy that masters multiple tasks and embodiments. Arnold combines behavior cloning and fine-tuning with PPO to achieve expert or super-expert performance in 14 challenging control tasks from dexterous object manipulation to locomotion. A key innovation is Arnold's sensorimotor vocabulary, a compositional representation of the semantics of heterogeneous sensory modalities, objectives, and actuators. Arnold leverages this vocabulary via a transformer architecture to deal with the variable observation and action spaces of each task. This framework supports efficient multi-task, multi-embodiment learning and facilitates rapid adaptation to novel tasks. Finally, we analyze Arnold to provide insights into biological motor control, corroborating recent findings on the limited transferability of muscle synergies across tasks.

## Introduction

Dexterous and adaptive motor control in humans and other animals serves as a central inspiration for robotics and artificial intelligence. While most robotics research has traditionally focused on motor-actuated systems (1, 2), muscle-like control is gaining momentum, supported by emerging hardware platforms (3–8). These muscle-based systems offer numerous advantages, including flexible weight distribution, efficient energy storage in tendons, and valuable opportunities for cross-pollination with computational neuroscience (9–15). In parallel, biomechanics simulators have also flourished (16–19), enabling in-silico experiments. Training musculoskeletal control policies presents unique challenges due to the large number of sensory inputs and actuators, as well as nonlinear muscle dynamics. These technical complications have motivated advances in reinforcement learning (RL), exploration, curriculum learning (CL) and imitation learning (IL) (11, 14, 20–29). By combining these techniques, previous works have succeeded in controlling biologically realistic models of the human body to reach strong performance for complex tasks such as object manipulation and locomotion (25, 30, 31). Unfortunately, these results are limited to one specific problem at a time, and different body parts or tasks require learning new policies from scratch. This approach complicates drawing systems-level conclusions about motor control in humans (32), because specialist policies can reproduce a limited repertoire of motor skills.

In this work, we address this limitation by developing **Arnold**, a generalist transformer policy to control the human body (Figure 1). Arnold leverages the inherent generalization capabilities of transformers to control different body parts across a variety of tasks. Unlike traditional task-specific controllers, our system utilizes a unified transformer network to handle diverse tasks seamlessly, ranging from single-finger control via skillful object manipulation to locomotion. As generic sequence-to-sequence models, transformers can handle a variable number of proprioceptive sensory streams and muscle actuators, according to the available embodiment. We emphasize that biological sensory-to-motor transformations have no linear structure, as they are part of a complex computational graph implemented by the neural and anatomical connections in the nervous system (32–34). In Arnold, we use compositional embeddings to capture the unique characteristics of each sensory stream and actuator, and represent its functional role in the sensory-to-motor transformation. Each input to the transformer is identified by a list of words, drawn from a *sensorimotor vocabulary*, specifying its functional role (Figure 2). This way, a few tokens encode shared representations characterizing each sensorimotor element. We trained Arnold using a combination of IL and RL (Figure 1). During the pretraining phase, Arnold learned to solve 14 tasks from the MyoSuite library (19) in parallel by imitating expert policies from prior works (14, 24, 29) via on-policy behavior cloning (OBC) (35). Unlike standard behavior cloning (BC), which trains a policy offline using a fixed dataset of expert demonstrations, OBC allows the student policy to actively interact with the environment, while the expert provides target actions (Figure 4). This setup ensures that the student is trained on state distributions it encounters during deployment, thereby reducing the distributional shift that arises when learning from data



**Figure 1.** Arnold is a generalist musculoskeletal policy that controls multiple human embodiments and solves 14 different motor control tasks. We train Arnold in three steps. **Step 1:** Arnold learns to imitate multiple experts in parallel using on-policy behavior cloning (OBC), thus building a general foundation. **Step 2:** Different versions of Arnold are specialized to individual tasks using PPO. These specialists might outperform the original experts. **Step 3:** The Arnold model from Step 1 learns to imitate these improved versions of itself via OBC. The specialized Arnolds become the new “experts”. The original Arnold model from Step 1 is then updated using OBC to imitate these improved experts. Overall, this creates a feedback loop where Arnold learns from improved versions of itself.

collected by other policies.

Overall, our contributions are four-fold:

- We introduce Arnold, a generalist transformer policy that succeeds at 14 motor control tasks involving biologically-realistic musculoskeletal models of human body parts, and biologically-plausible sensory feedback.
- We show that OBC, unlike traditional BC, can close the gap between the student and the expert policies, while RL fine-tuning can achieve super-expert performance.
- Arnold can efficiently learn novel control tasks with a fraction of the data necessary for randomly initialized networks.
- We find that, within a single task, the muscle activations output by Arnold lie on a low-dimensional subspace of the action space, but that this subspace is not shared across tasks.

## Related work

Our research lies at the intersection of three fields: motor control in musculoskeletal systems, multi-task learning, and control of multiple embodiments with a single policy.

**Muscle control.** Controlling models of the human body through muscles has long been an active research topic, spurred by the introduction of biomechanical simulators like OpenSim (16, 17) and more recently MyoSuite (19), and featured in numerous NeurIPS competitions (25, 30, 31, 36–39). In the domain of motion imitation using reference motion capture datasets, traditional techniques such as PPO (40) and model-based learning (41) were successfully extended to musculoskeletal control (26, 29). However, significant challenges remain in RL tasks that either lack reference motion data, or involve dexterous object manipulation. To this end, recent works investigated novel exploration techniques (20, 24), learned low-dimensional action spaces (23, 27), curricula (14, 25), world models (42) and reward shaping (28). In contrast, here we develop on-policy behavior cloning (OBC) to combine the expertise of multiple specialist policies into a single generalist policy.

**From single-task to multi-task policies.** Policy distillation is commonly used to tackle the challenge of multi-task learning, whereby knowledge is transferred from task-level teacher policies into a single student policy (43, 44), or task-specific policies coordinate around a central distilled policy (45). Subsequent work on multi-task policies extended this idea by leveraging demonstrations or specialist policies to guide and improve the fine-tuning of multi-task policies. These demonstrations can either be actively selected (46), generated by RL rollouts (47, 48), derived from diverse offline data sources (49), or focused on difficult tasks (50). While most research has focused on robotic systems, Arnold tackles multi-embodiment, multi-task learning in the inherently non-linear and high-dimensional domain of musculoskeletal dynamics.

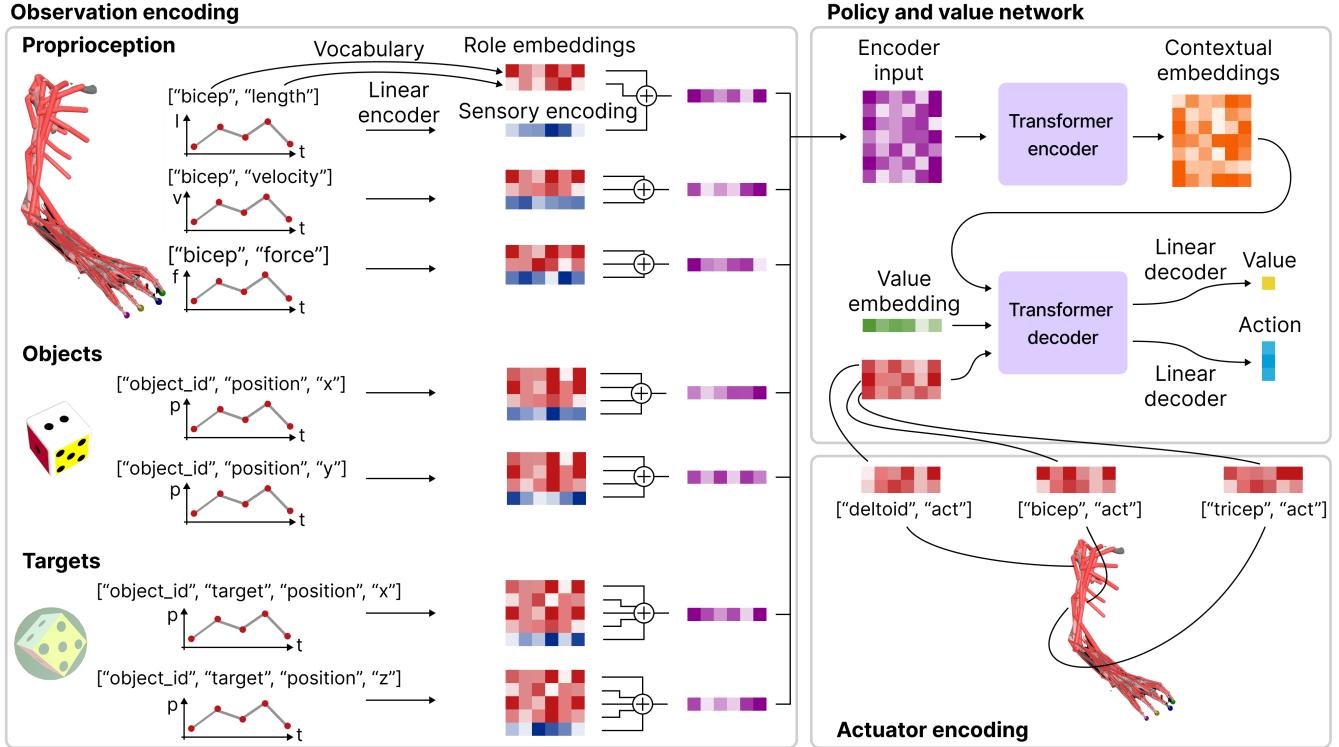
**Transformers in motor control.** Transformers have found application in reinforcement learning by modeling decision-making as a sequence-to-sequence problem (51). More relevant to our work is their capacity to flexibly process a variable number of sensory inputs and produce a corresponding set of motor commands. This architectural flexibility has been leveraged to handle variable-length and heterogeneous inputs (52–54), as well as variable embodiments (34, 55–57). Arnold is the first work

applying this idea to musculoskeletal models, providing a recipe to distill expert policies from multiple embodiments and tasks into a single agent that enables super-expert performance improvement through RL-finetuning.

## Results

**Problem formulation.** The core objective of this study is to design a controller to solve multiple Partially Observable Markov Decision Processes (POMDP) at once. Each POMDP represents a distinct motor control task, which in turn is characterized by three components: a musculoskeletal model, possible object interactions, and a task objective. Specifically, we tackle several complex, biologically-realistic motor control tasks from the MyoSuite library (19), ranging from the precision of finger movements and object-manipulation tasks, to the complex coordination required for human locomotion (Figure 1). The 14 tasks span four distinct musculoskeletal models, creating a comprehensive testbed that reflects some of the versatility of human motor control. At the simplest level, we consider precise joint control for different fingers and the elbow, escalating to the coordination required for object manipulation (Boading ball, die orient and pen reorient) using 39 hand muscles. The complexity of musculoskeletal control is exemplified in two coordination tasks: in Object Relocation, agents grasp and relocate differently-shaped objects through 63-muscle control, while in Walk to point, the goal is to locomote to arbitrary targets using an 80-muscle lower body. All tasks are described in detail in the Methods.

At each simulation step, the policy observes the proprioceptive state of the body in the form of muscle (length, velocity, force and activation) and joint (angular position and velocity) information, the configuration of the objects (position and orientation) and the task objective. This format of the proprioceptive feedback deviates from the original formulation in the MyoSuite that considers joint states (19), but is more biologically plausible (13). In all considered tasks, the objective is defined by one or more target locations or orientations, which the agent needs to reach either with body parts or with objects. The agent can interact with the environment by selecting the activation of each muscle, causing a contraction of variable intensity. The biomechanical simulator (MuJoCo (58)) transforms activations into physical forces through a Hill-type muscle model (59). Given these variables and depending on the task, the policy network needs to handle observation and action spaces of variable size and content. How can we deal with these diverse state and action spaces?



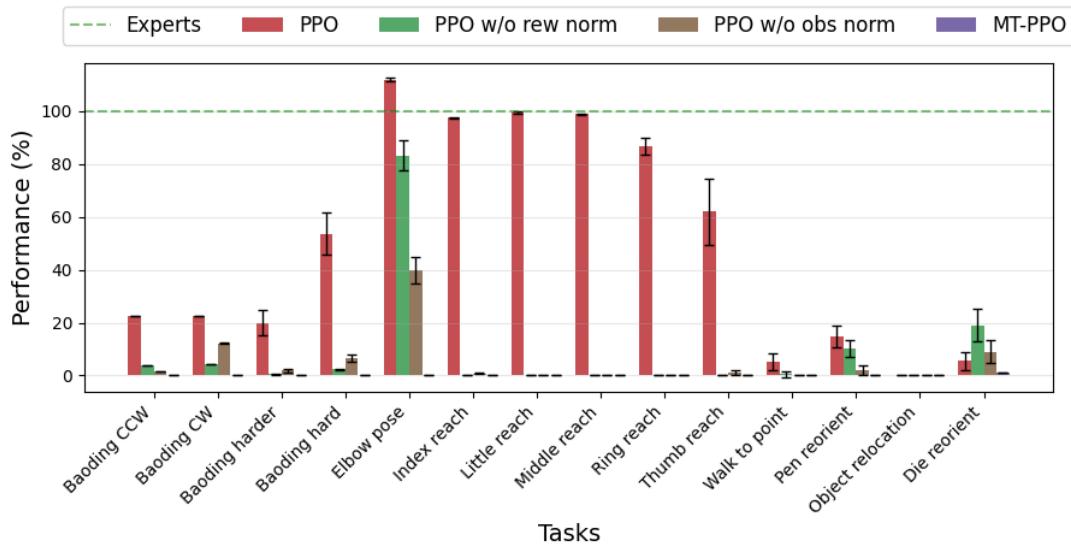
**Figure 2.** Architecture of the muscle transformer policy. Observation time series are transformed into sensory encodings by a linear transformation. Role embeddings are extracted from the sensorimotor vocabulary and added to the sensory encoding before they are processed by the transformer encoder. A transformer decoder processes actuator and value embeddings attending to the encoder’s output, to perform value estimation and the muscle actuation.

