Interpretable Domain Randomization for Reinforcement Learning with Disentangled Representations

Moritz Schneider, B.Sc. RWTH

The present work was submitted to the *Chair of Information Management in Mechanical Engineering* at the *Faculty of Mechanical Engineering* of *RWTH Aachen University*.

presented by	Moritz Schneider
Student ID no.:	345827
1st Examiner:	Prof. Dr. phil. Ingrid Isenhardt
Supervisor:	Vladimir Samsonov, M.Sc.

Aachen, 5. Oktober 2021

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ACCI Automatic Controlled Capacity Increase

Ada-GVAE Adaptive Group-based Variational Autoencoder

ADR Automatic Domain Randomization

AE autoencoder

AI artificial intelligence

CNN convolutional neural network

COBRA Curious Object-Based seaRch Agent

CPSU cart-pole swing-up

DA domain adaptation

DARLA DisentAngled Representation Learning Agent

DNN deep neural network

DR domain randomization

ELBO evidence lower bound

FoV factors of variation

i.i.d. independently and identically distributed

GVAE Group-based Variational Autoencoder

KL Kullback-Leibler

MDP Markov decision process

MC Monte Carlo

MLP multilayer perceptron

MSBE Mean Squared Bellman Error

MSE mean squared error

OOD out-of-distribution

PCA principal component analysis

RL (deep) reinforcement learning

SAC Soft Actor-Critic

SGD stochastic gradient descent

TD temporal-difference

VAE variational autoencoder

VI variational inference

w.r.t. with respect to

1

INTRODUCTION

> Robots will neither be common nor very good in 2014, but they will be in existence.
>
> Asimov [Asi64]

This quote from Isaac Asimov from the year 1964 predicted the state of robotics in 2014 quite accurate. Around 2013, Mnih et al. [Mni+13] showed impressive results combining reinforcement learning and deep learning that lead to a milestone in today's view of artificial intelligence (AI). This progress also entered the field of robotics, which increased the overall interest in the subfield of robot learning. The field of (deep) reinforcement learning (RL) enables robotic agents to learn complex skills in the real world purely by using images. On the other hand, RL algorithms require learning from experience that the robot autonomously collects itself and the learned control policy is usually not transparent. If the data collection is done in the real-world, the behavior of the robot might be unstable during the collection process. Therefore, collecting experience is often outsourced to simulations. However, since simulations make certain assumptions, data collected in them will always differ from the real world. Due to this distribution shift in the data, robot learning algorithms often fail to transfer to different domains like the real world. By randomizing various aspects of the simulation during training it is possible to increase the agent's generalization capabilities. A method which is referred to as domain randomization (DR). Problematically, the method is not well theoretically founded. While the technique is fairly intuitive, it does not explain what representation of the randomized simulation the robot has learned in the end. Moreover, it is difficult to interpret why the robot fails anyway. On the other hand, numerous methods exist to learn representations that are more transparent. I.e. disentangled representation learning is able to learn structured representations from high-dimensional inputs. They shall capture the underlying factors of variation (FoV) of the given scene in distinct independent dimensions. A property that makes them interpretable as the factors explain the data. DR and disentangled representations both rely on the generative factors of the task. Hence there is a strong theoretical relationship between the two methods. This relationship has been rarely considered until now and is investigated in this thesis. The assumption this thesis relies on is, that combining both methods should produce better and more interpretable representations. This might ensure that the representation is capable of capturing the important factors of the task allowing a robot

to transfer to the real world more reliably. If the agent fails to transfer to the real world the interpretation of the corresponding disentangled representation could indicate which factors led to the failure.

Furthermore, this thesis examines the question, if learned disentangled representations can be combined with DR to assist the perception of RL agents. Previous work on disentangled representation learning and RL separates the overall learning process into individual substeps for both learning kinds [Hig+17b; Wul+20; Trä+21]. On the other hand, work on vision-based and data efficient reinforcement learning demonstrates that learning a meaningful representation jointly with a reinforcement learning policy helps to learn the control task [Yar+21; LSA20]. Instead of training agents on a static range of environment factors, Akkaya et al. [Akk+19] shows that it is beneficial to train agents on a DR curriculum.

