Aalto University

Ghadirzadeh, Ali; Poklukar, Petra; Arndt, Karol; Finn, Chelsea; Kyrki, Ville; Kragic, Danica; Björkman, Mårten

Training and Evaluation of Deep Policies Using Reinforcement Learning and Generative Models

Training and Evaluation of Deep Policies Using Reinforcement Learning and Generative Models

Ali Ghadirzadeh[*]   GHADIRI@CS.STANFORD.EDU
Stanford University, CA 94305, USA

Petra Poklukar[*]   POKLUKAR@KTH.SE
KTH Royal Institute of Technology, Stockholm, Sweden

Karol Arndt   KAROL.ARNDT@AALTO.FI
Aalto University, Espoo, Finland

Chelsea Finn   CBFINN@CS.STANFORD.EDU
Stanford University, CA 94305, USA

Ville Kyrki   VILLE.KYRKI@AALTO.FI
Aalto University, Espoo, Finland

Danica Kragic   DANI@KTH.SE
KTH Royal Institute of Technology, Stockholm, Sweden

Mårten Björkman   CELLE@KTH.SE
KTH Royal Institute of Technology, Stockholm, Sweden

Abstract

We present a data-efficient framework for solving sequential decision-making problems which exploits the combination of reinforcement learning (RL) and latent variable generative models. The framework, called GenRL, trains deep policies by introducing an action latent variable such that the feed-forward policy search can be divided into two parts: (i) training a sub-policy that outputs a distribution over the action latent variable given a state of the system, and (ii) unsupervised training of a generative model that outputs a sequence of motor actions conditioned on the latent action variable. GenRL enables safe exploration and alleviates the data-inefficiency problem as it exploits prior knowledge about valid sequences of motor actions. Moreover, we provide a set of measures for evaluation of generative models such that we are able to predict the performance of the RL policy training prior to the actual training on a physical robot. We experimentally determine the characteristics of generative models that have most influence on the performance of the final policy training on two robotics tasks: shooting a hockey puck and throwing a basketball. Furthermore, we empirically demonstrate that GenRL is the only method which can safely and efficiently solve the robotics tasks compared to two state-of-the-art RL methods.

Keywords: reinforcement learning, policy search, robot learning, deep generative models, representation learning

---

[*]. denotes equal contribution

---

1. Introduction

Reinforcement learning (RL) can leverage modeling capability of generative models to solve complex sequential decision making problems more efficiently (Singh et al., 2020; Ghadirzadeh et al., 2017; Arndt et al., 2020). RL has been applied to end-to-end training of deep visuomotor robotic policies (Levine et al., 2016, 2018) but it is typically too data-inefficient especially when applied to tasks that provide only a terminal reward at the end of an episode. One way to alleviate the data-inefficiency problem in RL is by leveraging prior knowledge to reduce the complexity of the optimization problem. One prior that significantly reduces the data requirement is an approximation of the distribution from which valid action sequences can be sampled. Such distributions can be efficiently approximated by training generative models given a sufficient amount of valid action sequences.

The question is then how to combine powerful RL optimization algorithms with the modeling capability of generative models to improve the efficiency of the policy training? Moreover, which characteristics of the generative models are important for efficient policy training? A suitable generative model must capture the entire distribution of the training data to generate as many distinct motion trajectories as possible, while avoiding the generation of invalid trajectories outside the training dataset. The diversity of the generated data enables the policy to complete a given task for the entire set of goal states when training a goal-conditioned policy. On the other hand, adhering to the distribution of the training data ensures safety in generated trajectories which are running on a real robotic platform.

In this paper, we (i) propose a learning framework that exploits RL and generative models to solve sequential decision making problems and considerably improves the data-efficiency of deep policy training, and (ii) provide a set of measures to evaluate the quality of the latent space of different generative models regulated by the RL policy search algorithms, and use them as a guideline for the training of generative models such that the data-efficiency of the policy training can be further improved prior to actual training on a physical robot.

Regarding (i), we propose the *GenRL* learning framework that divides the sequential decision-making problem into the following sub-problems which can be solved more efficiently: (1) an unsupervised generative model training problem that approximates the distribution of motor actions, and (2) a trust-region policy optimization problem that solves a contextual multi-armed bandit without temporal credit assignment issue which exists in typical sequential decision-making problems.

Regarding (ii), we evaluate generative models based on (a) the quality and coverage of the samples they generate using the precision and recall metric (Kynkäänniemi et al., 2019), and (b) the quality of their latent representations by measuring their disentanglement using a novel measure called *disentanglement with precision and recall (DwPR)* which is derived from the above precision and recall metric. The measures can be applied to any task prior to the policy training as they do not leverage the end states obtained after execution of the generated trajectories on a robotic platform. Our hypothesis is that a generative model which is able to generate realistic samples that closely follow the training data (i.e. has high precision and high recall) and is well disentangled (i.e. has a latent space where individual dimensions control different aspects of the task) leads to a more sample efficient neural network policy training.

We experimentally evaluate the GenRL framework on two robotics tasks: shooting a hockey puck and throwing a basketball. To quantify the importance of (a) and (b) for a superior RL policy training performance and validate our hypothesis, we perform a detailed evaluation of several generative models, namely β-VAEs (Higgins et al., 2017) and InfoGANs (Chen et al., 2016), by calculating Pearson's R between their characteristics and the performance of the policy. Furthermore, we demonstrate the safety and data-efficiency of GenRL on the two tasks by comparing it to two state-of-the-art policy search algorithms, PPO (Schulman et al., 2017) and SAC (Haarnoja et al., 2018b). In summary, the advantages of the proposed GenRL framework are:

- It enables learning complex robotics skills such as throwing a ball by incorporating prior knowledge in terms of a distribution over valid sequences of actions, therefore, reducing the search space (Section 6.4).

- It helps to learn from sparse terminal rewards that are only provided at the end of successful episodes. The proposed formulation converts the sequential decision-making problem into a contextual multi-armed bandit (Section 3). Therefore, it alleviates the temporal credit assignment problem that is inherent in sequential decision-making tasks and enables efficient policy training with only terminal rewards.

- It enables safe exploration in RL by sampling actions only from the approximated distribution (Section 6.5). This is in stark contrast to the typical RL algorithms in which random actions are taken during the exploration phase.

- It provides a set of measures for evaluation of the generative model based on which it is possible to predict the performance of the RL policy training prior to the actual training (Section 6.6).

This paper provides a comprehensive overview of our earlier work for RL policy training based on generative models (Ghadirzadeh et al., 2017; Arndt et al., 2020; Chen et al., 2020; Hämäläinen et al., 2019; Bütepage et al., 2020) and is organized as follows: in Section 2, we provide an overview of the related work. We formally introduce the problem of policy training with generative models in Section 3, and describe how the GenRL framework is trained in Section 4. In Section 5 we first briefly overview β-VAEs and InfoGANs, and then define the evaluation measures used to predict the final policy training performance. We present the experimental results in Section 6 and discuss the conclusion and future work in Section 7. Moreover, for the sake of completeness, we describe the end-to-end training of the perception and control modules in Appendix A by giving a summary of the prior work (Levine et al., 2016; Chen et al., 2020). Note that this work provides a complete overview of the proposed GenRL framework and focuses more on the evaluation of the generative model. We refer the reader to our prior work (Ghadirzadeh et al., 2017) for further investigation of the data-efficiency of the proposed approach in training complex visuomotor skills.

2. Related Work

Our work addresses two types of problems: (a) policy training based on unsupervised generative model training and trust-region policy optimization, and (b) evaluation of generative

models to forecast the efficiency of the final policy training task. We introduce the related work for each of the problems in the following sections.

2.1 Data-efficient End-to-end Policy Training

In recent years, end-to-end training of policies using deep RL has gained in popularity in robotics research (Ghadirzadeh et al., 2017; Levine et al., 2016; Finn et al., 2016; Kalashnikov et al., 2018; Quillen et al., 2018; Singh et al., 2017; Devin et al., 2018; Pinto et al., 2017). However, deep RL algorithms are typically data-hungry and learning a general policy, i.e., a policy that performs well also for previously unseen inputs, requires a farm of robots continuously collecting data for several days (Levine et al., 2018; Finn and Levine, 2017; Gu et al., 2017; Dasari et al., 2019). The limitation of large-scale data collection has hindered the applicability of RL solutions to many practical robotics tasks. Recent studies tried to improve the data-efficiency by training the policy in simulation and transferring the acquired skills to the real setup (Quillen et al., 2018; Pinto et al., 2017; Abdolmaleki et al., 2020; Peng et al., 2018), a paradigm known as sim-to-real transfer learning. Sim-to-real approaches are utilized for two tasks in deep policy training: (i) training the perception model via randomization of the texture and shape of visual objects in simulation and using the trained model directly in the real world setup (zero-shot transfer) (Hämäläinen et al., 2019; Tobin et al., 2017), and (ii) training the policy in simulation by randomizing the dynamics of the task and transferring the policy to the real setup by fine-tuning it with the real data (few-shot transfer learning) (Arndt et al., 2020; Peng et al., 2018). However, challenges in the design of the simulation environment can cause large differences between the real and the simulated environments which hinder an efficient knowledge transfer between these two domains. In such cases, transfer learning from other domains, e.g., human demonstrations (Bütepage et al., 2020; Yu et al., 2018) or simpler task setups (Chen et al., 2020, 2018b), can help the agent to learn a policy more efficiently. In this work, we exploit human demonstrations to shape the robot motion trajectories by training generative models that reproduce the demonstrated trajectories. Following our earlier work (Chen et al., 2020), we exploit adversarial domain adaptation techniques (Tzeng et al., 2017, 2020) to improve the generality of the acquired policy when it is trained in a simple task environment with a small amount of training data. In the rest of this section, we review related studies that improve the data-efficiency and generality of RL algorithms by utilizing trust-region terms, converting the RL problem into a supervised learning problem, and trajectory-centered approaches that shape motion trajectories prior to the policy training.

Improving the policy while avoiding abrupt changes in the policy distribution after each update is known as the trust-region approach in policy optimization. Trust-region policy optimization (TRPO) (Schulman et al., 2015) and proximal policy optimization (PPO) (Schulman et al., 2017) are two variants of trust-region policy gradient methods that scale well to non-linear policies such as neural networks. The key component of TRPO and PPO is a surrogate objective function with a trust-region term based on which the policy can be updated and monotonically improved. In TRPO, the changes in the distributions of the policies before and after each update are penalized by a KL divergence term. Therefore, the policy is forced to stay in a trust-region given by the action distribution of the current policy. Our expectation-maximization (EM) formulation yields a similar trust-region term with the

difference being that it penalizes the changes in the distribution of the deep policy and a so-called variational policy that will be introduced as a part of our proposed optimization algorithm. Since our formulation allows the use of any policy gradient solution, we use the same RL objective function as in TRPO.

The EM algorithm has been used for policy training in a number of prior work (Neumann, 2011; Deisenroth et al., 2013; Levine and Koltun, 2013). The key idea is to introduce variational policies to decompose the policy training into two downstream tasks that are trained iteratively until no further policy improvement can be observed (Ghadirzadeh, 2018). The guided policy search (GPS) algorithm (Levine et al., 2016) divides the visuomotor policy training task into a trajectory optimization and a supervised learning problem. GPS alternates between two steps: (i) optimizing a set of trajectories by exploiting a trust-region term to stay close to the action distribution given by the deep policy, and (ii) updating the deep policy to reproduce the motion trajectories. Our EM solution differs from the GPS framework and earlier approaches in that we optimize the trajectories by regulating a generative model that is trained prior to the policy training. Training generative models enables the learning framework to exploit human expert knowledge as well as to optimize the policy given only terminal rewards as explained earlier.