**Muscle transformer architecture.** We cast the motor control problem as a translation of sensory input into motor commands (Figure 2). Here, sensory inputs are interpreted as words in the source language, and muscle activations as words in the target language and. To learn this mapping, we leverage a generic encoder-decoder transformer model (60). Each sensory modality

(e.g., muscle length and velocity, object position and orientation) is represented by a sensory encoding. By attending to all sensory modalities and their embeddings, the encoder network can generate contextual representations of the proprioceptive state of each muscle and joint, given the task objective and the state of the objects involved in the task (Figure 2). Before feeding the sensory embeddings to the transformer encoder, a *role* embedding is added to specify the source of the sensory data. The transformer decoder receives as input one embedding per muscle and outputs one vector for each of them, attending to the encoder network’s output. Finally, the decoder’s output is linearly mapped into muscle activations. When deploying the muscle transformer with an actor-critic algorithm like PPO (40), the decoder network also receives in input a *value* embedding, which is associated with a scalar output (see Methods for details on the architecture).

We remark that this network architecture can accept any number of input sensory modalities and output a control signal for any number of actuators, making it suitable for multi-task learning for different embodiments and environments (Figure 1). We detail the network’s hyperparameters in the Methods.

**Single-task imitation experiments.** Before launching multi-task experiments, we verified that Arnold could imitate expert policies in a single task. These experts are based on previous work and include winning policies from past NeurIPS competitions (14, 25, 29, 30) (see Methods). We found that Arnold could learn all tasks individually. Single-task training confirmed the intuition that certain policies require more steps to be imitated successfully (Figure S1). Some of these tasks are extremely challenging, and the expert policies themselves were obtained only after extensive experimentation with diverse training methods. To reduce the training time in the multi-task experiments, we used this information to allocate more parallel environments to the more challenging tasks (Table S1).



**Figure 3.** Performance comparison between three variants of PPO and Multitask-PPO (MT-PPO). Reward standardization (rew norm) and per-component observation standardization (obs norm) are ablated as variants of PPO. They are essential to reach satisfying performance on the “reach” tasks, but overall PPO fails to tackle the locomotion and manipulation tasks in contrast to the experts.

d

**Training Arnold on 14 tasks from scratch (Multi-task RL).** Given the success of this architecture for single-task imitation, we sought to answer if it can learn to carry out multiple tasks with potentially different embodiments in one policy. We emphasize that muscle transformer architecture is suitable for being trained with imitation learning or reinforcement learning. A widely known issue when learning multiple tasks for current ML models is catastrophic forgetting. A simple yet effective way to train a network on multiple tasks while mitigating catastrophic forgetting is to ensure that all tasks are represented throughout the entire training process. To achieve this, we used a parallel experience collection setup, where instances of each task environment run simultaneously and populate a shared rollout buffer (Figure 1: Step 1). Among the 14 tasks, some are more difficult than others. In particular, the *Baoding balls*, *Object relocate* and *Walk to point* tasks require significantly more environment interactions. Thus, as we discussed in the previous section, we allocated more computational resources to these tasks by dedicating more parallel environments to them (Table S1).

We trained Arnold for 50 M environment interactions, plus 5 M at a reduced learning rate, for three random seeds with PPO (see Methods). We selected PPO, as it achieved strong results on many MyoSuite challenges (e.g., (14, 25, 30)). However,

in the multi-task setting we found that PPO only succeeds for the easier environments (reaching tasks), while the more difficult ones (manipulation and locomotion) the experts performed substantially stronger (Figure 3). This result was expected, because the more challenging environments required advanced training strategies beyond standard RL (specifically curriculum learning, mixture of experts, and pre-training with motion imitation) to successfully train the original expert policies (see Methods).

Interestingly, reward standardization and per-component observation standardization contributed to the limited success of PPO, as removing either caused learning to collapse (Figure 3). We also tested Multitask-PPO (40, 61) (see Methods), which augments PPO with flattened observations and an additional task embedding, but this approach struggled to learn and showed no improvement in final reward accumulation. Overall, these results highlight that straightforward RL training with PPO is insufficient to master the diverse set of Arnold’s tasks from scratch, especially the more challenging ones.

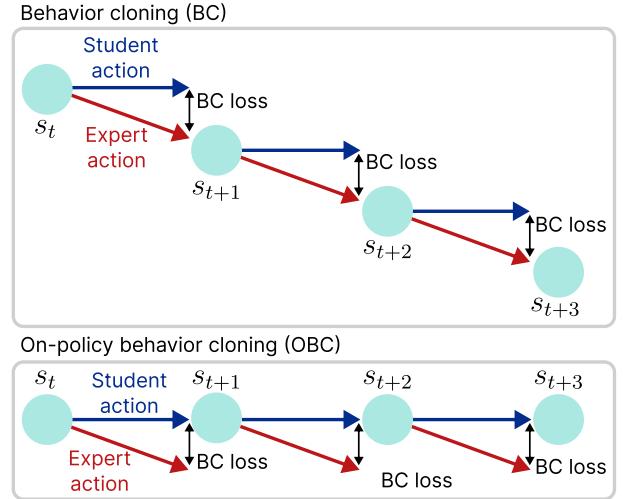
#### Training Arnold on 14 tasks (pre-training).

To address these limitations, we identified imitation learning as an effective strategy for creating a generalist policy. In standard BC (62–64), a student policy learns to mimic the actions of an expert from a dataset of state-action pairs collected from the expert’s interaction with the environment. However, the student policy often underperforms relative to the expert, primarily due to imitation error and the generalization gap (65). The latter is particularly relevant: since the training data is generated by the expert, it does not reflect the distribution of states encountered by the student during deployment. As a result, even small discrepancies in action selection can compound over time, leading the student into out-of-distribution states and significantly degrading performance. This distributional mismatch can be mitigated when both the expert policy and the environment are accessible. For example, DAgger (65) collects trajectories using the student policy and annotates them with the expert’s actions, thereby aligning the training state distribution with that of the states encountered during deployment. With OBC, we take this idea further by discarding the initial offline dataset entirely and training solely on data collected by the student policy based on expert feedback. OBC reframes imitation learning as an on-policy algorithm (Figure 4), as proposed in recent work for bidding (35). This structure naturally allows integration with on-policy reinforcement learning methods such as PPO. When a reward function is available, we can jointly optimize for both imitation and task reward by combining the behavior cloning loss with the policy gradient objective. We refer to this combined approach as OBC-PPO.

For a fair comparison, we maintained the same configuration for BC, OBC and OBC-PPO as for PPO including the imbalanced, parallel experience collection setup (see Methods). BC using expert trajectories provided stronger performance than PPO, but struggled in the Baoding Balls environments and failed at locomotion entirely (Figure 5A). OBC and OBC-PPO, instead, achieved near-expert performance in almost all tasks (Figure 5B).

**Arnold reaches super-expert performance with RL fine-tuning and self-distillation.** Expert policies might not be globally optimal, leaving room for further performance improvement. Inspired by fine-tuning techniques applied to large language models (66) and RL-finetuning with augmented loss in (67), we used the policies trained with OBC as a starting point for further RL training (Figure 1: Step 2). Fine-tuning on individual tasks enhanced performance in several of them (Figure S2). Using these improved policies as new experts, we distilled them back into the generalist policy (Figure 1: Step 3), achieving overall super-expert performance (Arnold, Figure 5A, B). Thus, we succeeded in training Arnold on 14 complex musculoskeletal tasks comprising different embodiments with expert or super-expert performance.

**What is the role of the different learning algorithm components in Arnold?** Arnold’s learning framework comprises several design components, whose importance was assessed through a loss-of function analysis (Figure 5B). Arnold achieved overall super-expert performance ( $105.13 \pm 1.60\%$  expert performance, 200 test episodes per environment, mean  $\pm$  s.e.m.). In contrast, OBC without fine-tuning achieved expert performance across all the 14 tasks ( $98.97 \pm 1.49\%$  expert performance, mean  $\pm$  s.e.m), yet significantly outperforming standard BC and PPO ( $80.89 \pm 7.49\%$  and  $50.05 \pm 10.74\%$ , respectively). Adding PPO’s surrogate policy gradient loss (OBC-PPO) only slightly boosted performance to  $99.34 \pm 1.89\%$  (not significant). Removing standardization per component signature caused a significant drop in performance for OBC ( $87.60 \pm 3.82\%$ ). The

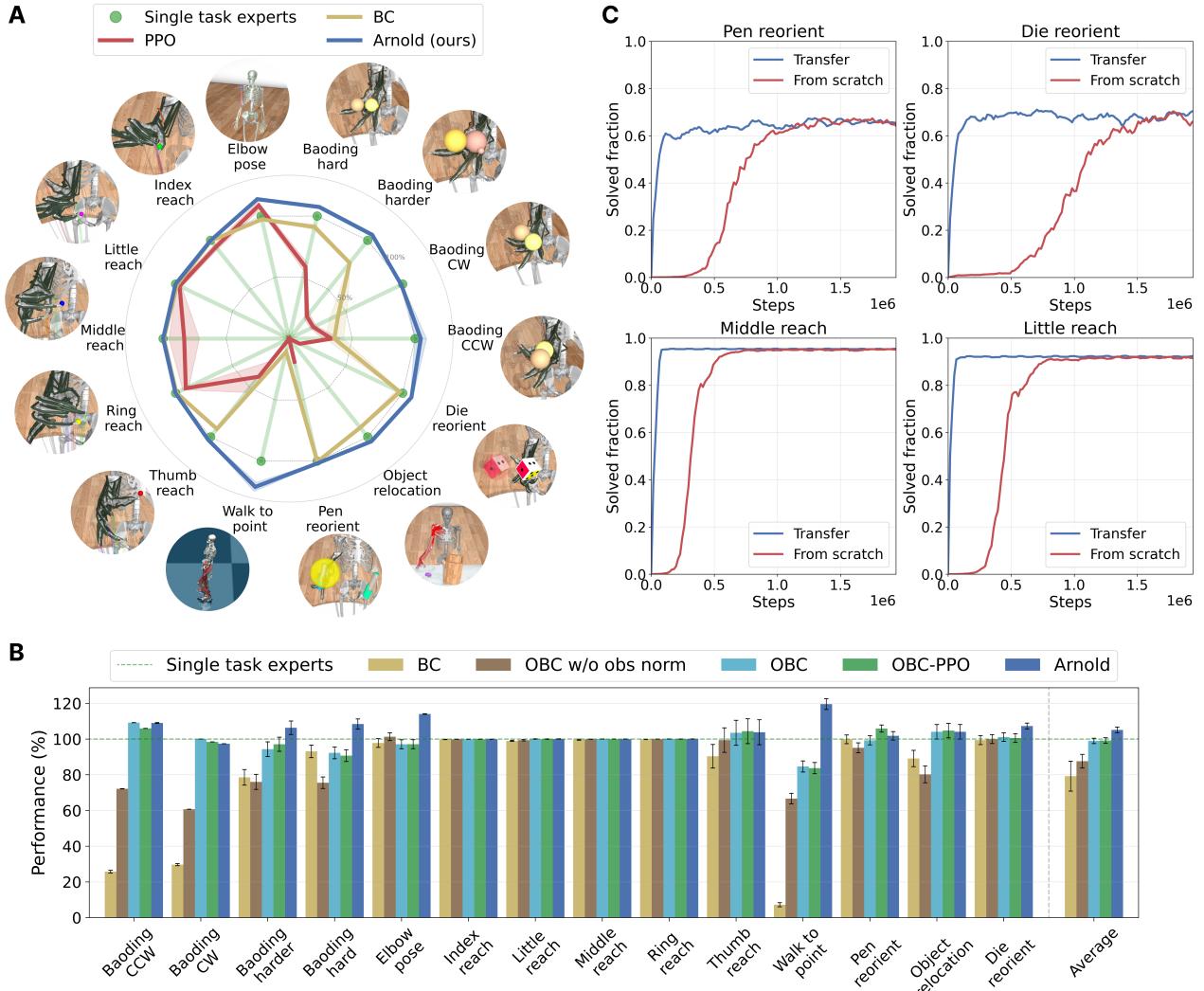


**Figure 4.** Difference between standard (top) and on-policy BC (OBC) (bottom). By using on-policy trajectories to train the student policy, we ensure that the transitions adhere to the same distribution as the ones the agent will encounter during deployment.

For a fair comparison, we maintained the same configuration for BC, OBC and OBC-PPO as for PPO including the imbalanced, parallel experience collection setup (see Methods). BC using expert trajectories provided stronger performance than PPO, but struggled in the Baoding Balls environments and failed at locomotion entirely (Figure 5A). OBC and OBC-PPO, instead, achieved near-expert performance in almost all tasks (Figure 5B).

**Arnold reaches super-expert performance with RL fine-tuning and self-distillation.** Expert policies might not be globally optimal, leaving room for further performance improvement. Inspired by fine-tuning techniques applied to large language models (66) and RL-finetuning with augmented loss in (67), we used the policies trained with OBC as a starting point for further RL training (Figure 1: Step 2). Fine-tuning on individual tasks enhanced performance in several of them (Figure S2). Using these improved policies as new experts, we distilled them back into the generalist policy (Figure 1: Step 3), achieving overall super-expert performance (Arnold, Figure 5A, B). Thus, we succeeded in training Arnold on 14 complex musculoskeletal tasks comprising different embodiments with expert or super-expert performance.