The major contribution of this thesis is a framework which combines these findings. To this end, this framework must incorporate specific steps: A vision module learns to produce disentangled representations using DR. At the same time those representations should be used to train a RL agent and to guide the DR sampling process so that the interplay between disentangled representation learning and DR creates a curriculum.

To built and explain such framework, the thesis is structured into several parts. In the first chapter the fundamentals of information theory, reinforcement learning, domain randomization and unsupervised representation learning are introduced. Subsequently, related publications on vision-based RL, robot learning and DR are reviewed. In particular, work that already combines reinforcement learning and disentangled representation learning is considered in Chapter 3. Based on these foundations, the problem statement for a curriculum-based DR algorithm with an emphasize on disentangled representation learning is presented in Chapter 4 and a theoretical framework that solves this problem is developed. In Chapter 5 an actual implementation of the framework is evaluated on a particular benchmark task against other baseline algorithms. Furthermore, different parameter settings are compared and an investigation on the interpretability of the learned representation is conducted. The approach is concluded in Chapter 6 as well as future work is discussed.

2

THEORETICAL BACKGROUND

This chapter describes the theoretical foundations that this thesis builds upon. The first of the four parts depicts the fundamentals of information theory since the methods of this thesis make use of those concepts. To be able to solve aforementioned problem as a (deep) reinforcement learning (RL) problem, the necessary framework and algorithms are introduced in Section 2.2. Since the main motivation of this thesis is to transfer a controller that was trained in simulation to another domain (i.e. the real-world or a different simulation), the problem of simulation-to-reality transfer is described in Section 2.3. This should provide a theoretical introduction which methods exist on how learned policies can be transferred from a simulation to a real robot. As the additional goal of this thesis is to learn more interpretable representations in form of disentangled representations using variational autoencoders (VAEs), the fundamentals of approximate variational inference and disentangled representation learning are summarized in Section 2.4.

2.1 FUNDAMENTALS OF INFORMATION THEORY

Information theory is an important concept throughout this thesis and it is the theory about measuring the information content that is present in a signal. Likewise the theory can also be seen as measuring the uncertainty about information. Shannon [Sha48] established this theory and defined the concept of entropy regarding information similar to the entropy in thermodynamics:

Definition (Entropy)**.** *The entropy $\mathcal{H}(X)$ of a discrete random variable X following a probability distribution P with probability mass function $p(x)$ is defined by*

$$\mathcal{H}(X) = -\mathbb{E}_{x\sim p}\left[\log p(x)\right] = -\sum_{x} p(x)\log p(x). \tag{2.1}$$

In other words this can be thought of the average amount of information required to describe the random variable X or more common the average uncertainty about this random variable X. The continuous case $\mathcal{H}(X) = -\int p(x)\log p(x)dx$ is known as *differential entropy.* [CT06, pp. 13–16, 243]

This definition using the logarithm shows that likely events have a low information content as they occur more often as less likely events that have a higher information content (thus the information measurement is more uncertain). Furthermore, the information

content between two independent random variables is additive due to the calculation rules of the logarithm. [GBC16, p. 71]

The entropy between two correlated random variables X and Y can be defined as the joint entropy between X and Y:

Definition (Joint Entropy)**.** *The joint entropy $\mathcal{H}(X, Y)$ of two discrete random variables X and Y following a joint probability distribution P with probability mass function $p(x, y)$ is defined by*

$$\mathcal{H}(X, Y) = -\mathbb{E}_{x,y\sim p}\left[\log p(x, y)\right] = -\sum_x \sum_y p(x, y) \log p(x, y). \tag{2.2}$$

This joint entropy is the union of the entropies of both random variables X and Y. The *mutual information* on the other hand is a measure of the information content that both random variables have in common. [CT06, pp. 16–17]

Definition (Mutual Information)**.** *The mutual information $I(X, Y)$ of two discrete random variables X and Y following a joint probability distribution P with joint probability mass function $p(x, y)$ and marginal probability mass functions $p(x)$ and $p(y)$ is defined by*

$$I(X, Y) = \mathbb{E}_{x,y\sim p}\left[\log \frac{p(x, y)}{p(x)p(y)}\right] = \sum_x \sum_y p(x, y) \log \frac{p(x, y)}{p(x)p(y)}. \tag{2.3}$$