Trajectory-centric approaches, such as dynamic movement primitives (DMPs), have been popular because of the ease of integrating expert knowledge in the policy training process via physical demonstration (Bahl et al., 2020; Peters and Schaal, 2006, 2008; Ijspeert et al., 2003, 2013; Hazara and Kyrki, 2019). However, such models are less expressive compared to deep neural networks. Moreover, these approaches cannot be used to train reactive policies where the action is adjusted in every time-step based on the observed sensory input (Haarnoja et al., 2018a). On the other hand, deep generative models can model complex dependencies within the data by learning the underlying data distribution from which realistic samples can be obtained. Furthermore, they can be easily accommodated in larger neural networks without affecting the data integrity. Our GenRL framework based on generative models enables training both feedback (reactive) and feedforward policies by adjusting the policy network architecture.

The use of generative models in robot learning has become popular in recent years (Zhou et al., 2020; Chen et al., 2022; Ghadirzadeh et al., 2017; Bütepage et al., 2020; Hämäläinen et al., 2019; Chen et al., 2020; Arndt et al., 2020; Lippi et al., 2020; Gothoskar et al., 2020; Igl et al., 2018; Buesing et al., 2018; Mishra et al., 2017; Ke et al., 2018; Hafner et al., 2018; Rhinehart et al., 2018; Krupnik et al., 2019) because of their low-dimensional and regularized latent spaces. However, latent variable generative models are mainly studied to train a long-term state prediction model that is used in the context of trajectory optimization and model-based reinforcement learning (Buesing et al., 2018; Mishra et al., 2017; Ke et al., 2018; Hafner et al., 2018; Rhinehart et al., 2018; Krupnik et al., 2019). Regulating generative models based on reinforcement learning to produce sequences of actions according to the visual state has first appeared in our prior work (Ghadirzadeh et al., 2017). Since then we applied the framework in different robotic tasks, e.g., throwing balls (Ghadirzadeh et al., 2017), shooting hockey-pucks (Arndt et al., 2020), pouring into mugs (Chen et al., 2020; Hämäläinen et al., 2019), and in a variety of problem domains, e.g., sim-to-real transfer learning (Hämäläinen et al., 2019; Arndt et al., 2020) and domain adaptation to acquire general policies (Chen et al., 2020). Recently, generative

models have also been used to train feedback RL policies based on normalizing flows (Dinh et al., 2016) to learn behavioral prior that transforms the original MDP into a simpler one for the RL policy training (Singh et al., 2020). In this work, we focus on RL tasks that can be best implemented using feed-forward policies, and study the properties of generative models that improve the efficiency of the RL policy training.

2.2 Evaluation of Generative Models

Although generative models have proved successful in many domains (Lippi et al., 2020; Brock et al., 2018; Wang et al., 2018; Wiewel and Yang, 2019; Xu and Tan, 2020) assessing their quality remains a challenging problem (Locatello et al., 2019; Poklukar et al., 2022, 2021). It involves analysing the quality of both latent representations and generated samples. Regarding the latter, generated samples and their diversity should resemble those obtained from the training data distribution. Early developed metrics such as IS (Salimans et al., 2016), FID (Heusel et al., 2017) and KID (Bińkowski et al., 2018) provided a promising start but were shown to be unable to separate between failure cases, such as mode collapse or unrealistic generated samples (Sajjadi et al., 2018; Kynkäänniemi et al., 2019). Instead of using a one-dimensional score, Sajjadi et al. (2018) proposed to evaluate the learned distribution by comparing the samples from it with the ground truth training samples using the notion of precision and recall. Intuitively, precision measures the similarity between the generated and real samples, while recall determines the fraction of the true distribution that is covered by the distribution learned by the model. The measure was further improved both theoretically and practically by Simon et al. (2019), while Kynkäänniemi et al. (2019) provides an explicit non-parametric variant of the original probabilistic approach. We use the precision and recall measure provided by Kynkäänniemi et al. (2019).

Regarding the assessment of the quality of the latent representation, a widely adopted approach is the measure of disentanglement (Higgins et al., 2018; Bengio et al., 2013; Tschannen et al., 2018). A representation is said to be disentangled if each latent component encodes exactly one ground truth generative factor present in the data (Kim and Mnih, 2018). Existing frameworks for both learning and evaluating disentangled representations (Higgins et al., 2017; Kim and Mnih, 2018; Eastwood and Williams, 2018; Chen et al., 2018a; Kumar et al., 2017) rely on the assumption that the ground truth factors of variation are known a priori and are independent. The core idea is to measure how changes in the generative factors affect the latent representations and vice versa. In cases when an encoder network is available, this is typically achieved with a classifier that was trained to predict which generative factor in the input data was held constant given a latent representation (Higgins et al., 2017; Kim and Mnih, 2018; Eastwood and Williams, 2018; Kumar et al., 2017; Chen et al., 2018a). In generative models without an encoder network, such as GANs, disentanglement is measured by visually inspecting the latent traversals provided that the input data are images (Chen et al., 2016; Jeon et al., 2021; Lee et al., 2020; Liu et al., 2019). However, these measures are difficult to apply when generative factors of variation are unknown or when manual visual inspection is not possible, both of which is the case with sequences of motor commands for controlling a robotic arm. We therefore define a measure of disentanglement that does not rely on any of these requirements. Our measure, *disentanglement with precision and recall (DwPR)*, is derived from the precision and recall measure introduced by

Kynkäänniemi et al. (2019) and estimates the level of disentanglement by measuring the diversity of trajectories generated when holding one latent dimension fixed. In contrast to existing measures, DwPR measures how changes in the latent space affect the generated trajectories in a fully unsupervised way.

3. Overview of the GenRL Framework

In this section, we provide an overview of the proposed GenRL approach. We consider a finite-horizon Markov decision process of length T, defined by a tuple $(\mathbf{S}, \mathbf{U}, p(s_{t+1}|s_t, u_t), p(s_0), r(s_T))$, where $\mathbf{S}$ is a set of states s, $\mathbf{U}$ is a set of M-dimensional motor actions u, $p(s_{t+1}|s_t, u_t)$ is the state transition probability, $p(s_0)$ is the initial state distribution, and $r(s_T)$ is the terminal reward function. The state contains information about the current configuration of the environment as well as the goal to reach in case a goal-conditioned policy has to be obtained. By abuse of notation, we will drop the explicit distinction between a random variable, e.g., s, and a concrete instance of it, in the rest of the paper and only write s when no confusions can arise.

We consider settings in which the agent only receives sparse terminal rewards at the end of an episode. Let r^* be the minimum reward required to successfully complete the task. We wish to find a policy $\pi_\Theta(\tau|s_0)$ which conditioned on an initial state, outputs a sequence of motor actions $\tau = \{u_t\}_{t=0:T-1}$ that results in high terminal rewards. The likelihood of a roll-out of states and actions $p_\Theta\{s_0, u_0, ..., s_{T-1}, u_{T-1}, s_T\}$ under the policy π_Θ is computed as $p(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, u_t)\pi_\Theta(u_t|s_0)$. The reward can then be found as $r = r(s_T)$. Considering the stochasticity of the terminal state, s_T, the terminal reward can be modeled by a conditional distribution $p(r|s_0, \tau)$ conditioned on the initial state s_0, and the sequence of actions τ. Here, the policy's objective is to maximize the likelihood of the terminal reward r being greater than r^*:

$$
\begin{aligned}
\Theta^* &= \underset{\Theta}{\operatorname{argmax}} \iint p(s_0)\, \pi_\Theta(\tau|s_0)\, p(r|s_0, \tau)\, ds\, d\tau \\
&= \underset{\Theta}{\operatorname{argmax}} \, \mathbb{E}_{s_0 \sim p(s_0), \tau \sim \pi_\Theta(\tau|s_0)}[\, p(r|s_0, \tau)\,],
\end{aligned}
\tag{1}
$$

where, we omitted the reward threshold r^* for simplicity. Our approach is based on training a generative model g_ϑ, parametrized by ϑ, that maps a low-dimensional latent action sample $\alpha \in \mathbb{R}^{N_\alpha}$ into a motion trajectory $\tau \in \mathbb{R}^{T \times M}$, where $N_\alpha \ll T \times M$. In other words, we assume that the search space is limited to the trajectories spanned by the generative model. In this case, the feed-forward policy search problem splits into two sub-problems: (i) finding the mapping $g_\vartheta(\alpha)$ and (ii) finding the sub-policy $\pi_\theta(\alpha|s)$, where $\Theta = [\theta, \vartheta]$. Instead of marginalizing over trajectories as in (1) we marginalize over the latent variable by exploiting the generative model

$$
\theta^* = \underset{\theta}{\operatorname{argmax}} \, \mathbb{E}_{s_0 \sim p(s_0), \alpha \sim \pi_\theta(\alpha|s_0)}[\, p(r|s_0, g_\vartheta(\alpha))\,].
\tag{2}
$$

An overview of our approach is shown in Figure 1. Once the models are trained the output of the policy π_Θ is found by first sampling from the sub-policy $\alpha \sim \pi_\theta(\alpha|s_0)$ given an initial state s_0 and then using the mapping g_ϑ to get the sequence of motor actions
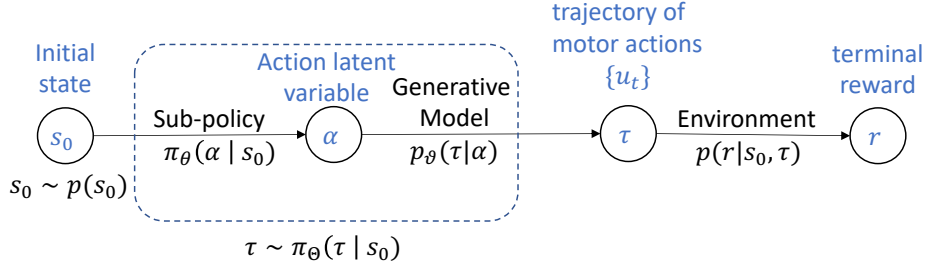
Figure 1: The architecture of the deep action-selection policy π_Θ based on latent-variable generative models. The policy consists of two models, the sub-policy $\pi_\theta(\alpha|s_0)$ that assigns a distribution over the action latent variable α conditioned on a given initial state s_0, and a generative model $\tau = g_\vartheta(\alpha)$ that maps a latent action sample $\alpha \sim \pi_\theta(\alpha|s_0)$ into a trajectory of motor actions τ. Given the initial state s_0 and the action trajectory τ that is executed on the robot, the environment returns a terminal reward r according to the reward probability $r \sim p(r|s_0, \tau)$.

$\tau = g_\vartheta(\alpha)$. The state s_0 and the generated trajectory τ are then given to the environment which outputs a terminal reward r. In the rest of the text we refer to the sub-policy as the *policy* and omit the parameters ϑ from the notation g_ϑ when they are not needed. In the following section, we introduce the expectation-maximization (EM) algorithm for training an action-selection policy using a generative model based on which different motor trajectories suitable for solving a given task can be generated.

4. Expectation-Maximization Policy Training

The EM algorithm is a well-suited approach to find the maximum likelihood solution to the intractable marginalization over the latent variable introduced in (1) and (2). We use the EM algorithm to find an optimal policy $\pi_{\theta^*}(\alpha|s_0)$ by first introducing a variational policy $q(\alpha|s_0)$ which can be a copy of the policy network. In this case, the variational policy is simply the next update for the policy. As the goal is to find an action trajectory τ that maximizes the reward probability $p(r|s_0, \tau)$, we start by expressing its logarithm as $\log p(r|s_0) = \int q(\alpha|s_0) \log p(r|s_0) d\alpha$, where we used the identity $\int q(\alpha|s_0) d\alpha = 1$ and omitted the conditioning on τ in the reward probability for simplicity. Following the EM derivation introduced in Neumann (2011) and using the identity $p(r|s_0) = p(r, \alpha|s_0)/p(\alpha|r, s_0)$, the expression can be further decomposed into

$$\log p(r|s_0) = \underbrace{\int q(\alpha|s_0) \log \frac{p(r, \alpha|s_0)}{q(\alpha|s_0)} d\alpha}_{\text{I}} + \underbrace{\int q(\alpha|s_0) \log \frac{q(\alpha|s_0)}{p(\alpha|r, s_0)} d\alpha}_{\text{II}} \tag{3}$$

The second term (II) is the Kullback-Leibler (KL) divergence $D_{KL}(q(\alpha|s_0) \,||\, p(\alpha|r, s_0))$ between distributions $q(\alpha|s_0)$ and $p(\alpha|r, s_0)$, which is a non-negative quantity. Therefore, the first term (I) provides a lower-bound for $\log p(r|s_0)$. Instead of maximizing the log-likelihood term directly, we maximize the lower bound using the EM algorithm. It is an

iterative procedure consisting of two steps known as the expectation (E-) and the maximization (M-) steps, introduced in the following sections.

