CoMic: Complementary Task Learning & Mimicry for Reusable Skills

Leonard Hasenclever [1] **Fabio Pardo** [2] **Raia Hadsell** [1] **Nicolas Heess** [1] **Josh Merel** [1]

Abstract

Learning to control complex bodies and reuse learned behaviors is a longstanding challenge in continuous control. We study the problem of learning reusable humanoid skills by imitating motion capture data and joint training with complementary tasks. We show that it is possible to learn reusable skills through reinforcement learning on 50 times more motion capture data than prior work. We systematically compare a variety of different network architectures across different data regimes both in terms of imitation performance as well as transfer to challenging locomotion tasks. Finally we show that it is possible to interleave the motion capture tracking with training on complementary tasks, enriching the resulting skill space, and enabling the reuse of skills not well covered by the motion capture data such as getting up from the ground or catching a ball.

1. Introduction

Learning policies that can produce complex motor behavior for articulated, physically simulated bodies is a difficult challenge that is central to artificial intelligence. Recent efforts have demonstrated that it is possible to learn robust policies for certain locomotion behaviors from scratch (Heess et al., 2017) in rich environments. However this can be very data inefficient or the quality of resulting behavior may be lacking. To alleviate this problem, it is desirable to be able to repurpose previously learned skills and use prior knowledge to facilitate and speed up learning on subsequent tasks. Various efforts have sought to design hierarchical architectures that enable reuse of skills (e.g. Heess et al. (2016); Florensa et al. (2017); Haarnoja et al. (2018)), often through factoring the problem into learning a high-level task-specific module and a reusable low-level controller.

[1]DeepMind, London [2]Imperial College, London, work done during an internship at DeepMind. Correspondence to: Leonard Hasenclever <leonardh@google.com>.

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

While hierarchical architectures seem especially promising in multi-task reinforcement learning, discovering robust behaviors via standard exploration techniques remains difficult. The pretrained skills could come from other, perhaps simpler, tasks (e.g. Riedmiller et al., 2018) but this requires careful task design and is difficult to scale to large numbers of tasks. Alternatively, the skills could be derived from expert demonstrations. In the case of humanoid control, these demonstrations can be acquired from the large amount of motion capture (mocap) data publicly available, and the resulting skills therefore consist of more natural movements (Merel et al., 2019b; Peng et al., 2019). While very effective, this approach may prevent the agent from solving tasks requiring other skills not well covered by the demonstrations.

In this paper, we explore this space further, studying the trade-offs of various existing approaches for representing and transferring reusable skills. How well skills can be learned and transferred to new tasks depends on a number of factors, including the nature and number of the pre-training tasks as well the structure and capacity of the architecture, which determines the ability to represent, re-use, and generalize skills. We attempt to disentangle the dimensions of the problem through a systematic comparison of architectures for skill representation, evaluated using different training regimes, transfer tasks, and varying quantities of diverse motion capture data. Simulated humanoid character control is a suitable problem domain, because humanoids can generate a rich set of behaviors. Furthermore, motion capture data gives us access to a very rich, dense and scalable set of pre-training tasks that we know can be solved. This allows us to isolate questions relating to the capacity of different skill architectures and the ease with which the learned skills can be used, generalized and adapted, reducing the confounding effect of other factors, such as exploration during the initial pre-training phase. In our experiments we demonstrate that simpler architectures tend to perform better than specialized ones, and that while a modest amount of motion capture data is sufficient to solve challenging locomotion tasks, asymptotic performance improves with data size.

Furthermore, we propose a flexible framework to simultaneously learn to solve goal-directed tasks while learning to track the motion capture clips. This approach allows us to combine motion capture data with additional training tasks to induce complementary behaviors. Our results show that

new skills can be discovered from additional tasks along with the ones from the reference dataset, and that their discovery is actually facilitated by the motion capture tracking compared to finding them from scratch. By training the low-level policy jointly on both tracking tasks and out-of-sample tasks, the resulting skill embedding space incorporates both the reference movements and those required to solve the complementary tasks.

2. Related Work

Across multiple disciplines involving complex control challenges, there has long been an interest in building reusable skills, using a wide range of approaches such as options (Sutton et al., 1999; Bacon et al., 2017; Barreto et al., 2019), movement primitives (Schaal et al., 2005; Paraschos et al., 2013; Mülling et al., 2013), or through scheduling libraries of control elements (Liu & Hodgins, 2017). Approaches involving learning and reuse of skills are broadly useful in control and robotics settings, either for humanoid control (Da Silva et al., 2014; Yang et al., 2018) or for manipulation, both in simulation (Rajeswaran et al., 2018) and on robotic platforms (Strudel et al., 2020; Wulfmeier et al., 2019).

In the context of continuous control, transferable motor skills can be obtained in various ways. A default method that does not make use of machine learning involves manually designing separate controllers for specific behaviors. While historically popular for applications, designing many case-specific controllers can become too difficult for complex problems. Moreover, while hand-designed controllers can be sequenced in novel ways, they may provide limited flexibility. Contemporary efforts have therefore moved towards learning based approaches. A first class of learning-based approaches involves carefully creating sets of tasks whose mastery leads to emergence of useful skills (Heess et al., 2016; James et al., 2018; Riedmiller et al., 2018; Hausman et al., 2018). And the expectation is that the skills that emerge to solve any particular task may be useful for other tasks in the set. A second class of approaches involves unsupervised learning of skills (Gregor et al., 2016; Florensa et al., 2017; Warde-Farley et al., 2019; Eysenbach et al., 2019), generally based on maximizing the diversity of the visited states. While these approaches require less engineering, they often depend upon the specification of strong priors on the dimensions that skills should affect, for example by prioritizing the body position in the plane to incentivize locomotion. Moreover, finding a large number of diverse behaviors may lead to skills that are not relevant for later tasks. A final alternative is to use imitation learning techniques to acquire skills from sources such as motion capture (Merel et al., 2017; 2019b) or even loosely structured "play" (Lynch et al., 2019).

