ASE: Large-Scale Reusable Adversarial Skill Embeddings for Physically Simulated Characters

XUE BIN PENG, University of California, Berkeley, USA and NVIDIA, Canada
YUNRONG GUO, NVIDIA, Canada
LINA HALPER, NVIDIA, Canada
SERGEY LEVINE, University of California, Berkeley, USA
SANJA FIDLER, University of Toronto, Canada and NVIDIA, Canada

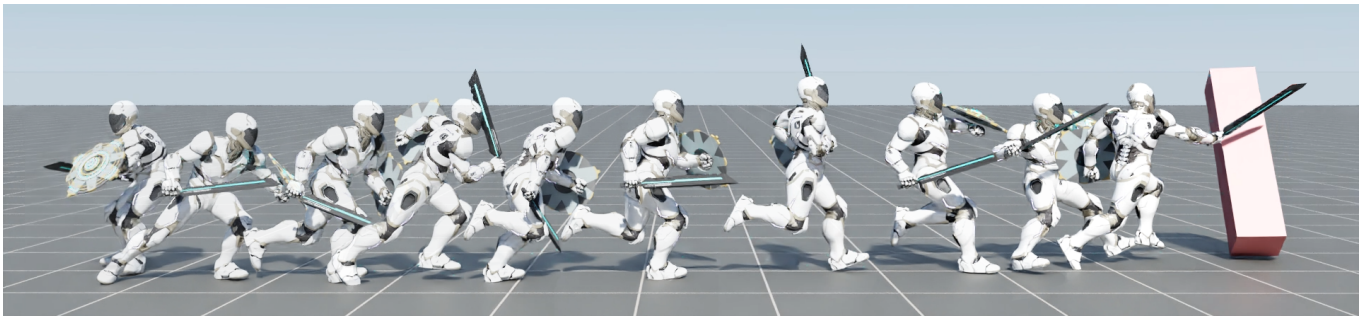


Fig. 1. Our framework enables physically simulated characters to learn versatile and reusable skill embeddings from large unstructured motion datasets, which can then be applied to produce life-like behaviors for new tasks. Here, a character is utlizing behaviors from a learned skill embedding in order to run to a target and knock it over.

The incredible feats of athleticism demonstrated by humans are made possible in part by a vast repertoire of general-purpose motor skills, acquired through years of practice and experience. These skills not only enable humans to perform complex tasks, but also provide powerful priors for guiding their behaviors when learning new tasks. This is in stark contrast to what is common practice in physics-based character animation, where control policies are most typically trained from scratch for each task. In this work, we present a large-scale data-driven framework for learning versatile and reusable skill embeddings for physically simulated characters. Our approach combines techniques from adversarial imitation learning and unsupervised reinforcement learning to develop skill embeddings that produce life-like behaviors, while also providing an easy to control representation for use on new downstream tasks. Our models can be trained using large datasets of unstructured motion clips, without requiring any task-specific annotation or segmentation of the motion data. By leveraging a massively parallel GPU-based simulator, we are able to train skill embeddings using over a decade of simulated experiences, enabling our model to learn a rich and versatile repertoire of skills. We show that a single pre-trained model can be effectively applied to perform a diverse set of new tasks. Our system also allows users to specify tasks through simple reward functions, and the skill embedding then enables the character to automatically synthesize complex and naturalistic strategies in order to achieve the task objectives.

CCS Concepts: • **Computing methodologies** → **Procedural animation**; *Control methods*; *Adversarial learning*.

Additional Key Words and Phrases: character animation, reinforcement learning, adversarial imitation learning, unsupervised reinforcement learning

ACM Reference Format:
Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. 2022. ASE: Large-Scale Reusable Adversarial Skill Embeddings for Physically Simulated Characters. *ACM Trans. Graph.* 41, 4, Article 94 (July 2022), 18 pages. https://doi.org/10.1145/3528223.3530110

1 INTRODUCTION

Humans are capable of performing an awe-inspiring variety of complex tasks by drawing on a vast repertoire of motor skills. This repertoire is built up over a lifetime of interaction with the environment, leading to general-purpose skills that can be widely reused to accomplish new tasks. This is at odds with what is conventional practice in physics-based character animation and reinforcement learning, where control policies are typically trained from scratch, to specialize in a single specific task. Developing more versatile and reusable models of motor skills can enable agents to solve tasks that would otherwise be prohibitively challenging to learn from scratch. However, manually constructing a sufficiently rich set of tasks and reward functions that can give rise to behaviors that are as diverse and versatile as those of humans would require an immense engineering effort.

How then can we endow agents with large and versatile repertoires of skills? Some inspiration may be drawn from domains such

Authors' addresses: Xue Bin Peng, xbpeng@berkeley.edu, University of California, Berkeley, USA and NVIDIA, Canada; Yunrong Guo, kellyg@nvidia.com, NVIDIA, Canada; Lina Halper, lhalper@nvidia.com, NVIDIA, Canada; Sergey Levine, svlevine@eecs.berkeley.edu, University of California, Berkeley, USA; Sanja Fidler, lhalper@nvidia.com, University of Toronto, Canada and NVIDIA, Canada.

0730-0301/2022/7-ART94
https://doi.org/10.1145/3528223.3530110

as computer vision and natural language processing, where large expressive models trained on massive datasets have been a central component of major advances [Brown et al. 2020; Deng et al. 2009; Doersch et al. 2015; He et al. 2016; Hinton and Salakhutdinov 2006; Jing and Tian 2021; Kingma and Welling 2014; Mikolov et al. 2013; Oord et al. 2018a]. Not only can these models solve challenging tasks, but they also provide powerful priors that can be reused for a wide range of downstream applications [Brown et al. 2020; Donahue et al. 2014; Lester et al. 2021; Raffel et al. 2020; Sermanet et al. 2018; Vinyals et al. 2014]. A similar data-driven paradigm may also be applicable for developing more general and versatile motor skills. Rather than laboriously designing a rich set of training tasks that leads to a flexible range of behaviors, we can instead provide the agent with a large unstructured motion dataset, containing examples of behaviors that we would like the agent to acquire. The agent can then be trained to perform a large variety of skills by imitating the behaviors depicted in the dataset. By modeling the learned skills with representations that are suitable for reuse, we may then be able to develop more capable and versatile motor skill models that can be re-purposed for a wide range of new applications.

To this end, we present adversarial skill embeddings (ASE), a scalable data-driven approach for learning general and reusable motor skills for physically simulated characters. Given a large dataset of unstructured motion clips, our system trains a low-level latent variable model to produce behaviors that resemble those shown in the dataset. Importantly, the learned skills need not exactly match any particular motion clip. Instead, our pre-training objective encourages the model to learn a diverse set of skills that exhibits the general behavioral characteristics depicted in the dataset. This is enabled by an information maximization objective, which spurs the model to discover diverse and distinct behaviors. Once trained, the low-level model can then be used to define an abstract action space for a high-level policy to perform new downstream tasks. By pre-training the low-level model with motion clips of naturalistic behaviors recorded from human actors, the model can be used to synthesize agile and life-like behaviors (see Figure 1), without requiring any additional motion data or extensive reward engineering.

The central contribution of this work is a scalable adversarial imitation learning framework that enables physically simulated characters to acquire large repertoires of complex and versatile motor skills, which can then be reused to perform a wide range of downstream tasks. Our system is able to learn behaviors from large unstructured motion datasets, containing over a hundred diverse motion clips. By leveraging Isaac Gym, NVIDIA's massively parallel GPU simulator, our system is able to pre-train general-purpose motor skill models using the equivalent of a decade of simulated experiences. We propose a number of important design decisions that increases the diversity of the skills acquired during the pre-training process, and improve the overall effectiveness of the models when transferred to downstream tasks. Furthermore, by pre-training the model to recover from random initial states, we can develop highly robust recovery strategies that can agilely and consistently recover from substantial external perturbations. Once trained, these recovery strategies can then be seamless integrated into new tasks, without any additional algorithmic overhead. Code and data for this work is available at https://xbpeng.github.io/projects/ASE/.

2 RELATED WORK

Developing controllers that can produce agile and life-like behaviors has been one of the core challenges of computer animation [Coros et al. 2010; Hodgins et al. 1995; Raibert and Hodgins 1991; Yin et al. 2007; Zordan and Hodgins 2002]. Optimization techniques, based on trajectory optimization or reinforcement learning, are some of the most widely used methods for synthesizing controllers for simulated characters [de Lasa et al. 2010; Levine and Koltun 2013; Mordatch et al. 2012; Tan et al. 2014; van de Panne et al. 1994; Yin et al. 2008]. These techniques synthesize controllers by optimizing an objective function, which encodes properties of a desired skill. While these methods are able to synthesize physically plausible motions for a wide variety of behaviors [Al Borno et al. 2013; Endo et al. 2005; Gehring et al. 2016; Tan et al. 2011; Tassa et al. 2012; Wampler et al. 2014], designing effective objectives that lead to naturalistic motions can often involve a tedious and labour-intensive development process. Heuristics derived from prior knowledge regarding the characteristics of natural behaviors can be incorporated into the objective in order to improve motion quality, such as enforcing energy efficiency, lateral symmetry, stability, and many more [Geyer et al. 2003; Mordatch et al. 2012; Wang et al. 2009; Yu et al. 2018]. However, these heuristics are generally not broadly applicable for all skills, and different skills often require different carefully curated sets of heuristics in order to produce life-like behaviors. Incorporating more biologically accurate simulation models can also improve motion quality [Geijtenbeek et al. 2013; Jiang et al. 2019; Wang et al. 2012], but may nonetheless produce unnatural behaviors without the appropriate objective functions [Song et al. 2020].

Data-driven methods: The difficulty of designing controllers and objective functions that produce naturalistic behaviors has motivated the widespread adoption of data-driven methods for physics-based character animation [Da Silva et al. 2008; Kwon and Hodgins 2017; Lee et al. 2010; Sharon and van de Panne 2005; Xu and Karamouzas 2021; Zordan and Hodgins 2002]. These techniques can produce highly life-like motions by imitating reference motion data. Many of these methods utilize some form of motion tracking, where controllers imitate reference motions by explicitly tracking the sequence of target poses specified by a motion clip [Lee et al. 2010; Libin Liu 2018; Liu et al. 2016, 2010; Peng et al. 2018; Sok et al. 2007]. However, it can be difficult to apply tracking-based techniques to imitate behaviors from large and diverse motion datasets. Composition of disparate skills often requires some form of a motion planner to select the appropriate motion for the character track in a given scenario [Bergamin et al. 2019; Liu et al. 2012; Park et al. 2019; Peng et al. 2017], which can be difficult to construct for complex tasks. More recently, Peng et al. [2021] proposed adversarial motion priors, which allow characters to perform tasks while imitating behaviors from large unstructured motion datasets. This enables characters to automatically compose and sequence different skills, without being constrained to explicitly track a particular motion clip. While these motion imitation methods can produce high quality results, they predominantly train models from scratch for each task. This *tabula-rasa* approach can incur significant drawbacks in terms of sample efficiency and can limit the complexity of tasks that characters can accomplish, requiring agents to relearn common behaviors, such

as walking, over and over again for each new task. Our work aims to learn reusable motor skills from large motion dataset, which can then be leveraged to synthesize naturalistic behaviors for new tasks without requiring additional motion data or retraining from scratch each time.

Hierarchical models: One way to reuse previously acquired skills for new tasks is by building a hierarchical model. This approach generally consists of two stages: a pre-training stage, where a collection of controllers are trained to specialize in different low-level skills (e.g., walking in different directions) [Coros et al. 2009; Hausman et al. 2018; Heess et al. 2016; Liu et al. 2012; Won et al. 2021], and a task-training stage, where the low-level controllers are integrated into a control hierarchy with a high-level controller that leverages the low-level skills to perform new downstream tasks (e.g., navigating to a target location) [Faloutsos et al. 2001; Ling et al. 2020; Liu and Hodgins 2017; Mordatch et al. 2010; Peng et al. 2017; Ye and Liu 2010]. General motion tracking models trained to imitate reference motion data can be used as effective low-level controllers [Chentanez et al. 2018; Wang et al. 2020; Won et al. 2020]. To apply these tracking models to downstream tasks, a high-level kinematic motion planner can be used to specify target motion trajectories for guiding the low-level controller towards completing a desired task [Bergamin et al. 2019; Park et al. 2019]. However, constructing such a motion planner typically requires maintaining a motion dataset for use on downstream tasks. Latent variable models trained via motion tracking can obviate the need for an explicit motion planner by allowing the high-level controller to steer low-level behaviors via latent variables [Hasenclever et al. 2020; Luo et al. 2020; Lynch et al. 2020; Merel et al. 2019, 2020; Peng et al. 2019; Wang et al. 2017]. While motion tracking can be an effective method for constructing low-level controllers, the tracking objective can ultimately limit a model's ability to produce behaviors that are not depicted in the original dataset, thereby potentially limiting a model's flexibility to develop more general and versatile skills. In this work, instead of using motion tracking, we present a more flexible pre-training approach that combines adversarial imitation learning and unsupervised reinforcement learning, which provides agents more flexibility in developing novel and versatile behaviors.