4.1 Expectation Step

The goal of the E-step is to maximize the lower bound indirectly by minimizing the KL-divergence with respect to the variational policy. The left hand side of Eq. 3 does not depend on the variational policy $q(\alpha|s)$. Therefore, the sum of (I) and (II) in the equation must remain the same when we only update the variational policy. As the result, reducing (II) in Eq. 3 should increase (I). Therefore, the E-Step objective, is to minimize (II) by optimizing q. Assuming that q is parametrized by ϕ, the E-step objective function is given by

$$\phi^* = \underset{\phi}{\operatorname{argmin}} D_{KL}(\, q_\phi(\alpha|s) \, || \, p(\alpha|r, s) \,)$$
$$= \underset{\phi}{\operatorname{argmax}} \mathbb{E}_{\alpha \sim q_\phi(\alpha|s)}[\log p(r|s, \alpha)] - D_{KL}(\, q_\phi(\alpha|s) \, || \, \pi_\theta(\alpha|s) \,), \qquad (4)$$

where, we used the Bayes rule $p(\alpha|r, s) = p(r|\alpha, s)p(\alpha|s)/p(r|s)$ and substituted $p(\alpha|s)$ by $\pi_\theta(\alpha|s)$. We can replace the original MDP with another one in which the action space is the latent space of the generative model. Besides, the horizon length of this new MDP is equal to one, $T = 1$, since the entire sequence of actions is now modeled by a single latent variable α. Therefore, we solve a contextual multi-armed bandit instead of a sequential decision making problem. The reward function can now be considered as $r(s, g(\alpha))$, or in short $r(s, \alpha)$. We can now use the policy gradient theorem to maximize the expected reward value $\mathbb{E}_{q_\phi(\alpha|s)}[r(s, \alpha)]$.

$D_{KL}(\, q_\phi(\alpha|s) \, || \, \pi_\theta(\alpha|s) \,)$ acts as a trust region term forcing q_ϕ not to deviate too much from the policy distribution π_θ. Therefore, we can apply policy search algorithms with trust region terms to optimize the objective given in (4). Following the derivations introduced in Schulman et al. (2015), we adopt TRPO objective for the E-step optimization

$$\phi^* = \underset{\phi}{\operatorname{argmax}} \mathbb{E}_{s \sim p(s_0), \alpha \sim \pi_\theta(\alpha|s)} \left[\frac{q_\phi(\alpha|s)}{\pi_\theta(\alpha|s)} A(s, \alpha) - D_{KL}(q_\phi(\alpha|s) \, || \, \pi_\theta(\alpha|s)) \right], \qquad (5)$$

where, $A(s, \alpha) = r(s, \alpha) - V_\pi(s)$ is the advantage function, $V_\pi(s) = \mathbb{E}_{\alpha \sim \pi_\theta(\alpha|s)}[r(s, \alpha)]$ is the value function and ϕ^* denotes the optimal solution for the given iteration. Note that the action latent variable α is always sampled from the policy $\pi_\theta(\alpha|s)$ and not from the variational policy $q_\phi(\alpha|s)$.

4.2 Maximization Step

The M-step yields a supervised learning objective with which we train the policy π_θ. It directly maximizes the lower bound (I) in (3) by optimizing the policy parameters θ while holding the variational policy q_ϕ constant. Following Deisenroth et al. (2013) and noting that the dynamics of the system $p(r|\alpha, s)$ are not affected by the choice of the policy parameters θ, we maximize (I) by minimizing the following KL-divergence

$$\theta^* = \underset{\theta}{\operatorname{argmin}} D_{KL}(\, q_\phi(\alpha|s) \, || \, \pi_\theta(\alpha|s) \,). \qquad (6)$$

In other words, the M-step updates the policy π_θ to match the distribution of the variational policy q_ϕ which was updated in the E-step. Similarly as in the E-step, θ^* denotes the optimal solution for the given iteration. However, for the variational policies with same network architecture as the policy network (which is the case for the current paper), the M-step can be simply copying the parameters of the variational policy ϕ onto the policy parameters θ.

A summary of EM policy training is given in Algorithm 1. In each iteration, a set of states $\{s_i\}$ is sampled from the initial state distribution $p(s)$. For each state s_i, a latent action sample α_i is sampled from the distribution given by the policy $\pi_\theta(\alpha|s_i)$. A generative model g is then used to map every latent action variable α_i into a full motor trajectory τ_i which is then deployed on the robot to get the corresponding reward value r_i. In the inner loop, the variational policy q_ϕ and the main policy π_θ are updated on batches of data.

Input : generative model g_ϑ, initial policy π_θ, initial value function V_π,
            batch size N
Output: trained π_θ
while *training π_θ* **do**
    for $i = 1, \ldots, N$ **do**
        sample states $s_i \sim p(s_0)$
        sample actions $\alpha_i \sim \pi_\theta(.|s_i)$
        generate motor actions $\tau_i \leftarrow g_\vartheta(\alpha_i)$
        obtain the reward $r_i \leftarrow r(s_i, \tau_i)$
    end
    repeat
        E-step:
            update the variational policy q_ϕ according to (5) given $\{s_i, \alpha_i, r_i\}_{i=1}^N$
        M-step:
            update the policy π_θ by copying the variational policy, i.e. $\theta \leftarrow \phi$
    until *training done*;
    update the value function $V = \mathrm{argmin}_{V'} \sum_{i=1}^N (V'(s_i) - r_i)^2$ given $\{s_i, r_i\}_{i=1}^N$
end

Algorithm 1: GenRL framework.

5. Generative Models in GenRL and Their Evaluation

So far we discussed how to train an action-selection policy based on the EM algorithm to regulate the action latent variable which is the input to a generative model. In this section, we review two prominent generative models, Variational Autoencoder (VAE) and Generative Adversarial Network (GAN), which we use in GenRL to generate sequences of actions required to solve the sequential decision-making problem. We then introduce a set of measures used to predict which properties of a generative model will influence the performance of the policy training.

5.1 Generative Models

We aim to model the distribution $p(\tau)$ of the motor actions that are suitable to complete a given task. To this end, we introduce a low-dimensional random variable α with a probability density function $p(\alpha)$ representing the latent actions which are mapped into unique trajectories τ by a generative model g. The model g is trained to maximize the likelihood $\mathbb{E}_{\tau \sim \mathcal{D}, \alpha \sim p(\alpha)}[p_\vartheta(\tau|\alpha)]$ of the training trajectories $\tau \in \mathcal{D}$ under the entire latent variable space.

5.1.1 Variational Autoencoders

A VAE (Kingma and Welling, 2014; Rezende et al., 2014) consists of encoder and decoder neural networks representing the parameters of the approximate posterior distribution $q_\varphi(\alpha|\tau)$ and the likelihood function $p_\vartheta(\tau|\alpha)$, respectively. The encoder and decoder neural networks, parametrized by φ and ϑ, respectively, are jointly trained to optimize the variational lower bound

$$\max_{\varphi, \vartheta} \mathbb{E}_{\alpha \sim q_\varphi(\alpha|\tau)}[\log p_\vartheta(\tau|\alpha)] - \beta D_{KL}(q_\varphi(\alpha|\tau)||p(\alpha)), \tag{7}$$

where the prior $p(\alpha)$ is the standard normal distribution and the parameter β (Higgins et al., 2017) a variable controlling the trade-off between the reconstruction fidelity and the structure of the latent space regulated by the KL divergence. A $\beta > 1$ encourages the model to learn more disentangled latent representations (Higgins et al., 2017).

5.1.2 Generative Adversarial Networks

A GAN model (Goodfellow et al., 2014) consists of a generator and discriminator neural network that are trained by playing a min-max game. The generative model g_ϑ, parametrized by ϑ, transforms a latent sample α sampled from the prior noise distribution $p(\alpha)$ into a trajectory $\tau = g_\vartheta(\alpha)$. The model needs to produce realistic samples resembling those obtained from the training data distribution $p(\tau)$. It is trained by playing an adversarial game against the discriminator network D_φ, parametrized by φ, which needs to distinguish a generated sample from a real one. The competition between the two networks is expressed as the following min-max objective

$$\min_\vartheta \max_\varphi \mathbb{E}_{\tau \sim p(\tau)}[\log D_\varphi(\tau)] + \mathbb{E}_{\alpha \sim p(\alpha)}[\log(1 - D_\varphi(g_\vartheta(\alpha)))]. \tag{8}$$

However, the original GAN formulation (8) does not impose any restrictions on the latent variable α and therefore the generator g_ϑ can use α in an arbitrary way. To learn disentangled latent representations we instead use InfoGAN (Chen et al., 2016) which is a version of GAN with an information-theoretic regularization added to the original objective. The regularization is based on the idea to maximise the mutual information $I(\alpha, g_\vartheta(\alpha))$ between the latent code α and the corresponding generated sample $g_\vartheta(\alpha)$. An InfoGAN model is trained using the following information minmax objective (Chen et al., 2016)

$$\min_{\vartheta, \psi} \max_\varphi \mathbb{E}_{\tau \sim p(\tau)}[\log D_\varphi(\tau)] + \mathbb{E}_{\alpha \sim p(\alpha)}[\log(1 - D_\varphi(g_\vartheta(\alpha)))]$$
$$- \lambda \mathbb{E}_{\alpha \sim p(\alpha), \tau \sim g_\vartheta(\alpha)}[\log Q_\psi(\alpha|\tau)], \tag{9}$$

where $Q_\psi(\alpha|\tau)$ is an approximation of the true unknown posterior distribution $p(\alpha|\tau)$ and λ a hyperparameter. In practice, Q_ψ is a neural network that models the parameters of a Gaussian distribution and shares all the convolutional layers with the discriminator network D_φ except for the last few output layers.

5.2 Evaluation of Generative Models in GenRL

We review the characteristics of generative models that may lead to effective policy training by measuring precision and recall as well as disentanglement. Our goal is to be able to judge the quality of the policy training by evaluating the generative models prior to the RL training. We relate the measures to the performance of the policy training in Section 6.6.

5.2.1 PRECISION AND RECALL

A generalized precision and recall measure for comparing distributions was introduced by Sajjadi et al. (2018), and further improved by Kynkäänniemi et al. (2019), to evaluate the quality of a distribution learned by a generative model g. It is based on the comparison of samples obtained from g with the samples from the ground truth reference distribution. In our case, the reference samples correspond to the training motor trajectories. Intuitively, *precision* measures the quality of the generated sequences of motor actions by quantifying how similar they are to the training trajectories. It determines the fraction of the generated samples that are realistic. On the other hand, *recall* evaluates how well the learned distribution covers the reference distribution and determines the fraction of the training trajectories that can be generated by the generative model. In the context of the policy training, we would like the output of π_Θ to be as similar as possible to the demonstrated motor trajectories. It is also important that π_Θ covers the entire state space as it must be able to reach different goal states from different task configurations. Therefore, the generative model should have both high precision and high recall scores.