**What is the role of the different learning algorithm components in Arnold?** Arnold’s learning framework comprises several design components, whose importance was assessed through a loss-of function analysis (Figure 5B). Arnold achieved overall super-expert performance ( $105.13 \pm 1.60\%$  expert performance, 200 test episodes per environment, mean  $\pm$  s.e.m.). In contrast, OBC without fine-tuning achieved expert performance across all the 14 tasks ( $98.97 \pm 1.49\%$  expert performance, mean  $\pm$  s.e.m), yet significantly outperforming standard BC and PPO ( $80.89 \pm 7.49\%$  and  $50.05 \pm 10.74\%$ , respectively). Adding PPO’s surrogate policy gradient loss (OBC-PPO) only slightly boosted performance to  $99.34 \pm 1.89\%$  (not significant). Removing standardization per component signature caused a significant drop in performance for OBC ( $87.60 \pm 3.82\%$ ). The

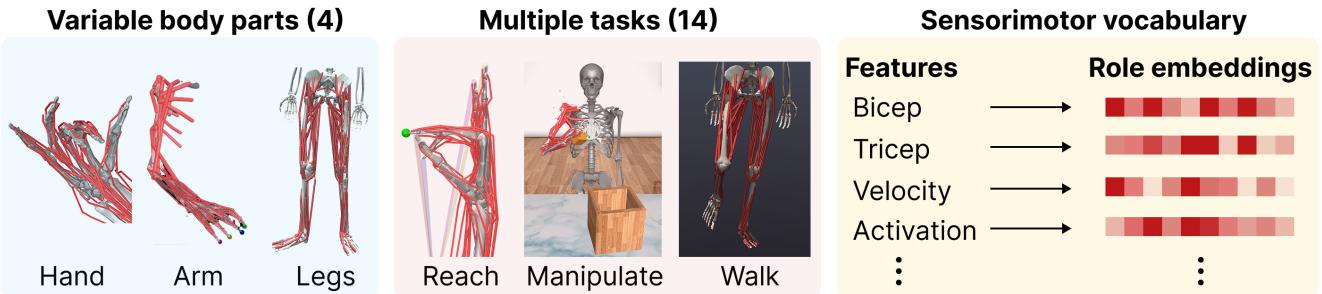


**Figure 5. A** Multi-task performance comparison between PPO, BC and Arnold, compared to the single task expert performance (mean  $\pm$  s.e.m. across 3 random seeds). **B** Single task and average performance of BC, OBC, OBC without observation normalization, OBC-PPO and Arnold (200 episodes, mean  $\pm$  standard error). **C** Learning curves of single-task training experiments, pre-trained network (blue) vs randomly initialized network (red). The pre-trained network experienced 50M steps of the 10 remaining tasks before being trained on Pen reorient, Die reorient, Middle reach and Little reach with OBC-PPO (one at a time). It learns the unseen tasks considerably faster than a random network, suggesting that it learned transferrable representations in the pre-training phase.

performance differences are mainly due to the object manipulation and locomotion environments, which are strongly affected by small errors of the control policy (Figure 5B). However, in comparison to PPO, the drop is much more mild (Figure 3).

**Arnold is more data-efficient for novel tasks.** A generalist policy should encode reusable principles of motor control in its neural network, facilitating learning motor control tasks different from the ones observed during training. Arnold's sensorimotor vocabulary further promotes shared representations across tasks. For instance, in a new reaching task not encountered during training (e.g., using a different finger), Arnold took advantage of the role embeddings learned for other tasks. We assessed the transfer ability of Arnold by training it on ten tasks with OBC and then resuming training on four held out tasks (Little reach, Middle reach, Pen reorient, Die reorient). In all the downstream tasks, the pretrained network reached the expert performance noticeably earlier than specialists (Figure 5C).

**What is the role of the learned sensorimotor vocabulary?** For multi-task learning with different embodiments and tasks, encoding the meaning of all the actuators and sensory modalities is challenging, because the number of input types to the policy network can grow quickly. This effect is especially relevant in musculoskeletal environments, which are high-dimensional to begin with. We illustrate this with the *MyoLeg* model (19) of the *Walk to point* task (29). The musculoskeletal model features 80 muscles and 28 joints, each of them associated with multiple proprioceptive feedback channels, composing (together with auxiliary variables) a 288-dimensional observation space and a 80-dimensional action space. This single task would require 368 role embeddings; combining multiple tasks in this manner is thus impractical. For this reason, we defined a compositional



**Figure 6.** Arnold's sensorimotor vocabulary includes 214 tokens that can represent the sensory input and motor output for 14 different tasks involving 4 different embodiments.

*sensorimotor vocabulary* containing a set of words that can be arranged to describe a wide variety of sensory inputs and outputs (Figure 6). We found that 214 elements are sufficient to characterize all inputs and outputs of the 14 Myosuite tasks. We combined these tokens to describe complex sensory input by summing them. For example, the role embedding of the length of the right soleus muscle is the sum of the "right", "soleus", "muscle", and "length" embeddings (see Methods). This compositional scheme allowed us to reduce the size of the sensorimotor vocabulary by taking advantage of physiological symmetries and shared semantics (e.g., lateral body symmetry and shared muscle features), promoting reusable representations for the same sensorimotor element across tasks.

Does it also have performance advantages? For comparison, we constructed a task-specific token embedding scheme based on the same vocabulary principles, in which each task maintains its own set of tokens thus minimizing the token sharing between tasks, and evaluated our compositional method against it. We found that task-specific embeddings cause a significant performance decrease compared to the compositional sensorimotor vocabulary on several tasks (Table 1), suggesting that token sharing promotes effective representations of sensory and motor components.

**Table 1.** Performance drop in success rate when using a disjoint vocabulary, compared to our proposed shared sensorimotor vocabulary for tasks with the biggest changes (↓ indicates performance drop). Results for one seed.

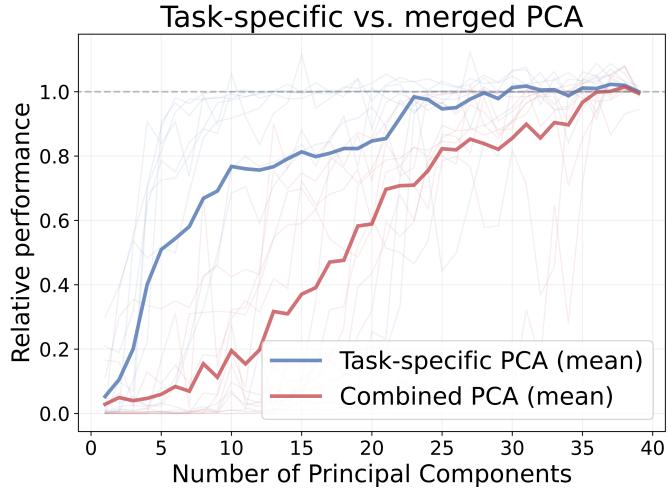
Tasks	Baoding CCW	Baoding hard	Baoding harder	Object relocation	Die Reorient
Success rate change	~35% ↓	~19% ↓	~11% ↓	~37% ↓	~10% ↓

Lastly, we also verified that a short time window (5 time steps), as we used in all our experiments, is sufficient for Arnold (see Methods and Figure S3).

**Analysis of motor universality.** Humans and many other animals embody the notion of generalist motor control. Despite their over-actuated, high-dimensional motor systems consisting of hundreds of muscles, they are capable of controlling their bodies across a wide range of motor tasks. To explain this seeming paradox, muscle synergies have been proposed as a mechanism by which the brain controls muscles through a reduced, universal motor space rather than specifying individual, task-specific activations (68–73). These synergies can be interpreted as basis vectors for muscle control—fixed activation patterns that are reused across motor tasks. While experiments have provided evidence for this mechanism in animals (68, 74, 75), artificial agents like Arnold offer a unique opportunity for systematic investigation of motor universality. Thus, creating muscle-driven generalists like Arnold is crucial for neuroscience research as biomechanics simulators also enable measures of success that go beyond the classic signal reconstruction approach in muscle synergy research (14). How can we quantify whether Arnold and other future generalist agents learn control policies that align with the muscle synergies hypothesis? Specifically, do learned policies exhibit low-dimensional muscle coordination patterns that generalize across tasks, and can we identify universal synergistic structures that emerge from multi-task training?

To investigate whether Arnold deploys transferable muscle synergies, we performed control subspace inactivation (CSI) analysis (14). We collected muscle activations generated by Arnold while solving 11 MyoHand tasks and used principal component analysis (PCA) to identify the underlying control structure (see Methods). We analyzed the data in two ways: (1) computing principal components for individual tasks separately, and (2) computing shared principal components across all tasks combined.

When projecting control signals onto task-specific principal component subspaces, we found that just a few synergies were sufficient for successful task completion (Figure 7), particularly in reaching tasks (Figure S4). This demonstrates that Arnold effectively compresses muscle activity into low-dimensional subspaces when considered within individual tasks. However, when we applied principal components computed across all tasks (Figures 7 and S4), significantly more components were required to achieve comparable performance. This stark difference reveals that while Arnold achieves dimensionality reduction



**Figure 7.** Average relative task performance across the 11 MyoHand tasks, when the control signal is projected on action subspaces generated by progressively more principal components (from 1 to 39 muscles, left to right). Blue: task-specific principal components. Red: shared principal components. Shaded lines: task-specific graphs. Plots per task can be found in Figure S4.

within tasks, these low-dimensional subspaces are highly task-specific and show limited transferability across the task set. To determine whether Arnold discovered novel transferable strategies through multi-task training, we repeated the analysis using muscle activations from single-task expert policies. Remarkably, the principal components derived from these expert policies produced similar performance profiles to those from Arnold’s activations (Figure S5). This indicates that Arnold, despite three rounds of training including self-distillation, did not leverage any transferable components beyond those already present in the expert policies.

We note that CSI is a functional measure considering relative performance as a function of dimensionality as opposed to traditional signal-based metrics such as cumulative explained variance (EV). When comparing both measures one can appreciate that EV often underestimates the functional dimensionality of the task (Figure S6).

Are these results robust across training seeds? To investigate this, we sought to compare three different relationships: (1) synergy differences between task pairs within a single Arnold checkpoint, (2) differences between Arnold checkpoints trained with different random seeds, and (3) differences between Arnold checkpoints and expert checkpoints. To assess these relationships we used Projected Variance Difference (PWD) and Principal Angle Difference (PAD) measures (see Methods). The results revealed that task-specific differences dominate over policy differences (Figure S7), indicating that the policy has learned distinct muscle activation patterns tailored to specific manipulation requirements. In contrast, differences between Arnold checkpoints and between Arnold and expert policies were substantially smaller (Figure S7), suggesting that: (1) the fundamental motor control structure remains consistent between Arnold checkpoints and expert checkpoints, and (2) Arnold’s learned synergies are closer to expert patterns than tasks are to each other.

Overall, this suggests two key insights: First, the fundamental motor control structure remains consistent between Arnold and expert checkpoints. Second, Arnold’s learned synergies are closer to expert patterns than tasks are to each other, confirming that the policy has learned appropriate task-specific solutions rather than discovering a universal motor representation (even after self-distillation).

## Discussion

We introduced Arnold, a generalist policy for musculoskeletal motor control. Our results show that a single monolithic agent can achieve expert-level performance across diverse tasks involving different body parts, objects, and objectives with a sensorimotor vocabulary. This required adapting behavior cloning, self-distillation and carefully designing a multi-task learning framework.

**Relevance for artificial embodied intelligence.** This study tackles the problem of multi-task learning in realistic motor control environments, with complex musculoskeletal systems, high-dimensional actuation, and rich sensory input. This setting presents unique challenges, requiring specialized techniques to discover viable policies (20, 23–27, 29–31). We sidestepped this problem by leveraging the collective effort of previous research and we used policies trained with methods such as curriculum learning and motion imitation as teachers for Arnold. Thanks to this approach, we reduced the challenge to distilling multiple policies into a single generalist network. We showed that naive behavior cloning led to poor results in those environments where precision and dexterity are paramount. In contrast, OBC achieved expert performance in all the considered tasks. We exceeded expert performance by fine-tuning Arnold with PPO on individual tasks and distilling it back into a unified network. Arnold also mastered novel tasks after observing few expert demonstrations, suggesting that multi-task learning is a viable strategy to encapsulate transferrable principles of motor control in a policy network. We believe that this framework can scale to richer task sets and more complete biomechanical models, advancing the development of embodied artificial intelligence with human-level motor skills.

**Relevance for neuroscience.** A hallmark of natural embodied intelligence is the ability to quickly adapt and transfer motor strategies to new tasks and domains (76, 77). Here, we considered muscle synergies as a framework to characterize the compression of these strategies onto generalizable low-dimensional components. We found evidence of compression in individual motor tasks, but not of transferability. We hypothesize that these results are due to Arnold’s computational capacity to faithfully imitate the single-task experts during OBC, without the need to uncover universal strategies, and that this universality may arise as a necessary property when we scale up the number and complexity of tasks.

Indeed, this limitation may stem from the inherent dissimilarity of the tasks we considered. The manipulation tasks may be too diverse for advantageous knowledge transfer, suggesting that either more similar task sets or different training approaches may be needed to achieve true motor universality. The absence of emergent transferable synergies, despite the multi-task setup shown here, raises important questions about the conditions necessary for universal motor representations to develop in artificial systems, and whether the muscle synergy hypothesis as observed in biological systems requires additional constraints or mechanisms not captured in our current training paradigm.