Calculating the mutual information for a single random variable X results in the original entropy definition $I(X, X) = \mathcal{H}(x)$ and therefore is often referred to as *self-information*. [CT06, pp. 19–21]

In the context of machine learning, a probability distribution Q with probability mass function $q(x)$ is often estimated given samples from the real probability distribution P with probability mass function $p(x)$. Typically, the inefficiency of using Q instead of P is of interest. The *Kullback-Leibler* (KL) *divergence* or *relative entropy* between two distributions can be interpreted as a distance between those:

Definition (Kullback-Leibler Divergence)**.** *The Kullback-Leibler divergence between two probability mass functions $p(x)$ and $q(x)$ of a random variable X is defined by*

$$D_{KL}(p \parallel q) = \mathbb{E}_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right] = \sum_x p(x) \log \frac{p(x)}{p(y)}. \tag{2.4}$$

The relative entropy is always non-negative and only 0 if and only if $P = Q$. The term $D_{KL}(p \parallel q)$ is defined as *forward* KL divergence whereas $D_{KL}(q \parallel p)$ is defined as *reverse* KL divergence. Both cases shall be distinguished as the KL divergence is not symmetric. Because of this property and since it does not satisfy the triangle inequality, the relative entropy is not a true distance measure. [CT06, p. 19]

All of those definitions can be transferred from discrete to continuous random variables by replacing the sums with integrals over the random variables.

2.1.1 *Information Bottleneck Method*

Shannon [Sha48] developed the fundamentals to compress and transmit data but focused less on extracting relevant information. In contrast, Tishby et al. [TPB99] point out that information theory also provides concepts to quantify relevant information:

To extract relevant information from a signal, a quantization $\tilde{X}$ of a signal X is needed which preserves maximum information about another signal Y, i.e. to extract a representation $\tilde{X}$ containing maximal information about an object Y in a given image X. The code $\tilde{X}$ is referred to as *bottleneck*. Using the mutual information functional of Equation 2.3, it is possible to measure information about Y in $\tilde{X}$:

$$I(\tilde{X}, Y) = \sum_{\tilde{x}} \sum_{y} p(\tilde{x}, y) \log \frac{p(\tilde{x}, y)}{p(\tilde{x})p(y)} \tag{2.5}$$

The original signal X is compressed in the bottleneck $\tilde{X}$, therefore the information content of $\tilde{X}$ cannot contain as much information like X about Y:

$$I(\tilde{X}, Y) \leq I(X, Y) \tag{2.6}$$

To this end, solutions of $\tilde{X}$ that contain maximum meaningful information about the signal Y while minimizing the information content from the original signal X are desired. This can be quantified by the functional

$$\min \mathcal{L} = \min I(\tilde{X}, X) - \beta I(\tilde{X}, Y) \tag{2.7}$$

in which the hyperparameter β serves as a Lagrange multiplier. High values of the hyperparameter β yield more meaningful representations while values of β near 0 result in better compressed solutions. [TPB99]

2.2 REINFORCEMENT LEARNING

In recent years much work has been done in the research area of RL. Many large-scale projects (like [Sil+16; Sil+17b; Sil+17a; And+19; Akk+19; Tea19; Ber+19]) were performed to show the potential of the created algorithms. In addition, many tools arised to train RL agents (e.g. [Dha+17; Bro+16; Tas+20; Bea+16; Joh+16]). Furthermore, many subdisciplines like meta-learning, multi-task learning, imitation learning, hierarchical Learning, etc. exist to solve different aspects of the overall reinforcement learning problem. RL can be successfully applied on many low dimensional control tasks but can also be used with high dimensional pixel inputs [Mni+13; Mni+16; And+19; Akk+19] making it the framework of choice for this thesis.