The improved measure introduced by Kynkäänniemi et al. (2019) is based on an approximation of the manifolds underlying the training and generated data. In particular, given a set $\boldsymbol{T} \in \{\boldsymbol{T_r}, \boldsymbol{T_g}\}$ of either real training trajectories $\boldsymbol{T_r}$ or generated trajectories $\boldsymbol{T_g}$, the corresponding manifold is estimated by forming hyperspheres around each trajectory $\tau \in \boldsymbol{T}$ with radius equal to its kth nearest neighbour $\text{NN}_k(\tau, \boldsymbol{T})$. To determine whether or not a given novel trajectory τ lies within the volume of the approximated manifold we define a binary function

$$f(\tau, \boldsymbol{T}) = \begin{cases} 1 & \text{if } ||\tau - \tau||_2 \leq ||\tau - \text{NN}_k(\tau, \boldsymbol{T})||_2 \text{ for at least one } \tau \in \boldsymbol{T} \\ 0 & \text{otherwise.} \end{cases}$$

By counting the number of generated trajectories $\tau_g \in \boldsymbol{T_g}$ that lie on the manifold of the real data $\boldsymbol{T_r}$ we obtain the *precision*, and similarly the *recall* by counting the number of real trajectories $\tau_r \in \boldsymbol{T_r}$ that lie on the manifold of the generated data $\boldsymbol{T_g}$

$$\text{precision}(\boldsymbol{T}_r, \boldsymbol{T}_g) = \frac{1}{|\boldsymbol{T}_g|} \sum_{\tau_g \in \boldsymbol{T_g}} f(\tau_g, \boldsymbol{T}_r) \quad \text{and} \quad \text{recall}(\boldsymbol{T}_r, \boldsymbol{T}_g) = \frac{1}{|\boldsymbol{T}_r|} \sum_{\tau_r \in \boldsymbol{T_r}} f(\tau_r, \boldsymbol{T}_g).$$
(10)

In our experiments, we use the original implementation provided by Kynkäänniemi et al. (2019) directly on the trajectories as opposed to their representations as suggested in their paper.

5.2.2 Disentanglement with Precision and Recall

In addition to evaluating the quality of the generated samples, we also assess generative models by the quality of their latent action representations in terms of disentanglement. We evaluate it with a novel measure, called *disentanglement with precision and recall (DwPR)*, that we derive from the precision and recall metric introduced in Section 5.2.1 as described below.

A disentangled representation of the motor data obtained from the latent space of a generative model can be defined as the one in which every end state of the system is controllable by one latent dimension determined by the vector basis of the latent space. For example, consider a task where the goal is to shoot a hockey puck to a specific position on a table top. We say that a latent representation given by a generative model g is well-disentangled if there exists a basis of the latent space with respect to which each dimension controls one axis of the end position of the hockey puck. Our hypothesis is that the more disentangled the representation is, the more efficient is the policy training. We experimentally evaluate the effect of disentangled representations on the performance of the policy training in Section 6.6.

Our measure of disentanglement is based on the following observation: if representations given by g are well disentangled, then setting one latent dimension to a fixed value should result in limited variation in the corresponding generated trajectories $\boldsymbol{T_g}$. For example, if the 1st latent dimension controls the distance of the end position of the hockey puck to a fixed origin then setting it to a fixed value should limit the set of possible end positions of the puck to a certain spherical range (assuming the polar coordinates in the end state space). In this case, assuming that g well reflects the true data distribution, we should obtain a high precision score since the limited end positions are close to the true ones but a low recall since these do not cover all possibilities. We present the details below.

Our approach can be described in two phases. In phase 1, we generate the two sets of generated trajectories on which we run the precision and recall metrics from Section 5.2.1. Firstly, for a fixed latent dimension $l \in \{1, \ldots, N_\alpha\}$ we perform a series of $D \in \mathbb{N}$ *latent interventions* where we set the lth component of a latent code α_l to a fixed value I_d, $\alpha_l = I_d$ for $d = 1, \ldots, D$. Each intervention $\alpha_l = I_d$ is performed on a set of n samples sampled from the prior distribution $p(\alpha)$. We denote by $\boldsymbol{T}_g^{l-I_d}$ the set of n generated trajectories corresponding to the latent samples on which we performed the intervention $\alpha_l = I_d$. We define $\boldsymbol{T}_g^l = \cup_{d=1}^D \boldsymbol{T}_g^{l-I_d}$ to be the union of the trajectories corresponding to all D interventions performed on the fixed latent dimension l. By construction, $\boldsymbol{T}_g^l$ consists of $n \cdot D$ trajectories. For example, $D = 5$ latent interventions on the 2-st latent dimension yield the set $\boldsymbol{T}_g^2 = \boldsymbol{T}_g^{2-I_1} \cup \cdots \cup \boldsymbol{T}_g^{2-I_5}$. Secondly, we define a set $\boldsymbol{T}_g^{p(\alpha)}$ of $n \cdot D$ generated trajectories corresponding to randomly sampled latent samples from the prior distribution $p(\alpha)$. We then treat $\boldsymbol{T}_g^{p(\alpha)}$ and $\boldsymbol{T}_g^l$ as $\boldsymbol{T}_r$ and $\boldsymbol{T}_g$ in (10), respectively, and calculate precision$(\boldsymbol{T}_g^{p(\alpha)}, \boldsymbol{T}_g^l)$ and recall$(\boldsymbol{T}_g^{p(\alpha)}, \boldsymbol{T}_g^l)$ for every latent dimension l. Note that

we consider $\boldsymbol{T}_g^{p(\alpha)}$ as the reference set $\boldsymbol{T}_r$ because we wish to evaluate the disentanglement relative to the generative capabilities of each individual model g.

In phase 2, our goal is to aggregate the precision and recall values obtained in phase 1 to measure the magnitude of the disentanglement of the latent dimensions. To this end, we define

$$\delta(l) = \text{precision}(\boldsymbol{T}_g^{p(\alpha)}, \boldsymbol{T}_g^l) - \text{recall}(\boldsymbol{T}_g^{p(\alpha)}, \boldsymbol{T}_g^l) \tag{11}$$

for each latent dimension l. Since a well disentangled generative model g should achieve high precision but low recall on $\boldsymbol{T}_g^{p(\alpha)}$ and $\boldsymbol{T}_g^l$, the higher the $\delta(l)$ value, the higher the magnitude of disentanglement for the latent dimension l. Note that this definition correctly reflects even cases where the trajectories $\boldsymbol{T}_g^l$ exhibit very low variance as this will be reflected in a low precision, thus yielding a low $\delta(l)$. Due to the nature of the robotics tasks considered in our experiments in Section 6, where the end state spaces are two dimensional, we compare our models based on the highest two $\delta(l)$ values obtained for each g. Let $\delta_1(g)$ and $\delta_2(g)$ denote the highest and second highest $\delta(l)$ values, respectively. To compare the magnitude of the two most disentangled latent dimensions among the models, we use $\delta_1(g)$ and the average $\widetilde{\delta}(g) := \frac{\delta_1(g) + \delta_2(g)}{2}$. Note that we generally cannot determine whether the two latent dimensions control the same aspect of the end state space without executing the obtained trajectories on the robot.

6. Experiments

We experimentally validated our approach by:

1. evaluating data efficiency of GenRL (Section 6.4),

2. evaluating safety of GenRL (Section 6.5), and

3. determining the characteristics of generative models that contribute to a more data-efficient policy training (Section 6.6).

We performed several experiments on two simulated tasks: *hockey task* and *basketball task*, described in Section 6.1 and compared the performance of the GenRL framework to two state-of-the-art policy search algorithms, PPO (Schulman et al., 2017) and SAC (Haarnoja et al., 2018b) in Sections 6.4 and 6.5. Also, we provide details on the training and evaluation of the generative models in Section 6.2 and the corresponding EM policy training in Section 6.3.

6.1 Experimental Setup and Implementation Details

In this section, we provide details of the experimental setup and describe the considered hockey and basketball tasks.

As illustrated in Figure 2, we evaluated GenRL by training goal-conditioned policies for two robotic tasks, shooting a hockey puck and throwing a basketball.

In the **Hockey** task, the goal is to learn to shoot a hockey puck to a target position (illustrated by the red dot in Figure 2) by a hockey stick which is attached to the robot

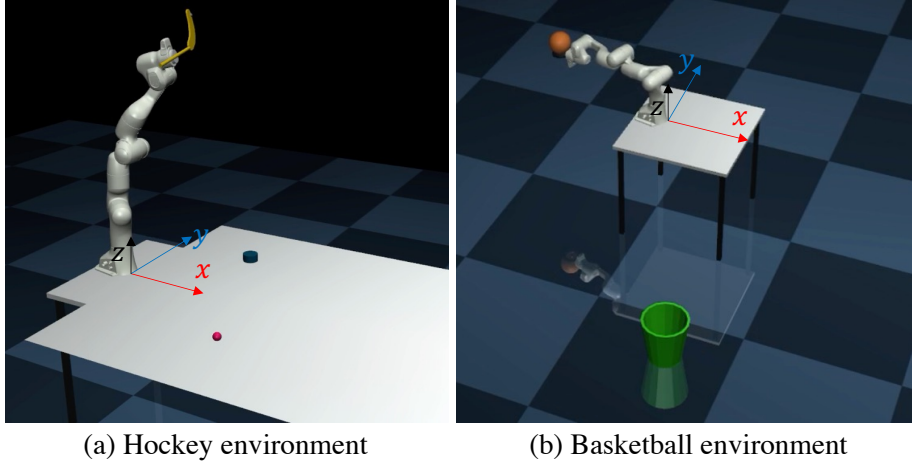(a) Hockey environment   (b) Basketball environment

Figure 2: The hockey shooting and basketball throwing environments used in the experiments. The Mujoco engine is used to simulate the robotic environments.

end-effector. The target is randomly placed at a position (x, y) in a rectangular area with $x \in [0.65, 1.3]$ meters, and $y \in [-0.4, 0.4]$ meters from the robot. The feed-forward policy, conditioned on the xy coordinates of the goal position, generates a sequence of actions consisting of 7 motor actions for 69 time-steps. The *feedback* policies, conditioned on the xy coordinate of the goal position (2-dimensional) and the joint position (7-dimensional) and velocity of the robot (7-dimensional), assign a distribution over the next motor action (7-dimensional) which is the commanded change in the joint position of the robot. The agent receives a terminal reward at the end of each episode which is calculated as the negative Euclidean distance between final position of the puck and the target.

In the **Basketball** task, the goal of the robot is to throw a ball to a basket which is randomly placed at a radius of one to eight meters from the robot, and at the angle formed by the basket center and the x-axis ranging from $-\pi/2$ to $\pi/2$. The feed-forward policy, conditioned on the xy coordinates of the basket, generates a sequence of actions consisting of 7 motor actions for 78 time-steps. On the other hand, the *feedback* policies, which are conditioned on the xy coordinate of the basket and the joint position and velocity of the robot (14 dimensions), assign a distribution over the next motor action (7 dimensions). In this environment, the reward is calculated as follows: at each timestep where the ball passes the height of the basket, the Euclidean distance to the goal in the horizontal plane is calculated and stored. At the end of the episode, the minimum of these distances is used as the terminal reward (with a minus sign). This allows the agent to learn complex throwing strategies (such as ones where the ball bounces on the floor multiple times before landing in the basket) without making the reward excessively sparse.

6.2 Generative Model Training and Evaluation

The generative models are represented by neural networks that map a low-dimensional action latent variable α into a $n_{\text{joints}} \times n_{\text{time-steps}}$ dimensional vector representing n_{joints} motor

	N_α	β	reward			N_α	λ	reward	
			hockey	*basketball*				*hockey*	*basketball*
VAE1	2	0.01	-0.267	-0.106	GAN1	2	0.1	-0.144	-0.767
VAE2	2	0.005	-0.085	-0.081	GAN2	2	1.5	-0.129	-0.469
VAE3	2	0.001	-0.040	-0.081	**GAN3**	2	3.5	-0.119	-1.183
VAE4	3	0.01	-0.266	-0.109	GAN4	3	0.1	-0.155	-0.413
VAE5	3	0.005	-0.060	-0.119	**GAN5**	3	1.5	-0.068	-0.357
VAE6	3	0.001	-0.033	-0.093	**GAN6**	3	3.5	-0.101	-1.019
VAE7	6	0.01	-0.266	-0.088	**GAN7**	6	0.1	-0.159	-0.337
VAE8	6	0.005	-0.065	-0.108	**GAN8**	6	1.5	-0.153	-0.254
VAE9	6	0.001	-0.038	-0.104	GAN9	6	3.5	-0.135	-1.344

Table 1: Training details of the generative models used in our experiments. We varied the latent dimensionality N_α and hyperparameters β in case of β-VAEs and λ in case of InfoGANs affecting the disentanglement. We additionally report the maximum reward achieved by each of these models over 3 independent runs (see Section 6.3). We marked three best performing models per task in gray.

actions and $n_{\text{time-steps}}$ time-steps. For all tasks $n_{\text{joints}} = 7$, while $n_{\text{time-steps}} = 69$ for the hockey task and $n_{\text{time-steps}} = 78$ for the basketball task. For all the models, we varied the dimension of the latent space $N_\alpha \in \{2, 3, 6\}$. For each task and each dimension N_α, we trained *i)* 3 β-VAE models with $\beta \in 0.01, 0.005, 0.001$ which are referred to as *VAE1-9* as reported in Table 1 (left column), and *ii)* 3 InfoGAN models with $\gamma \in 0.1, 1.5, 3.5$ which are referred to as *GAN1-9* (right column). In total, we considered 9 β-VAE models and 9 Info-GAN models per task. We refer the reader to Appendix B for the complete training details and the exact architecture of the models, as well as to Appendix C for visualized examples of generated trajectories. The prior distribution $p(\alpha)$ was chosen to be the standard normal $N(0, 1)$ in case of β-VAEs, and uniform $U(-1, 1)$ in case of InfoGANs. We evaluated all the generative models using the precision and recall as well as disentanglement measures introduced in Section 5.2 with hyperparameters described below.