Howsoever learned, a common architectural motif among recent approaches is for there to be an embedding space that reflects the learned set of skills (Heess et al., 2016). This can be repurposed in the context of hierarchical control schemes where, for a new task, a high-level controller produces actions via the embedding space. Embedding vectors can be used to trigger either a specific plan to reach a goal (Lynch et al., 2019), a behavior for an entire episode (Wang et al., 2017; Eysenbach et al., 2019), a segment in an episode (Hausman et al., 2018) or a temporally correlated behavior specified at every time step (Merel et al., 2019b). For the latent space to be easy to control for a high-level controller, the distribution of latent vectors is generally regularized to be well distributed and smooth. This regularization can also be viewed from an information bottleneck perspective (Goyal et al., 2019; 2020). The resulting skills are usually frozen and the reuse is limited by the behaviors that can be triggered via the latent space (Haarnoja et al., 2018).

Articulated humanoids are an example of a high-dimensional body that is particularly complex to control. Our present work is situated most closely among the various efforts to produce robust, natural, and reusable motor behavior for physically simulated humanoids. In this setting, approaches which generate behavior through tracking of motion capture demonstrations tend to produce the most robust and realistic movements (Liu et al., 2010; Peng et al., 2018; Chentanez et al., 2018; Bergamin et al., 2019). Moreover, there has been a longstanding recognition in this field that skills should be able to be repurposed for new tasks (Faloutsos et al., 2001; Liu & Hodgins, 2017). We build most directly on recent efforts involving large scale tracking of motion capture data (Chentanez et al., 2018) as well as efforts to build reusable skills from these data (Merel et al., 2019b; Peng et al., 2019).

Our joint training approach is conceptually related to previous work in supervised learning on learning shared embedding spaces for multiple tasks or modalities (e.g. Frome et al., 2013; Kiros et al., 2014). Another related strand of research are semi-supervised approaches, both in supervised learning (e.g. Kingma et al., 2014) and in RL (Finn et al., 2016), that leverage a large number of unlabelled examples to improve task performance based on a small number of labelled examples. In a similar spirit, we use a large number of task agnostic mocap demonstrations to improve data efficiency when learning tasks of interest for which experience is accompanied by rewards.

3. Approach

This paper aims to study the following question: How can we best learn, represent and reuse diverse skills?

Learning reusable skills To this end, we learn low-dimensional skill embedding spaces via imitation using

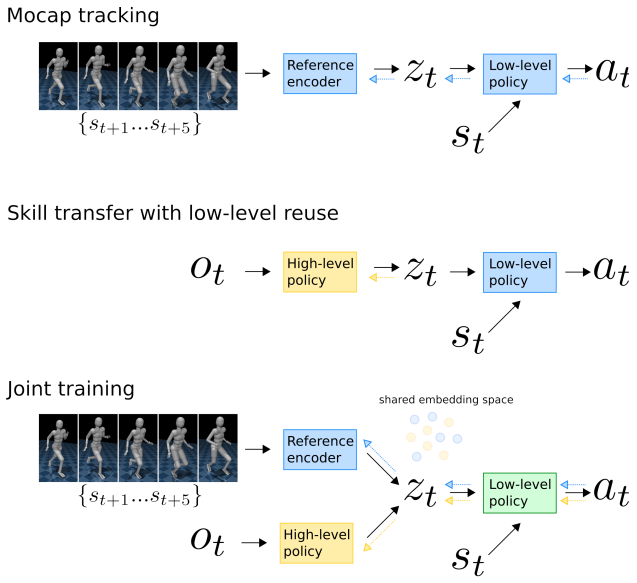

Figure 1. Schema depict skill learning via mocap tracking with a reference encoder producing a stochastic embedding z_t which conditions a low-level policy; skill transfer to a new task with a frozen low-level policy; and joint training with complementary tasks during the skill learning phase to enrich the resulting skill space.

reinforcement learning and use the resulting low-level policies on challenging transfer tasks. All architectures considered in this work use a *reference* encoder $\pi_{\text{HL}}(z_t|s_t, s_t^{\text{ref}})$, which, at time t encodes the desired future target states $s_t^{\text{ref}} = (\hat{s}_{t+1}, \dots, \hat{s}_{t+5})$ from the motion capture reference data and current state s_t into a stochastic embedding z_t, used to condition a low-level policy π_{LL}. π_{LL} also receives proprioceptive information about the state of the body as an input and produces a distribution over actions a_t (see Panel 1 in figure 1). To learn the skill embedding spaces, we train a policy to imitate motion capture demonstrations via reinforcement learning. We use a reward that compares the current pose of the humanoid to the target pose from the motion capture reference (see section 4) and maximize the sum of discounted rewards using V-MPO (Song et al., 2020), an on-policy variant of Maximum a Posteriori Policy Optimization (Abdolmaleki et al., 2018). In the case of mixture low-level policies (described below), which we train with an on-policy variant of RHPO (Wulfmeier et al., 2019). This is equivalent to V-MPO with an additional KL constraint on the mixture weights. We regularize the latent embedding with a standard Gaussian prior by adding a KL loss term to the V-MPO losses:

$$\beta \mathbb{E}_\pi \left[\sum_t KL(\pi_{\text{HL}}(z_t|s_t, s_t^{\text{ref}}) \| \mathcal{N}(0, I)) \right], \qquad (1)$$

where the coefficient β controls the strength of the regularization.

Transferring skills to new tasks Having trained on the mocap tracking task, we can then transfer the low-level policy to a new task by reusing the learned embedding space as a new action space. To do this we freeze the parameters of the low-level policy and learn a new high-level policy that outputs a latent embedding z_t at each time step (see Panel 2 in figure 1). This high-level policy can take as inputs task observations o_t. Acting in the learned embedding space heavily biases the resulting behavior to the behaviors present in the motion capture data. This enables much more coherent exploration. For example, we observe that often a randomly initialized high-level policy already results in naturalistic movements for an extended period of time while randomly initialized policies in the raw action space tend to fall immediately.

Joint training with complementary tasks On the other hand, the bias towards behaviors from the motion capture data can also hurt performance. Skills that are relevant for a transfer task but not well covered by the motion capture data may be impossible to learn. For example, getting up from the ground is underrepresented in many motion capture datasets and a hard skill to learn but clearly desirable in transfer tasks. To tackle this problem we propose a joint training approach: during the skill learning phase we interleave the mocap tracking task with other, complementary, out-of-sample tasks that ensure that specific skills are well represented in the embedding space (see Panel 3 in figure 1). Since complementary tasks do not share the same observation as the mocap tracking task, and optimize for different rewards, we train a separate high-level policy acting in the learned embedding space but use the same low-level policy. This approach forces the embedding space to represent behaviors induced by the complementary tasks.