This section shall serve as a overview of the most important concepts and algorithms in the context of this thesis following the notation of the standard work Sutton et al. [SB18], François-Lavet et al. [Fra+18], and Szepesvári [Sze10]. This starts by describing the basic concepts of reinforcement learning which characterize our system and ends in

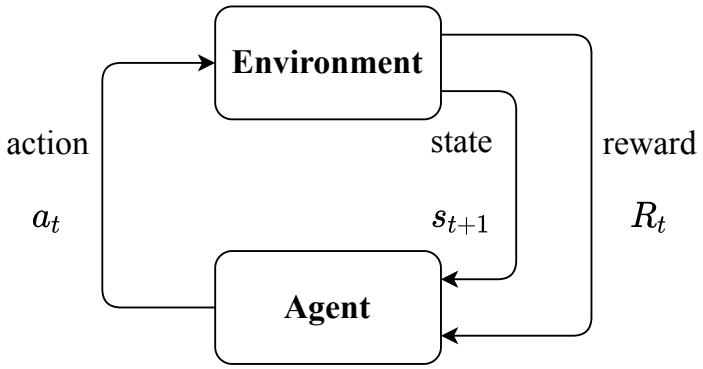


Figure 2.1: The closed-loop interaction in a MDP. [SB18, p. 48])

the description of more sophisticated algorithms which will be used as learning framework later in this thesis.

2.2.1 *Markov Decision Processes*

Reinforcement learning considers system which can be described as a Markov decision process (MDP) [Bel57]. In principle all reinforcement learning algorithms are build around the formalization of MDPs or a subtype of it. Sutton et al. [SB18] describes them as "[...] a mathematical idealized form of the reinforcement learning problem." [SB18, p. 47]. MDPs can also be seen as capsuled environments with which the learning agent interact. This process can be seen in Figure 2.1. Everything outside the agent itself will be from now on summarized as the *environment*. Mathematically these environments can be described as a 5-tuple.

Definition (Markov decision process)**.** *A Markov decision process (MDP) is a 5-tuple* $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma \rangle$ *which consists of*

- *a set of states* $\mathcal{S}$*,*
- *a set of actions* $\mathcal{A}$ *which the agent is able to take,*
- *a reward function* $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{R} \subset \mathbb{R}$ *and*
- *state-transition probabilities* $\mathcal{T}(s', r|s, a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ *with the initial distribution of starting states* $\mathcal{T}(S_0)$
- γ *as a discount factor for the reward over time which reflects the planning horizon of the agent and must be in range* $[0, 1]$*.*

The variables s and a correspond to the state and action at a timestep t whereas s' is the state at the subsequent timestep $t + 1$. Similar the reward is defined $r(s, a, s') = \mathbb{E}[R_{t+1}|S_t = s, A_t = a, S_{t+1} = s']$. With a trial-and-error approach, the algorithm shall solve the underlying MDP by learning an optimal policy π which maximizes the cumulative reward $G_0 = \sum_{t=0}^{T} R_t$ in the finite horizon case or the cumulative discounted reward

$G_0 = \sum_{t=0}^{\infty} \gamma^t \cdot R_t$ in the infinite horizon case. Both types of cumulative rewards will be called episodic return in the following with G_t as the episodic return calculated from timestep t. [SB18, pp. 48–56]

A policy can either be a mapping from a state to the action itself in the deterministic case $\pi(s) : \mathcal{S} \to \mathcal{A}$ or a mapping to a distribution of actions in the stochastic case $\pi(s, a) : \mathcal{S} \times \mathcal{A} \to [0, 1]$. At each time step t the agent chooses an action A_t resulting in the subsequent state S_{t+1} with reward R_{t+1}. This transition is based on the transition probabilities $\mathcal{T}$ which completely describe the dynamics of the environment but might not be deterministic and also not known in most cases. This dynamics might be induced by a simulation engine or the real-world. Via the experienced reward signals of the selected trajectories the agent shall improve his decision-making policy. [SB18, p. 58]