6.2.1 PRECISION AND RECALL

For each generative model g, we randomly sampled 15000 samples from the latent prior distribution $p(\alpha)$. The corresponding set of the generated trajectories $\boldsymbol{T_g}$ was compared to a set $\boldsymbol{T_r}$ comprised of 15000 randomly chosen training trajectories which were sampled only once and fixed for all the models. The neighbourhood size k was set to 3 as suggested by Kynkäänniemi et al. (2019).

6.2.2 DISENTANGLEMENT

For each generative model g, we performed $D = 5$ interventions on every latent dimension $l \in \{1, \ldots, N_\alpha\}$ and sampled $n = 10000$ latent vectors for each D. Thus, the resulting sets $\boldsymbol{T}_g^l$ contained $n \cdot D = 50000$. We additionally generated 50000 trajectories from the
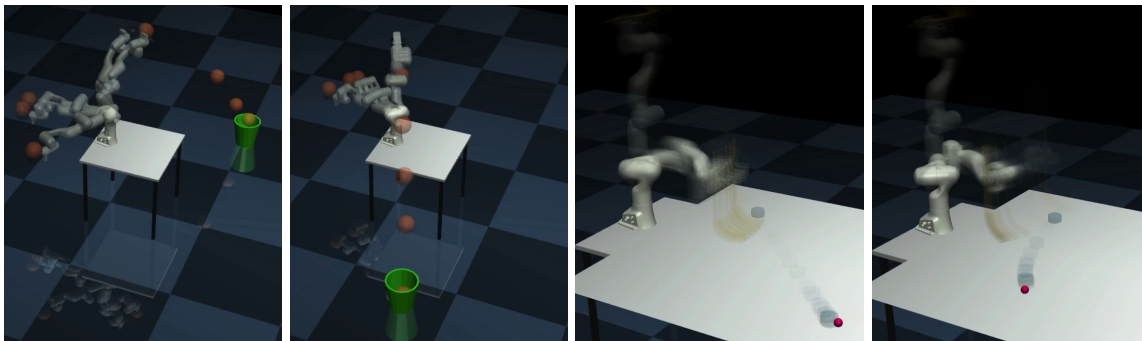
Figure 3: Examples of the performance of policies trained with the GenRL method on the basketball and hockey tasks. For the basketball task, the trained policy can successfully throw the ball into the green basket placed at different random locations, and for the hockey task, the policy can shoot the puck to random target positions illustrated by the red dots.

corresponding prior $p(\alpha)$ used as the reference set $\boldsymbol{T}_g^{p(\alpha)}$, which we kept fixed for each g. As in precision and recall, we used the neighbourhood size $k = 3$.

6.3 EM Policy Training

GenRL trained generative models with offline action data containing 40K episodes of 7-dimensional motor actions, each containing 69 time-steps for the hockey task and 78 time-steps for the basketball task. We used the combination of human demonstration and expert trajectory shaping to collect the motor action data. First, we collected a few human demonstrations through kinesthetic teaching and then added random values to some of the robot joint velocities to generate different hits or throws, e.g., with different strengths or different orientations. For each generative model and each task, we trained one policy with three different random seeds. The obtained average training performances together with the standard deviations are shown in Figures 4 and 5 for the hockey and basketball tasks, respectively. Using these results, we labeled each generative model with the maximum reward achieved during the EM policy training across all three random seeds. The values are reported Table 1 where we marked the best performing models per task in gray. In general, we observe that while VAE3 and GAN6 performed well on both tasks, VAEs generally performed better than GAN models. Figure 3 illustrates examples of GenRL performance at the deployment phase. For the basketball task, the policy trained with GenRL can successfully throw the ball into baskets placed randomly at different locations. Also, for the hockey task, the trained policy can successfully shoot the puck to different random target positions.

6.4 Comparison to Other Policy Search Methods

We compared GenRL with two state-of-the-art policy search algorithms, PPO (Schulman et al., 2017) and SAC (Haarnoja et al., 2018b), in terms of the data efficiency and safety
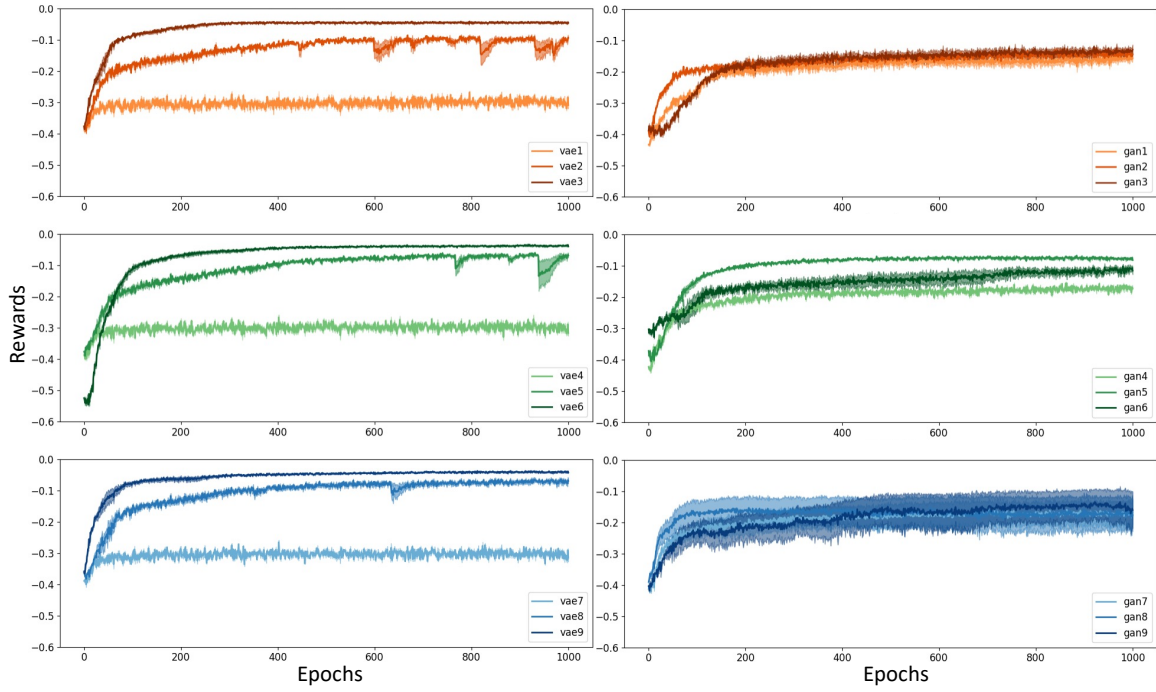
Figure 4: Policy training performance for the hockey task. We report the average reward together with the standard deviation obtained during EM policy training across three different random seeds.
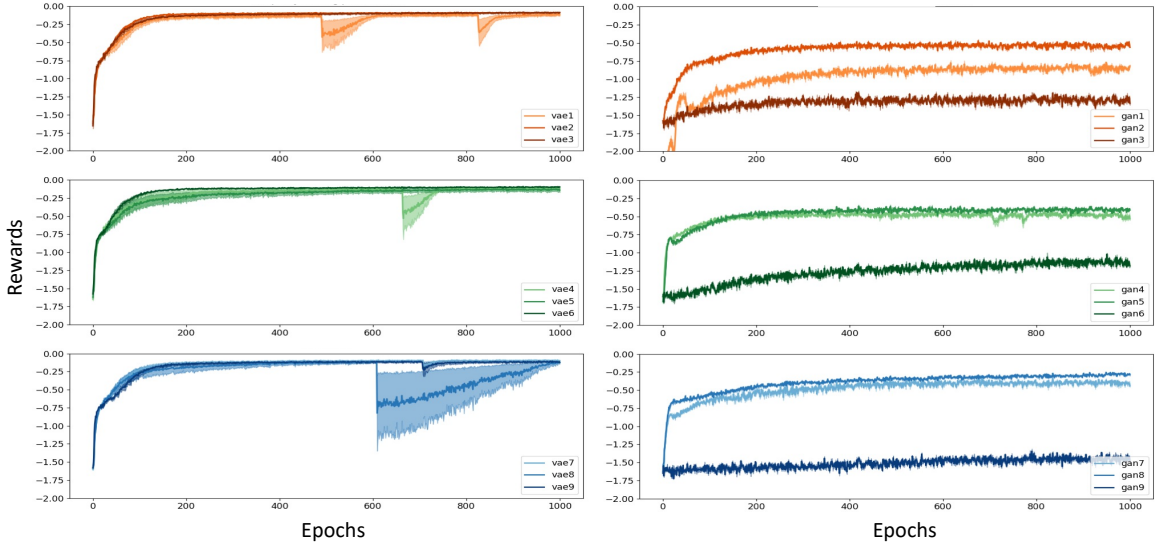

Figure 5: Policy training performance for the basketball task. We report the average reward together with the standard deviation obtained during EM policy training across three different random seeds.
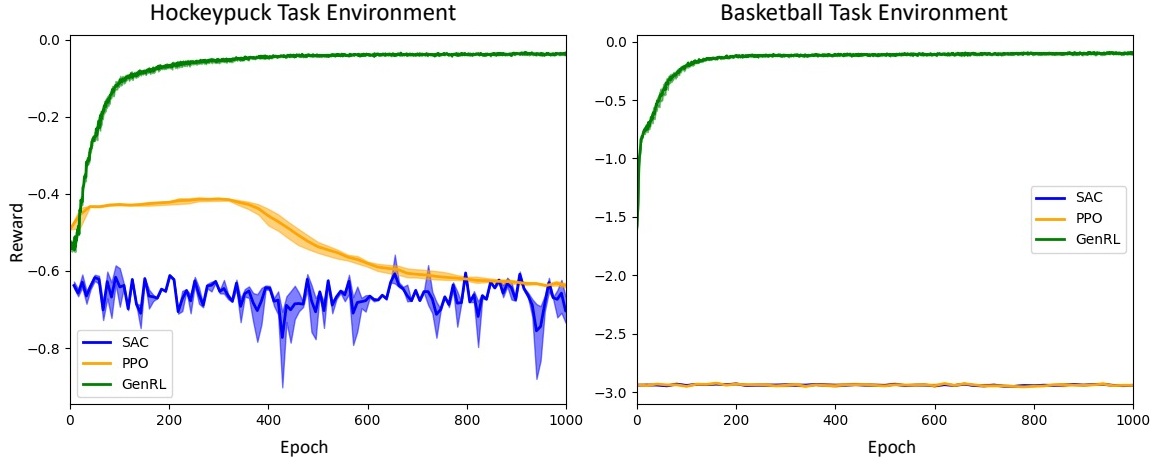
Figure 6: The average policy training performance of GenRL, PPO and SAC on the hockey and basketball tasks. We report the average reward together with the standard deviation (y-axis) obtained across three different random seeds.

of RL exploration. To make fair comparisons, we also pre-trained PPO and SAC with the same amount of data used to train the generative model of the GenRL. However, for the PPO and SAC methods, we needed to provide corresponding state transition and reward data in addition to the action data. We pre-trained a goal-conditioned PPO policy with the behavior cloning approach provided the pre-recorded state-action pairs at every time-step and the goal state. On the other hand, SAC was pre-trained by adding the the goal states and the state-action pairs to its replay buffer. In this way, we ensured that PPO and SAC receive the same amount of data for pre-training the policy.