Unsupervised reinforcement learning: In the unsupervised reinforcement learning setting, agents are not provided with an explicit task objective. Instead, the goal is for the agent to learn skills autonomously by optimizing an *intrinsic* objective derived from the agent's own past experiences. These intrinsic objectives typically motivate the agent to seek novelty or diversity, which can be quantified using surrogate metrics, such as errors from model predictions [Achiam and Sastry 2017; Burda et al. 2019; Pathak et al. 2017, 2019; Stadie et al. 2015], state visitation counts [Bellemare et al. 2016; Florensa et al. 2017; Fu et al. 2017; Strehl and Littman 2008; Tang et al. 2017], or entropy of the agent's state distribution [Hazan et al. 2019; Liu and Abbeel 2021b]. In this work, we will leverage a class of techniques based on maximizing mutual information between abstract skills and their resulting behaviors [Baumli et al. 2020; Eysenbach et al. 2019; Gregor et al. 2017; Liu and Abbeel 2021a; Sharma et al. 2020]. While unsupervised reinforcement learning has shown promising results on relatively low-dimensional problems, when applied to more complex settings with large numbers of degrees-of-freedom, these unsupervised reinforcement learning techniques often fail to discover useful behaviors. Furthermore, without additional prior knowledge, it is unlikely that unsupervised reinforcement learning techniques alone will discover naturalistic, life-like behaviors that resemble the motions of humans or animals. Therefore, our models are trained using a combination of an adversarial imitation learning objective and an unsupervised information maximization objective. Adversarial imitation learning allows our models to mimic realistic motions from user-provided data, while unsupervised reinforcement learning techniques allow the model to learn skill representations that are more directable and mitigate the common problem of mode-collapse associated with adversarial training methods by promoting more diverse behaviors.

3 REINFORCEMENT LEARNING BACKGROUND

In our framework, both pre-training and transfer tasks will be modeled as reinforcement learning problems, where an agent interacts with an environment according to a policy π in order to optimize a given objective [Sutton and Barto 1998]. At each time step t, the agent observes the state $\mathbf{s}_t$ of the system, then samples an action from a policy $\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)$. The agent then executes the action, which leads to a new state $\mathbf{s}_{t+1}$, sampled according to the dynamics of the environment $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, and a scalar reward $r_t = r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$. The agent's objective is to learn a policy that maximizes its expected discounted return $J(\pi)$,

$$J(\pi) = \mathbb{E}_{p(\tau|\pi)}\left[\sum_{t=0}^{T-1} \gamma^t r_t\right], \tag{1}$$

where $p(\tau|\pi) = p(\mathbf{s}_0)\prod_{t=0}^{T-1} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\pi(\mathbf{a}_t|\mathbf{s}_t)$ represents the likelihood of a trajectory $\tau = \{\mathbf{s}_0, \mathbf{a}_0, r_0, \mathbf{s}_1, ..., \mathbf{s}_{T-1}, \mathbf{a}_{T-1}, r_{T-1}, \mathbf{s}_T\}$ under π. $p(\mathbf{s}_0)$ is the initial state distribution, T denotes the time horizon of a trajectory, and $\gamma \in [0, 1]$ is a discount factor. The reward function thus provides an interface through which users can specify the task that an agent should perform. However, designing effective reward functions that elicit the desired behaviors from an agent often involves a tedious design process. Therefore, constructing a sufficiently rich set of tasks that leads to diverse and sophisticated motor skills can present a daunting engineering endeavour.

4 OVERVIEW

In this paper, we present adversarial skill embeddings (ASE), a scalable data-driven approach for learning reusable motor skills for physically simulated characters. An overview of our framework is provided in Figure 2. The training processes consists of two stages: a pre-training stage, and a task-training stage. During pre-training, the character is given a dataset of motion clips $\mathcal{M} = \{m^i\}$, which provides examples of the kinds of behaviors that the agent should learn. Each motion clip $m^i = \{\mathbf{s}_t^i\}$ is represented as a sequence of states that depicts a particular behavior. This dataset is used by an adversarial imitation learning procedure to train a low-level skill-conditioned policy $\pi(\mathbf{a}|\mathbf{s}, \mathbf{z})$, which maps latent variables $\mathbf{z}$ to behaviors that resemble motions shown in the dataset. Unlike most prior motion imitation techniques [Lee et al. 2010; Liu et al. 2010;

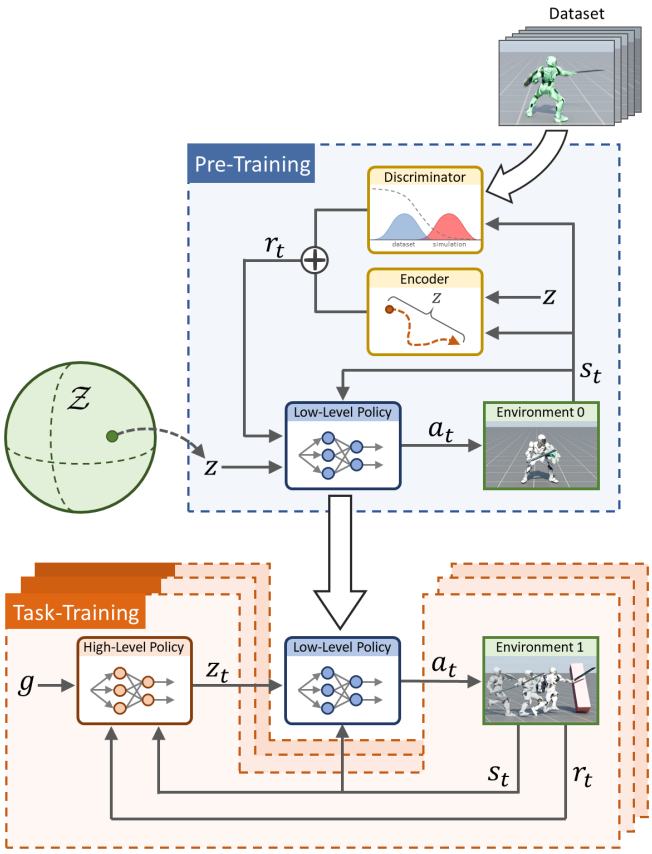


Fig. 2. The ASE framework consists of two stages: pre-training and transfer. During pre-training a low-level policy $\pi(\mathbf{a}|\mathbf{s},\mathbf{z})$ is trained to map latent skills $\mathbf{z}$ to behaviors that resemble motions depicted in a dataset. The policy is trained to model a diverse repertoire of skills by using a reward function that combines an adversarial imitation objective, specified by a discriminator D, and an unsupervised skill discovery, specified by an encoder q. After pre-training, π can be transferred to new tasks by using a task-specific high-level policy $\omega(\mathbf{z}|\mathbf{s},\mathbf{g})$ to specify latent variables $\mathbf{z}$ for directing the low-level policy towards accomplishing a task-specific goal $\mathbf{g}$.

Merel et al. 2020; Peng et al. 2018], our models need not exactly match any particular motion clip in the dataset. Instead, the objective is for a policy to discover a diverse and versatile set of skills that exhibits the general characteristics of the motion data. Then in the task-training stage, the low-level policy is reused to perform new tasks by training a task-specific high-level policy $\omega(\mathbf{z}|\mathbf{s},\mathbf{g})$, which specifies latent variables for directing the low-level policy towards completing the task objectives. The high-level policy is conditioned on a task-specific goal $\mathbf{g}$, and can be trained without any motion data. But since the low-level policy was trained with a motion dataset, it allows the character to produce naturalistic behaviors even in the absence of motion data.

5 ADVERSARIAL SKILL EMBEDDINGS

In ASE, the skills are modeled by a skill-conditioned policy $\pi(\mathbf{a}|\mathbf{s},\mathbf{z})$, where each skill is represented by a latent variable $\mathbf{z} \in \mathcal{Z}$ sampled according to a prior over latent skills $\mathbf{z} \sim p(\mathbf{z})$. An effective skill model should produce realistic behaviors, while also providing a *directable* skill representation that can be conveniently utilized to perform new downstream tasks. This is accomplished by combining a motion imitation objective with an unsupervised skill discovery objective, which encourages the agent to develop a set of diverse skills that conforms to the behavioral characteristics specified by a dataset of motion clips. Given a motion dataset $\mathcal{M}$, the pre-training objective is given by:

$$\max_{\pi} \quad -D_{\text{JS}}\left(d^{\pi}(\mathbf{s},\mathbf{s}')\middle\|d^{\mathcal{M}}(\mathbf{s},\mathbf{s}')\right) + \beta\, I\left(\mathbf{s},\mathbf{s}';\mathbf{z}\middle|\pi\right). \tag{2}$$

The first term is the imitation objective, which encourages the policy to produce realistic behaviors by matching the marginal state-transition distribution of the dataset, where D_{JS} denotes the Jensen-Shannon divergence, $d^{\pi}(\mathbf{s},\mathbf{s}')$ represents the marginal distribution of state transitions induced by π, and $d^{\mathcal{M}}(\mathbf{s},\mathbf{s}')$ is the marginal state-transition distribution of the motion dataset $\mathcal{M}$. Intuitively, this objective encourages a low-level policy to reproduce the *entire* distribution of behaviors from which the motion data were sampled, rather than simply imitating the individual motion clips. An effective skill embedding, should therefore be able to generalize to behaviors that are not depicted in the motion clips, but still conform to the general characteristics of the dataset. The second term is the unsupervised skill discovery objective, which encourages the policy to develop a diverse and distinct set of skills by maximizing the mutual information between a skill $\mathbf{z}$ and the resulting behaviors, as represented by state transitions. β is a manually specified coefficient. This objective encourages the low-level policy to be *directable*, such that different values of $\mathbf{z}$ leads to distinct and predictable behaviors. Computing these quantities exactly is generally intractable. In the following subsections, we will discuss techniques for constructing a practical approximation of this objective.

5.1 Imitation Objective

Since computing the Jensen-Shannon divergence can be computationally challenging, we will leverage a variational approximation in the form of a GAN-like framework [Goodfellow et al. 2014]. First, we introduce a discriminator $D(\mathbf{s},\mathbf{s}')$, which is trained to classify if a given state transition $(\mathbf{s},\mathbf{s}')$ is from the dataset or was produced by the agent,

$$\min_{D} \quad -\mathbb{E}_{d^{\mathcal{M}}(\mathbf{s},\mathbf{s}')}\left[\log\left(D\left(\mathbf{s},\mathbf{s}'\right)\right)\right] - \mathbb{E}_{d^{\pi}(\mathbf{s},\mathbf{s}')}\left[\log\left(1-D\left(\mathbf{s},\mathbf{s}'\right)\right)\right]. \tag{3}$$

The policy can then be trained using rewards specified by $r_t = -\log\left(1 - D\left(\mathbf{s}_t,\mathbf{s}_{t+1}\right)\right)$. It can be shown that this adversarial training procedure minimizes the Jensen-Shannon divergence between $d^{\pi}(\mathbf{s},\mathbf{s}')$ and $d^{\mathcal{M}}(\mathbf{s},\mathbf{s}')$ [Ke et al. 2021; Nowozin et al. 2016]. This objective is similar to the one presented by Peng et al. [2021], and can be interpreted as training a policy to produce behaviors that appear to the discriminator as being indistinguishable from motions shown in the dataset. But unlike Peng et al. [2021], which leverages a discriminator to shape the behavioral style of policies trained from scratch for a particular task, our framework utilizes a discriminator to learn reusable skill embeddings, which can later be used to

perform new tasks. However, simply matching the behavioral distribution of the data, does not ensure that the low-level policy learns a *directable* skill representation that can be effectively reused for new tasks. In the next subsection, we will present a skill discovery objective for acquiring more directable skill representations.

5.2 Skill Discovery Objective

The imitation objective encourages the policy to produce behaviors that resemble the dataset, but it does not ensure that π learns a skill representation that is easy to control and amenable for reuse on downstream tasks. Furthermore, the adversarial training procedure described in Section 5.1 is prone to mode-collapse, where the policy reproduces only a narrow range of behaviors depicted in the original dataset. To address these shortcomings, we incorporate a skill discovery objective, commonly used in unsupervised reinforcement learning, to encourage the policy to model more diverse and distinct behaviors. The objective aims to maximize the mutual information $I(\mathbf{s}, \mathbf{s}'; \mathbf{z}|\pi)$ between the state transitions $(\mathbf{s}, \mathbf{s}')$ produced by a policy π, and a latent skill variable $\mathbf{z}$, drawn from a distribution of skills $\mathbf{z} \sim p(\mathbf{z})$. The intuition for this choice of objective can be made more apparent if we consider the definition of mutual information,

$$I(\mathbf{s}, \mathbf{s}'; \mathbf{z}|\pi) = \mathcal{H}(\mathbf{s}, \mathbf{s}'|\pi) - \mathcal{H}(\mathbf{s}, \mathbf{s}'|\mathbf{z}, \pi). \tag{4}$$

Maximizing the mutual information can be interpreted as maximizing the *marginal* state-transition entropy $\mathcal{H}(\mathbf{s}, \mathbf{s}'|\pi)$ induced by π, while minimizing the *conditional* state-transition entropy $\mathcal{H}(\mathbf{s}, \mathbf{s}'|\mathbf{z}, \pi)$ produced by a particular skill $\mathbf{z}$. In other words, the policy should learn a set of skills that produces diverse behaviors, while each individual skill should produce a distinct behavior.