Low-Level Architectures It is a priori unclear which network architectures might permit effective transfer of skills. The simplest choice is a monolithic feedforward architecture but there are many possible hierarchical architectures that introduce additional structure by organizing the low-level policy into primitives. Two recent, representative examples are mixture distributions (Wulfmeier et al., 2019) and product distributions (Peng et al., 2019). The promise of such modular architectures is compositionality: different primitives could specialize to different skills which can then be flexibly composed by varying the primitive weights. Whether this hope is realized remains unclear and is likely highly task-dependent. Modularity is another possible benefit of mixture or product architectures. Both potentially allow the addition of new primitives and importantly allow primitives to specialize which may help represent wide ranges of skills.

Another plausible alternative is a recurrent low-level policy which can model temporal correlation in behavior and allows for temporally extended 'default' behaviors.

Guided by these considerations, we compare four representative network architectures for the low-level policy in our experiments: a fully-connected neural network (MLP), a recurrent neural network (LSTM), a mixture of Gaussians policy and a product of Gaussians policy.

The mixture distribution with C components is given by:

$$\pi_{\text{LL}}(a_t|s_t, z_t) = \sum_{i=1}^{C} w_i(s_t, z_t)\phi(a_t|\mu_i(s_t), \sigma_i(s_t)), \quad (2)$$

where $\phi(\cdot|\mu, \sigma)$ is the density of a normal distribution with diagonal covariance and with mean μ and standard deviation σ. Similarly the (unnormalized) product distribution is given by:

$$\pi_{\text{LL}}(a_t|s_t, z_t) \propto \prod_{i=1}^{C} \phi(a_t|\mu_i(s_t), \sigma_i(s_t))^{w_i(s_t,z_t)}. \quad (3)$$

For Gaussian factors, the resulting product distribution is Gaussian with mean and variance given analytically as function of w_i, μ_i, σ_i. Following Wulfmeier et al. (2019); Peng et al. (2019), we do not provide the latent embedding z_t to the primitives to encourage specialization. In our experiments both the mixture and the product policy networks have a shared torso followed by per-component one hidden layer MLPs. In order to keep the capacity of the low-level architectures similar, we choose networks with roughly similar numbers of parameters. We found that adding a KL regularizer made training product distributions much more stable.

4. Tasks

In this work, we perform our architecture comparisons and demonstrate our joint training approach in the context of humanoid continuous motor control tasks. Because motor control of high-dimensional, physically simulated bodies is difficult, it can be efficient to reuse skills; however, reuse is only helpful if previously acquired skills are relevant for subsequent challenges. To that end, given a distribution of reference motions, we consider tasks that we expect can be solved by reusing skills found in the motion capture dataset as well as tasks which cannot. All tasks involve simulated physics using MuJoCo (Todorov et al., 2012). We use a humanoid body adapted from the "CMU humanoid" available at dm_control/locomotion (Merel et al., 2019a). We adjusted limb lengths, masses, and dynamic properties of the body to make it more consistent with an average human. See figure 2 for an image of the body in the context of our tracking task.

Tracking task The primary task we employ to learn skills in this work is the multi-clip tracking task. Our task is available in the dm_control/locomotion package. The task is similar to the ones employed in various other efforts to produce policies that track motion capture data (Peng et al., 2018; Merel et al., 2019a; Chentanez et al., 2018; Peng et al., 2019). The task can be described in terms of episode initializations, the instructions provided by the environment to the agent, the reward function, and termination criteria. At the start of each episode we randomly select a starting frame from all frames in the underlying set of clips (excluding the last 10 frames from each clip). At the beginning of each episode the humanoid is initialized to the target pose in the selected frame. We provide target reference poses to the policy as an instruction of where to go; specifically, we provide a short snippet of future target states, $\{s_{t+1}, ...s_{t+5}\}$ (similarly to Chentanez et al., 2018; Merel et al., 2019b). At every timestep, the reward corresponds to a similarity function that compares the current pose to the target pose. Our reward function contains five different terms:

$$r = \frac{1}{2}r_{\text{trunc}} + \frac{1}{2}\left(0.1r_{\text{com}} + r_{vel} + 0.15r_{\text{app}} + 0.65r_{\text{quat}}\right)$$

The first reward term r_{trunc} penalizes large deviations from the reference in joint angles and the euclidean position of a set of 13 different body parts:

$$\varepsilon := \|b_{\text{pos}} - b^{\text{ref}}_{\text{pos}}\|_1 + \|q_{\text{pos}} - q^{\text{ref}}_{\text{pos}}\|_1$$
$$r_{\text{trunc}} = 1 - \frac{\varepsilon}{\tau}$$

where b_{pos} and $b^{\text{ref}}_{\text{pos}}$ correspond to the body positions of the simulated character and the mocap reference and q_{pos} and $q^{\text{ref}}_{\text{pos}}$ correspond to the joint angles. This reward terms is linked to the termination condition of our tracking task. Given a termination threshold τ, we terminate an episode if $\varepsilon > \tau$. Note that this ensures that $r_{\text{trunc}} \in [0, 1]$. We found that including this termination condition and the coupled reward speeds up training on larger clip sets but does not by itself lead to visually appealing behavior. The second reward term is similar to the objective proposed in Peng et al. (2018) with terms penalizing deviations in terms of the center of mass, the joint angle velocities, the end effector positions and the joint orientations but uses slightly different weights. Please refer to the supplementary material for further details.