Figure 6 shows the average training performance for three random seeds of SAC, PPO and GenRL for the hockey and basketball task. As shown, PPO and SAC cannot train a policy that can solve the tasks, and training with the pre-collected data does not help. For the basketball task (right), the robot looses the ball at the beginning of episodes because of the shaky motion caused by sampling actions from the PPO or SAC policies. This is not a problem for GenRL because it uses the generative model to generate smooth trajectories of actions to properly throw the ball. For the hockey task (left), pre-training the policy with behavior cloning helps at the beginning of the training. However, further training with PPO worsen the learning performance most likely because it is hard to learn from the sparse rewards provided for the PPO training phase. Moreover, SAC completely fails to learn a policy for the hockey task even though its replay buffer is initialized by high-quality demonstration data. GenRL, on the other hand, consistently achieves significantly better performance on both of the tasks even though its generative model is only trained with action data, and not with all data containing state-action transitions and rewards.
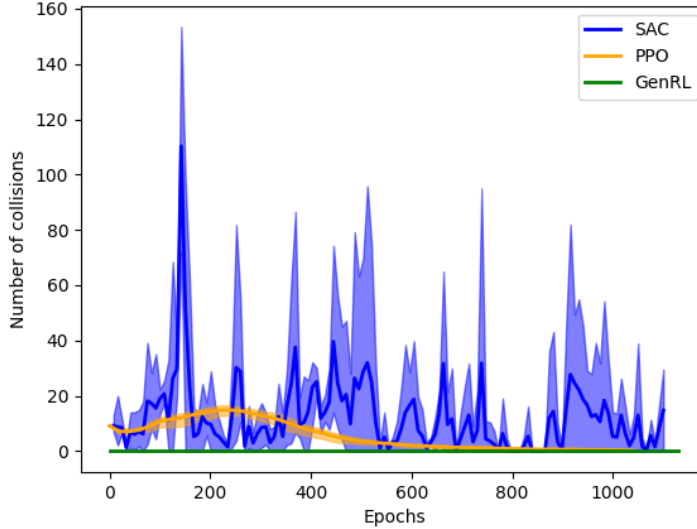
19

Figure 7: The average number of unsafe collisions during policy training with GenRL, PPO and SAC. We report the average number of time-steps per episode during which the hockey blade is in contact with the table top for the hockey task.

6.5 Safety of Policy Training with GenRL

In this section, we study the safety aspect of policy training on physical systems. Safe policy training is particularly important for real-robot applications to avoid damages to the robot and its environment due to the trial-and-error exploration of RL training. For this experiment, we train policies using GenRL, PPO and SAC on the hockey task, and report the average number of time-steps per episode during which the hockey blade is in contact with the table top. The collision between the blade and the table top is a proper indication of the safety of the policy training algorithm in our hockey task setting because collisions between the blade and the table top are more likely than any other type of collisions in this setting, and in real world, training must be stopped when such collisions happen.

Similar to the experiments introduced in the previous section, we pre-train PPO and SAC with the same amount of data used to train the generative model of the GenRL. We train policies using all three methods for a fixed number of epochs. Figure 7 illustrates the result for the GenRL, PPO and SAC methods. Most importantly, during the entire training time with GenRL, there is no collision between the table top and the blade. According to our collision tracker algorithm, the only collision is hitting the puck with the hockey blade. However, during policy training with PPO and SAC, the blade hits the table top for many time-steps which indicates that these methods may not be suitable for real-robot training. Therefore, considering both the performance and safety, policy training with GenRL is a better alternative than SAC and PPO when training on similar robotic tasks.

hockey		Precision	Recall	δ_1	$\widetilde{\delta}$
VAE	Pearson's R	-0.052	0.959	-0.533	0.979
	p-value	0.893	0.000	0.140	0.000
GAN	Pearson's R	-0.485	0.526	0.505	0.393
	p-value	0.186	0.146	0.165	0.295
ALL	Pearson's R	-0.068	0.737	0.082	0.833
	p-value	0.789	0.000	0.746	0.000

Table 2: Correlation results between the performance of the policy reported in Table 1 and resulting evaluation measures described in Section 5.2 for the generative models trained on the hockey task. We report Pearson correlation coefficient R (the higher the absolute value, the better ↑) with the corresponding p-value.

basketball		Precision	Recall	δ_1	$\widetilde{\delta}$
VAE	Pearson's R	-0.936	0.571	-0.059	0.131
	p-value	0.000	0.108	0.880	0.738
GAN	Pearson's R	-0.035	0.811	0.539	0.591
	p-value	0.929	0.008	0.134	0.094
ALL	Pearson's R	0.434	0.907	0.648	0.629
	p-value	0.072	0.000	0.004	0.005

Table 3: Correlation results between the performance of the policy reported in Table 1 and resulting evaluation measures described in Section 5.2 for the generative models trained on the basketball task. We report Pearson correlation coefficient R (the higher the absolute value, the better ↑) with the corresponding p-value.

6.6 Evaluation of Generative Models

In this section, we empirically determine the characteristics of generative models that contribute to a more efficient policy training for the hockey and basketball tasks. We evaluated the trained β-VAE and InfoGAN models (Section 6.2) by measuring precision and recall and disentanglement measures introduced in Section 5.2. Using these models we trained several RL policies (as described in Section 6.3) with the proposed GenRL framework presented in Section 4, and investigated the relation between the properties of the generative models and the performance of the policy. *Our analysis in the remained of this section empirically shows that generative models that are able to cover the distribution of the training trajectories well and additionally have well disentangled latent action representations capturing all aspects of the robotic task result in a successful policy training when combined with GenRL.*

We first studied the correlation of each individual evaluation measure to the performance of the policy training separately for the basketball and hockey tasks. In particular, we calculated the Pearson's correlation between the values obtained from each evaluation measure and the corresponding generative model label derived from the performance of the policy as described in Section 6.3. We report the Pearson's correlation coefficients (the
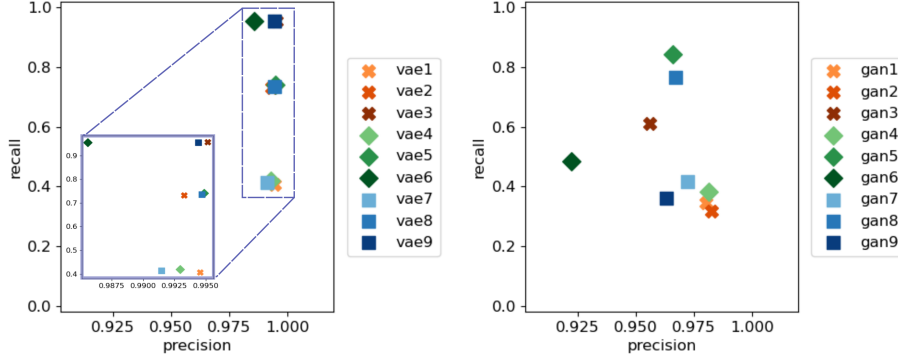
Figure 8: Precision and recall scores for VAE (left) and GAN models (right) trained on the hockey task.

higher, the better ↑) together with the corresponding p-values (the lower, the better ↓), where we consider a correlation to be significant if $p < 0.005$. Furthermore, we studied the same correlation but without differentiating between the tasks. Note that our aim was not to train perfect generative models since the variation in the evaluation results among different models enabled us to investigate which of their characteristics are important for the training of GenRL.

6.6.1 PRECISION AND RECALL

To obtain a successful policy, we expect the generative models to output trajectories that are as similar to the demonstrated motor trajectories as possible. This is measured with precision and recall scores which are shown in Figures 8 and 9 for the hockey and basketball tasks, respectively. For the *hockey* task (Figure 8), we observe that VAEs have higher precision (x-axis) and on average higher recall (y-axis) than GANs. In Table 2, we observe a positive correlation of the recall to the policy performance for VAEs, while for GANs neither precision nor recall were found significant. This is likely due to the fact that all GAN models achieve very similar maximum reward (see Table 1). However, we observe that GAN5, GAN6 and GAN3, which achieve the highest maximum reward, have the highest recall among the models. Same holds for VAE6, VAE9 and VAE3 models.

Similarly, for the *basketball* task (Figure 9), we observe that VAEs have both very high precision and recall, while GANs exhibit lower recall. Especially low recall is obtained for GAN3, GAN6 and GAN9 models which were all trained with $\lambda = 3.5$ which encourages larger disentanglement. These models all achieved worst reward (see Table 1). Note that in general the precision of GAN models trained on basketball task is lower than of those trained on the hockey task. In Table 3, we observe that precision is negatively correlated to the policy performance in case of VAEs. We hypothesize that this is a spurious correlation originating from the importance of the high recall. For GANs, we instead observe a positive correlation for recall. When the type of the model was disregarded in the correlation computation (rows ALL in Tables 2 and 3), *we observe recall to be an important characteristic*
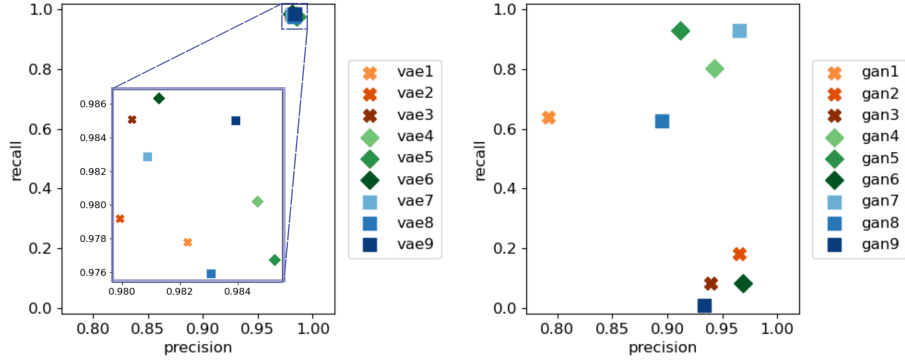
Figure 9: Precision and recall scores for VAE (left) and GAN models (right) trained on the basketball task.
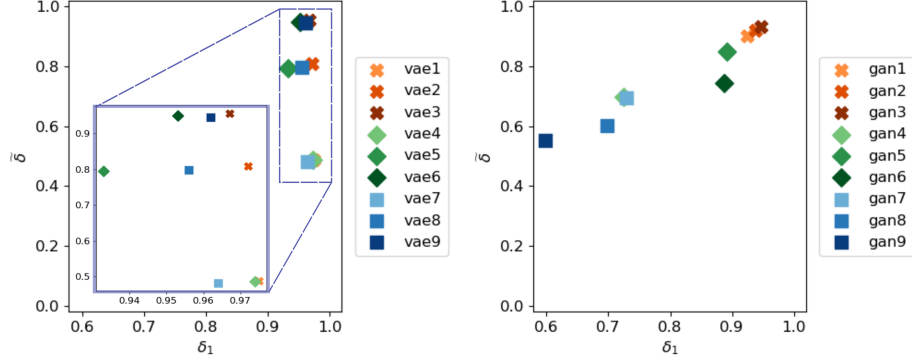

Figure 10: Disentanglement scores for VAE (left) and GAN models (right) trained on the hockey task.

of the generative models since it is positively correlated to the performance of the policy for both tasks.

6.6.2 Disentanglement

Next, we evaluated whether disentangled generative models yield a better policy trained with GenRL. Note again that we considered $\boldsymbol{T}_g^{p(\alpha)}$ as the reference set since we wished to determine how disentangled each model is with respect to its own modelling capabilities. We report the exact disentanglement scores $\delta_1, \widetilde{\delta}$ together with the respective precision and recall values in Tables 4 and 5, and provide visualization of δ_1 and $\widetilde{\delta}$ in Figures 10 and 11 for the hockey and basketball tasks, respectively.