Unfortunately, computing Equation 4 is intractable in most domains. Therefore, we will approximate $I(\mathbf{s}, \mathbf{s}'; \mathbf{z}|\pi)$ using a variational lower bound presented by Gregor et al. [2017] and Eysenbach et al. [2019]. But unlike these prior methods, instead of learning a discrete set of skills, our policy will be trained to model a continuous latent space of skills, which is more amenable to interpolation and blending between different behaviors. To derive this objective, we first note that determining the marginal entropy $\mathcal{H}(\mathbf{s}, \mathbf{s}')$ is also generally intractable. A more practical objective can be obtained by taking advantage of the symmetry of mutual information,

$$\begin{aligned} I(\mathbf{s}, \mathbf{s}'; \mathbf{z}|\pi) &= I(\mathbf{z}; \mathbf{s}, \mathbf{s}'|\pi) && (5)\\ &= \mathcal{H}(\mathbf{z}) - \mathcal{H}(\mathbf{z}|\mathbf{s}, \mathbf{s}', \pi). && (6) \end{aligned}$$

This decomposition removes the need to estimate $\mathcal{H}(\mathbf{s}, \mathbf{s}'|\pi)$, and instead we now only need to determine the entropy over skills $\mathcal{H}(\mathbf{z})$. Note that, since we are free to define $p(\mathbf{z})$ and it is independent of π, $\mathcal{H}(\mathbf{z})$ is effectively a constant and does not influence the optimization process. Then, to obtain a tractable approximation of $\mathcal{H}(\mathbf{z}|\mathbf{s}, \mathbf{s}', \pi)$, we will introduce a variational approximation $q(\mathbf{z}|\mathbf{s}, \mathbf{s}')$ of the conditional skill distribution $p(\mathbf{z}|\mathbf{s}, \mathbf{s}', \pi)$. Since the cross-entropy between p and q is an upper bound on the entropy of p, $\mathcal{H}(p) \leq \mathcal{H}(p, q)$, we obtain the following variational lower bound on $I(\mathbf{s}, \mathbf{s}'; \mathbf{z}|\pi)$,

$$\begin{aligned} I(\mathbf{s}, \mathbf{s}'; \mathbf{z}|\pi) &= \mathcal{H}(\mathbf{z}) + \mathbb{E}_{p(\mathbf{z})}\mathbb{E}_{p(\mathbf{s},\mathbf{s}'|\pi,\mathbf{z})}\left[\log p(\mathbf{z}|\mathbf{s}, \mathbf{s}', \pi)\right] && (7)\\ &\geq \max_q \mathcal{H}(\mathbf{z}) + \mathbb{E}_{p(\mathbf{z})}\mathbb{E}_{p(\mathbf{s},\mathbf{s}'|\pi,\mathbf{z})}\left[\log q(\mathbf{z}|\mathbf{s}, \mathbf{s}')\right], && (8) \end{aligned}$$

where the lower bound is tight if $q = p$. We will refer to $q(\mathbf{z}|\mathbf{s}, \mathbf{s}')$ as the *encoder*. This skill discovery objective encourages a policy to produce distinct behaviors for each skill $\mathbf{z}$, such that the encoder can easily recover the $\mathbf{z}$ that produced a particular behavior.

5.3 Surrogate Objective

Using the above variational approximations, we can construct a surrogate objective that approximates Equation 2,

$$\begin{aligned} \arg\max_\pi \quad \mathbb{E}_{p(\mathbf{z})}\mathbb{E}_{p(\tau|\pi,\mathbf{z})}\Bigg[\sum_{t=0}^{T-1}\gamma^t\big(&-\log\left(1 - D\left(\mathbf{s}_t, \mathbf{s}_{t+1}\right)\right)\\ &+ \beta \log q\left(\mathbf{z}|\mathbf{s}_t, \mathbf{s}_{t+1}\right)\big)\Bigg]. \end{aligned} \tag{9}$$

The reward for the policy at each timestep is then specified by

$$r_t = -\log\left(1 - D(\mathbf{s}_t, \mathbf{s}_{t+1})\right) + \beta \log q\left(\mathbf{z}_t|\mathbf{s}_t, \mathbf{s}_{t+1}\right). \tag{10}$$

This objective in effect encourages a model to develop a set of distinct skills, which also produce behaviors that resemble the dataset. A similar objective was previously proposed by Chen et al. [2016] for learning disentangled representations for image synthesis with adversarial generative networks. Similar techniques have also been applied to acquire disentangled skill representations for imitation learning [Hausman et al. 2017; Li et al. 2017]. However, these methods have generally only been effective for low-dimensional systems, such as driving and simple RL benchmarks, and have not shown to be effective for more complex domains. Furthermore, the skill embeddings learned by these prior methods have not been demonstrated as being effective for hierarchical control. In this work, we show that our method can in fact learn a rich repertoire of sophisticated motor skills for complex physically simulated characters, and in the following sections, we propose a number of design decisions that improve the quality of the resulting motions, as well as allow the skills to be effectively utilized for hierarchical control.

6 LOW-LEVEL POLICY

Although the method described in Section 5 provides a conceptually elegant approach for learning skill embeddings, a number of challenges need to be addressed in order to learn effective skill representations from large datasets in practice. In this section, we detail design improvements for training an effective low-level policy $\pi(\mathbf{a}|\mathbf{s}, \mathbf{z})$, including techniques for preventing low-quality out-of-distribution samples from the latent space, improving stability during training, improving the responsiveness of the low-level policy, and developing robust recovery strategies that can be seamlessly integrated into downstream tasks.

6.1 Latent Space

First, we consider the design of the latent space of skills $\mathcal{Z}$ and the prior over skills $p(\mathbf{z})$. In GAN frameworks, a common choice is to model the latent distribution using a Gaussian $p(\mathbf{z}) = \mathcal{N}(0, I)$. However, this design results in an unbounded latent space, where latents that are far from the origin of $\mathcal{Z}$ may produce low-quality samples. Since the latent space will be used as the action space for a high-level policy $\omega(\mathbf{z}|\mathbf{s}, \mathbf{g})$, an unbounded $\mathcal{Z}$ can lead ω to select latents that are far from the origin, which may result in unnatural

motions. To better ensure high-quality samples, bounded latent spaces have also been used, where $p(\mathbf{z})$ can be modeled with a uniform distribution $\mathcal{U}[-1, 1]$ or a truncated Gaussian [Brock et al. 2019]. In this work, we will model the latent space as a hypersphere $\mathcal{Z} = \{\mathbf{z} : ||\mathbf{z}|| = 1\}$ [Karras et al. 2019], and $p(\mathbf{z})$ will be a uniform distribution on the surface of the sphere. Samples can be drawn from $p(\mathbf{z})$ by normalizing samples from a standard Gaussian distribution,

$$\bar{\mathbf{z}} \sim \mathcal{N}(0, I), \qquad \mathbf{z} = \bar{\mathbf{z}}/||\bar{\mathbf{z}}||. \tag{11}$$

This provides the model with a bounded latent space, which can reduce the likelihood of unnatural behaviors arising from out-of-distribution latents. As we will discuss in Section 7, this choice of latent space can also facilitate exploration for downstream tasks.

6.2 Skill Encoder

Since the the latent space is modeled as a hypersphere, the skill encoder will be modeled using a von Mises-Fisher distribution,

$$q(\mathbf{z}|\mathbf{s}, \mathbf{s}') = \frac{1}{Z}\exp\left(\kappa\, \mu_q(\mathbf{s}, \mathbf{s}')^T \mathbf{z}\right), \tag{12}$$

which is the analogue to the Gaussian distribution on the surface of a sphere. $\mu_q(\mathbf{s}, \mathbf{s}')$ is the mean of the distribution, which must be normalized $||\mu_q(\mathbf{s}, \mathbf{s}')|| = 1$, Z is a normalization constant, and κ is a scaling factor. The encoder can then be trained by maximizing the log-likelihood of samples $(\mathbf{z}, \mathbf{s}, \mathbf{s}')$ collected from the policy,

$$\max_q \quad \mathbb{E}_{p(\mathbf{z})}\mathbb{E}_{d^\pi(\mathbf{s},\mathbf{s}'|\mathbf{z})}\left[\kappa\, \mu_q(\mathbf{s}, \mathbf{s}')^T z\right], \tag{13}$$

where $d^\pi(\mathbf{s}, \mathbf{s}'|\mathbf{z})$ represents the likelihood of observing a state transition under π given a particular skill $\mathbf{z}$.

6.3 Discriminator

Adversarial imitation learning is known to be notoriously unstable. To improve training stability and quality of the resulting motions, we incorporate the gradient penalty regularizers used by Peng et al. [2021]. The discriminator is trained using the following objective,

$$\begin{aligned}\min_D \quad & -\mathbb{E}_{d^{\mathcal{M}}(\mathbf{s},\mathbf{s}')}\left[\log\left(D\left(\mathbf{s}, \mathbf{s}'\right)\right)\right] - \mathbb{E}_{d^\pi(\mathbf{s},\mathbf{s}')}\left[\log\left(1 - D\left(\mathbf{s}, \mathbf{s}'\right)\right)\right] \\ & + w_{\text{gp}}\, \mathbb{E}_{d^{\mathcal{M}}(\mathbf{s},\mathbf{s}')}\left[\left\|\nabla_\phi D(\phi)\Big|_{\phi=(\mathbf{s},\mathbf{s}')}\right\|^2\right],\end{aligned} \tag{14}$$

where w_{gp} is a manually specified coefficient.

6.4 Responsive Skills

When reusing pre-trained skills to perform new tasks, the high-level policy $\omega(\mathbf{z}|\mathbf{s}, \mathbf{g})$ specifies latents $\mathbf{z}_t$ at each timestep to control the behavior of the low-level policy $\pi(\mathbf{a}|\mathbf{s}, \mathbf{z})$. A responsive low-level policy should change its behaviors according to changes in $\mathbf{z}$. However, the objective described in Equation 9 can lead to unresponsive behaviors, where π may perform different behaviors depending on the initial $\mathbf{z}_0$ selected at the start of an episode, but if a new latent $\mathbf{z}'$ is selected at a later timestep, then the policy may ignore $\mathbf{z}'$ and continue performing the same behavior specified by $\mathbf{z}_0$. This lack of responsiveness can hamper a character's ability to perform new tasks and agilely respond to unexpected perturbations.

To improve the responsiveness of the low-level policy, we propose two modifications to the objective in Equation 9. First, instead of conditioning π on a fixed $\mathbf{z}$ over an entire episode, we will instead construct a sequence of latents $\mathbf{Z} = \{\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_{T-1}\}$, and condition π on a different latent $\mathbf{z}_t$ at each timestep t. The sequence of latents is constructed such that a latent $\mathbf{z}$ is repeated for multiple timesteps, before a new latent is sampled from $p(\mathbf{z})$ and repeated for multiple subsequent timesteps. This encourages the model to learn to transition between different skills.

To further encourage the model to produce different behaviors for different latents, we incorporate a diversity objective similar to the one proposed by Yang et al. [2019] to mitigate mode-collapse of conditional GANs. These modifications lead to the following pre-training objective,

$$\begin{aligned}\arg\max_\pi \quad & \mathbb{E}_{p(\mathbf{Z})}\mathbb{E}_{p(\tau|\pi,\mathbf{Z})}\left[\sum_{t=0}^{T-1}\gamma^t\Big(-\log\left(1 - D\left(\mathbf{s}_t, \mathbf{s}_{t+1}\right)\right) \right.\right. \\ & \left.\left. + \beta \log q\left(\mathbf{z}_t|\mathbf{s}_t, \mathbf{s}_{t+1}\right)\Big)\right] \\ & - w_{\text{div}}\, \mathbb{E}_{d^\pi(\mathbf{s})}\mathbb{E}_{\mathbf{z}_1,\mathbf{z}_2\sim p(\mathbf{z})}\left[\left(\frac{D_{\text{KL}}\left(\pi(\cdot|\mathbf{s}, \mathbf{z}_1), \pi(\cdot|\mathbf{s}, \mathbf{z}_2)\right)}{D_{\mathbf{z}}\left(\mathbf{z}_1, \mathbf{z}_2\right)} - 1\right)^2\right],\end{aligned} \tag{15}$$

with w_{div} being a manually specified coefficient. The last term is the diversity objective, which stipulates that if two latents $\mathbf{z}_1$ and $\mathbf{z}_2$ are similar under a distance function $D_{\mathbf{z}}$, then the policy should produce similar action distributions, as measured under the KL-divergence. Conversely, if $\mathbf{z}_1$ and $\mathbf{z}_2$ are different, then the resulting action distributions should also be different. In our implementation, the distance function is specified by $D_{\mathbf{z}}(\mathbf{z}_1, \mathbf{z}_2) = 0.5(1 - \mathbf{z}_1^T\mathbf{z}_2)$, which reflects the cosine distance between $\mathbf{z}_1$ and $\mathbf{z}_2$, since the latents lie on a sphere. This diversity objective is reminiscent of the loss used in multidimensional scaling [Kruskal 1964], and we found it to be more stable than the objective proposed by Yang et al. [2019]. Note, the reward for the policy at each time step depends only on the encoder q and discriminator D. The diversity objective is only applied during gradient updates.

6.5 Robust Recovery Strategies

A common failure case for physically simulated characters is losing balance and falling when subjected to perturbations. In these situations, it would be favorable to have characters that can automatically recover and resume performing a task. Therefore, in addition to training the low-level policy to imitate behaviors from a dataset, π is also trained to recover from a large variety of fallen configurations. During pre-training, the character has a 10% probability of being initialized in a random fallen state at the start of each episode. The fallen states are generated by dropping the character from the air at random heights and orientations. This simple strategy then leads to robust recovery strategies that can consistently recover from significant perturbations. By incorporating these recovery strategies into the low-level controller, π can be conveniently reused to allow the character to automatically recover from perturbations when performing downstream tasks, without requiring the character to be explicitly trained to recover from falling for each new task.