Locomotion tasks To evaluate the reusability of the learned low-level policies we use three challenging locomotion tasks from the DM Control locomotion task library (Merel et al., 2019a; Tassa et al., 2020). Locomotion tasks form a natural, well-motivated test bed for humanoid skills that is within reach of current methods. The first task we consider is a sparse *go-to-target* task, which involves locomoting to an instructed target position in an open area, and

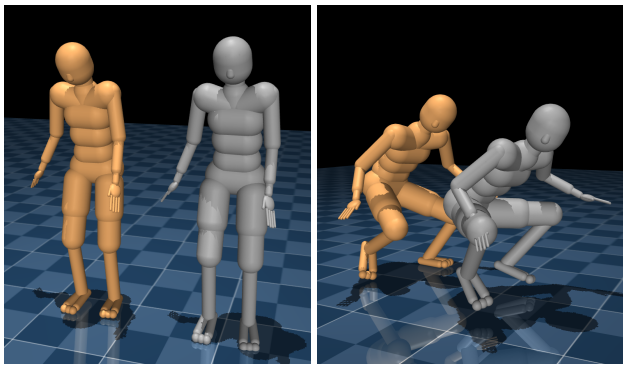

Figure 2. Multi-clip motion capture tracking task with physically simulated humanoid (bronze) and offset mocap reference pose (grey). Note how the imperfect reference pose self-intersects.

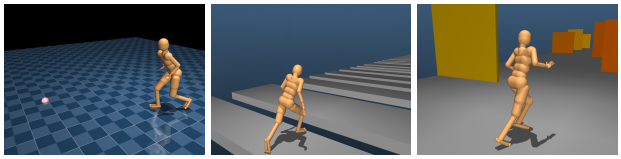

Figure 3. We evaluate transfer of trained low-level policies to challenging locomotion tasks: (left) go-to-target (center) gaps (right) walls.

learning from sparse rewards when the target is obtained. The other two tasks are dense reward tasks in which the humanoid agent has to traverse an obstacle course consisting of variable length *walls* or *gaps*, guided by first-person visual observations.

Out-of-sample tasks Some motor skills may not be available, or may be underrepresented, in the motion capture dataset. For example, the CMU motion capture database has relatively few clips involving getting up from the ground (compared to, e.g. walking). After training a policy that is capable of tracking a subset of the motion capture library, the reusable low-level controller may not provide an advantage on a task that requires the humanoid to get up from the ground – rather, the low-level controller may even bias learning away from such movements. Consequently, we also consider two representative *out-of-sample* tasks that involve movements that were either underrepresented or absent from our tracking reference data. The *get-up and stand* task consists of initializing the humanoid in various poses (about 5 % lying on the ground and 95% floating slighly above the ground in a standing pose which induces falling in a variety of different ways) and rewarding the humanoid for being in a standing pose with a reward of $\exp(-(h - h_{\text{target}})^2)$, where h is the head height of the humanoid and h_{target} is the target height corresponding to standing. For perfect task performance the humanoid has to both avoid falling when initialized in a standing pose and learn to get up from the ground when initialized lying down. A second task involves catching a ball thrown towards the humanoid. This task is similar to the one considered by Merel et al. (2020). In this task, the humanoid is initialized standing with the ball thrown towards it at varying angles and velocities. The reward function consists primarily of a negative reward if the humanoid falls or the ball touches the ground, combined with comparatively quite small shaping rewards to encourage the humanoid to stand upright and to touch the ball with its hands. In the data we consider, while arm movements are present, there is no data of catching a ball. Please refer to the supplementary material for additional details.

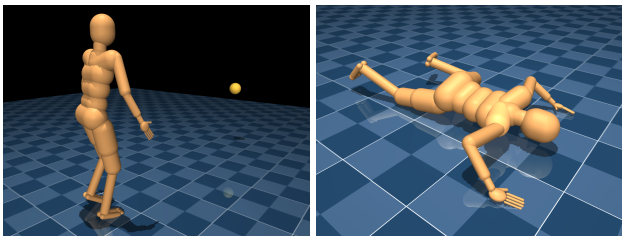

Figure 4. Some skills are not well covered by most motion capture datasets. To obtain a rich skill space we train jointly with complementary tasks such as getting up or catching a ball.

5. Experiments

In our experiments, we compare a number of different architectures across a range of data regimes with a focus on architectures that permit reuse on transfer tasks. Prior work (Merel et al., 2019b) on learning reusable skills from large motion capture datasets used a two-stage approach of imitation learning on short segments of single motion capture clips followed by a supervised distillation phase. Our training setting allows training reusable controllers through RL without a second distillation phase (Merel et al., 2019b) on up to 3.5 hours of diverse motion capture data. This approach is significantly simpler, faster and cheaper than the two stage approach taken by Merel et al., 2019b.

We restrict ourselves to architectures with a bottlenecked latent space (Hausman et al., 2018; Merel et al., 2019b) that serves as a new action space in transfer tasks. For the mixture and product architectures the high-level policy acts in the space of primitive mixture weights or exponents. Our experiments aim to answer the following research questions:

- Does the size of the latent space affect tracking performance and reusability in transfer tasks?
- How does KL regularization in the latent space impact tracking performance and reusability?
- How do different architectures compare in different regimes both in terms of tracking performance and

reusability?

Using reinforcement learning to learn reusable skills directly gives researchers larger control over the exact training setting. To highlight this flexibility, we show that we can train reusable controllers in a multitask set-up where the motion capture tracking task is interleaved with other tasks that endow the resulting controller with specific skills that would be difficult to acquire through motion capture alone, such as the ability to get up from the ground from a wide range of different positions.

5.1. Data Regimes

Different architectures inject different inductive biases and the effect of these model choices may be more pronounced in small data regimes. Hence, we compare different architectures across four different data regimes: a two minute set of clips containing walking behaviors with various turns (*"Walking"*), a two minute set of clips containing walking behaviors as well as running and jumping (*"Running"*), a 40 minute set of locomotion clips (*"Locomotion"*) as well as a 3.5 hour set of clips with a wide range of locomotion behaviors and hand movements (*"Large"*). All motion capture data was obtained from the CMU mocap database. [1]

5.2. Scaling Up Motion Capture Tracking

We find that several factors affect how well a policy can be trained on multiple clips. Firstly, we found on-policy reinforcement learning easier to scale to large motion capture datasets. Secondly, better learning of the value function improved learning speed. Specifically, we explored two different approaches to improve value learning. We explored giving the value function access to a one-hot encoding of the reference clip id. Secondly, we exploit the linear nature of the reward function and learn a separate value function per reward term (e.g. Van Seijen et al., 2017) (see figure 5). We found that learning a separate value function per reward term significantly speeds up learning whereas conditioning on the clip id only results in a modest additional improvement. We use both of these improvements in all further experiments. In line with previous work (Peng et al., 2018; Merel et al., 2019a), we observed that the reward function needs to be chosen carefully for best results although it is worth noting that concurrent work (Abdolmaleki et al., 2020) alleviates this difficulty.