For the *hockey* task (Figure 10), we observe that all generative models attain similar average $\widetilde{\delta}$ of two most disentangled latent dimensions (y-axis) and that GAN4, GAN7-9 models (right) attain worse maximum disentanglement δ_1 (x-axis) than VAEs (left). These

hockey	VAE1	VAE2	VAE3	VAE4	VAE5	VAE6	VAE7	VAE8	VAE9
Precision	1.000	0.988	0.979	1.000	0.984	0.984	1.000	0.988	0.986
Recall	0.025	0.016	0.012	0.026	0.051	0.032	0.036	0.032	0.024
δ_1	0.975	0.972	0.967	0.974	0.933	0.953	0.964	0.956	0.962
Precision	1.000	0.989	0.984	1.000	0.988	0.984	1.000	0.986	0.985
Recall	1.000	0.344	0.04	1.000	0.332	0.037	1.000	0.346	0.055
δ_2	0.000	0.645	0.945	0.000	0.656	0.947	0.000	0.640	0.930
$\widetilde{\delta}$	0.487	0.808	0.956	0.487	0.794	0.950	0.482	0.798	0.946

hockey	GAN1	GAN2	GAN3	GAN4	GAN5	GAN6	GAN7	GAN8	GAN9
Precision	0.966	0.965	0.960	0.975	0.957	0.952	0.961	0.949	0.937
Recall	0.041	0.027	0.013	0.251	0.065	0.066	0.232	0.250	0.340
δ_1	0.925	0.938	0.947	0.724	0.892	0.887	0.729	0.698	0.598
Precision	0.981	0.954	0.955	0.978	0.952	0.953	0.964	0.943	0.94
Recall	0.103	0.051	0.034	0.304	0.145	0.348	0.303	0.434	0.434
δ_2	0.878	0.904	0.921	0.674	0.807	0.604	0.661	0.509	0.506
$\widetilde{\delta}$	0.901	0.921	0.934	0.699	0.850	0.746	0.695	0.603	0.552

Table 4: Disentanglement scores for VAE (top rows) and GAN (bottom rows) models obtained on the hockey task. We report the precision and recall values calculated as in Equation (11) that yield the highest two values δ_1 and δ_2. We additionally report their average $\widetilde{\delta}$.
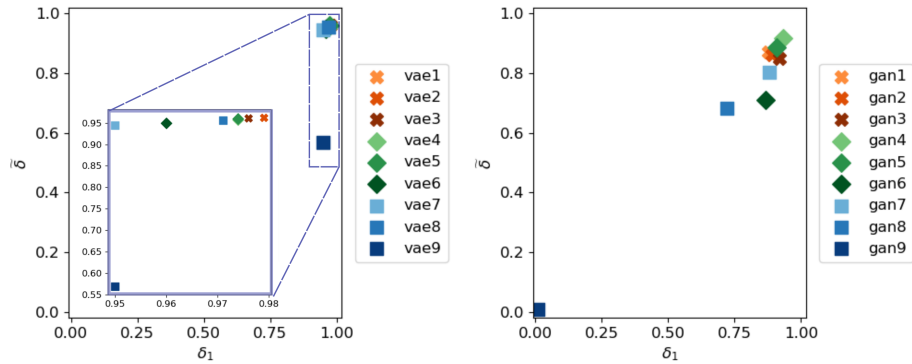

Figure 11: Disentanglement scores for VAE (left) and GAN models (right) trained on the basketball task.

basketball	VAE1	VAE2	VAE3	VAE4	VAE5	VAE6	VAE7	VAE8	VAE9
Precision	0.983	0.988	0.985	0.989	0.984	0.974	0.984	0.991	0.986
Recall	0.009	0.009	0.008	0.015	0.010	0.015	0.034	0.020	0.036
δ_1	0.974	0.979	0.976	0.974	0.974	0.960	0.950	0.971	0.950
Precision	0.986	0.986	0.986	0.979	0.984	0.983	0.985	0.987	0.984
Recall	0.042	0.041	0.040	0.038	0.037	0.045	0.045	0.045	0.796
δ_2	0.944	0.945	0.946	0.941	0.946	0.938	0.940	0.942	0.189
$\widetilde{\delta}$	0.959	0.962	0.961	0.958	0.960	0.949	0.945	0.956	0.570

basketball	GAN1	GAN2	GAN3	GAN4	GAN5	GAN6	GAN7	GAN8	GAN9
Precision	0.953	0.962	0.985	0.965	0.95	0.976	0.955	0.944	1.000
Recall	0.073	0.071	0.067	0.032	0.042	0.109	0.075	0.221	0.989
δ_1	0.879	0.891	0.918	0.933	0.908	0.867	0.880	0.723	0.011
Precision	0.957	0.966	0.989	0.949	0.956	0.995	0.951	0.925	1.000
Recall	0.096	0.134	0.205	0.043	0.092	0.440	0.222	0.281	0.998
δ_2	0.862	0.832	0.784	0.905	0.864	0.555	0.729	0.644	0.002
$\widetilde{\delta}$	0.871	0.861	0.851	0.919	0.886	0.711	0.804	0.683	0.006

Table 5: Disentanglement scores for VAE (top rows) and GAN (bottom rows) models obtained on the basketball task. We report the precision and recall values calculated as in Equation (11) that yield the highest two values δ_1 and δ_2. We additionally report their average $\widetilde{\delta}$.

all tasks		Precision	Recall	δ_1	$\widetilde{\delta}$
ALL	Pearson's R	0.479	0.650	0.538	0.458
	p-value	0.003	0.000	0.001	0.005

Table 6: Correlation results between the performance of the policy reported in Table 1 and resulting evaluation measures described in Section 5.2 for all generative models on both basketball and hockey task. We report Pearson correlation coefficient R (the higher, the better ↑) with the corresponding p-value.

models are trained with larger dimension of the latent space, $N_\alpha = 3$ and 6, respectively. We observe that GAN5, GAN6 and GAN3 models, which achieve the best maximum reward, all have both high δ_1 and high $\widetilde{\delta}$. Together with the precision and recall results visualized in Figure 8, this suggest that the models necessarily need to have both high recall and be well disentangled. We obtain similar results for VAEs, where we observe a lower average disentanglement $\widetilde{\delta}$ for VAE1, VAE4 and VAE7, which result in worse policy performance as seen in Table 1. These models are trained with $\beta = 0.01$ which should learn more disentangled latent representations. However, we observe in Table 4 (top) that $\beta = 0.01$ seems to highly disentangle only one latent dimension but ignoring the others. This is on par with observations in Table 2, where we observe a positive correlation of $\widetilde{\delta}$ to the policy performance for VAEs. Moreover, we observe that VAE3, VAE6 and VAE9 achieve highest δ_1 and $\widetilde{\delta}$, which are the models with highest precision and recall scores and best performing policy.

For the *basketball* task (Figure 11), we firstly observe that all VAEs achieve high δ_1 and $\widetilde{\delta}$ except for VAE9 (see also Table 5) which also achieves one of the lowest maximum reward. Secondly, we observe that GANs obtain lower $\widetilde{\delta}$ than VAEs. We observe that the best performing GAN4, GAN5 and GAN7 achieve high disentanglement values. These models also achieve the highest precision and recall values (see again Figure 9, right). We observe the worst disentanglement scores for GAN9 which most likely originate from the poor recall of the model. When investigating the correlation of δ_1 and $\widetilde{\delta}$ to the performance of the policy in Table 3, we find that the correlation is not significant when we distinguish the type of the generative models (rows VAE and GAN) due to the dominance of the recall measure. However, when analyzing the correlation without distinguishing the type of the model, we observe a positive correlation for both scores (row ALL). *Therefore, we conclude that disentanglement is an additionally beneficial characteristic for the generative models that obtain sufficiently high recall.*

6.6.3 General evaluation

Lastly, we evaluated the correlation of the evaluation measures to the policy performance for all the models and both tasks. The results, shown in Table 6, support our earlier conclusions. We observe the highest correlation for recall followed by δ_1. We also observe a weaker positive correlation for precision and $\widetilde{\delta}$. *Therefore, we conclude that the latent action representations should capture all aspects of the robotic task in a disentangled manner, while it is less important to capture them precisely.*

26

7. Conclusion

We presented an RL framework that combined with generative models trains deep visuomotor policies in a data-efficient manner. The generative models are integrated with the RL optimization by introducing a latent variable α that is a low-dimensional representation of a sequence of motor actions. Using the latent action variable α, we divided the optimization of the parameters Θ of the deep visuomotor policy $\pi_\Theta(\tau|s)$ into two parts: optimizing the parameters ϑ of a generative model $p_\vartheta(\tau|\alpha)$ that generates valid sequences of motor actions, and optimizing the parameters θ of a sub-policy $\pi_\theta(\alpha|s)$, where $\Theta = [\theta, \vartheta]$. The sub-policy parameters θ are found using the EM algorithm, while generative model parameters ϑ are trained unsupervised to optimize the objective corresponding to the chosen generative model. In summary, the complete framework consists of two data-efficient subsequent tasks, training the generative model p_ϑ, and training the sub-policy π_θ.

Moreover, we provided a set of measures for evaluating the quality of the generative models regulated by the RL policy search algorithms such that we can predict the performance of the deep policy training π_Θ prior to the actual training. In particular, we used the precision and recall measure (Kynkäänniemi et al., 2019) to evaluate the quality of the generated samples, and adjusted it to define a novel measure called disentanglement with precision and recall (DwPR) that additionally evaluates the quality of the latent space of the generative model p_ϑ. We experimentally demonstrated the predictive power of these measures on two tasks in simulation, shooting hockey puck and throwing a basketball, using a set of different VAE and GAN generative models. Regardless of the model type, we observe that the most beneficial latent action representations are those that capture all aspects of the robotic task in a disentangled manner.

Acknowledgments

Lin(N_α, 128) + BN + ReLU
Lin(128, 256) + BN + ReLU
Lin(256, 512) + BN + ReLU
Lin(512, InputDim)

Table 7: Architecture of the generator neural network.

Shared layers	Lin(InputDim, 256) + ReLU
	Lin(256, 128) + ReLU
discriminator	Lin(128, 1) + Sigmoid
Qnet	Lin(128, 64)
	Lin(64, N_α)

Table 8: Architecture of the discriminator and Qnet neural networks.

Appendix A. End-to-end Training of Perception and Control

The EM policy training algorithm presented in Section 4 updates the deep policy using the supervised learning objective function introduced in (6) (the M-step objective). Similar to GPS (Levine et al., 2016), the EM policy training formulation enables simultaneous training of the perception and control parts of the deep policy in an end-to-end fashion. In this section, we describe two techniques that can improve the efficiency of the end-to-end training.

A.1 Input Remapping Trick

The input remapping trick (Levine et al., 2016) can be applied to condition the variational policy q on a low-dimensional compact state representation, z, instead of the high-dimensional states s given by the sensory observations, e.g., camera images. The policy training phase can be done in a controlled environment such that extra measures other than the sensory observation of the system can be provided. These extra measures can be for example the position of a target object on a tabletop. Therefore, the image observations s can be paired with a compact task-specific state representation z such that z is used in the E-step for updating the variational policy $q_\phi(\alpha|z)$, and s in the M-step for updating the policy $\pi_\theta(\alpha|s)$.

A.2 Domain Adaptation for Perception Training

Domain adaptation techniques, e.g., adversarial methods (Chen et al., 2020), can improve the end-to-end training of visuomotor policies with limited robot data samples. The unlabeled task-specific images, captured without involving the robot, can be exploited in the M-step to improve the generality of the visuomotor policy to manipulate novel task objects in cluttered backgrounds.