ALGORITHM 1: ASE Pre-Training

input $\mathcal{M}$: dataset of reference motions
$D \leftarrow$ initialize discriminator
$q \leftarrow$ initialize encoder
$\pi \leftarrow$ initialize policy
$V \leftarrow$ initialize value function

while not done **do**
  $\mathcal{B} \leftarrow \emptyset$ initialize data buffer
  for trajectory $i = 1, ..., m$ **do**
    $\mathbf{Z} \leftarrow$ sample sequence of latents $\{\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_{T-1}\}$ from $p(\mathbf{z})$
    $\tau^i \leftarrow \{\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, ..., \mathbf{s}_T\}$ collect trajectory with π and $\mathbf{Z}$
    record $\mathbf{Z}$ in τ^i
    for time step $t = 0, ..., T-1$ **do**
      $r_t \leftarrow -\log\left(1 - D(\mathbf{s}_t, \mathbf{s}_{t+1})\right) + \beta \log q\left(\mathbf{z}_t | \mathbf{s}_t, \mathbf{s}_{t+1}\right)$
      record r_t in τ^i
    end for
    store τ^i in $\mathcal{B}$
  end for

  Update encoder:
  for update step = 1, ..., n **do**
    $b^\pi \leftarrow$ sample batch of K transitions $\{(\mathbf{s}_j, \mathbf{s}'_j, \mathbf{z}_j)\}_{j=1}^K$ from $\mathcal{B}$
    update q according to Equation 13 using b^π
  end for

  Update discriminator:
  for update step = 1, ..., n **do**
    $b^\mathcal{M} \leftarrow$ sample batch of K transitions $\{(\mathbf{s}_j, \mathbf{s}'_j)\}_{j=1}^K$ from $\mathcal{M}$
    $b^\pi \leftarrow$ sample batch of K transitions $\{(\mathbf{s}_j, \mathbf{s}'_j)\}_{j=1}^K$ from $\mathcal{B}$
    update D according to Equation 14 using $b^\mathcal{M}$ and b^π
  end for

  update V and π according to Equation 15 using data from $\mathcal{B}$
end while

6.6 Pre-Training

Algorithm 1 provides an overview of the ASE pre-training process for the low-level policy. At the start of each episode, a sequence of latents $\mathbf{Z} = \{\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_{T-1}\}$ is sampled from the prior $p(\mathbf{z})$. A trajectory τ^i is collected by conditioning the policy π on $\mathbf{z}_t$ at each timestep t. The agent receives a reward r_t at each timestep, calculated from the discriminator D and encoder q according to Equation 10. Once a batch of trajectories has been collected, minibatches of transitions $(\mathbf{s}_j, \mathbf{s}'_j, \mathbf{z}_j)$ are sampled from the trajectories and used to update the encoder according to Equation 13. The discriminator is updated according to Equation 14 using minibatches of transitions $(\mathbf{s}_j, \mathbf{s}'_j)$ sampled from the agent's trajectories and the motion dataset $\mathcal{M}$. Finally, the recorded trajectories are used to update the policy. The policy is trained using proximal policy optimization (PPO) [Schulman et al. 2017], with advantages computed using GAE(λ) [Schulman et al. 2015], and the value function is updated using TD(λ) [Sutton and Barto 1998].

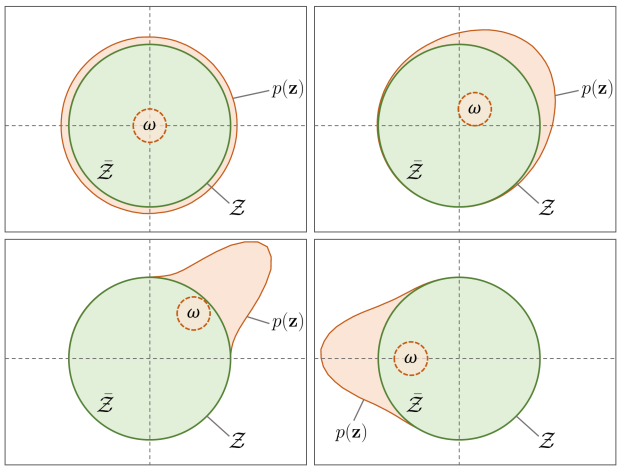


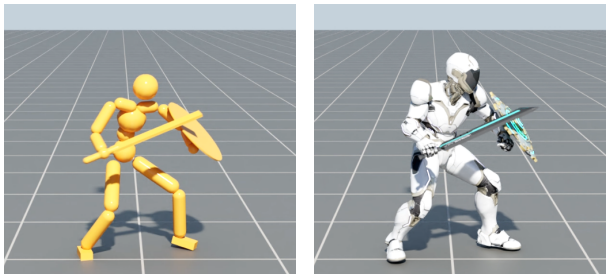
Fig. 3. The *unnormalized* latent space $\bar{\mathcal{Z}}$ is used as the action space for the high-level policy ω. Initializing the action distribution at the origin of $\bar{\mathcal{Z}}$ allows ω to sample skills uniformly from the *normalized* latent space $\mathcal{Z}$. By shifting the action distribution closer or further from the origin of $\bar{\mathcal{Z}}$, ω can increase or decrease the entropy over skills in the normalized space $\mathcal{Z}$.

7 HIGH-LEVEL POLICY

After pre-training, the low-level policy $\pi(\mathbf{a}|\mathbf{s}, \mathbf{z})$ can be applied to downstream tasks by training a task-specific high-level policy $\omega(\mathbf{z}|\mathbf{s}, \mathbf{g})$, which receives as input the state of the character $\mathbf{s}$ and a task-specific goal $\mathbf{g}$, then outputs a latent $\mathbf{z}$ for directing the low-level policy. In this section, we detail design decisions for improving exploration of skills and motion quality on downstream tasks.

7.1 High-Level Action Space

When the character is first presented with a new task, it has no knowledge of which skill will be most effective. Therefore, during early stages of training, the high-level policy should sample skills uniformly from $\mathcal{Z}$ in order to explore a diverse variety of behaviors. As training progresses, the policy should hone in on the skills that are more effective for the task, and assign lower likelihoods to skills that are less effective. Our choice of a spherical latent space provides a convenient structure that can directly encode this exploration-exploitation trade-off into the action space for the high-level policy. This can be accomplished by using the *unnormalized* latents $\bar{\mathbf{z}} \in \bar{\mathcal{Z}}$ as the action space for ω. The high-level policy is then defined as a Guassian in the unnormalized space $\omega(\bar{\mathbf{z}}|\mathbf{s}, \mathbf{g}) = \mathcal{N}\left(\mu_\omega(\mathbf{s}, \mathbf{g}), \Sigma_\omega\right)$. The actions from ω are projected onto $\mathcal{Z}$ by normalizing the actions $\mathbf{z} = \bar{\mathbf{z}}/||\bar{\mathbf{z}}||$, before being passed to the low-level policy. An illustrative example of this sampling scheme is available in Figure 3. At the start of training, the action distribution of ω is initialized close to the origin of $\bar{\mathcal{Z}}$, $\omega(\bar{\mathbf{z}}|\mathbf{s}, \mathbf{g}) \approx \mathcal{N}\left(0, \Sigma_\omega\right)$, which allows ω to sample skill uniformly from the normalized latent space $\mathcal{Z}$. As training progress, the mean $\mu_\omega(\mathbf{s}, \mathbf{g})$ can be shifted away from the origin as needed by the gradient updates, thereby allowing ω to specialize, and increase the likelihood of more effective skills. By shifting the action distribution closer or further from the origin of $\bar{\mathcal{Z}}$, the ω can increase or decrease its entropy over skills in $\mathcal{Z}$ respectively.



(a) Simulation Model (b) Visualization Model

Fig. 4. Our framework is used to learn skill embeddings for a 37 degrees-of-freedom humanoid character, equipped with a sword and shield.

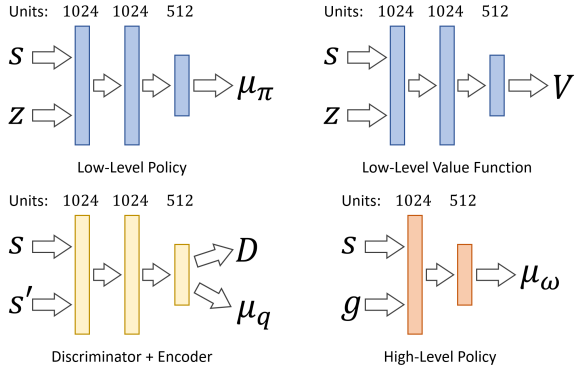


Fig. 5. Network architectures used to model various components of the system. All networks are comprised of fully-connected layers with ReLU activations for hidden layers. The discriminator D and encoder q are modeled by the same network with separate output layers.

7.2 Motion Prior

While π is trained to produce natural behaviors for any latent skill $\mathbf{z}$, when reusing the low-level policy on new tasks, it is still possible for ω to specify *sequences* of latents that lead to unnatural behaviors. This is especially noticeable when the actions from ω changes drastically between timesteps, leading the character to constantly change the skill that it is executing, which can result in unnatural jittery movements. Motion quality on downstream tasks can be improved by reusing the discriminator $D(\mathbf{s}, \mathbf{s}')$ from pre-training as a portable motion prior when training the high-level policy. The reward for the high-level policy is then specified by a combination of a task-reward $r^G(\mathbf{s}, \mathbf{a}, \mathbf{s}', \mathbf{g})$ and a style-reward from the discriminator, in a similar manner as Peng et al. [2021],

$$r_t = w_G\, r^G(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, \mathbf{g}) - w_S \log\left(1 - D(\mathbf{s}_t, \mathbf{s}_{t+1})\right), \tag{16}$$

with w_G and w_S being manually specified coefficients. Note that the parameters of the discriminator are fixed after pre-training, and are not updated during task-training. Therefore, no motion data is needed when training on new tasks. Prior works have observed that training a policy against a fixed discriminator often leads to unnatural behaviors that exploit idiosyncrasies of the discriminator [Peng et al. 2021]. However, we found that the low-level policy π sufficiently constrains the behaviors that can be produced by the character, such that this kind of exploitation is largely eliminated.

8 MODEL REPRESENTATION

To evaluate the effectiveness of our framework, we apply ASE to develop reusable motor skills for a complex 3D simulated humanoid character, with 37 degrees-of-freedom. An illustration of the character is available in Figure 4. The character is similar to the one used by Peng et al. [2021], but our character is additionally equipped with a sword and shield. The sword is attached to its right hand via a 3D spherical joint, and the shield is attached to its left arm with a fixed joint. In this section, we detail modeling decisions for various components of the system.

8.1 States and Actions

The state $\mathbf{s}_t$ consists of a set of features that describes the configuration of the character's body. The features include:

- Height of the root from the ground.
- Rotation of the root in the character's local coordinate frame.
- Linear and angular velocity of the root in the character's local coordinate frame.
- Local rotation of each joint.
- Local velocity of each joint.
- Positions of the hands, feet, shield, and tip of the sword in the character's local coordinate frame.

The root is designated to be character's pelvis. The character's local coordinate frame is defined with the origin located at the root, the x-axis oriented along the root link's facing direction, and the y-axis aligned with the global up vector. The 1D rotation of revolute joints are encoded using a scalar value, representing the rotation angle. The 3D rotation of the root and spherical joints are encoded using two 3D vector corresponding to the tangent $\mathbf{u}$ and normal $\mathbf{v}$ of the link's local coordinate frame expressed in the link parent's coordinate frame [Peng et al. 2021]. Combined, these features result in a 120D state space. Each action $\mathbf{a}_t$ specifies target rotations for PD controllers positioned at each of the character's joints. Like Peng et al. [2021], the target rotation for 3D spherical joints are encoded using a 3D exponential map [Grassia 1998]. These action parameters result in a 31D action space.

8.2 Network Architecture

A schematic illustration of the network architectures used to model the various components of the system is provided in Figure 5. The low-level policy is modeled by a neural network that maps a state $\mathbf{s}$ and latent $\mathbf{z}$ to a Gaussian distribution over actions $\pi(\mathbf{a}|\mathbf{s}, \mathbf{z}) = \mathcal{N}(\mu_\pi(\mathbf{s}, \mathbf{z}), \Sigma_\pi)$, with an input-dependent mean $\mu_\pi(\mathbf{s}, \mathbf{z})$ and a fixed diagonal covariance matrix Σ_π. The mean is specified by a fully-connected network with 3 hidden layers of [1024, 1024, 512] units, followed by linear output units. The value function $V(\mathbf{s}, \mathbf{z})$ is modeled by a similar network, but with a single linear output unit. The encoder $q(\mathbf{z}|\mathbf{s}, \mathbf{s}')$ and discriminator $D(\mathbf{s}, \mathbf{s}')$ are jointly modeled by a single network, with separate output units for the mean of the encoder distribution $\mu_q(\mathbf{s}, \mathbf{s}')$ and the output of the discriminator. The output units of the encoder is normalized such that $||\mu_q(\mathbf{s}, \mathbf{s}')|| = 1$, and the output of the discriminator consists of a single sigmoid unit.

(a) Sword Swing (b) Shield Bash (c) Kick (d) Turn (e) Crouched Walk (f) Jog

Fig. 6. Simulated character performing skills generated by random samples from the latent space. The low-level policy learns to model a diverse array of skills.

The high-level policy $\omega(\bar{\mathbf{z}}|\mathbf{s},\mathbf{g}) = \mathcal{N}\left(\mu_\omega(\mathbf{s},\mathbf{g}), \Sigma_\omega\right)$ is modeled using 2 hidden layers with [1024, 512] units, followed by linear output units for the mean $\mu_\omega(\mathbf{s},\mathbf{g})$. Outputs from the high-level policy specify unnormalized latents $\bar{\mathbf{z}}$, which are then normalized $\mathbf{z} = \bar{\mathbf{z}}/||\bar{\mathbf{z}}||$ before being passed to the low-level policy π. ReLU activations are used for all hidden units [Nair and Hinton 2010].

9 TASKS

Once a low-level policy has been trained to model a large variety of skills, it can then be reused to solve new downstream tasks. The corpus of tasks are designed to evaluate our model's ability to perform a diverse array of skills, compose disparate skills in furtherance of high-level task objectives, and the precision in which the model can control the character's movements. We show that the pre-trained low-level policy enable our characters to produce naturalistic motions using only simple task-reward functions.

Reach: We start with a simple reach task to evaluate the accuracy with which the model can control a character's low-level movements for tasks that require more fine-grain precision. The objective for this task is to position the tip of the sword at a target location $\mathbf{x}^*$. The goal input for the policy $\mathbf{g}_t = \tilde{\mathbf{x}}_t^*$ records the target location $\tilde{\mathbf{x}}_t^*$ in the character's local coordinate frame. The task-reward is calculated according to:

$$r_t^G = \exp\left(-5 \left\|x^* - x_t^{\text{sword}}\right\|^2\right), \tag{17}$$

where x_t^{sword} denotes the position of the sword tip at timestep t. The target is placed randomly within 1m of the character's root. This task represents a form of physics-based data-driven inverse-kinematics.