Animals, including humans, are generalist agents, excelling at diverse tasks such as locomotion and manipulation. How their biological neural networks within their environment achieve this feat (33) is a yet-unresolved puzzle, considering how hard it is for artificial agents to perform in novel environments or conditions. Recent studies investigated how flexible and compositional representations may emerge in the brain by training artificial neural networks on multiple abstract tasks (78–80). Our proposed embodied multi-task framework and the learned generalist policy can be used to extend and challenge these ideas in the context of motor control. Arnold’s architecture draws loose inspiration from cortico-striatal circuits, which are thought to provide a universal computational framework where specialization emerges from input patterns rather than architectural differences (81, 82). By implementing learned role-embeddings, Arnold demonstrates how a single neural architecture can develop diverse motor competencies. However, as we discussed, Arnold did not develop the transferable motor synergies one might expect from this biological inspiration. While the application of generic transformers to motor control has not been without criticism (77), here we focus on the compositional capacity of transformers rather than draw parallels between artificial and biological implementation details.

Interestingly, the distilled model (Arnold) receives more biologically plausible feedback than the original policies, yet maintains equivalent or superior performance. This represents a significant advantage over typical RL approaches in sensorimotor modeling, which rely on abstract joint positions and velocities rather than realistic proprioceptive signals based on muscle states—as is standard in environments like MyoSuite. Our approach therefore offers excellent potential for developing biologically plausible models of the proprioceptive system, complementing direct modeling approaches (12, 13, 15). Furthermore, generalist policies like Arnold enable the development of task-driven neural networks to reverse-engineer sensorimotor control. Indeed, we believe that the sensorimotor vocabulary provides a principled framework for identifying which inductive biases to build into neural architectures of the sensorimotor system.

**Limitations and future work.** While Arnold surpassed expert performance using RL, it struggled to discover new control policies without demonstrations. Training with PPO from scratch only succeeded at the simpler tasks of the benchmark, confirming that tasks such as *Object relocation* and *Walk to point* require carefully crafted curricula or reference motions (14, 29, 30). These results further motivate future research in skill discovery, proposing exploration techniques that generate new task-relevant behaviors.

The Fitts and Posner model describes skill acquisition through three distinct stages: cognitive, associative, and autonomous (83). Currently, Arnold represents an interesting contrast to this framework—by self-distilling from expert policies (some developed through cognitive insights and reinforcement learning (14)), Arnold effectively bypasses the cognitive stage and operates primarily in the associative and autonomous phases of skill learning. Future work could systematically explore this computational analogy for musculoskeletal motor skills.

While Arnold, trained on ten different tasks, proved to be a better starting point for learning unseen tasks than random initialization, it was not able to solve these tasks zero-shot (Figure 5C). Including more training tasks might lead to zero-shot performance improvement, but at this stage the policy is not yet able to quickly retarget previously learned skills to novel tasks. This is in contrast with the adaptation ability of humans and animals, which can often solve novel tasks without learning (76, 84). Whether this is due to the large prior experience or to specific mechanisms of biological motor control is an open research question. Notably, all results in this work were achieved with only around 55M training iterations (approximately 5.5 days of compute with CPU-based simulator), highlighting both the efficiency of our approach and the potential for even greater gains under larger-scale training.

Overall, Arnold establishes a rigorous foundation for the development of a generalist agent for musculoskeletal control with human-level motor control skills. Future iterations of this work will aim to ease interaction with Arnold, e.g., via natural language (29) and contrast its computation to neural dynamics (85). We envision a future in which motor control studies will be paired with simulation experiments, without the need of machine learning expertise, opening novel avenues for neuroscientific research.

**Acknowledgments:** We thank members of the Mathis Group for Computational Neuroscience and AI for helpful feedback. This work was funded by the EPFL School of Life Sciences Summer Research Program (B.A.) and Swiss SNF grant (310030\_212516).

## Methods

Here, we formalize the multi-task control problem and describe the implementation of the muscle transformer policy. In particular, we detail how Arnold can deal with variable observation and action spaces and outline the implementation of OBC as well as other learning algorithms for multiple tasks.

**Musculoskeletal tasks and experts.** In all our experiments, we considered the musculoskeletal models of the MyoSuite library (19). MyoSuite is implemented in MuJoCo (58) and features biologically-realistic models of multiple body parts ported from OpenSim (16, 17). The advantage of MyoSuite compared to OpenSim is its speed, as it simulates these models three orders of magnitude faster, enabling our large-scale experiments.

The 14 tasks featured in this work are based on four different musculoskeletal models (Figure 1):

**MyoElbow** is the simplest model of the four, with six muscles and one joint. The only task posed for this model involves reaching a target elbow angle (Elbow pose). The expert for this task was trained by Chiappa et al. (24) with Lattice-PPO.

**MyoHand** comprises a detailed model of a human hand, with 39 muscles and 23 joints, featured in eleven tasks. Five of them consist in reaching a randomly sampled point with a finger. We trained single task experts using Lattice-PPO (24, 40). The other six are object manipulation tasks. In *Pen reorient* and *Die reorient*, the hand is supposed to rotate a pen or a die, respectively, to reach a target pose.

The experts for these tasks are from Chiappa et al. (24). The last four tasks involve the rotation of two Baoding balls, clockwise and counter-clockwise, with progressively more randomness in the prescribed trajectories and in the physical properties of the balls (*hard* and *harder*). The experts for the Baoding tasks are the winning solutions to the 2022 NeurIPS MyoChallenge (25). A targeted curriculum learning strategy resembling part-to-whole practice was developed for this purpose (14).

**MyoArm** is an extension of the MyoHand that includes the upper arm, pectoral and shoulder muscles. It has 63 muscles and 38 joints. *Relocate* is implemented using this model. The agent needs to grasp an object of a dynamically generated shape, lift it and place it inside a box.

The expert is the winning solution of the 2023 NeurIPS MyoChallenge (30). We leveraged Lattice-PPO as well as a curriculum learning strategy.

**MyoLeg** is a model of the human lower body with 80 muscles and 27 joints. It is used in the *Walk to point* task, in which the agent has to walk towards a random target location. Training an expert involved imitation of human motion data, a mixture of experts architecture in addition to RL fine-tuning (29).

Furthermore, we provide additional details on each task and environment. Firstly, the task descriptions:

- Baoding CCW: the muscle hand needs to rotate two baoding balls counter-clockwise. The initial phase and target rotation speed is always the same.
- Baoding CW: the muscle hand needs to rotate two baoding balls clockwise. The initial phase and target rotation speed is always the same.
- Baoding hard: the muscle hand needs to rotate two baoding balls. The initial phase is always the same. The rotation direction and target rotation speed is variable.
- Baoding harder: the muscle hand needs to rotate two baoding balls. The initial phase, rotation direction and target rotation speed is variable.
- Elbow pose: the elbow can rotate to point the hand to a random target location.
- Index reach: the muscle hand needs to point the tip of the index finger to a random target location.
- Little reach: the muscle hand needs to point the tip of the little finger to a random target location.
- Middle reach: the muscle hand needs to point the tip of the index finger to a random target location.
- Ring reach: the muscle hand needs to point the tip of the ring finger to a random target location.
- Thumb reach: the muscle hand needs to point the tip of the thumb to a random target location.
- Walk to point: the muscle leg needs to move the body to a random location without falling.
- Pen reorient: the muscle hand needs to rotate a pen (cylinder) to a random desired orientation.
- Object relocation: the muscle hand and arm together need to pick-up an random object and put it into a box.

- Die reorient: the muscle hand needs to rotate a die (cube) to a random desired orientation.

Secondly, we provide details on the task and reward parameters (Table 2 and Table 3).

**Table 2.** Task and reward parameters of Elbow pose, Finger reach, Pen reorient, Die reorient and Object relocation.

Task	Elbow pose Parameter	Value	Finger reach Parameter	Value	Pen reorient Parameter	Value	Die reorient Parameter	Value	Object relocation Parameter	Value
Specification	Max steps	100	Max steps	100	Max steps	100	Max steps	150	Max steps	150
	Pose th.	.175	Pose th.	.35	Frame skip	5	Pos th.	inf	Pos th.	.1
	Target dst.	1					Rot th.	.262	Rot th.	inf
							Goal pos.	(0, 0)	Qpos noise	.01
							Goal rot.	-.785 ~ .785	Tgt. xyz	(0, -.45, .9) ~ (.3, -.1, 1.05)
							Frame skip	5	Tgt. rpy	(-.2, -.2, -.2) ~ (.2, .2, .2)
									Obj. xyz	(-.1, -.35, 1) ~ (.1, -.15, 1)
									Obj. geom.	(.015, .015, .015) ~ (.025, .025, .025)
									Obj. mass	.05 ~ .2
									Obj. fric.	(.8, .004, .00008) ~ (1.2, .006, .000012)
Reward	Pose	1	Reach	1	Pos align diff	100	Pos dist.	2	Solved	20
	Solved	1	Solved	1	Rot align diff	100	Rot dist.	.2	Pos dist.	10
	Action reg.	1	Action reg.	1	Alive	1	Pos align diff	100	Palm dist.	1
					Solved	1	Rot align diff	10	Tip dist.	.1
							Alive	1		
							Solved	2		

**Table 3.** Task and reward parameters of Baoding CW, Baoding CCW, Baoding hard, Baoding harder and Walk to point.

Task	Baoding CW Parameter	Value	Baoding CCW Parameter	Value	Baoding hard Parameter	Value	Baoding harder Parameter	Value	Walk to point Parameter	Value
Specification	Max steps	200	Max steps	200	Max steps	200	Max steps	200	Max steps	150
	Goal period	5	Goal period	5	Goal period	4 ~ 6	Goal period	4 ~ 6		
	Init. phase	1.57	Init. phase	1.57	Init. phase	1.57	Init. phase	1.57		
	Lim. init. angle	0	Lim. init. angle	0	Lim. init. angle	0	Lim. init. angle	3.14		
				0	Goal x	.02 ~ .03	Goal x	.02 ~ .03		
					Goal y	.022 ~ .032	Goal y	.022 ~ .032		
					Obj. size	.018 ~ .024	Obj. size	.018 ~ .024		
					Obj. mass	.03 ~ .3	Obj. mass	.03 ~ .3		
					Obj. fric.	(.2, .001, .00002)	Obj. fric.	(.2, .001, .00002)		
Reward	Pos dist. 1	1	Pos dist. 1	1	Pos dist. 1	1	Pos dist. 1	1	W-energy	.1
	Pos dist. 2	1	Pos dist. 2	1	Pos dist. 2	1	Pos dist. 2	1	W-upright	.1
	Alive	1	Alive	1	Alive	1	Alive	1	K-energy	.05
	Solved	5	Solved	5	Solved	5	Solved	5		

**Muscle transformer architecture.** Each sensory modality (e.g., muscle length and velocity, object position and orientation) is represented by a short time series (5 steps), mapped into a sensory embedding of a fixed size by a linear encoder. We adopted this simple, yet powerful solution leveraging encoder-decoder transformers (60) because we found no advantage in using attention over time. In order to prove that the simple approach of only including a short time series (5 steps) in the observation is enough, we tested modified versions of Arnold (Figure S3). This model is identical to the original model except that it has an additional layer of attention over time. Attention allows the model to consider a history of 96 timesteps. The results showed that although the modified version did better on some of the tasks for randomized experts (expert actions are sampled from the random distribution the expert outputs), it is not comparable with the original method when the experts are fixed. The reason may be that, with a larger history, the model can learn to better denoise when learning from randomized experts, resulting in a better reward. But the tasks themselves itself might not require as much history to achieve a high reward, thus the increased model complexity hinders the training when learning from deterministic experts.

Thus, Arnold’s architecture consists of a transformer encoder-decoder network. While the encoder network is shared, two separate decoders output the action and the state value (Table 4).

When deploying the muscle transformer with an actor-critic algorithm like PPO (40), the decoder network also receives in input a *value* embedding, which is associated with a scalar output. We remark that this network architecture can accept any number of input sensory modalities and output a control signal for any number of actuators, making it suitable for multi-task learning for different embodiments and environments (Figure 1). We detail the network’s hyperparameters in Table 4.

We remark that we did not implement Q-value estimation, because it is not required by PPO, which we used in our experiments. However, we hypothesize that it could be implemented in a similar way, either by running the encoder with additional action input (corresponding to the action whose Q-value needs to be estimated), or directly feeding it to the decoder. In discrete action spaces, instead, a method such as Q-learning could be implemented by providing tokens for each admissible action as input to the decoder.