The M-step is updated to include an extra loss function to adapt data from the two different domains: (i) unlabeled images and (ii) robot visuomotor data. The images must contain only one task object in a cluttered background, possibly different than the task object used by the robot during the policy training. Given images from the two domains, the basic idea is to extract visual features such that it is not possible to detect the source of the features. More details of the method can be found in our recent work in (Chen et al., 2020).
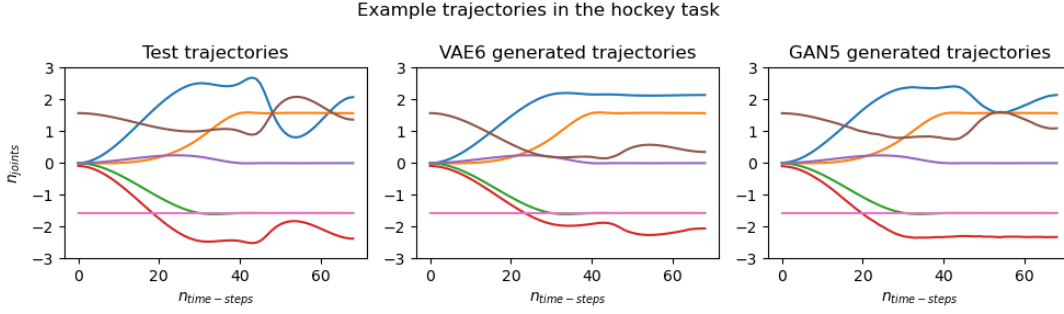
Figure 12: Example of a test trajectory (left), VAE6 generated trajectory (middle) and GAN5 generated trajectory (right) for the hockey task.

Appendix B. Training Details of Generative Models

In this section, we report architectural and training details for generative models used in our experiments. The architectures of the decoders is shown in Table 7, where we used Lin and BN to refer to the linear and batch normalization layers, respectively. The InputDim denotes the dimension of the trajectories and is equal to $69 \cdot 7$ and $78 \cdot 7$ for the hockey and basketball tasks, respectively. Architectural choices related to each type of generative model are discussed below. All the models were trained for 5000 epochs with learning rate fixed to $2e-4$ using batch size 256. For our evaluation, we chose the best performing models based on minimum squared error computed between 100 randomly generated trajectories and trajectories in the test datasets not used during training.

B.1 Variational Autoencoder

The encoder neural network is symmetric to the decoder with two output linear layers of size N_α representing the mean and the log standard deviation of the approximate posterior distribution.

B.2 InfoGAN

The architecture of the generator, discriminator and the neural network parametrizing $Q_\phi(\alpha|\tau)$ are summarised in Tables 7 and 8. Learning rates of the optimizers for the generator and discriminator networks were fixed to $2e-4$.

Appendix C. Examples of Generated Trajectories

In this section, we visualize examples of testing trajectories as well as generated ones for both hockey (Figure 12) and basketball tasks (Figure 13).
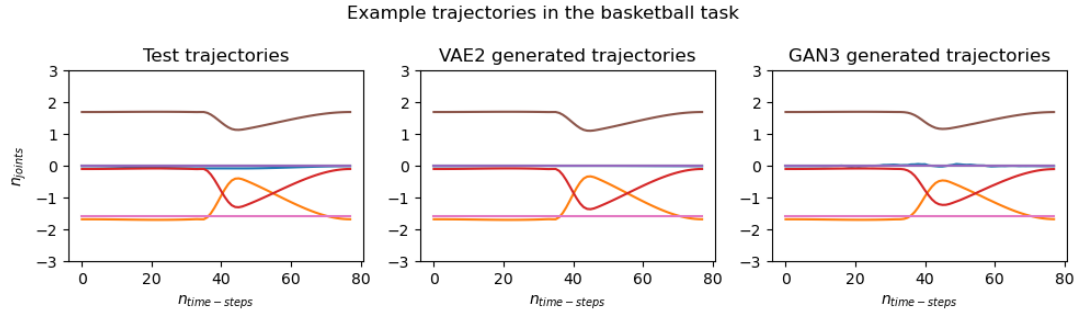
Figure 13: Example of a test trajectory (left), VAE2 generated trajectory (middle) and GAN3 generated trajectory (right) for the basketball task.

References

Abbas Abdolmaleki, Sandy H Huang, Leonard Hasenclever, Michael Neunert, H Francis Song, Martina Zambelli, Murilo F Martins, Nicolas Heess, Raia Hadsell, and Martin Riedmiller. A distributional view on multi-objective policy optimization. *arXiv preprint arXiv:2005.07513*, 2020.

Karol Arndt, Murtaza Hazara, Ali Ghadirzadeh, and Ville Kyrki. Meta reinforcement learning for sim-to-real domain adaptation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2725–2731. IEEE, 2020.

Shikhar Bahl, Mustafa Mukadam, Abhinav Gupta, and Deepak Pathak. Neural dynamic policies for end-to-end sensorimotor learning. *Advances in Neural Information Processing Systems*, 33:5058–5069, 2020.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8): 1798–1828, 2013.

Mikołaj Bińkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *International Conference on Learning Representations*, 2018.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Lars Buesing, Theophane Weber, Sébastien Racaniere, SM Eslami, Danilo Rezende, David P Reichert, Fabio Viola, Frederic Besse, Karol Gregor, Demis Hassabis, et al. Learning and querying fast generative models for reinforcement learning. *arXiv preprint arXiv:1802.03006*, 2018.

Judith Bütepage, Ali Ghadirzadeh, Özge Öztimur Karadağ, Mårten Björkman, and Danica Kragic. Imitating by generating: Deep generative models for imitation of interactive tasks. *Frontiers in Robotics and AI*, 7:47, 2020.

Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 2610–2620, 2018a.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.

Xi Chen, Ali Ghadirzadeh, John Folkesson, Mårten Björkman, and Patric Jensfelt. Deep reinforcement learning to acquire navigation skills for wheel-legged robots in complex environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3110–3116. IEEE, 2018b.

Xi Chen, Ali Ghadirzadeh, Mårten Björkman, and Patric Jensfelt. Adversarial feature training for generalizable robotic visuomotor control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1142–1148. IEEE, 2020.

Xi Chen, Ali Ghadirzadeh, Tianhe Yu, Yuan Gao, Jianhao Wang, Wenzhe Li, Bin Liang, Chelsea Finn, and Chongjie Zhang. Latent-variable advantage-weighted policy optimization for offline rl. *arXiv preprint arXiv:2203.08949*, 2022.

Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019.

Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

Coline Devin, Pieter Abbeel, Trevor Darrell, and Sergey Levine. Deep object-centric representations for generalizable robot learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7111–7118. IEEE, 2018.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Cian Eastwood and Christopher KI Williams. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*, 2018.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519. IEEE, 2016.

Ali Ghadirzadeh. *Sensorimotor Robot Policy Training using Reinforcement Learning*. PhD thesis, KTH Royal Institute of Technology, 2018.

Ali Ghadirzadeh, Atsuto Maki, Danica Kragic, and Mårten Björkman. Deep predictive policy training using reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2351–2358. IEEE, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Nishad Gothoskar, Miguel Lázaro-Gredilla, Abhishek Agarwal, Yasemin Bekiroglu, and Dileep George. Learning a generative model for robot control using visual feedback. *arXiv preprint arXiv:2003.04474*, 2020.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, and Sergey Levine. Composable deep reinforcement learning for robotic manipulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6244–6251. IEEE, 2018a.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018b.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

Aleksi Hämäläinen, Karol Arndt, Ali Ghadirzadeh, and Ville Kyrki. Affordance learning for end-to-end visuomotor robot control. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1781–1788. IEEE, 2019.

Murtaza Hazara and Ville Kyrki. Transferring generalizable motor primitives from simulation to real world. *IEEE Robotics and Automation Letters*, 4(2):2172–2179, 2019.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems 30*, pages 6626–6637. Curran Associates, Inc., 2017.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018.

Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. *arXiv preprint arXiv:1806.02426*, 2018.

Auke J Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning attractor landscapes for learning motor primitives. In *Advances in neural information processing systems*, pages 1547–1554, 2003.

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

Insu Jeon, Wonkwang Lee, Myeongjang Pyeon, and Gunhee Kim. Ib-gan: Disentangled representation learning with information bottleneck generative adversarial networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7926–7934, 2021.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. In *2nd Conference on Robot Learning (CoRL)*, 2018.

Nan Rosemary Ke, Amanpreet Singh, Ahmed Touati, Anirudh Goyal, Yoshua Bengio, Devi Parikh, and Dhruv Batra. Modeling the long term future in model-based reinforcement learning. In *International Conference on Learning Representations*, 2018.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2014.

Orr Krupnik, Igor Mordatch, and Aviv Tamar. Multi agent reinforcement learning with multi-step generative models. *arXiv preprint arXiv:1901.10251*, 2019.

Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. *arXiv preprint arXiv:1711.00848*, 2017.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *Advances in Neural Information Processing Systems*, pages 3927–3936, 2019.

Wonkwang Lee, Donggyun Kim, Seunghoon Hong, and Honglak Lee. High-fidelity synthesis with disentangled representation. *arXiv preprint arXiv:2001.04296*, 2020.

Sergey Levine and Vladlen Koltun. Variational policy search via trajectory optimization. In *Advances in neural information processing systems*, pages 207–215, 2013.

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

Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

Martina Lippi, Petra Poklukar, Michael C Welle, Anastasiia Varava, Hang Yin, Alessandro Marino, and Danica Kragic. Latent space roadmap for visual action planning of deformable and rigid object manipulation. *arXiv preprint arXiv:2003.08974*, 2020.

Bingchen Liu, Yizhe Zhu, Zuohui Fu, Gerard de Melo, and Ahmed Elgammal. Oogan: Disentangling gan with one-hot sampling and orthogonal regularization, 2019.

Francesco Locatello, Stefan Bauer, Mario Lučić, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Frederic Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning*, 2019. Best Paper Award.

Nikhil Mishra, Pieter Abbeel, and Igor Mordatch. Prediction and control with temporal segment models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2459–2468. JMLR. org, 2017.

Gerhard Neumann. Variational inference for policy search in changing situations. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 817–824, 2011.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1–8. IEEE, 2018.

Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2219–2225. IEEE, 2006.

Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017.

Petra Poklukar, Anastasiia Varava, and Danica Kragic. Geomca: Geometric evaluation of data representations. In *International Conference on Machine Learning*, pages 8588–8598. PMLR, 2021.

Petra Poklukar, Vladislav Polianskii, Anastasiia Varava, Florian T. Pokorny, and Danica Kragic Jensfelt. Delaunay component analysis for evaluation of data representations. In *International Conference on Learning Representations*, 2022.

Deirdre Quillen, Eric Jang, Ofir Nachum, Chelsea Finn, Julian Ibarz, and Sergey Levine. Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6284–6291. IEEE, 2018.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Int. Conf. Mach. Learn.*, pages 1278–1286, 2014.

Nicholas Rhinehart, Rowan McAllister, and Sergey Levine. Deep imitative models for flexible inference, planning, and control. *arXiv preprint arXiv:1810.06544*, 2018.

Mehdi SM Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. In *Advances in Neural Information Processing Systems*, pages 5228–5237, 2018.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2234–2242. Curran Associates, Inc., 2016.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Loic Simon, Ryan Webster, and Julien Rabin. Revisiting precision recall definition for generative modeling. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5799–5808, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Avi Singh, Larry Yang, and Sergey Levine. Gplac: Generalizing vision-based robotic skills using weakly labeled images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5851–5860, 2017.

Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. In *International Conference on Learning Representations*, 2020.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

Michael Tschannen, Olivier Bachem, and Mario Lucic. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*, 2018.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

Eric Tzeng, Coline Devin, Judy Hoffman, Chelsea Finn, Pieter Abbeel, Sergey Levine, Kate Saenko, and Trevor Darrell. Adapting deep visuomotor representations with weak pairwise constraints. In *Algorithmic Foundations of Robotics XII*, pages 688–703. Springer, 2020.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans.

In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

F. Wiewel and B. Yang. Continual learning for anomaly detection with variational autoencoder. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3837–3841, 2019.

W. Xu and Y. Tan. Semisupervised text classification by variational autoencoder. *IEEE Transactions on Neural Networks and Learning Systems*, 31(1):295–308, 2020.

Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018.

Wenxuan Zhou, Sujay Bajracharya, and David Held. Plas: Latent action space for offline reinforcement learning. *arXiv preprint arXiv:2011.07213*, 2020.