Speed: To evaluate the model's ability to utilize different locomotion skills, we consider a target speed task, where the objective is for the character to travel along a target direction $\mathbf{d}^*$ at a target speed v^*. The goal is represented by $\mathbf{g}_t = \left(\tilde{\mathbf{d}}_t^*, v^*\right)$, with $\tilde{\mathbf{d}}_t^*$ being the target direction in the character's local coordinate frame. The task-reward is calculated according to:

$$r_t^G = \exp\left(-0.25 \left(v^* - \mathbf{d}^* \cdot \dot{\mathbf{x}}_t^{\text{root}}\right)^2\right), \tag{18}$$

where $\dot{\mathbf{x}}_t^{\text{root}}$ is the velocity of the character's root. The target speed is selected randomly between $v^* \in [0, 7]$m/s.

Steering: In the steering task, the objective is for the character to travel along a target direction, while facing a target heading direction $\mathbf{h}^*$. The goal is given by $\mathbf{g}_t = \left(\tilde{\mathbf{d}}_t^*, \tilde{\mathbf{h}}_t^*\right)$, with $\tilde{\mathbf{h}}_t^*$ being the local heading direction. The task-reward is calculated according to:

$$r_t^G = 0.7 \exp\left(-0.25 \left(v^* - \mathbf{d}^* \cdot \dot{\mathbf{x}}_t^{\text{root}}\right)^2\right) + 0.3\, \mathbf{h}^* \cdot \mathbf{h}_t^{\text{root}}, \tag{19}$$

where $\mathbf{h}_t^{\text{root}}$ is the heading direction of the root, and the target speed is set to $v^* = 1.5$m/s.

Location: In this task, the objective is for the character to move to a target location $\mathbf{x}^*$. The goal $\mathbf{g}_t = \tilde{\mathbf{x}}_t$ records the target location $\tilde{\mathbf{x}}_t$ in the character's local frame. The task-reward is then given by:

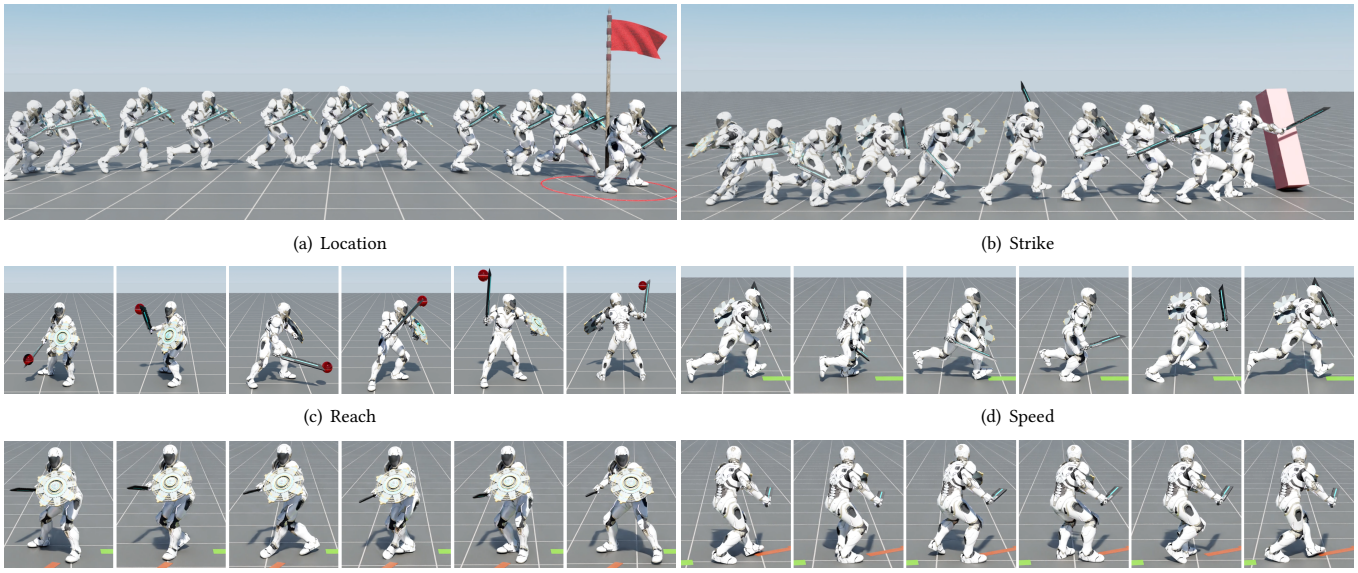
$$r_t^G = \exp\left(-0.5 \left\|x^* - x_t^{\text{root}}\right\|^2\right), \tag{20}$$

with x_t^{root} being the location of the character's root.

Strike: Finally, to evaluate the model's effectiveness in composing disparate skills, we consider a strike task, where the objective is for the character to knock over a target object with its sword. The episode terminates if any body part makes contact with the target, other than the sword. The goal $\mathbf{g}_t = (\tilde{\mathbf{x}}_t^*, \tilde{\dot{\mathbf{x}}}_t^*, \tilde{q}_t^*, \tilde{\dot{q}}_t^*)$ records the position of the target $\tilde{\mathbf{x}}_t^*$, its rotation $\tilde{q}_t^*$, linear velocity $\tilde{\dot{\mathbf{x}}}_t^*$, and angular velocity $\tilde{\dot{q}}_t^*$. All features are recorded in the character's local coordinate frame. The reward is then calculated according to:

$$r_t^G = 1 - \mathbf{u}^{\text{up}} \cdot \mathbf{u}_t^*, \tag{21}$$

where $\mathbf{u}^{\text{up}}$ is the global up vector, and $\mathbf{u}_t^*$ is the local up vector of the target object expressed in the global coordinate frame.

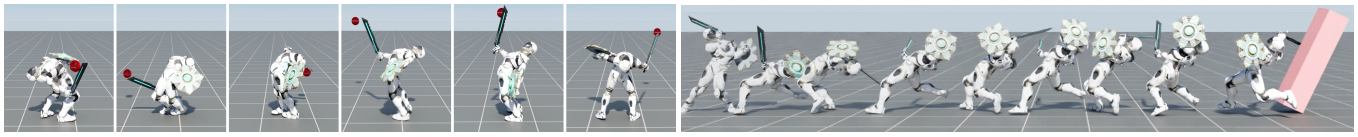


(a) Location (b) Strike

(c) Reach (d) Speed

(e) Steering: Walking Sideways (f) Steering: Walking Backwards

Fig. 7. Simulated character performing tasks using skills from a pre-trained low-level policy. The character can be directed to perform various tasks using simple reward functions, and the low-level policy then enables the character to achieve the task objectives by using naturalistic behaviors.



(a) Reach (b) Strike

Fig. 8. Policies that are trained from scratch for each task, without using the low-level policy, often develop unnatural behaviors.

10 RESULTS

We evaluate the effectiveness of our framework by using ASE to develop skill embeddings that enable a complex simulated humanoid to perform a variety of motion control tasks. First, we show that ASE can learn a rich latent embedding of diverse and sophisticated skills from large unstructured motion datasets containing over a hundred motion clips. Once trained, the learned skill model can then be reused to perform new tasks in a natural and like-like manner. Composition of disparate skills emerge automatically from the model, without requiring intricate reward shaping. Motions produced by our system are best viewed in the supplementary video.

10.1 Experimental Setup

All environments are simulated using Isaac Gym [Makoviychuk et al. 2021], a high-performance GPU-based physics simulator. During training, 4096 environments are simulated in parallel on a single NVIDIA V100 GPU, with a simulation frequency of 120Hz. The low-level policy operates at 30Hz, while the high-level policy operates at 6Hz. All neural networks are implemented using PyTorch [Paszke et al. 2019]. The low-level policy is trained using a custom motion dataset of 187 motion clips, provided by Reallusion [Reallusion 2022]. The dataset contains approximately 30 minutes of motion data that depict a combination of everyday locomotion behaviors, such as walking and running, as well as motion clips that depict a gladiator wielding a sword and shield. The use of a high-performance simulator allows our models to be trained with large volumes of simulated data. The low-level policy is trained with over 10 billion samples, which is approximately 10 years in simulated time, requiring about 10 days on a single GPU. A batch of 131072 samples is collected per update iteration, and gradients are computed using mini-batches of 16384 samples. Gradient updates are performed using the Adam optimizer with a stepsize of 2×10^{-5} [Kingma and Ba 2015]. Detailed hyperparameter settings are available in Appendix A.

10.2 Low-Level Skills

The ASE pre-training process is able to develop expressive low-level policies that can perform a diverse array of complex skills. Examples of the learned skills are shown in Figure 6. Conditioning the policy on random latents $\mathbf{z}$ leads to a large variety of naturalistic and agile behaviors, ranging from common locomotion behaviors, such as walking and running, to highly dynamic behaviors, such as sword swings and shield bashes. All of these skills are modeled by a single low-level policy. During pre-taining, the motion dataset is treated as a homogeneous set of state transitions, without any segmentation or organization of the clips into distinct skills. Despite this lack of prescribed structure, our model learns to organize the

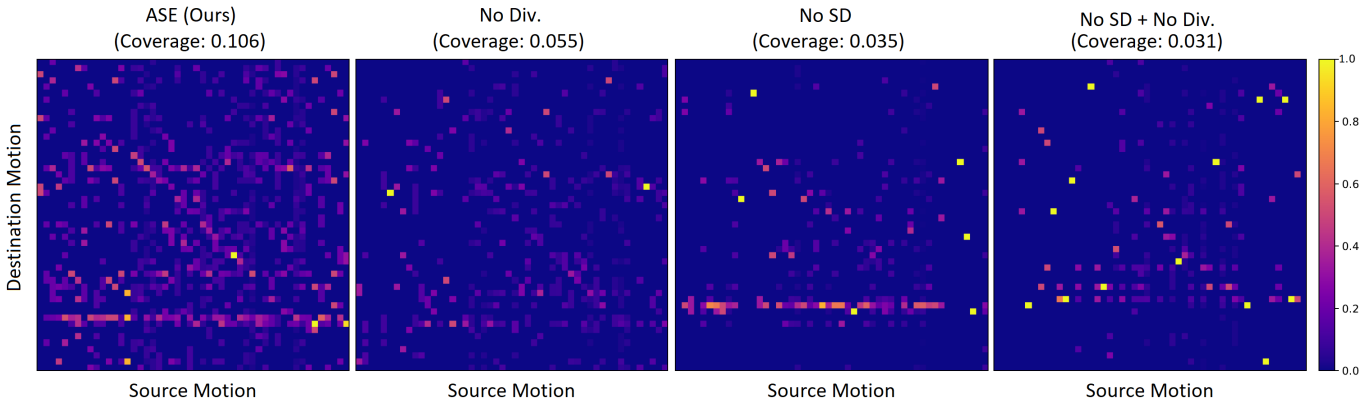


Fig. 9. Probabilities of transitioning between different motion clips under various models. Each column represents the probabilities of transitioning from a source motion to each destination motion. Results are shown for the 50 most frequently matched motion clips. Coverage represents the portion of the total number of possible transitions that was observed from a model. ASE exhibits denser connections between different motions. Denser connections can lead to more responsive behaviors and improve a model's ability to compose different skills when performing new tasks. The number of transitions decrease drastically when the skill discovery objective (No SD) and the diversity objective (No Div.) are removed.

different behaviors into a structured skill embedding, where each latent produces a semantically distinct skill, such as sword swings vs. shield bashes. This structure is likely a result of the skill discovery objective, which encourages the low-level policy to produce distinct behaviors for each latent $\mathbf{z}$, such that the encoder can more easily recover the original $\mathbf{z}$. Changing $\mathbf{z}$ partway through a trajectory also leads the character to transition to different skills. The policy is able to synthesize plausible transitions, even when the particular transitions may not be present in the original dataset.

10.3 Tasks

To demonstrate the effectiveness of the pre-trained skill embeddings, we apply the pre-trained low-level policy to a variety of downstream tasks. Separate high-level policies are trained for each task, while the same low-level policy is used for all tasks. Figure 7 illustrates behaviors learned by the character on each task. ASE learns to model a versatile corpus of skills that enables the character to effectively accomplish diverse task objectives. Though each task is represented by a simple reward function that specifies only a minimal criteria for the particular task, the learned skill embedding automatically gives rise to complex and naturalistic behaviors. In the case of the *Reach* task, the reward simply stipulates that the character should move the tip of its sword to a target location. But the policy then learns to utilize various life-like full-body postures in order to reach target. The skill embedding provides the high-level policy with fine-grain control over the character's low-level movements, allowing the character to closely track the target, with an average error of 0.088 ± 0.046m. For the *Steering* task, the character learns to utilize different forward, backward, and sideways walking behaviors to follow the target directions. When training the character to move at a target speed, the policy is able to transition between various locomotion gaits according to the desired speed. When the target speed is 0m/s, the character learns to stand still. As the target speed increases ($\sim$ 2m/s), the character transitions to a crouched walking gait, and then breaks into a fast upright running gait at the fastest

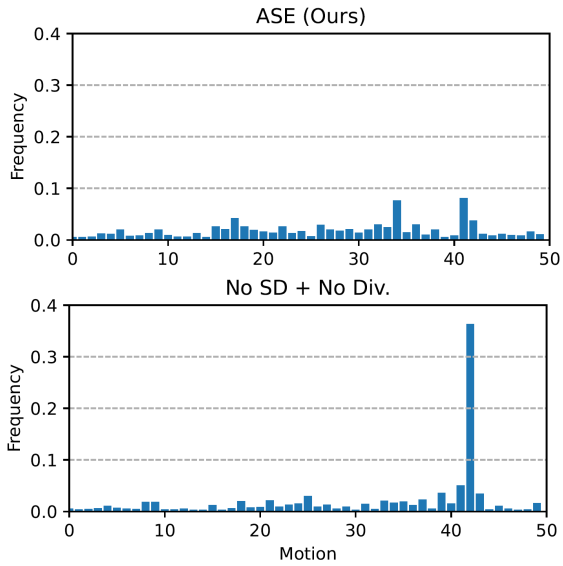


Fig. 10. Frequencies at which the low-level policy produces motions that match individual clips in the dataset. Results are shown for the 50 most frequently matched motion clips. ASE produces diverse behaviors that more evenly covers the dataset. Without the skill discovery objective (No SD) and the diversity objective (No Div.), the policy produces less diverse behaviors and is more prone to collapsing to a single behavior.

speeds ($\sim$ 7m/s). This composition of disparate skills is further evident in the *Strike* task, where the policy learns to utilize running behaviors to quickly move to the target. Once it is close to the target, the policy quickly transitions to a sword swing behavior in order to hit the target. After the target has been successfully knocked over, the character concludes by transitioning to an idle stance.