**Table 4.** Transformer hyperparameters and parameter counts

Hyperparameter	Encoder	Action Decoder	Value Decoder
Embedding size	128	128	128
Feedforward dimension	512	512	512
Number of heads	4	4	4
Number of layers	6	6	6
Dropout	0	0	0
Activation	ReLU	ReLU	ReLU
LayerNorm $\epsilon$	$10^{-5}$	$10^{-5}$	$10^{-5}$
Pre-norm (norm_first)	True	True	True
#Parameters	1,189,888	1,587,584	1,587,584
<b>Additional Components</b>			
Tokenizer (Linear)	896		
Positional Encoder (Embedding)	27,392		
Action Net	129		
Value Net	129		
Log Std Net	129		
Total additional parameters	28,676		
<b>Total Parameters</b>	4,393,732		

**Sensorimotor vocabulary.** Transformer networks are positionally invariant, meaning that the order of the input vectors does not change the output values. Typically, positional encoding is used to break positional invariance (60, 86–88). In our setting, the observation values denote sensory elements which vary in number and ordering across tasks (Figure 1). The policy network needs to distinguish between these elements, which represent different sensory modalities and anatomical locations. To this end, we propose the use of learnable input-specific role embeddings to encode the identity of the data source. As highlighted in the main text, we created a sensorimotor vocabulary for this purpose. The vocabulary used for encoding sensory and motor tokens (Figure 6) is organized into semantically meaningful categories:

- **Muscle Names**

- *Finger muscles*: extn, adabR, adabL, mflx, dflx, FDS2–5, FDP2–5, EDC2–5, EDM, EIP, EPL, EPB, FPL, APL, OP
- *Hand/forearm muscles*: ECRL, ECRB, ECU, FCR, FCU, PL, PT, PQ, RI2–RI5, LU\_RB2–LU\_RB5, UI\_UB2–UI\_UB5
- *Elbow muscles*: TRIIlong, TRIIlat, TRIImed, BIClong, BICshort, BRA
- *Upper arm/shoulder muscles*: DELT1–3, SUPSP, INFSP, SUBSC, TMIN, TMAJ, PECM1–3, LAT1–3, CORB, ANC, SUP, BRD
- *Leg muscles*: addbrev, addlong, addmagDist, addmagIsch, addmagMid, addmagProx, bflh, bfsh, edl, ehl, fdl, fhl, gaslat, gasmed, glmax1–3, glmed1–3, glmin1–3, grac, iliacus, perbrev, perlong, piri, psoas, recfem, sart, semimem, semiten, soleus, tfl, tibant, tibpost, vasint, vaslat, vasmed

- **Joint Names**

- *Finger joints*: IFadb, IFmcp, IFpip, IFdip, cmc\_abduction, cmc\_flexion, mp\_flexion, ip\_flexion, mcp2–5\_flexion, mcp2–5\_abduction, pm2–5\_flexion, md2–5\_flexion
- *Hand/wrist joints*: r\_elbow flex, pro\_sup, deviation, flexion
- *Arm joints*: sternoclavicular\_r2–r3, unrotscap\_r2–r3, acromioclavicular\_r1–r3, unrothum\_r1–r3, elv\_angle, shoulder\_elv, shoulder1\_r2, shoulder\_rot, elbow\_flexion
- *Leg joints*: hip\_flexion, hip\_adduction, hip\_rotation, knee\_angle, knee\_angle\_translation1–2, knee\_angle\_rotation2–3,

knee\_angle\_beta\_translation1–2, knee\_angle\_beta\_rotation1,  
 ankle\_angle, subtalar\_angle, mtp\_angle

- Targets and Contact Points
  - THtip, IFtip, MFtip, RFtip, LFtip, pelvis, foot, toes, root, contact
- Object IDs and Types
  - *Object types*: object, pen, ball, die
  - *Object IDs*: id\_1–10
- Semantic Tokens
  - muscle, joint, tip, target, error, value, position, velocity, acceleration, force, activation, linear, angular
- Coordinate and Direction Terms
  - x, y, z, OBJTx, OBJTy, OBJTz, OBJRx, OBJRy, OBJRz, left, right
- Special Tokens
  - padding

**Training details.** All training experiments were carried out on a GPU cluster featuring A100 GPUs. We could run two training experiments in parallel on a single node, to maximally utilize the GPU memory.

- We ran PPO, BC, OBC and OBC-PPO using the same network architecture and hyperparameters (Table 5).
- OBC-PPO combines imitation loss (L2), the surrogate policy gradient loss of PPO, the value loss, and an entropy loss to encourage exploration.
- PPO, instead, sets the imitation loss coefficient to 0, while OBC sets the policy gradient and entropy coefficients to 0.
- For BC, we used the same setup as OBC, but collected the rollouts suing to the expert policy’s actions.

All training runs consisted of 50M steps at fixed learning rate. The policy gradient objective does not pair well with a large learning rate, thus we reduced it for OBC-PPO and PPO. After the first 50M steps, we ran BC, OBC and OBC-PPO for an additional 5M steps with a smaller learning rate, which boosted the performance in certain tasks. We did not achieve the same improvement with PPO. Due to the low performance of Multitask-PPO (40, 61) after 50M steps, we did not train it for an additional 5M steps. Each of these experiments was performed for three different random seeds. One full PPO run required approximately 4 days, while a run of OBC, OBC-PPO or BC required approximately 5 days. The overhead was due to the computation of the expert actions.

**Multi-task learning.** Training a policy across diverse environments poses unique challenges, requiring a flexible architecture and careful handling of observations, rewards, and action variability. Below, we outline the key components of our approach.

**Experience collection.** To expose the policy to diverse transitions at each training step, we execute all tasks in parallel (Figure 1: Step 1). Each task is assigned to a separate process, which returns one observation per step. Observations are padded to enable batching, passed through the policy, and the resulting actions are routed back to their respective environments. Transitions are stored in a shared rollout buffer, containing exclusively data generated by the current policy.

**Observation standardization.** Standardizing inputs (zero mean, unit variance) often accelerates learning. In RL, this requires maintaining running statistics over observed states, as no fixed dataset exists. In a multi-task setting, the meaning of observation components varies across tasks, so we maintain normalization statistics per *observation signature*, i.e., a descriptor of the sensory modality. For example, all ["right", "soleus", "muscle", "length"] inputs share the same statistics regardless of task. This method improves both learning efficiency and final performance (Figures 3 and 5B), while preventing the number of standardization coefficients from increasing excessively.

**Reward standardization.** Different reward ranges across tasks complicate learning a unified value function. We normalize returns per task by using task-specific mean and standard deviation of cumulative rewards, thus ensuring stable value targets. While a task-specific reward standardization requires task identifiers at training time, the policy remains agnostic to task identity during deployment, where rewards are not provided. Removing this normalization significantly impairs PPO training (Figure 3). Although OBC does not require a value function, learning it facilitates subsequent fine-tuning with PPO.

**Table 5.** Hyperparameters across different training methods

Hyperparameter	PPO	BC	OBC	OBC-PPO	RL Fine-tuning
Initial standard deviation	1.0	1.0	1.0	1.0	1e-3
Batch size	128	128	128	128	128
Entropy coefficient	1e-6	0.0	0.0	1e-6	1e-6
Value function coefficient	0.5	0.5	0.5	0.5	0.5
Policy gradient coefficient	1.0	0.0	0.0	1.0	1.0
Imitation coefficient	0.0	1.0	1.0	1.0	0.0
Initial learning rate	2e-5	1e-3	1e-3	1e-4	2e-6
Reduced learning rate	n/a	1e-5	1e-5	1e-5	n/a
Rollout steps	512	512	512	512	512
Training epochs	3	3	3	3	3
Expert rollout	n/a	Yes	No	No	n/a
Discount factor (gamma)	0.99	0.99	0.99	0.99	0.99
GAE lambda	0.95	0.95	0.95	0.95	0.95
Clip range	0.2	0.2	0.2	0.2	0.2
Max gradient norm	0.5	0.5	0.5	0.5	0.5
Standardize advantage	True	True	True	True	True
Standardize observations	True	True	True	True	True
Imitation loss	n/a	MSE	MSE	MSE	n/a

**Task-specific exploratory noise.** PPO uses a stochastic policy where each action component is sampled from a Gaussian distribution (40). Typically, standard deviations are learned via backpropagation, to adapt their magnitude to the sensitivity of each component to noise and to the component’s scale. However, in multi-task settings, the action semantics vary. To address this, the policy outputs each action component’s standard deviation as  $\sigma = \tilde{\sigma} \text{sm}(\mathbf{x}^\top E)N_A$ , where  $\tilde{\sigma} \in \mathbb{R}$  is a global learnable scalar,  $\text{sm}$  denotes softmax,  $E \in \mathbb{R}^{N_T \times N_A}$  are the transformer decoder embeddings and  $\mathbf{x} \in \mathbb{R}^{N_T}$  maps each embedding to a noise magnitude. Here,  $N_T$  denotes the embedding size and  $N_A$  the number of actuators. This formulation allows the policy to scale exploratory noise per component according to the task-specific action semantics.

**RL fine-tuning and self-distillation.** Expert policies might not be globally optimal, leaving room for further performance improvement. Inspired by fine-tuning techniques applied to large language models (66) and RL fine-tuning with augmented loss (67), we use a policy trained with OBC as a starting point for one RL training per task (Figure 1: Step 2). This fine-tuning phase generates multiple new policies, some featuring super-expert performance in specific tasks (Figure fig:performanceA,B). We use these new policies as experts for multi-task learning with OBC, repeating the previous distillation procedure into the original Arnold policy, thus achieving overall super-expert performance with a single policy (Figure 1: Step 3).

For the fine-tuning phase of Arnold, we resumed the training of OBC after 55M steps, replacing the imitation loss with the policy gradient and entropy losses (the value loss was active at all times). In the fine-tuning experiments, OBC was trained on a single task at a time, generating one new expert per task. Fine-tuning improved the performance compared to the expert in Walk to point, Elbow pose, Baoding hard, Baoding harder, and Die reorient (Figure S2). Crucially, in order to achieve a performance improvement we had to reset the standard deviation of the action distribution to  $10^{-3}$  (during the pre-training phase it had approached 0) and reduce the learning rate to  $2 \times 10^{-6}$ . We then replaced the previous experts with these improved versions and ran additional 10M steps of OBC, leading to the final Arnold policy.

**Muscle synergy analysis and Control subspace inactivation (CSI).** We collected muscle activations generated by Arnold while solving the 11 tasks involving the MyoHand model, and we used this dataset to find the principal components of the muscle activation signal. We ran PCA both on the muscle activations of the 11 tasks together and on the activations of individual tasks. We ran 100 episodes per task projecting the control signal on a reduced subspace, spanned by the  $N$  most important principal components for  $N$  up to 39, the number of muscles of MyoHand (Figures 7 and S4).

Details for CSI and the EV analysis can be found in Chiappa et al. (14).

To compare pairs of subspaces (Figure S7), we used Projected Variance Difference (PWD) and Principal Angle Difference (PAD) measures, which we adopted from Todorov and Ghahramani (89).