5.3. Latent Space Dimensionality and KL Regularization Strength

When training reusable low-level controllers, an important choice is the size of the learned latent space. On the one hand, choosing a smaller size reduces the action space in transfer tasks and could lead to faster learning. On the other hand, a smaller latent space can represent fewer behaviors and might impair transfer performance. In addition, training the low-level controller on the mocap tracking task is likely to be harder, the smaller the learned latent space. Similar considerations apply to regularization in the learned latent space. Stronger regularization should be beneficial in transfer tasks since random exploration in the latent space at the beginning of training will be closer to the distribution of embeddings encountered during tracking, leading to more coherent exploration. However, too strong a regularization might impair the tracking performance and reduce the number of behaviors represented in the latent space.

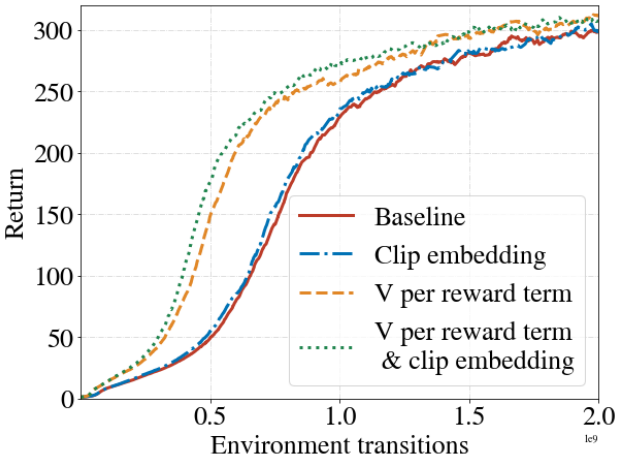

Figure 5. Learning separate value functions per reward component speeds up training but does not affect asymptotic performance. The figure shows results for an MLP low-level on the "Locomotion" dataset with $D = 60$ and $\beta = 1 \times 10^{-4}$.

To study these questions we trained MLP low-level controllers on the "Locomotion" clip set with latent space dimensions $D = 20, 40$ and 60 and a range of KL regularization strengths β. The tracking performance is shown in table 1. It remains roughly constant for different values of D and for a range of different values of β and begins to degrade slightly for $\beta = 5 \times 10^{-4}$. Even higher values of β lead to substantially reduced performance.

Next, we evaluate the trained low-level controllers on our transfer tasks. Figure 6 shows the performance on the go-to-target task. Across values of D, we find that the most highly regularized low-level policies encounter rewards fastest. This reflects the initial exploration behavior with a randomly initialized high-level controller. For the most regularized controller the initial behaviors is to take a few steps while less regularized policies lead to almost immediate falling. Note that for high values of D, the regularization strength

[1] mocap.cs.cmu.edu

β	1×10^{-5}	1×10^{-4}	2×10^{-4}	5×10^{-4}
D=20	0.69	0.69	0.69	0.62
D=40	0.69	0.69	0.70	0.59
D=60	0.70	0.69	0.70	0.62

Table 1. Average tracking reward per step for various latent space dimensionalities D and KL regularization strengths β after 6×10^9 environment transitions processed.

β	1×10^{-5}	1×10^{-4}	2×10^{-4}	5×10^{-4}
D=20	514 ± 6	550 ± 5	551 ± 5	578 ± 5
D=40	536 ± 5	532 ± 6	520 ± 5	576 ± 6
D=60	527 ± 5	447 ± 5	614 ± 7	628 ± 5

Table 2. Mean return with standard errors on the gaps transfer task for various embedding space sizes D and KL regularization strengths β. The returns are averaged over the logged rewards of a single run between 1.45×10^9 and 1.5×10^9 environment transitions processed.

has a particularly strong effect on transfer learning speed. The transfer result to the walls and gaps tasks are shown in tables 2 and 3. On the walls task, all low-level controllers perform quite well but the most highly regularized models perform best. On the gaps task the picture is less clear; but there is slight trend for more highly regularized models to perform best.

5.4. Architecture Comparison

Next, we compare a variety of different architectures across four different data regimes ranging from two minutes worth of motion capture data to 3.5 hours. Figure 7 shows the performance of different architectures on the motion capture tracking task. For each architecture we compared a number of variants and present results for the models performing best on the transfer tasks since this is our primary motivation in this work. For details on the different network architectures, please refer to the supplementary material. Across all data sets, we found that LSTM and MLP low-level policies tended to be easier to train and outperform mixture and product low-level policies. While the asymptotic performance on the two smallest motion capture datasets is similar, we found it difficult to achieve good tracking performance on the larger motion capture datasets with product and mixture low-level policies. One possible reason is the fact that, following Peng et al. (2019); Wulfmeier et al. (2019), the primitives are not conditioned on the latent variable to encourage specialization, making it harder to learn on a large motion capture datasets. The LSTM low-level policy performs best on the mocap tracking task since it can model temporal structure in the behavior and overfit to difficult reference clips. See supplementary video 1 for examples of the tracking performance of the LSTM architecture on our largest dataset.

β	1×10^{-5}	1×10^{-4}	2×10^{-4}	5×10^{-4}
D=20	576 ± 4	635 ± 4	643 ± 4	817 ± 3
D=40	641 ± 5	591 ± 4	621 ± 4	785 ± 4
D=60	647 ± 4	524 ± 4	592 ± 4	788 ± 4

Table 3. Mean return with standard errors on the walls transfer task for various embedding space sizes D and KL regularization strengths β. The returns are averaged over the logged rewards of a single run between 0.95×10^9 and 1×10^9 environment transitions processed

Next, we evaluated the trained low-level policies on the locomotion transfer tasks (see tables 4, 5 and 6). Across all transfer tasks, we found the MLP low-level policy to perform best. The LSTM performed less well, perhaps reflecting the fact that it can model temporal structure independent of the latent space, limiting its controllability. In our setting mixture and product policies performed worse than the MLP across all data regimes despite considerable tuning effort. See supplementary video 2 for examples of good performance on the locomotion transfer tasks.