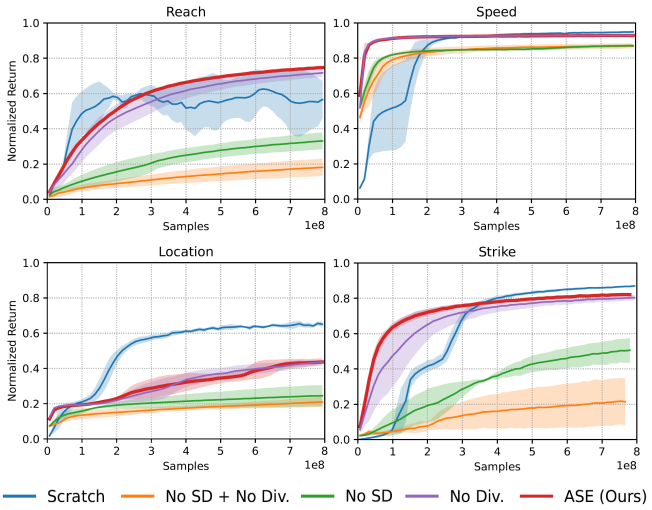


Fig. 11. Learning curves comparing performance on downstream tasks using different low-level policies. We compare ASE to policies that are trained from scratch for each tasks (Scratch), as well as to low-level policies trained without the skill discovery objective (No SD), without the diversity objective (No Div.), and with both objectives disabled (No SD + No Div.). The skill discovery objective is crucial for learning effective skill representations. The policies trained from scratch often achieve higher returns by exploiting unnatural behaviors (see Figure 8)

10.4 Dataset Coverage and Transitions

Learning skill embeddings that can reproduce a wide range of behaviors is crucial for building general and reusable skill models. However, mode-collapse remains a persistent problem for adversarial imitation learning algorithms. To evaluate the diversity of the behaviors produced by our model, we compare motions produced by the low-level policy to motions from the original dataset. First, a trajectory τ is generated by conditioning π on a random latent $\mathbf{z} \sim p(\mathbf{z})$. Then, for every transition $(\mathbf{s}_t, \mathbf{s}_{t+1})$ in the trajectory, we find a motion clip in the dataset $m^* \in \mathcal{M}$ that contains a transition that most closely matches the particular transition from π,

$$m^* = \arg\min_{m^i \in \mathcal{M}} \min_{(\mathbf{s},\mathbf{s}') \in m^i} ||\bar{\mathbf{s}}_t - \bar{\mathbf{s}}||^2 + ||\bar{\mathbf{s}}_{t+1} - \bar{\mathbf{s}}'||^2. \quad (22)$$

Note, $\bar{\mathbf{s}}$ represents the normalized state features of $\mathbf{s}$, normalized using the mean and standard deviation of state features from the motion data. This matching process is repeated for every transition in a trajectory, and the motion clip that contains the most matches will be specified as the clip that best matches the trajectory τ. Figure 10 records the frequencies at which π produces trajectories that matched each motion clip in the dataset across 1000 trajectories. We compare the distribution of trajectories generated by our ASE model to an ablated model that was trained without the skill discovery objective (Section 5.2) and the diversity objective (Equation 15). The ASE model produces a diverse variety of behaviors that more evenly covers the motions in the dataset. The model trained without the skill discovery objective and the diversity object produces less diverse behaviors, where a large portion of the trajectories matched a single motion clip, corresponding to an idle standing motion.

Table 1. Performance of the different skill models when applied to various tasks. Performance is recorded as the normalized return, with 0 being the minimum possible return per episode, and 1 being the maximum. The returns are averaged across 3 models using different pre-trained low-level policies, with 4096 episodes per model. The policies trained from scratch achieve higher returns on most tasks by utilizing unnatural behaviors.

Task	Scratch	No SD + No Div.	No SD	No Div.	ASE (Ours)
Reach	0.56 ± 0.11	0.18 ± 0.05	0.33 ± 0.05	0.72 ± 0.02	**0.75 ± 0.01**
Speed	**0.95 ± 0.01**	0.87 ± 0.01	0.87 ± 0.01	0.93 ± 0.01	0.93 ± 0.01
Steering	**0.94 ± 0.01**	0.72 ± 0.01	0.74 ± 0.02	0.90 ± 0.01	0.90 ± 0.01
Location	**0.67 ± 0.01**	0.22 ± 0.04	0.25 ± 0.06	0.47 ± 0.01	0.45 ± 0.01
Strike	**0.87 ± 0.01**	0.21 ± 0.13	0.50 ± 0.07	0.80 ± 0.02	0.82 ± 0.01

In addition to producing diverse skills, the low-level policy should also learn to transition between the various skills, so that the skills can be more easily composed and sequenced to perform more complex tasks. To evaluate a model's ability to transition between different skills, we generate trajectories by conditioning π on two random latents $\mathbf{z}_S$ and $\mathbf{z}_D$ per trajectory. A trajectory is then generated by first conditioning π on $\mathbf{z}_S$ for 150 to 200 timesteps, then π is conditioned on $\mathbf{z}_D$ for the remainder for the trajectory. We will refer to the two sub-trajectories produced by the different latents as the source trajectory τ_S and the destination trajectory τ_D. The two sub-trajectories are separately matched to motion clips in the dataset, following the same procedure in Equation 22, to identify the source motion m^S and destination m^D. We repeat this process for about 1000 trajectories, and record the probability of transitioning between each pair of motion clips in Figure 9. We compare ASE to models trained without the skill discovery objective (No SD), without the diversity objective (No Div.), and without both objectives (No SD + No Div.). Coverage denotes the portion of transitions out of all possible pairwise transitions observed from trajectories produced by a model $\left(\text{coverage} = \frac{\text{transitions from model}}{\text{all possible transitions}}\right)$. Our ASE model generates much denser connections between different skills, and exhibits about 10% of all possible transitions. Removing the skill discovery objective and diversity objective results in less responsive models that exhibit substantially fewer transitions between skills.

To determine the performance impact of these design decisions, we compare the task performance achieved using various low-level policies trained with and without the skill discovery objective and diversity objective. Figure 11 compares the learning curves of the different models, and Table 1 summarizes the average return of each model. The skill discovery objective is crucial for learning an effective skill representation. Without this objective, the skill embedding tends to produce less diverse and distinct behaviors, which in turn leads to a significant deterioration in task performance. While the diversity objective did lead to denser connections between skills (Figure 9), removing this objective (No Div.) did not seem to have a large impact on performance on our suite of tasks. For most of the tasks we considered, the highest returns are achieved by policies that were trained from scratch for each task. These policies often utilize unnatural behaviors in order to better maximize the reward function, such as adopting highly energetic sporadic movements to propel the character more quickly towards the target in the *Location*

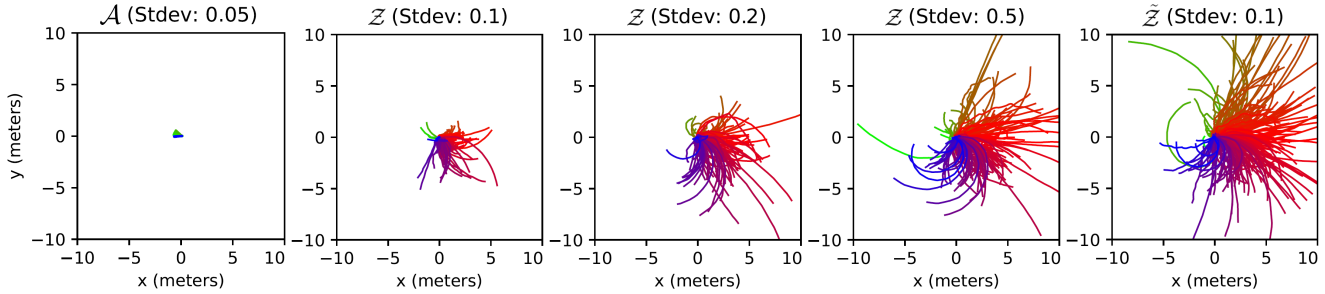


Fig. 12. Trajectories of the character's root produced by random exploration with different action spaces for the high-level policy. Random exploration in the original action space $\mathcal{A}$ does not produce semantically meaningful behaviors, and tends to cause the character to fall after a few timesteps. Our method of using the unnormalized latent space $\tilde{\mathcal{Z}}$ as the action space allows the policy to explore more structured and diverse behaviors. Using the normalized latent space $\mathcal{Z}$ can lead to less diverse behaviors, and a much larger standard deviation is needed for the action distribution to produce similar diversity.

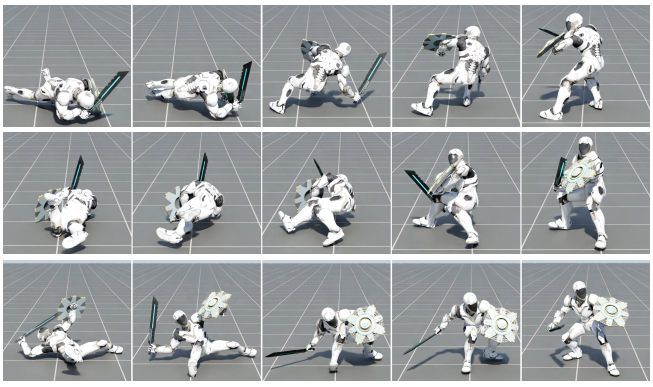


Fig. 13. The low-level policy can consistently recover after falling. The dataset does not contain motion clips that depict get up behaviors, however our model still develops plausible recovery strategies.

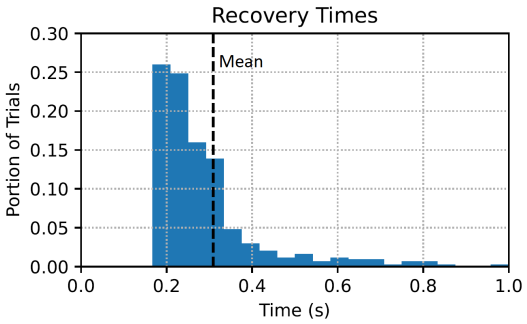


Fig. 14. Time required for the character to get back up after falling when subjected to random perturbation forces. The low-level policy is able to consistently recover after falling across 500 trials, requiring an average recovery time of 0.31s and a maximum of 4.1s.

and *Strike* tasks. Examples of these behaviors are shown in Figure 8. Training from scratch also tends to require more samples compared policies trained using ASE, since the model needs to repeatedly relearn common skills for each task, such as maintaining balance and locomotion. ASE is able to achieve high returns on the suite of tasks, while also producing more natural and life-like behaviors.

10.5 High-Level Action Space

To evaluate the effects of different choices of action space for the high-level policy, we visualize the behaviors produced by random exploration in different action spaces. Figure 12 illustrates trajectories produced by the different action spaces, and the corresponding motions can be viewed in the supplementary video. First, we consider the behaviors produced by random exploration in the original action space of the system $\mathcal{A}$. This strategy does not produce semantically meaning behaviors, and often just leads to the character falling after a few timesteps. Next, we consider behaviors produced by sampling from a Gaussian distribution in the unnormalized latent space $\tilde{\mathcal{Z}}$, with a standard deviation of 0.1, as described in Section 7.1. Positioning the Gaussian at the origin of $\tilde{\mathcal{Z}}$ allows the policy to uniformly sample latents from the normalized latent space $\mathcal{Z}$, leading to a diverse range of behaviors that travel in different directions. Finally, we have trajectories produced by directly sampling in the normalized latent space $\mathcal{Z}$ with various standard deviations $\sigma = [0.1, 0.2, 0.5]$. This sampling strategy limits the policy to selecting latents within a local region on the surface of a hypersphere, which can lead to less diverse behaviors. More diverse behaviors can be obtained by drastically increasing the standard deviation of the action distribution. However, a large action standard deviation can hamper learning, and additional mechanisms are needed to decrease the standard deviation over time.

10.6 Motion Prior

Reusing the pre-trained discriminator as a motion prior when training a high-level policy can improve motion quality. The task-reward function is combined with the style-reward using weights $w_G = 0.9$ and $w_S = 0.1$ respectively. A comparison of the motions produced with and without a motion prior is available in the supplementary video. Without the motion prior, the character is prone to producing unnatural jittering behaviors and other extraneous movements. With the motion prior, the character produces much smoother motions, and exhibits more natural transitions between different skills.