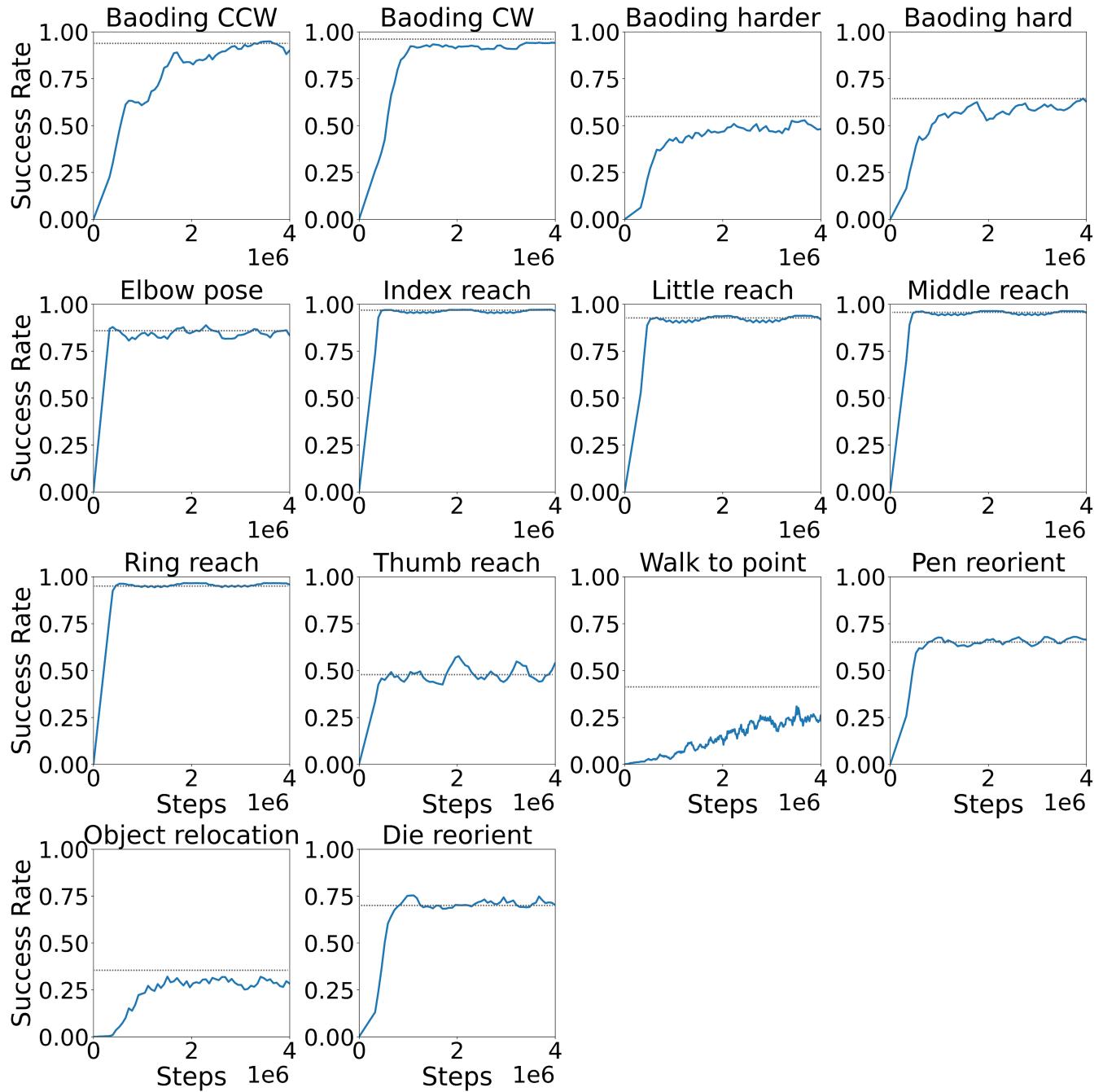
1. Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.
2. Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.
3. S. Jacobsen, E. Iversen, D. Knutti, R. Johnson, and K. Biggers. Design of the Utah/M.I.T. Dextrous Hand. In *1986 IEEE International Conference on Robotics and Automation Proceedings*, volume 3, pages 1520–1532, April 1986. doi: 10.1109/ROBOT.1986.1087395.
4. Vikash Kumar, Zhe Xu, and Emanuel Todorov. Fast, strong and compliant pneumatic actuation for dexterous tendon-driven hands. *arXiv preprint arXiv:2013.6630771*, 2013.
5. Oncay Yasa, Yasunori Toshimitsu, Mike Y. Michelis, Lewis S. Jones, Miriam Filippi, Thomas Buchner, and Robert K. Katzschmann. An Overview of Soft Robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 6(Volume 6, 2023):1–29, May 2023. ISSN 2573-5144. doi: 10.1146/annurev-control-062322-100607.
6. Kenneth Shaw, Ananya Agarwal, and Deepak Pathak. LEAP Hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *arXiv preprint arXiv:2309.06440*, 2023.
7. Allegro Hand | robot hand. <https://www.allegrohand.com>.
8. Clemens C. Christoph, Maximilian Eberlein, Filippos Katsimalis, Arturo Roberti, Aristotelis Sympetheros, Michel R. Vogt, Davide Liconti, Chenyu Yang, Barnabas Gavin Cangan, Ronan J. Hinchet, and Robert K. Katzschmann. Orca: An open-source, reliable, cost-effective, anthropomorphic robotic hand for uninterrupted dexterous task learning. *arXiv preprint arXiv:2504.04259*, 2025.
9. Jun Zhang, Jun Sheng, Ciarán T. O'Neill, Conor J Walsh, Robert J Wood, Jee-Hwan Ryu, Jaydev P Desai, and Michael C Yip. Robotic artificial muscles: Current progress and future perspectives. *IEEE transactions on robotics*, 35(3):761–781, 2019.
10. Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.
11. Isabell Wochner, Pierre Schumacher, Georg Martius, Dieter Büchler, Syn Schmitt, and Daniel Haeufle. Learning with Muscles: Benefits for Data-Efficiency and Robustness in Anthropomorphic Tasks. In *Proceedings of The 6th Conference on Robot Learning*, pages 1178–1188. PMLR, March 2023.
12. Kai J Sandbrink, Pranav Mamidanna, Claudio Michaelis, Matthias Bethge, Mackenzie W Mathis, and Alexander Mathis. Contrasting action and posture coding with hierarchical deep neural network models of proprioception. *Elife*, 12:e81499, 2023.
13. Marin Vargas, Alessandro, Axel Bisi, Alberto S. Chiappa, Chris Versteeg, Lee E. Miller, and Alexander Mathis. Task-driven neural network models predict neural dynamics of proprioception. *Cell*, 187(7):1745–1761.e19, March 2024. ISSN 0092-8674, 1097-4172. doi: 10.1016/j.cell.2024.02.036.
14. Alberto Silvio Chiappa, Pablo Tano, Nisheet Patel, Abigail Ingster, Alexandre Pouget, and Alexander Mathis. Acquiring musculoskeletal skills with curriculum-based reinforcement learning. *Neuron*, 112(23):3969–3983.e5, December 2024. ISSN 0896-6273. doi: 10.1016/j.neuron.2024.09.002.
15. Adriana Perez Rotondo, Merkourios Simos, Florian David, Sebastian Pigeon, Olaf Blanke, and Alexander Mathis. Deep-learning models of the ascending proprioceptive pathway are subject to illusions. *Experimental Physiology*, 2025.
16. Scott L Delp, Frank C Anderson, Allison S Arnold, Peter Loan, Ayman Habib, Chand T John, Eran Guendelman, and Darryl G Thelen. OpenSim: Open-source software to create and analyze dynamic simulations of movement. *IEEE transactions on biomedical engineering*, 54(11):1940–1950, 2007.
17. Ajay Seth, Michael Sherman, Jeffrey A Reinbolt, and Scott L Delp. OpenSim: A musculoskeletal modeling and simulation framework for in silico investigations and exchange. *Procedia lutam*, 2:212–232, 2011.
18. Thomas Geijtenbeek. Scone: Open source software for predictive simulation of biological motion. *Journal of Open Source Software*, 4(38):1421, 2019.
19. Vittorio Caggiano, Huawei Wang, Guillaume Durandau, Massimo Sartori, and Vikash Kumar. MyoSuite: A contact-rich simulation suite for musculoskeletal motor control. In *Learning for Dynamics and Control Conference*, pages 492–507. PMLR, 2022.
20. Pierre Schumacher, Daniel Haeufle, Dieter Büchler, Syn Schmitt, and Georg Martius. DEP-RL: Embodied exploration for reinforcement learning in overactuated and musculoskeletal systems. In *The Eleventh International Conference on Learning Representations*, 2022.
21. Vittorio La Barbera, Fabio Pardo, Yuval Tassa, Monica Daley, Christopher Richards, Petar Kormushev, and John Hutchinson. Ostrichrl: A musculoskeletal ostrich simulation to study bio-mechanical locomotion. *arXiv preprint arXiv:2112.06061*, 2021.
22. Seungmoon Song, Łukasz Kidziński, Xue Bin Peng, Carmichael Ong, Jennifer Hicks, Sergey Levine, Christopher G Atkeson, and Scott L Delp. Deep reinforcement learning for modeling human locomotion control in neuromechanical simulation. *Journal of neuroengineering and rehabilitation*, 18(1):126, 2021.
23. C. Berg, V. Caggiano, and Vikash Kumar. Sar: Generalization of physiological agility and dexterity via synergistic action representation. *arXiv preprint arXiv:2307.03716*, 2023.
24. Alberto Silvio Chiappa, Alessandro Marin Vargas, Ann Huang, and Alexander Mathis. Latent exploration for Reinforcement Learning. *Advances in Neural Information Processing Systems*, 36:56508–56530, December 2023.
25. Vittorio Caggiano, Guillaume Durandau, Huawei Wang, Alberto Chiappa, Alexander Mathis, Pablo Tano, Nisheet Patel, Alexandre Pouget, Pierre Schumacher, Georg Martius, Daniel Haeufle, Yiran Geng, Boshi An, Yifan Zhong, Jiaming Ji, Yuanpei Chen, Hao Dong, Yaodong Yang, Rahul Siripurapu, Luis Eduardo Ferro Diez, Michael Kopp, Vihang Patil, Sepp Hochreiter, Yuval Tassa, Josh Merel, Randy Schultheis, Seungmoon Song, Massimo Sartori, and Vikash Kumar. Myochallenge 2022: Learning contact-rich manipulation using a musculoskeletal hand. In Marco Ciccone, Gustavo Stolovitzky, and Jacob Albrecht, editors, *Proceedings of the NeurIPS 2022 Competitions Track*, volume 220 of *Proceedings of Machine Learning Research*, pages 233–250. PMLR, 28 Nov–09 Dec 2022.
26. Yusen Feng, Xiyan Xu, and Libin Liu. MuscleVAE: Model-based controllers of muscle-actuated characters. *arXiv preprint arXiv:2312.07340*, 2023.
27. Kaibo He, Chenhui Zuo, Chengtian Ma, and Yanan Sui. DynSyn: Dynamical synergistic representation for efficient learning and control in overactuated embodied systems. *arXiv preprint arXiv:2407.11472*, 2024.
28. Pierre Schumacher, Thomas Geijtenbeek, Vittorio Caggiano, Vikash Kumar, Syn Schmitt, Georg Martius, and Daniel FB Haeufle. Emergence of natural and robust bipedal walking by learning from biologically plausible objectives. *iScience*, 2025.
29. Merkourios Simos, Alberto Silvio Chiappa, and Alexander Mathis. Reinforcement learning-based motion imitation for physiologically plausible musculoskeletal motor control. *arXiv preprint arXiv:2503.14637*, 2025.
30. Vittorio Caggiano, Guillaume Durandau, Huiyi Wang, Chun Kwang Tan, Pierre Schumacher, Huawei Wang, Alberto Silvio Chiappa, Alessandro Marin Vargas, Alexander Mathis, Jungdam Won, Jungnam Park, Gunwoo Park, Beomsoo Shin, Minsueng Kim, Seungbum Koo, Zhuo Yang, Wei Dang, Heng Cai, Jianfei Song, Seungmoon Song, Massimo Sartori, and Vikash Kumar. Myochallenge 2023: Towards human-level dexterity and agility. 2024.
31. Vittorio Caggiano, Guillaume Durandau, Seungmoon Song, Chun Kwang Tan, Huiyi Wang, Balint Hodossy, Pierre Schumacher, Letizia Gionfida, Massimo Sartori, and Vikash Kumar. MyoChallenge 2024: Physiological Dexterity and Agility in Bionic Humans. In *NeurIPS 2024 Competition Track*, April 2024.
32. Richard A. Schmidt, Timothy D. Lee, Carolee J. Winstein, Gabriele Wulf, and Howard N. Zelaznik. *Motor Control and Learning: A Behavioral Emphasis*. Human Kinetics, Champaign, IL, 6 edition, 2018. ISBN 9781492547754. With Web Resource.
33. Paul Cisek. Evolution of behavioural control from chordates to primates. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 377(1844):20200522, December 2021. doi: 10.1098/rstb.2020.0522.
34. Alberto Silvio Chiappa, Alessandro Marin Vargas, and Alexander Mathis. DMAP: A Distributed Morphological Attention Policy for learning to

- locomote with a changing body. *Advances in Neural Information Processing Systems*, 35:37214–37227, December 2022.
35. Alberto Silvio Chiappa, Briti Gangopadhyay, Zhao Wang, and Shingo Takamatsu. Auto-bidding in real-time auctions via oracle imitation learning (oil). *arXiv preprint arXiv:2412.11434*, 2024.
  36. Łukasz Kidziński, Sharada P. Mohanty, Carmichael F. Ong, Jennifer L. Hicks, Sean F. Carroll, Sergey Levine, Marcel Salathé, and Scott L. Delp. Learning to Run Challenge: Synthesizing Physiologically Accurate Motion Using Deep Reinforcement Learning. In Sergio Escalera and Markus Weimer, editors, *The NIPS '17 Competition: Building Intelligent Systems*, pages 101–120, Cham, 2018. Springer International Publishing. ISBN 978-3-319-94042-7. doi: 10.1007/978-3-319-94042-7\_6.
  37. Łukasz Kidziński, Sharada Prasanna Mohanty, Carmichael Ong, Zhenwei Huang, Shuchang Zhou, Anton Pechenko, Adam Stelmazczyk, Piotr Jarosik, Mikhail Pavlov, Sergey Kolesnikov, Sergey Plis, Zhibo Chen, Zhizheng Zhang, Jiale Chen, Jun Shi, Zhuobin Zheng, Chun Yuan, Zhihui Lin, Henryk Michalewski, Piotr Mitoś, Błażej Osiński, Andrew Melnik, Malte Schilling, Helge Ritter, Sean Carroll, Jennifer Hicks, Sergey Levine, Marcel Salathé, and Scott Delp. Learning to run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments. *arXiv preprint arXiv:1804.00361*, 2018.
  38. Łukasz Kidziński, Carmichael Ong, Sharada Prasanna Mohanty, Jennifer Hicks, Sean F. Carroll, Bo Zhou, Hongsheng Zeng, Fan Wang, Rongzhong Lian, Hao Tian, Wojciech Jaśkowski, Garrett Andersen, Odd Rune Lykkebø, Nihat Engin Toklu, Pranav Shyam, Rupesh Kumar Srivastava, Sergey Kolesnikov, Oleksii Hrinchuk, Anton Pechenko, Mattias Ljungström, Zhen Wang, Xu Hu, Zehong Hu, Minghui Qiu, Jun Huang, Aleksei Shpilman, Ivan Sosin, Oleg Svidchenko, Aleksandra Malysheva, Daniel Kudenko, Lance Rane, Aditya Bhatt, Zhengfei Wang, Penghui Qi, Zeyang Yu, Peng Peng, Quan Yuan, Wenxin Li, Yunsheng Tian, Ruihan Yang, Pingchuan Ma, Shauharda Khadka, Somdeb Majumdar, Zach Dwiel, Yinyin Liu, Evren Tumer, Jeremy Watson, Marcel Salathé, Sergey Levine, and Scott Delp. Artificial intelligence for prosthetics - challenge solutions. *arXiv preprint arXiv:1902.02441*, 2019.
  39. Łukasz Kidziński, Carmichael Ong, Sharada Prasanna Mohanty, Jennifer Hicks, Sean Carroll, Bo Zhou, Hongsheng Zeng, Fan Wang, Rongzhong Lian, Hao Tian, et al. Artificial intelligence for prosthetics: Challenge solutions. In *The NeurIPS'18 Competition: From Machine Learning to Intelligent Conversations*, pages 69–128. Springer, 2020.
  40. John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
  41. Heyuan Yao, Zhenhua Song, Baoquan Chen, and Libin Liu. Controlvae: Model-based learning of generative controllers for physics-based characters. *ACM Trans. Graph.*, 41(6):183:1–183:16, 2022. doi: 10.1145/3550454.3555434.
  42. Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.
  43. Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2016.
  44. Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2016.
  45. Yee Teh, Victor Bapst, Wojciech M. Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
  46. Marco Bagatella, Jonas Hübotter, Georg Martius, and Andreas Krause. Active fine-tuning of generalist policies. *arXiv preprint arXiv:2410.05026*, 2024.
  47. Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2018.
  48. Charles Xu, Qiyang Li, Jianlan Luo, and Sergey Levine. RLDG: Robotic generalist policy distillation via reinforcement learning. *arXiv preprint arXiv:2412.09858*, 2024.
  49. Hiroki Furuta, Yusuke Iwasawa, Yutaka Matsuo, and S. Gu. A system for morphology-task generalization via unified representation and behavior distillation. *International Conference on Learning Representations*, 2022. doi: 10.48550/arXiv.2211.14296.
  50. Zhiwei Jia, Xuanlin Li, Zhan Ling, Shuang Liu, Yiran Wu, and Hao Su. Improving Policy Optimization with Generalist-Specialist Learning. In *Proceedings of the 39th International Conference on Machine Learning*, pages 10104–10119. PMLR, June 2022.
  51. Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.
  52. Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.
  53. Fatemeh Zargarbashi, Jin Cheng, Dongho Kang, Robert Sumner, and Stelian Coros. Robotkeyframing: Learning locomotion with high-level objectives via mixture of dense and sparse rewards. *Conference on Robot Learning*, 2024. doi: 10.48550/arXiv.2407.11562.
  54. Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.
  55. Brandon Trabucco, Mariano Philipp, and Glen Berseth. AnyMorph: Learning transferable policies by inferring agent morphology. *International Conference on Machine Learning*, pages 21677–21691. PMLR, 2022.
  56. Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
  57. Carmelo Sferrazza, Dun-Ming Huang, Fangchen Liu, Jongmin Lee, and Pieter Abbeel. Body transformer: Leveraging robot embodiment for policy learning. *arXiv preprint arXiv:2408.06316*, 2024.
  58. 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.
  59. Jack M. Winters. Hill-Based Muscle Models: A Systems Engineering Perspective. In Jack M. Winters and Savio L-Y. Woo, editors, *Multiple Muscle Systems: Biomechanics and Movement Organization*, pages 69–93. Springer, New York, NY, 1990. ISBN 978-1-4613-9030-5. doi: 10.1007/978-1-4613-9030-5\_5.
  60. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
  61. Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan C. Julian, Karol Hausman, Chelsea Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *Conference on Robot Learning*, 2019.
  62. Dean A. Pomerleau. ALVINN: An Autonomous Land Vehicle in a Neural Network. In *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.
  63. Donald Michie, Michael Bain, and Jean E. Hayes-Michie. Cognitive models from subcognitive skills. In M. Grimble, S. McGhee, and P. Mowforth, editors, *Knowledge-Based Systems in Industrial Control*. Peter Peregrinus, Stevenage, 1990.
  64. Claude Sammut, Scott Hurst, Dana Kedzier, and Donald Michie. Learning to fly. In D. Sleeman and P. Edwards, editors, *Proceedings of the Ninth International Conference on Machine Learning*, pages 385–393, San Francisco, 1992. Morgan Kaufmann.
  65. Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning.