Go To Target	Walking (2min)	Running (2min)	Locomotion (40min)	Large (220min)
MLP	$\mathbf{151 \pm 6}$	$\mathbf{213 \pm 6}$	$\mathbf{261 \pm 4}$	$\mathbf{280 \pm 7}$
LSTM	19 ± 1	30 ± 1	181 ± 4	264 ± 5
Mixture	19 ± 1	84 ± 2	100 ± 3	177 ± 4
Product	73 ± 3	13 ± 1	142 ± 3	154 ± 3

Table 4. Mean return with standard errors on the go to target transfer task for different architectures and data regimes. The returns are averaged over the logged rewards between 2.9×10^9 and 3.0×10^9 environment transitions processed and three runs. Across all data regimes the MLP low-level policy performs best.

Gaps	Walking (2min)	Running (2min)	Locomotion (40min)	Large (220min)
MLP	$\mathbf{461 \pm 8}$	$\mathbf{500 \pm 8}$	$\mathbf{628 \pm 5}$	$\mathbf{556 \pm 3}$
LSTM	77 ± 1	217 ± 3	261 ± 2	266 ± 2
Mixture	144 ± 1	421 ± 4	506 ± 4	433 ± 3
Product	164 ± 1	216 ± 1	174 ± 2	417 ± 6

Table 5. Mean return on the gaps transfer task for different architectures and data regimes. The returns are averaged over the logged rewards of one run between 1.4×10^9 and 1.5×10^9 environment transitions processed. Across all data regimes the MLP low-level policy performs best.

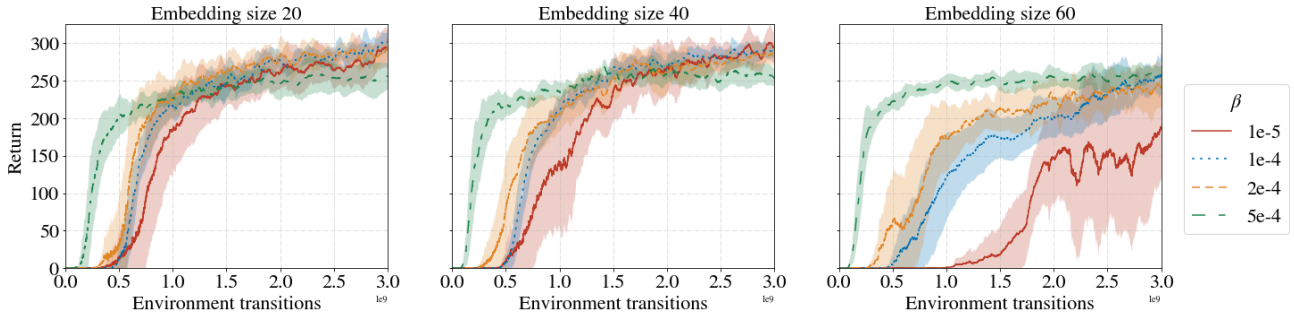

Figure 6. Go-to-target transfer performance for various latent space dimensionalities D and KL regularization strengths β (averaged over 5 runs). More highly regularized models encounter rewards faster but achieve slightly lower asymptotic performance.

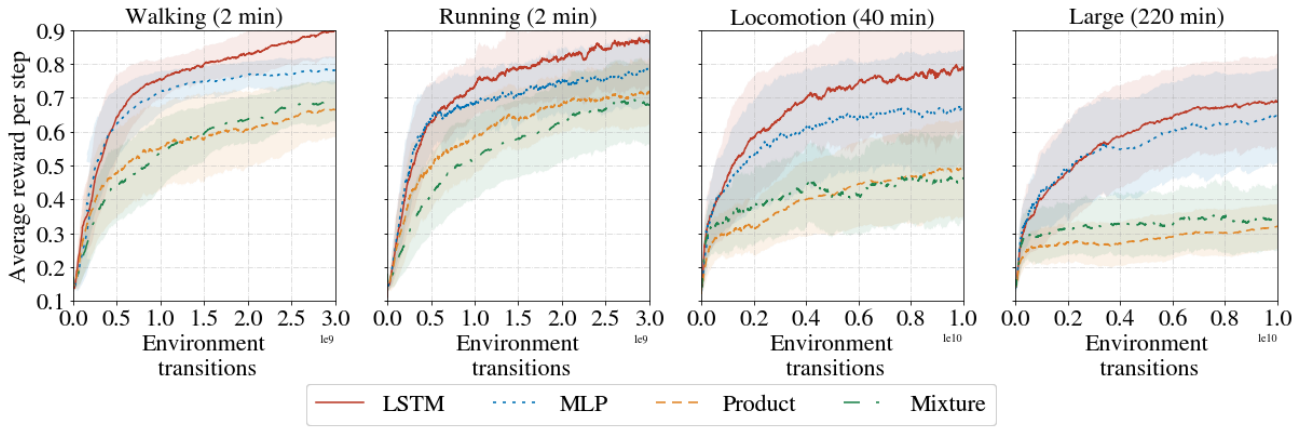

Figure 7. Architecture comparison across various data regimes. While modular architectures were competitive in small data regimes we found it very challenging to scale them up to large motion capture datasets.

Walls	Walking (2min)	Running (2min)	Locomotion (40min)	Large (220min)
MLP	$\mathbf{626 \pm 5}$	$\mathbf{569 \pm 5}$	$\mathbf{807 \pm 6}$	622 ± 5
LSTM	152 ± 2	368 ± 5	660 ± 7	$\mathbf{725 \pm 6}$
Mixture	159 ± 2	351 ± 4	512 ± 11	679 ± 10
Product	391 ± 5	264 ± 4	638 ± 7	647 ± 6

Table 6. Mean return on the walls transfer task for different architectures and data regimes. The returns are averaged over the logged rewards of one run between 0.95×10^9 and 1×10^9 environment transitions processed. The MLP low-level policy performs best in most settings.

5.5. Joint training with complementary tasks

We consider joint training with two tasks that are not well covered by the motion capture data: getting up from the ground (present in only a few short clips) and catching a ball (completely absent).