10.7 Robust Recoveries

During the pre-training stage, the low-level policy is trained to recover from random initial states, which leads the policy to develop robust recovery strategies that enable the character to consistent get back up after falling. These recovery strategies can then be conveniently transferred to new tasks by simply reusing the low-level policy. This allows the character to agilely execute recovery behaviors in order to get up after a fall, and then seamless transition back to performing a given task. These behaviors emerge automatically from the low-level policy, without requiring explicit training to recover from perturbations for each new task. Examples of the learned recovery strategies are available in Figure 13. To evaluate the effectiveness of these recovery strategies, we apply random perturbation forces to the character's root and record the time required for the character to recover after a fall. Figure 14 shows the amount of time needed for recoveries across 500 trials. A force between [500, 1000]N in a random direction is applied for 0.5s to the character's root. A fall is detected whenever a body part makes contact with the ground, excluding the feet. A successful recovery is detected once the character's root recovers back to a height above 0.5m and its head is above 1m. The policy is able to successfully recover from all falls, with a mean recovery time of 0.31s and a maximum recovery time of 4.1s. Even though the dataset does not contain any motion clips that depict get up behaviors, the low-level policy is still able to discover plausible recovery strategies, such as using its hands to break a fall and get back up more quickly. However, the character can still exhibit some unnatural recovery behaviors, such as using overly energetic spins and flips. Including motion data for more life-like recovery strategies will likely help to further improve the realism of these motions.

11 DISCUSSION AND FUTURE WORK

In this work, we presented adversarial skill embeddings, a scalable data-driven framework for learning reusable motor skills for physics-based character animation. Our framework enables characters to learn rich and versatile skill embeddings by leveraging large unstructured motion datasets, without requiring any task-specific annotation or segmentation of the motion clips. Once trained, the skill embeddings can be reused to synthesize naturalistic behaviors for a diverse array of downstream tasks. Users can specify tasks through simple high-level reward functions, and the pre-trained low-level policy then allows the character automatically utilize and compose life-like behaviors in furtherance of the task objectives.

Our system demonstrates that large scale adversarial imitation learning can be an effective paradigm for developing general-purpose motor skill models for a wide range of sophisticated behaviors. However, like many GAN-based techniques, our training procedure for the low-level policy is still prone to mode-collapse. While the skill discovery objective can greatly improve the diversity of behaviors learned by the model, the policy still cannot fully capture the rich variety of behaviors depicted in a large motion dataset. Exploring techniques to mitigate mode-collapse could lead to more expressive and versatile skill embeddings that enable characters to tackle more complex tasks, as well as to synthesize more graceful and agile behvaiors. The ASE objective detailed in Section 5 provides a general pre-training objective, where different approximations can be used for each component. We proposed using a GAN-based approximation for the distribution matching objective, and a particular variational approximation for the mutual information objective. Alternative approximations can be used, such as flow models [Dinh et al. 2017], diffusion models [Sohl-Dickstein et al. 2015], and contrastive predictive coding [Oord et al. 2018b], which can present trade-offs that provide a rich design space for future exploration. While our model is able to perform a large array of skills, the downstream tasks that we have explored in our experiments are still relatively simple. Applying ASE to more challenging tasks in complex environments that require composition of a wider array of skills will help to better understand the capabilities of these models. While our framework is able to leverage massively parallel simulators to train skill embeddings with years of simulated data, this process is highly sample intensive compared to the efficiency with which humans can explore and acquire new skills. We would be interested in exploring techniques that can better replicate the skill acquisitions of humans, which may improve the efficiency of the training process and further enhance the capabilities of the acquired skills. Finally, the low-level policy can still occasionally produce unnatural motions, such as jittering, sudden jerks, and overly energetic recovery behaviors. We are interested in exploring methods to mitigate these artifacts and further improve the realism of the generated motions, such as incorporating motion data for natural recovery strategies and integrating energy efficiency objectives into the pre-training stage [Peng et al. 2021; Yu et al. 2018]. Despite these limitations, we hope this work will help pave the way towards developing large-scale and widely reusable data-driven control models for physics-based character animation.

ACKNOWLEDGMENTS

We would like to thank Reallusion[1] for providing reference motion data for this project, the anonymous reviewers for their feedback, Charles Zhou, Dane Johnston, David Dixon, Simon Yuen, Viktor Makoviychuk, and Gavriel State for their support for this work.

REFERENCES

Joshua Achiam and Shankar Sastry. 2017. Surprise-Based Intrinsic Motivation for Deep Reinforcement Learning. *CoRR* abs/1703.01732 (2017). arXiv:1703.01732 http://arxiv.org/abs/1703.01732

M. Al Borno, M. de Lasa, and A. Hertzmann. 2013. Trajectory Optimization for Full-Body Movements with Complex Contacts. *IEEE Transactions on Visualization and Computer Graphics* 19, 8 (2013), 1405–1414. https://doi.org/10.1109/TVCG.2012.325

Kate Baumli, David Warde-Farley, Steven Hansen, and Volodymyr Mnih. 2020. Relative Variational Intrinsic Control. *CoRR* abs/2012.07827 (2020). arXiv:2012.07827 https://arxiv.org/abs/2012.07827

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying Count-Based Exploration and Intrinsic Motivation. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2016/file/afda332245e2af431fb7b672a68b659d-Paper.pdf

Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. 2019. DReCon: Data-Driven Responsive Control of Physics-Based Characters. *ACM Trans. Graph.* 38, 6, Article 206 (Nov. 2019), 11 pages. https://doi.org/10.1145/3355089.3356536

Andrew Brock, Jeff Donahue, and Karen Simonyan. 2019. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *International Conference on Learning Representations*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell,

[1] actorcore.reallusion.com

Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *CoRR* abs/2005.14165 (2020). arXiv:2005.14165 https://arxiv.org/abs/2005.14165

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. 2019. Exploration by random network distillation. In *International Conference on Learning Representations*.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2016/file/7c9d0b1f96aebd7b5eca8c3edaa19ebb-Paper.pdf

Nuttapong Chentanez, Matthias Müller, Miles Macklin, Viktor Makoviychuk, and Stefan Jeschke. 2018. Physics-Based Motion Capture Imitation with Deep Reinforcement Learning. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games* (Limassol, Cyprus) *(MIG '18)*. Association for Computing Machinery, New York, NY, USA, Article 1, 10 pages. https://doi.org/10.1145/3274247.3274506

Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2009. Robust Task-based Control Policies for Physics-based Characters. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 28, 5 (2009), Article 170.

Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2010. Generalized Biped Walking Control. *ACM Transctions on Graphics* 29, 4 (2010), Article 130.

M. Da Silva, Y. Abe, and J. Popovic. 2008. Simulation of Human Motion Data using Short-Horizon Model-Predictive Control. *Computer Graphics Forum* (2008).

Martin de Lasa, Igor Mordatch, and Aaron Hertzmann. 2010. Feature-Based Locomotion Controllers. *ACM Transactions on Graphics* 29, 3 (2010).

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2017. Density estimation using Real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=HkpbnH9lx

Carl Doersch, Abhinav Gupta, and Alexei A. Efros. 2015. Unsupervised Visual Representation Learning by Context Prediction. In *2015 IEEE International Conference on Computer Vision (ICCV)*. 1422–1430. https://doi.org/10.1109/ICCV.2015.167

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In *Proceedings of the 31st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 32)*, Eric P. Xing and Tony Jebara (Eds.). PMLR, Bejing, China, 647–655. https://proceedings.mlr.press/v32/donahue14.html

Gen Endo, Jun Morimoto, Takamitsu Matsubara, Jun Nakanishi, and Gordon Cheng. 2005. Learning CPG Sensory Feedback with Policy Gradient for Biped Locomotion for a Full-Body Humanoid. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3* (Pittsburgh, Pennsylvania) *(AAAI'05)*. AAAI Press, 1267–1273.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. 2019. Diversity is All You Need: Learning Skills without a Reward Function. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SJx63jRqFm

Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. 2001. Composable Controllers for Physics-Based Character Animation. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 251–260. https://doi.org/10.1145/383259.383287

Carlos Florensa, Yan Duan, and Pieter Abbeel. 2017. Stochastic Neural Networks for Hierarchical Reinforcement Learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=B1oK8aoxe

Justin Fu, John Co-Reyes, and Sergey Levine. 2017. EX2: Exploration with Exemplar Models for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/1baff70e2669e8376347efd3a874a341-Paper.pdf

Christian Gehring, Stelian Coros, Marco Hutler, Dario Bellicoso, Huub Heijnen, Remo Diethelm, Michael Bloesch, Péter Fankhauser, Jemin Hwangbo, Mark Hoepflinger, and Roland Siegwart. 2016. Practice Makes Perfect: An Optimization-Based Approach to Controlling Agile Motions for a Quadruped Robot. *IEEE Robotics & Automation Magazine* (02 2016), 1–1. https://doi.org/10.1109/MRA.2015.2505910

Thomas Geijtenbeek, Michiel van de Panne, and A. Frank van der Stappen. 2013. Flexible Muscle-Based Locomotion for Bipedal Creatures. *ACM Transactions on Graphics* 32, 6 (2013).

Hartmut Geyer, Andre Seyfarth, and Reinhard Blickhan. 2003. Positive force feedback in bouncing gaits? *Proc. Royal Society of London B: Biological Sciences* 270, 1529 (2003), 2173–2183.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2672–2680. http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

F. Sebastin Grassia. 1998. Practical Parameterization of Rotations Using the Exponential Map. *J. Graph. Tools* 3, 3 (March 1998), 29–48. https://doi.org/10.1080/10867651.1998.10487493

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. 2017. Variational Intrinsic Control. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net.

Leonard Hasenclever, Fabio Pardo, Raia Hadsell, Nicolas Heess, and Josh Merel. 2020. CoMic: Complementary Task Learning & Mimicry for Reusable Skills. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 4105–4115. https://proceedings.mlr.press/v119/hasenclever20a.html

Karol Hausman, Yevgen Chebotar, Stefan Schaal, Gaurav Sukhatme, and Joseph J Lim. 2017. Multi-Modal Imitation Learning from Unstructured Demonstrations using Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/632cee946db83e7a52ce5e8d6f0fed35-Paper.pdf

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. 2018. Learning an Embedding Space for Transferable Robot Skills. In *International Conference on Learning Representations*.

Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. 2019. Provably Efficient Maximum Entropy Exploration. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 2681–2691. https://proceedings.mlr.press/v97/hazan19a.html

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778. https://doi.org/10.1109/CVPR.2016.90

Nicolas Heess, Gregory Wayne, Yuval Tassa, Timothy P. Lillicrap, Martin A. Riedmiller, and David Silver. 2016. Learning and Transfer of Modulated Locomotor Controllers. *CoRR* abs/1610.05182 (2016). arXiv:1610.05182

Geoffrey Hinton and Ruslan Salakhutdinov. 2006. Reducing the Dimensionality of Data with Neural Networks. *Science* 313, 5786 (2006), 504 – 507.

Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. 1995. Animating human athletics. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1995, Los Angeles, CA, USA, August 6-11, 1995*, Susan G. Mair and Robert Cook (Eds.). ACM, 71–78. https://doi.org/10.1145/218380.218414

Yifeng Jiang, Tom Van Wouwe, Friedl De Groote, and C. Karen Liu. 2019. Synthesis of Biologically Realistic Human Motion Using Joint Torque Actuation. *ACM Trans. Graph.* 38, 4, Article 72 (July 2019), 12 pages. https://doi.org/10.1145/3306346.3322966

L. Jing and Y. Tian. 2021. Self-Supervised Visual Feature Learning With Deep Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 43, 11 (nov 2021), 4037–4058. https://doi.org/10.1109/TPAMI.2020.2992393

Tero Karras, Samuli Laine, and Timo Aila. 2019. A Style-Based Generator Architecture for Generative Adversarial Networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4396–4405. https://doi.org/10.1109/CVPR.2019.00453

Liyiming Ke, Sanjiban Choudhury, Matt Barnes, Wen Sun, Gilwoo Lee, and Siddhartha Srinivasa. 2021. Imitation learning as f-divergence minimization. In *Algorithmic Foundations of Robotics XIV: Proceedings of the Fourteenth Workshop on the Algorithmic Foundations of Robotics 14*. Springer International Publishing, 313–329.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1312.6114

J.B. Kruskal. 1964. Nonmetric multidimensional scaling: A numerical method. *Psychometrika* 29, 2 (1964), 115–129.

Taesoo Kwon and Jessica K. Hodgins. 2017. Momentum-Mapped Inverted Pendulum Models for Controlling Dynamic Human Motions. *ACM Trans. Graph.* 36, 4, Article 145d (Jan. 2017), 14 pages. https://doi.org/10.1145/3072959.2983616

Yoonsang Lee, Sungeun Kim, and Jehee Lee. 2010. Data-Driven Biped Control. *ACM Trans. Graph.* 29, 4, Article 129 (July 2010), 8 pages. https://doi.org/10.1145/1778765.1781155

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. *CoRR* abs/2104.08691 (2021). arXiv:2104.08691 https://arxiv.org/abs/2104.08691

Sergey Levine and Vladlen Koltun. 2013. Guided Policy Search. In *Proceedings of the 30th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 28)*, Sanjoy Dasgupta and David McAllester (Eds.). PMLR, Atlanta, Georgia, USA, 1–9. https://proceedings.mlr.press/v28/levine13.html

Yunzhu Li, Jiaming Song, and Stefano Ermon. 2017. InfoGAIL: Interpretable Imitation Learning from Visual Demonstrations. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/2cd4e8a2ce081c3d7c32c3cde4312ef7-Paper.pdf

Jessica Hodgins Libin Liu. August 2018. Learning Basketball Dribbling Skills Using Trajectory Optimization and Deep Reinforcement Learning. *ACM Transactions on Graphics* 37, 4 (August 2018).

Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel van de Panne. 2020. Character Controllers Using Motion VAEs. *ACM Trans. Graph.* 39, 4 (2020).

Hao Liu and Pieter Abbeel. 2021a. APS: Active Pretraining with Successor Features. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 6736–6747. https://proceedings.mlr.press/v139/liu21b.html

Hao Liu and Pieter Abbeel. 2021b. Behavior From the Void: Unsupervised Active Pre-Training. *CoRR* abs/2103.04551 (2021). arXiv:2103.04551 https://arxiv.org/abs/2103.04551

Libin Liu and Jessica Hodgins. 2017. Learning to Schedule Control Fragments for Physics-Based Characters Using Deep Q-Learning. 36, 4, Article 42a (jun 2017), 14 pages. https://doi.org/10.1145/3072959.3083723

Libin Liu, Michiel van de Panne, and KangKang Yin. 2016. Guided Learning of Control Graphs for Physics-Based Characters. *ACM Transactions on Graphics* 35, 3 (2016).