*arXiv preprint arXiv:1011.0686*, 2011.

66. DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiaoshi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuan Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shufeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanja Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Chen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv: 2501.12948*, 2025.
67. Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2018.
68. Ferdinando A Mussa-Ivaldi, Simon F Giszter, and Emilio Bizzi. Linear combinations of primitives in vertebrate motor control. *Proceedings of the National Academy of Sciences*, 91(16):7534–7538, 1994.
69. Andrea d’Avella and MMCM Tresch. Modularity in the motor system: Decomposition of muscle patterns as combinations of time-varying synergies. *Advances in neural information processing systems*, 14, 2001.
70. 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.
71. Matthew C Tresch and Anthony Jarc. The case for and against muscle synergies. *Current opinion in neurobiology*, 19(6):601–607, 2009.
72. Cristiano Alessandro, Ioannis Delis, Francesco Nori, Stefano Panzeri, and Bastien Berret. Muscle synergies in neuroscience and robotics: From input-space to task-space perspectives. *Frontiers in computational neuroscience*, 7:43, 2013.
73. Gerald E Loeb. Learning to use muscles. *Journal of human kinetics*, 76(1):9–33, 2021.
74. Matthew C Tresch and Emilio Bizzi. Responses to spinal microstimulation in the chronically spinalized rat and their relationship to spinal systems activated by low threshold cutaneous stimulation. *Experimental brain research*, 129:401–416, 1999.
75. Simon A Overduin, Andrea d’Avella, Jose M Carmena, and Emilio Bizzi. Microstimulation activates a handful of muscle synergies. *Neuron*, 76(6): 1071–1077, 2012.
76. Mackenzie Weygandt Mathis. Adaptive intelligence: leveraging insights from adaptive behavior in animals to build flexible ai systems. *arXiv preprint arXiv:2411.15234*, 2024.
77. James AR Marshall and Andrew B Barron. Are transformers truly foundational for robotics? *npj Robotics*, 3(1):9, 2025.
78. W. Jeffrey Johnston and Stefano Fusi. Abstract representations emerge naturally in neural networks trained to perform multiple tasks. *Nature Communications*, 14(1):1040, February 2023. ISSN 2041-1723. doi: 10.1038/s41467-023-36583-0.
79. Laura N. Driscoll, Krishna Shenoy, and David Sussillo. Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. *Nature Neuroscience*, 27(7):1349–1363, July 2024. ISSN 1546-1726. doi: 10.1038/s41593-024-01668-6.
80. Reidar Riveland and Alexandre Pouget. Natural language instructions induce compositional generalization in networks of neurons. *Nature Neuroscience*, 27(5):988–999, May 2024. ISSN 1546-1726. doi: 10.1038/s41593-024-01607-5.
81. G. E. Alexander, M. R. DeLong, and P. L. Strick. Parallel organization of functionally segregated circuits linking basal ganglia and cortex. *Annual Review of Neuroscience*, 9:357–381, 1986.
82. Terrence J Sejnowski. Large language models and the reverse turing test. *Neural computation*, 35(3):309–342, 2023.
83. Paul Morris Fitts and Michael I. Posner. *Human Performance*. Brooks/Cole Publishing Company, Belmont, California, 1967.
84. Daniel M. Wolpert, Jörn Diedrichsen, and J. Randall Flanagan. Principles of sensorimotor learning. *Nature reviews. Neuroscience*, 12(12): 739–751, December 2011. ISSN 1471-0048. doi: 10.1038/nrn3112.
85. Krishna V Shenoy, Maneesh Sahani, and Mark M Churchland. Cortical control of arm movements: a dynamical systems perspective. *Annual review of neuroscience*, 36(1):337–359, 2013.
86. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Naacl*, 2019.
87. Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.
88. Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2023.
89. 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.

## Appendix



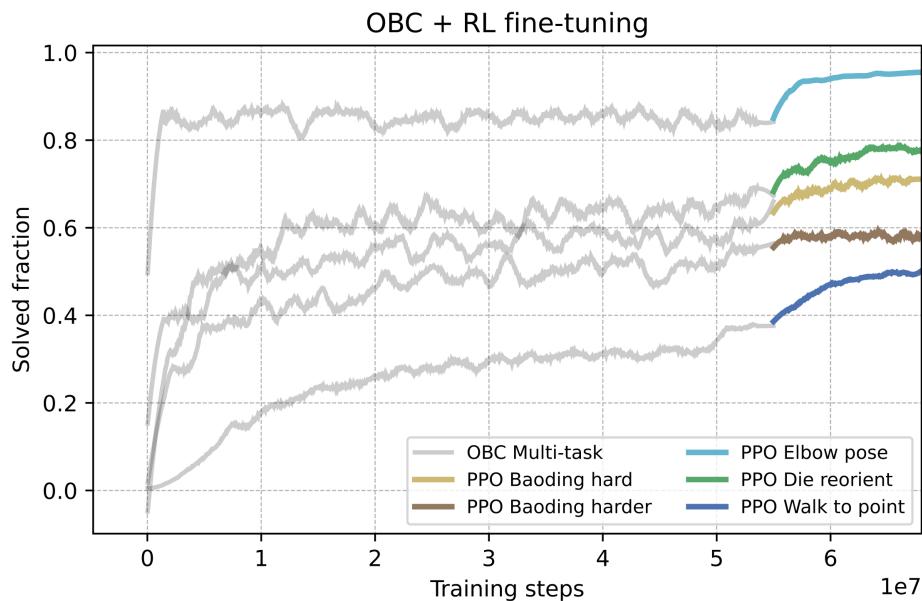
**Figure S1.** Single-task imitation learning curves with Arnold's policy network architecture trained with PPO. As can be seen, certain tasks require longer training than the others to reach the expert's performance. We used these graphs as a metric to allocate more parallel environments to tasks that required more training steps in a single task setting.

**Table S1.** Number of environments per task. We picked these heuristic numbers based on the single-task imitation experiments.

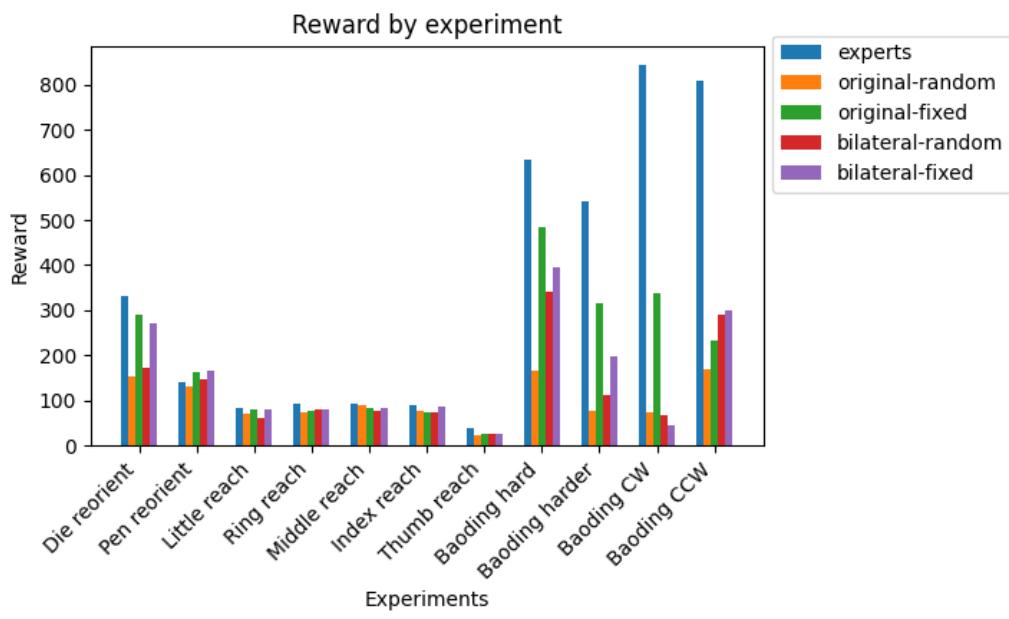
Environment Name	Count
Elbow pose	2
Thumb reach	2
Index reach	2
Middle reach	2
Ring reach	2
Little reach	2
Pen reorient	2
Die reorient	2
Baoding CW	2
Baoding CCW	6
Baoding hard	6
Baoding harder	6
Object relocate	6
Walk to point	10

**Table S2.** Performance drop in success rate when using a disjoint vocabulary, compared to our proposed shared sensorimotor vocabulary on all ( $\downarrow$  indicates performance drop).

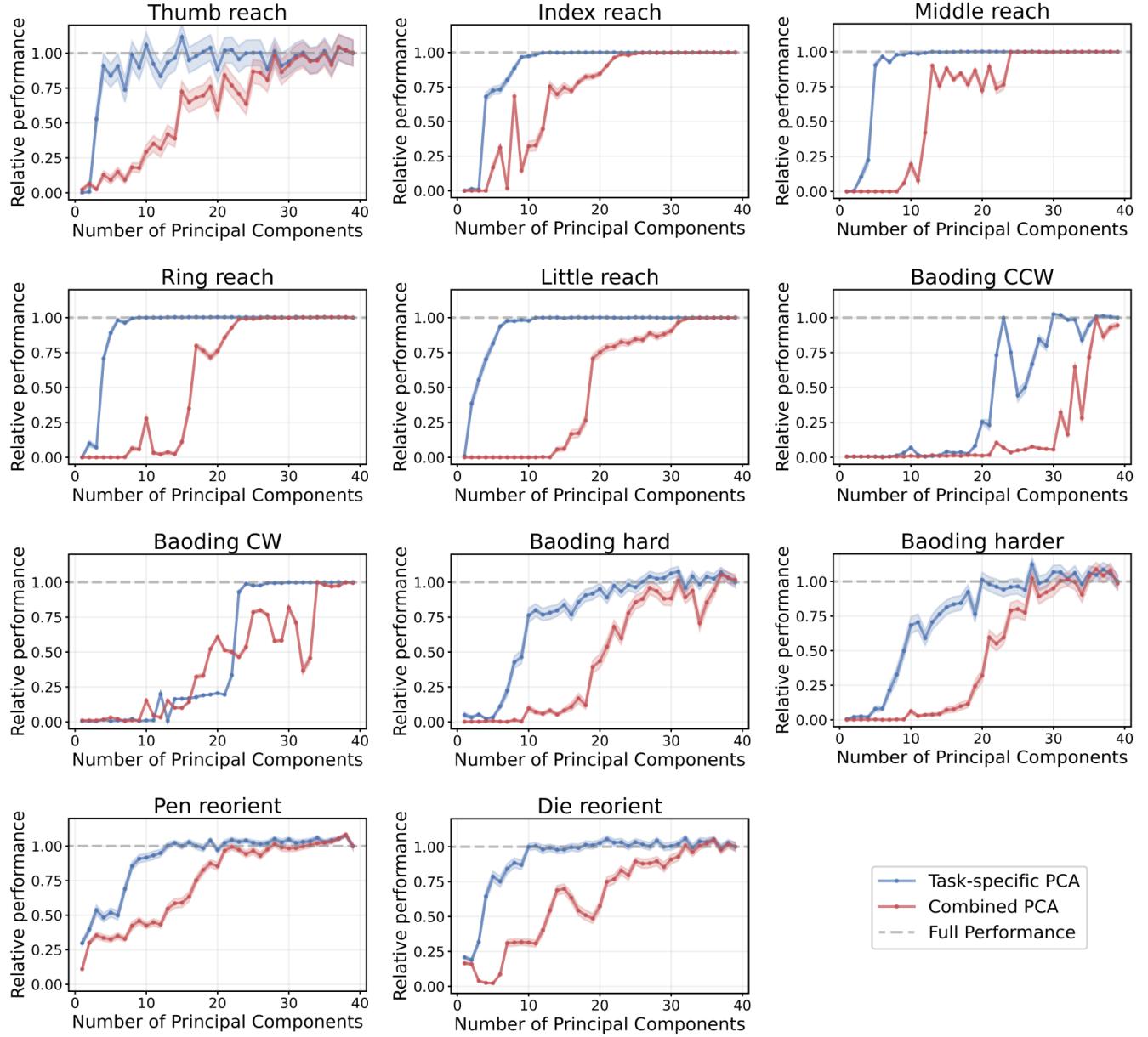
Tasks	Baoding CCW	Baoding hard	Baoding harder	Object relocation	Die Reorient
Success rate change	$\sim 35\% \downarrow$	$\sim 19\% \downarrow$	$\sim 11\% \downarrow$	$\sim 37\% \downarrow$	$\sim 10\% \downarrow$