For these experiments we train a low-level policy on the largest motion capture clip set with an MLP low-level policy and an additional task. For joint training with the get up and stand task, at the beginning of each episode, the mocap tracking task is selected with 90 % probability and the get-up task is selected with 10 % probability. The ball catching task is sampled 20 % of the time. We find that in both settings, joint training slightly degrades tracking performance but it remains possible to transfer the learned skills to the locomotion transfer tasks with similar performance. We also investigate the transfer performance on the joint training tasks (see figure 8). In the case of get up and stand, the jointly trained low-level policy almost instantly solves the task. Indeed *even a randomly initialized high-level policy* will rarely fall and sometimes spontaneously get up from the ground. The low-level policy without joint training learns much more slowly and only learns to not fall but not to get up from the ground. In addition, the jointly trained low-level policy solves the task in a visually human-like fashion. Similarly, in the case of ball catching, with joint training the task is solved faster and with higher performance. See supplementary video 3 .

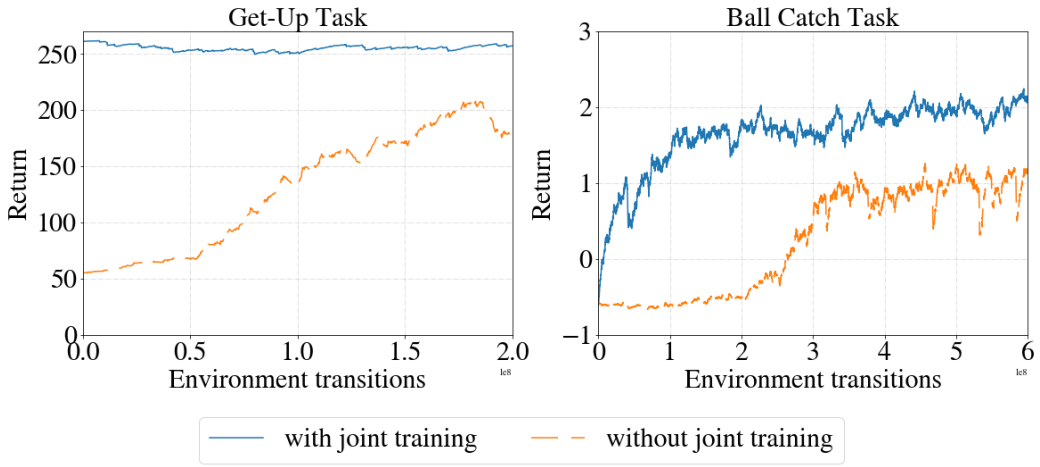

Figure 8. Performance on the complementary tasks with and without joint training. In the case of get up and stand, the jointly trained low-level policy almost instantly solves the task. *Even a randomly initialized high-level policy* will rarely fall and sometimes spontaneously get up from the ground.

6. Discussion

Learning reusable skills is a crucial challenge for control. This paper explored how to learn diverse, reusable skills via imitation and joint training on complementary tasks. Furthermore we studied which architectures facilitate effective skill reuse. Prior work (Merel et al., 2019b; 2020) on learning reusable skills from large motion capture datasets showed that it is possible to train reusable skills combining reinforcement learning and distillation. In these papers, expert policies are trained to track short segments of individual mocap clips. Expert trajectories are then distilled into a single model in a supervised step with a low-level policy that can be transferred to new tasks. In constrast, in this paper we train the low-level policy directly via RL on large sets of mocap data. This is significantly simpler, faster and cheaper than the two stage approach taken by Merel et al., 2019b; 2020. In addition, focusing on reinforcement learning only provides greater flexibility in terms of the training setting (as evidenced by our joint trained results).

We exhibited a setting in which it is possible to train reusable skills on large corpora of motion capture data. We compared a variety of different embedding space sizes and regularization strengths for a simple feed-forward architecture and found that transfer results are relevant insensitive to the embedding space size and strong regularization helps transfer on sparse reward tasks and has less of an effect on performance in dense reward tasks. Furthermore, we systematically compared a variety of different network architectures across a range of different data regimes and found that a simple feed-forward architecture worked best in our setting. In particular, we found that, despite considerable effort, it was difficult to scale modular architecture with several primitives to large motion capture datasets. Indeed, even in the low data regime a simple feed-forward low-level policy outperformed modular architectures. This result is at odds with results by Peng et al. (2019) and we speculate that a reason for this discrepancy is the more challenging nature of our tasks as well as the more realistic body. We leave a further investigation to future work. We found that the tasks we considered could be solved with reasonable performance using only about two minutes worth of motion capture data but asymptotic performance improved with the size of the underlying motion capture dataset.

In addition, we showed that it is possible to *jointly train* on complementary tasks providing an easy way for practitioners to ensure that skills not well covered by motion capture data are well represented in the learned embedding space. Future work in this space could include leveraging richer environments than considered in this paper, or combining imitation approaches with unsupervised skill discovery. Ultimately, we believe that the general approach of learning skills both via imitation and on complementary tasks jointly is very promising, offering the best of both worlds.

Acknowledgements

We would like to thank Greg Wayne for helpful feedback on an early draft of this paper, Yuval Tassa for help with MuJoCo and Saran Tunyasuvunakool for infrastructure and opensourcing support.

The data used in this project was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217.

References

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations (ICLR)*, 2018.

Abdolmaleki, A., Huang, S. H., Hasenclever, L., Neunert, M., Song, H. F., Zambelli, M., Martins, M. F., Heess, N., Hadsell, R., and Riedmiller, M. A distributional view on multi-objective policy optimization. In *ICML*, 2020.

Bacon, P.-L., Harb, J., and Precup, D. The option-critic architecture. In *Conference on Artificial Intelligence (AAAI)*, 2017.

Barreto, A., Borsa, D., Hou, S., Comanici, G., Aygün, E., Hamel, P., Toyama, D., Mourad, S., Silver, D., Precup, D., et al. The option keyboard: Combining skills in reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Bergamin, K., Clavet, S., Holden, D., and Forbes, J. R. Drecon: data-driven responsive control of physics-based characters. *Transactions on Graphics (TOG)*, 2019.