Libin Liu, KangKang Yin, Michiel van de Panne, and Baining Guo. 2012. Terrain runner: control, parameterization, composition, and planning for highly dynamic motions. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 154.

Libin Liu, KangKang Yin, Michiel van de Panne, Tianjia Shao, and Weiwei Xu. 2010. Sampling-based contact-rich motion control. *ACM Trans. Graph.* 29, 4, Article 128 (July 2010), 10 pages. https://doi.org/10.1145/1778765.1778865

Ying-Sheng Luo, Jonathan Hans Soeseno, Trista Pei-Chun Chen, and Wei-Chao Chen. 2020. CARL: Controllable Agent with Reinforcement Learning for Quadruped Locomotion. *ACM Trans. Graph.* 39, 4, Article 38 (July 2020), 10 pages. https://doi.org/10.1145/3386569.3392433

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. 2020. Learning Latent Plans from Play. In *Proceedings of the Conference on Robot Learning (Proceedings of Machine Learning Research, Vol. 100)*, Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (Eds.). PMLR, 1113–1132. https://proceedings.mlr.press/v100/lynch20a.html

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. 2021. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning. *CoRR* abs/2108.10470 (2021). arXiv:2108.10470 https://arxiv.org/abs/2108.10470

Josh Merel, Leonard Hasenclever, Alexandre Galashov, Arun Ahuja, Vu Pham, Greg Wayne, Yee Whye Teh, and Nicolas Heess. 2019. Neural Probabilistic Motor Primitives for Humanoid Control. In *International Conference on Learning Representations*.

Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. 2020. Catch & Carry: Reusable Neural Controllers for Vision-Guided Whole-Body Tasks. *ACM Trans. Graph.* 39, 4, Article 39 (jul 2020), 14 pages. https://doi.org/10.1145/3386569.3392474

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

Igor Mordatch, Martin de Lasa, and Aaron Hertzmann. 2010. Robust Physics-Based Locomotion Using Low-Dimensional Planning. In *ACM SIGGRAPH 2010 Papers* (Los Angeles, California) *(SIGGRAPH '10)*. Association for Computing Machinery, New York, NY, USA, Article 71, 8 pages. https://doi.org/10.1145/1833349.1778808

Igor Mordatch, Emanuel Todorov, and Zoran Popović. 2012. Discovery of Complex Behaviors through Contact-Invariant Optimization. *ACM Trans. Graph.* 31, 4, Article 43 (July 2012), 8 pages. https://doi.org/10.1145/2185520.2185539

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning* (Haifa, Israel) *(ICML'10)*. Omnipress, Madison, WI, USA, 807–814.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. 2016. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2016/file/cedebb6e872f539bef8c3f919874e9d7-Paper.pdf

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018a. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018b. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. 2019. Learning Predict-and-Simulate Policies from Unorganized Human Motion Data. *ACM Trans. Graph.* 38, 6, Article 205 (Nov. 2019), 11 pages. https://doi.org/10.1145/3355089.3356501

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. 2017. Curiosity-driven Exploration by Self-supervised Prediction. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 2778–2787. https://proceedings.mlr.press/v70/pathak17a.html

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. 2019. Self-Supervised Exploration via Disagreement. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 5062–5071. https://proceedings.mlr.press/v97/pathak19a.html

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. DeepMimic: Example-guided Deep Reinforcement Learning of Physics-based Character Skills. *ACM Trans. Graph.* 37, 4, Article 143 (July 2018), 14 pages. https://doi.org/10.1145/3197517.3201311

Xue Bin Peng, Glen Berseth, Kangkang Yin, and Michiel Van De Panne. 2017. DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning. *ACM Trans. Graph.* 36, 4, Article 41 (July 2017), 13 pages. https://doi.org/10.1145/3072959.3073602

Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. 2019. MCP: Learning Composable Hierarchical Control with Multiplicative Compositional Policies. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 3681–3692. http://papers.nips.cc/paper/8626-mcp-learning-composable-hierarchical-control-with-multiplicative-compositional-policies.pdf

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. 2021. AMP: Adversarial Motion Priors for Stylized Physics-Based Character Control. *ACM Trans. Graph.* 40, 4, Article 1 (July 2021), 15 pages. https://doi.org/10.1145/3450626.3459670

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21, 140 (2020), 1–67. http://jmlr.org/papers/v21/20-074.html

Marc H. Raibert and Jessica K. Hodgins. 1991. Animation of Dynamic Legged Locomotion. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '91)*. Association for Computing Machinery, New York, NY, USA, 349–358. https://doi.org/10.1145/122718.122755

Reallusion. 2022. 3D Animation and 2D Cartoons Made Simple. http://www.reallusion.com.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2015. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *CoRR* abs/1506.02438 (2015). arXiv:1506.02438

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017). arXiv:1707.06347 http://arxiv.org/abs/1707.06347

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. 2018. Time-Contrastive Networks: Self-Supervised Learning from Video. *Proceedings of International Conference in Robotics and Automation (ICRA)* (2018). http://arxiv.org/abs/1704.06888

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. 2020. Dynamics-Aware Unsupervised Discovery of Skills. In *International Conference on Learning Representations*.

Dana Sharon and Michiel van de Panne. 2005. Synthesis of Controllers for Stylized Planar Bipedal Walking. In *Proc. of IEEE International Conference on Robotics and Animation*.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 37)*, Francis Bach and David Blei (Eds.). PMLR, Lille, France,

2256–2265. https://proceedings.mlr.press/v37/sohl-dickstein15.html

Kwang Won Sok, Manmyung Kim, and Jehee Lee. 2007. Simulating Biped Behaviors from Human Motion Data. *ACM Trans. Graph.* 26, 3 (July 2007), 107–es. https://doi.org/10.1145/1276377.1276511

Seungmoon Song, Łukasz Kidziński, Xue Bin Peng, Carmichael Ong, Jennifer Hicks, Sergey Levine, Christopher G. Atkeson, and Scott L. Delp. 2020. Deep reinforcement learning for modeling human locomotion control in neuromechanical simulation. *bioRxiv* (2020). https://doi.org/10.1101/2020.08.11.246801 arXiv:https://www.biorxiv.org/content/early/2020/08/12/2020.08.11.246801.full.pdf

Bradly C. Stadie, Sergey Levine, and Pieter Abbeel. 2015. Incentivizing Exploration In Reinforcement Learning With Deep Predictive Models. *CoRR* abs/1507.00814 (2015). arXiv:1507.00814 http://arxiv.org/abs/1507.00814

Alexander L. Strehl and Michael L. Littman. 2008. An Analysis of Model-Based Interval Estimation for Markov Decision Processes. *J. Comput. Syst. Sci.* 74, 8 (dec 2008), 1309–1331. https://doi.org/10.1016/j.jcss.2007.08.009

Richard S. Sutton and Andrew G. Barto. 1998. *Introduction to Reinforcement Learning* (1st ed.). MIT Press, Cambridge, MA, USA.

Jie Tan, Yuting Gu, C. Karen Liu, and Greg Turk. 2014. Learning Bicycle Stunts. *ACM Trans. Graph.* 33, 4, Article 50 (July 2014), 12 pages. https://doi.org/10.1145/2601097.2601121

Jie Tan, Yuting Gu, Greg Turk, and C. Karen Liu. 2011. Articulated Swimming Creatures. *ACM Trans. Graph.* 30, 4, Article 58 (jul 2011), 12 pages. https://doi.org/10.1145/2010324.1964953

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. 2017. #Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3a20f62a0af1aa152670bab3c602feed-Paper.pdf

Yuval Tassa, Tom Erez, and Emanuel Todorov. 2012. Synthesis and stabilization of complex behaviors through online trajectory optimization.. In *IROS*. IEEE, 4906–4913. http://dblp.uni-trier.de/db/conf/iros/iros2012.html#TassaET12

Michiel van de Panne, Ryan Kim, and Eugene Flume. 1994. Virtual Wind-up Toys for Animation. In *Proceedings of Graphics Interface '94*. 208–215.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2014. Show and Tell: A Neural Image Caption Generator. http://arxiv.org/abs/1411.4555 cite arxiv:1411.4555.

Kevin Wampler, Zoran Popović, and Jovan Popović. 2014. Generalizing Locomotion Style to New Animals with Inverse Optimal Regression. *ACM Trans. Graph.* 33, 4, Article 49 (July 2014), 11 pages.

Jack M. Wang, David J. Fleet, and Aaron Hertzmann. 2009. Optimizing Walking Controllers. In *ACM SIGGRAPH Asia 2009 Papers* (Yokohama, Japan) *(SIGGRAPH Asia '09)*. Association for Computing Machinery, New York, NY, USA, Article 168, 8 pages. https://doi.org/10.1145/1661412.1618514

Jack M. Wang, Samuel R. Hamner, Scott L. Delp, and Vladlen Koltun. 2012. Optimizing Locomotion Controllers Using Biologically-Based Actuators and Objectives. *ACM Trans. Graph.* 31, 4, Article 25 (July 2012), 11 pages. https://doi.org/10.1145/2185520.2185521

Tingwu Wang, Yunrong Guo, Maria Shugrina, and Sanja Fidler. 2020. UniCon: Universal Neural Controller For Physics-based Character Motion. arXiv:2011.15119 [cs.GR]

Ziyu Wang, Josh Merel, Scott Reed, Greg Wayne, Nando de Freitas, and Nicolas Heess. 2017. Robust Imitation of Diverse Behaviors. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 5326–5335.

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2020. A Scalable Approach to Control Diverse Behaviors for Physically Simulated Characters. *ACM Trans. Graph.* 39, 4, Article 33 (jul 2020), 12 pages. https://doi.org/10.1145/3386569.3392381

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2021. Control Strategies for Physically Simulated Characters Performing Two-Player Competitive Sports. *ACM Trans. Graph.* 40, 4, Article 146 (jul 2021), 11 pages. https://doi.org/10.1145/3450626.3459761

Pei Xu and Ioannis Karamouzas. 2021. A GAN-Like Approach for Physics-Based Imitation Learning and Interactive Character Control. 4, 3, Article 44 (sep 2021), 22 pages. https://doi.org/10.1145/3480148

Dingdong Yang, Seunghoon Hong, Yunseok Jang, Tiangchen Zhao, and Honglak Lee. 2019. Diversity-Sensitive Conditional Generative Adversarial Networks. In *International Conference on Learning Representations*.

Yuting Ye and C. Karen Liu. 2010. Optimal Feedback Control for Character Animation Using an Abstract Model. In *ACM SIGGRAPH 2010 Papers* (Los Angeles, California) *(SIGGRAPH '10)*. Association for Computing Machinery, New York, NY, USA, Article 74, 9 pages. https://doi.org/10.1145/1833349.1778811

KangKang Yin, Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2008. Continuation Methods for Adapting Simulated Skills. *ACM Trans. Graph.* 27, 3 (2008).

KangKang Yin, Kevin Loken, and Michiel van de Panne. 2007. SIMBICON: Simple Biped Locomotion Control. *ACM Trans. Graph.* 26, 3 (2007), Article 105.

Wenhao Yu, Greg Turk, and C. Karen Liu. 2018. Learning Symmetric and Low-Energy Locomotion. *ACM Trans. Graph.* 37, 4, Article 144 (July 2018), 12 pages. https://doi.org/10.1145/3197517.3201397

Victor Brian Zordan and Jessica K. Hodgins. 2002. Motion Capture-Driven Simulations That Hit and React. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Antonio, Texas) *(SCA '02)*. Association for Computing Machinery, New York, NY, USA, 89–96. https://doi.org/10.1145/545261.545276

A HYPERPARAMETERS

Hyperparameter settings used during pre-training of the low-level policy are available in Table 2, and the hyperparameters for task-training of the high-level policy are available in Table 3. During pre-training, the trajectories are generated by conditioning the low-level policy π on a random sequence of latents $\mathbf{Z} = \{\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_{T-1}\}$ sampled according to $p(\mathbf{z})$. The sequence of latents is constructed such that a latent $\mathbf{z}$ is repeated for multiple timesteps, before a new latent is sampled from $p(\mathbf{z})$ and repeated for multiple subsequent timesteps. Each latent is kept fixed for between 1 and 150 timesteps before being changed. During task-training, the high-level policy is queried at 6Hz, while the low-level policy operates at 30Hz. The latents specified by the high-level policy is therefore repeated for 5 steps for the low-level policy. This can help improve motion quality by lowering the rates at which the high-level policy can change the skill being executed by the low-level policy, reducing the prevalence of unnatural jittery behaviors.

Table 2. ASE hyperparameters for training low-level policy.

Parameter	**Value**
$\dim(\mathcal{Z})$ Latent Space Dimension	64
Σ_π Action Distribution Variance	0.0025
β Skill Discovery Objective Weight	0.5
w_{gp} Gradient Penalty Weight	5
w_{div} Diversity Objective Weight	0.01
κ Encoder Scaling Factor	1
Samples Per Update Iteration	131072
Policy/Value Function Minibatch Size	16384
Discriminator/Encoder Minibatch Size	4096
γ Discount	0.99
Adam Stepsize	2×10^{-5}
GAE(λ)	0.95
TD(λ)	0.95
PPO Clip Threshold	0.2
T Episode Length	300

Table 3. ASE hyperparameters for training high-level policy.

Parameter	**Value**
w_G Task-Reward Weight	0.9
w_S Style-Reward Weight	0.1
Σ_π Action Distribution Variance	0.01
Samples Per Update Iteration	131072
Policy/Value Function Minibatch Size	16384
γ Discount	0.99
Adam Stepsize	2×10^{-5}
GAE(λ)	0.95
TD(λ)	0.95
PPO Clip Threshold	0.2
T Episode Length	300