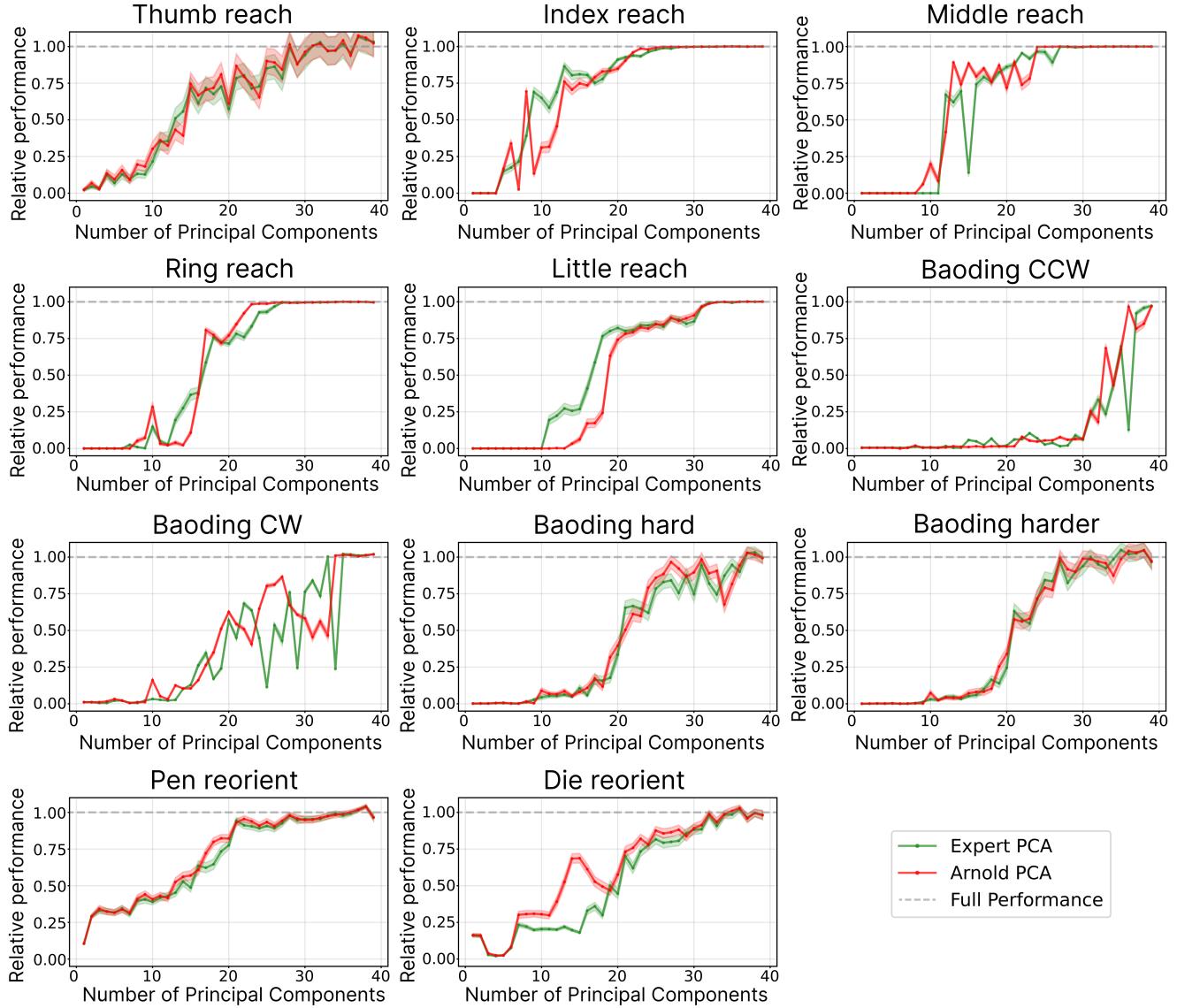
**Figure S2.** Learning curves of multi-task OBC (50M steps plus 5M steps at reduced learning rate) followed by RL fine-tuning on single tasks. We plotted the graphs relative to those tasks where RL fine-tuning leads to a performance improvement. The performance is measured as *solved fraction*, corresponding to the fraction of time steps in which the target configuration is achieved (e.g., balls on targeted for the Baoding tasks, humanoid at destination for Walk to point). The policies resulting from the fine-tuning experiments were used as new experts for Arnold.



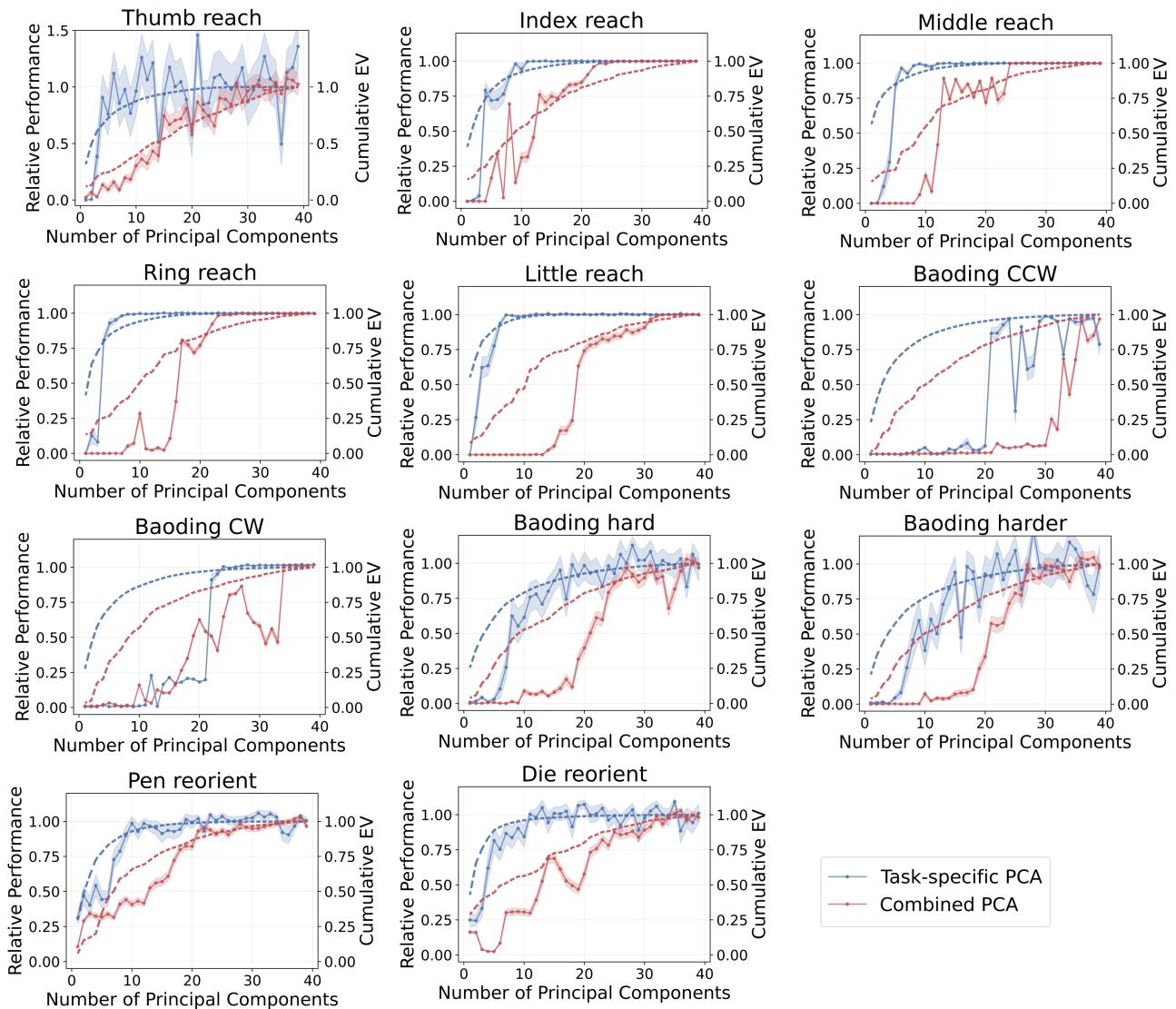
**Figure S3.** Experiment conducted to compare the performance of original model and a modified model which also does self-attention on the time axis. **original-random** is training our original method on a randomized expert, **original-fixed** is instead trained on a deterministic expert. **bilateral-random** and **bilateral-fixed** is the same experiments done with the modified transformer.



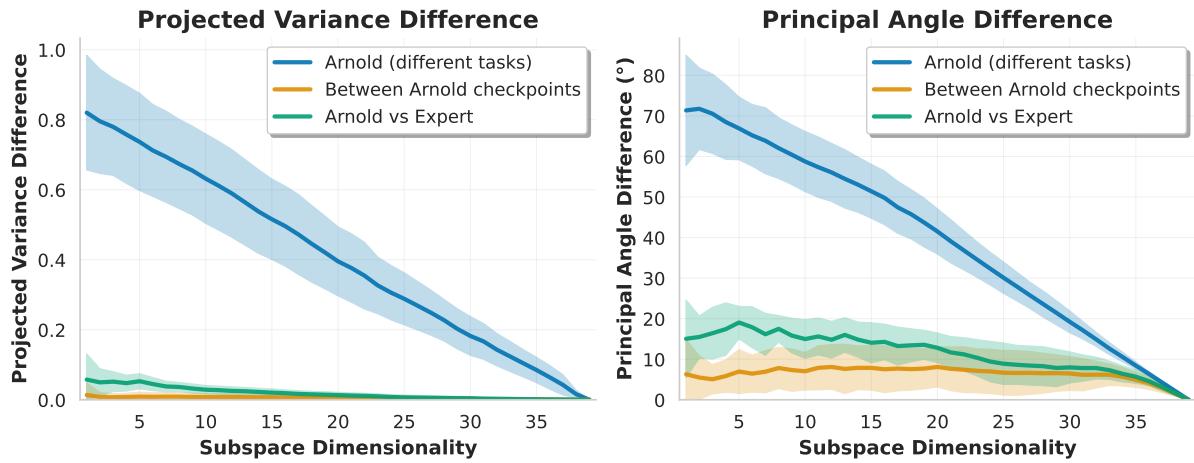
**Figure S4.** Comparison between the performance for varying action space dimensionality, when the control subspace is defined by task-specific principal components (blue) or combining all tasks (red).



**Figure S5.** Comparison between the performance for varying action space dimensionality, when the control subspace is defined by principal components extracted from actions generated by Arnold (red) or single-task specialists (green), with actions from all tasks combined.



**Figure S6.** Comparison between CSI and cumulative explained variance (EV) as measures of control space dimensionality. CSI results are shown in solid lines for task-specific (blue) and task-shared (red) principal components, and EV results are respectively shown in dashed lines.



**Figure S7.** Comparison of muscle synergy patterns across Arnold checkpoints and expert checkpoints using Principal Angle Difference (PAD) and Projected Variance Difference (PWD) metrics. **Left:** PVD measures how well synergies from one condition can explain variance in another's muscle activation patterns. **Right:** PAD quantifies angular differences between synergy subspaces, with  $90^\circ$  indicating completely orthogonal (maximally different) patterns. Three comparisons are shown: Arnold checkpoint across different task pairs (blue), between different Arnold training checkpoints on same tasks (orange), and Arnold checkpoints versus expert checkpoints (green). Shaded regions represent  $\pm 1$  standard deviation. The results demonstrate that task-specific differences within a single policy (blue) exceed inter-policy differences, indicating that Arnold has learned distinct muscle synergy patterns for different manipulation tasks under the time-invariant synergy model assumptions.