Chentanez, N., Müller, M., Macklin, M., Makoviychuk, V., and Jeschke, S. Physics-based motion capture imitation with deep reinforcement learning. In *International Conference on Motion, Interaction, and Games (MIG)*. ACM, 2018.

Da Silva, B. C., Baldassarre, G., Konidaris, G., and Barto, A. Learning parameterized motor skills on a humanoid robot. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2014.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations (ICLR)*, 2019.

Faloutsos, P., Van de Panne, M., and Terzopoulos, D. Composable controllers for physics-based character animation. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001.

Finn, C., Yu, T., Fu, J., Abbeel, P., and Levine, S. Generalizing skills with semi-supervised reinforcement learning. *arXiv preprint arXiv:1612.00429*, 2016.

Florensa, C., Duan, Y., and Abbeel, P. Stochastic neural networks for hierarchical reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017.

Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Ranzato, M. A., and Mikolov, T. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*. 2013.

Goyal, A., Islam, R., Strouse, D., Ahmed, Z., Larochelle, H., Botvinick, M., Levine, S., and Bengio, Y. Transfer and exploration via the information bottleneck. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJg8yhAqKm.

Goyal, A., Bengio, Y., Botvinick, M., and Levine, S. The variational bandwidth bottleneck: Stochastic evaluation on an information budget. In *International Conference on Learning Representations*, 2020.

Gregor, K., Rezende, D. J., and Wierstra, D. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Haarnoja, T., Hartikainen, K., Abbeel, P., and Levine, S. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pp. 1851–1860, 2018.

Hausman, K., Springenberg, J. T., Wang, Z., Heess, N., and Riedmiller, M. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations (ICLR)*, 2018.

Heess, N., Wayne, G., Tassa, Y., Lillicrap, T., Riedmiller, M., and Silver, D. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

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

James, S., Bloesch, M., and Davison, A. J. Task-embedded control networks for few-shot imitation learning. In *Conference on Robot Learning (CoRL)*, 2018.

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pp. 3581–3589, 2014.

Kiros, R., Salakhutdinov, R., and Zemel, R. S. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014.

Liu, L. and Hodgins, J. Learning to schedule control fragments for physics-based characters using deep q-learning. *Transactions on Graphics (TOG)*, 2017.

Liu, L., Yin, K., van de Panne, M., Shao, T., and Xu, W. Sampling-based contact-rich motion control. In *SIGGRAPH*. ACM, 2010.

Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., and Sermanet, P. Learning latent plans from play. In *Conference on Robot Learning (CoRL)*, 2019.

Merel, J., Tassa, Y., TB, D., Srinivasan, S., Lemmon, J., Wang, Z., Wayne, G., and Heess, N. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*, 2017.

Merel, J., Ahuja, A., Pham, V., Tunyasuvunakool, S., Liu, S., Tirumala, D., Heess, N., and Wayne, G. Hierarchical visuomotor control of humanoids. In *International Conference on Learning Representations (ICLR)*, 2019a.

Merel, J., Hasenclever, L., Galashov, A., Ahuja, A., Pham, V., Wayne, G., Teh, Y. W., and Heess, N. Neural probabilistic motor primitives for humanoid control. In *International Conference on Learning Representations (ICLR)*, 2019b.

Merel, J., Tunyasuvunakool, S., Ahuja, A., Tassa, Y., Hasenclever, L., Pham, V., Erez, T., Wayne, G., and Heess, N. Catch & carry: Reusable neural controllers for vision-guided whole-body tasks. In *SIGGRAPH 2020*, 2020.

Mülling, K., Kober, J., Kroemer, O., and Peters, J. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 2013.

Paraschos, A., Daniel, C., Peters, J. R., and Neumann, G. Probabilistic movement primitives. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

Peng, X. B., Abbeel, P., Levine, S., and van de Panne, M. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *Transactions on Graphics (TOG)*, 2018.

Peng, X. B., Chang, M., Zhang, G., Abbeel, P., and Levine, S. Mcp: Learning composable hierarchical control with multiplicative compositional policies. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

Riedmiller, M., Hafner, R., Lampe, T., Neunert, M., Degrave, J., Wiele, T., Mnih, V., Heess, N., and Springenberg, J. T. Learning by playing solving sparse reward tasks from scratch. In *International Conference on Machine Learning (ICML)*, 2018.

Schaal, S., Peters, J., Nakanishi, J., and Ijspeert, A. Learning movement primitives. In *Robotics research. the eleventh international symposium*. Springer, 2005.

Song, H. F., Abdolmaleki, A., Springenberg, J. T., Clark, A., Soyer, H., Rae, J. W., Noury, S., Ahuja, A., Liu, S., Tirumala, D., Heess, N., Belov, D., Riedmiller, M., and Botvinick, M. M. V-MPO: On-policy maximum a posteriori policy optimization for discrete and continuous control. In *International Conference on Learning Representations (ICLR)*, 2020.

Strudel, R. A. M., Pashevich, A., Kalevatykh, I., Laptev, I., Sivic, J., and Schmid, C. Learning to combine primitive skills: A step towards versatile robotic manipulation. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2020.

Sutton, R. S., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 1999.

Tassa, Y., Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., and Heess, N. dm_control: Software and tasks for continuous control, 2020.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012.

Van Seijen, H., Fatemi, M., Romoff, J., Laroche, R., Barnes, T., and Tsang, J. Hybrid reward architecture for reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 5392–5402, 2017.

Wang, Z., Merel, J. S., Reed, S. E., de Freitas, N., Wayne, G., and Heess, N. Robust imitation of diverse behaviors. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Warde-Farley, D., de Wiele, T. V., Kulkarni, T., Ionescu, C., Hansen, S., and Mnih, V. Unsupervised control through non-parametric discriminative rewards. In *International Conference on Learning Representations (ICLR)*, 2019.

Wulfmeier, M., Abdolmaleki, A., Hafner, R., Springenberg, J. T., Neunert, M., Hertweck, T., Lampe, T., Siegel, N., Heess, N., and Riedmiller, M. A. Regularized hierarchical policies for compositional transfer in robotics. *arXiv preprint arXiv:1906.11228*, 2019.

Yang, C., Yuan, K., Merkt, W., Komura, T., Vijayakumar, S., and Li, Z. Learning whole-body motor skills for humanoids. In *International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018.