A necessary characterization of MDPs is the *Markovian property* [SB18; Fra+18, pp. 49, 235]. A decision process is *markovian* if all future aspects of the environment depend on the information of the current observation only and that information from the past which is not included in the current observation does not make a difference in the dynamics of future timesteps:

$$\mathbb{P}(S_{t+1}|S_t, A_t) = \mathbb{P}(S_{t+1}|S_0, A_0, \ldots, S_t, A_t), \quad \text{and} \tag{2.8}$$

$$\mathbb{P}(R_{t+1}|S_t, A_t) = \mathbb{P}(R_{t+1}|S_0, A_0, \ldots, S_t, A_t) \tag{2.9}$$

With this property the probability of an entire trajectory $\tau = (S_0, A_0, \ldots, S_T, A_T)$ using policy π is defined as

$$\mathbb{P}(\tau) = \mathcal{T}(S_0) \cdot \prod_{t=0}^{T} \mathcal{T}(S_{t+1}, R_{t+1}|S_t, A_t)\pi(A_t|S_t) \tag{2.10}$$

2.2.2 *Value Functions*

In many cases it is useful to know how valuable a specific state or state-action pair is to make good decisions for further steps. Therefore, most reinforcement learning policies incorporate some type of value function as a planning or guiding instrument. Since the return is the main objective of the algorithm, the value of a state can be expressed as the expected return the agent might achieve.

Definition (State-value function)**.** *The state-value function $V^\pi(s) : \mathcal{S} \to \mathbb{R}$ is the expected return from state s when following policy π*

$$V^\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] \tag{2.11}$$

This definition can be further extended by incorporating the action variable a.

Definition (Action-value function)**.** *The action-value function* $Q^\pi(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *is the expected return of following policy* π *after selecting action* a *in state* s

$$Q^\pi(s,a) = \mathbb{E}_\pi \left[G_t | S_t = s, A_t = a \right] \tag{2.12}$$

Both are connected by the following equation:

$$Q^*(s,a) = \mathbb{E} \left[R_{t+1} + \gamma V^*(S_t) | S_t = s, A_t = a \right] \tag{2.13}$$

The optimal value functions are then determined by

$$V^*(s) = \max_\pi V^\pi(s) \qquad \text{and} \qquad Q^*(s,a) = \max_\pi Q^\pi(s,a) \tag{2.14}$$

On the basis of the afterwards achieved rewards the value function can be optimized to estimate the true value of the corresponding state (and action). Both types of value functions can simply be estimated by using *Monte Carlo (MC)* methods through tracking and averaging the achieved returns G_t from each single state s (and if required performing action a at this state) and then following policy π:

$$V^\pi(S_t) \leftarrow V^\pi(S_t) + \alpha \left[G_t - V^\pi(S_t) \right] = (1-\alpha) V^\pi(s) + \alpha G_t \tag{2.15}$$

Besides, *temporal-difference (TD)* methods can be used also. More precisely this entails updating the value of the function not on the basis of the return G_t of a trajectory but on the basis of the directly achieved reward R_t and the estimates of the function values of the next states (bootstrapping):

$$V^\pi(S_t) \leftarrow V^\pi(S_t) + \alpha \left[R_t + \gamma V^\pi(S_{t+1}) - V^\pi(S_t) \right] = V^\pi(S_t) + \alpha \delta_t \tag{2.16}$$

The optimized TD-error δ_t at timestep t is defined as:

$$\delta_t = R_t + \gamma V^\pi(S_{t+1}) - V^\pi(S_t) \tag{2.17}$$

Equation 2.15, Equation 2.16 and Equation 2.17 likewise apply to the action-value function $Q(s,a)$. [SB18, pp. 58, 59, 119, 120, 121]

2.2.2.1 *Bellman equations*

Given the complete MDP, the value function for a policy π can be calculate recursively using the Bellman equation

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s',r} \mathcal{T}(s', r | s, a) \left[r + \gamma V^\pi(s') \right] \tag{2.18}$$

by applying the Bellman operator $B^\pi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ to V^π. This operator is defined as

$$\left(B^\pi V \right)(s) = \sum_a \pi(a|s) \sum_{s',r} \mathcal{T}(s', r | s, a) \left[r + \gamma V(s') \right] \forall s \in \mathcal{S}, v : \mathcal{S} \to \mathbb{R} \tag{2.19}$$

and can be applied to any value function as it yields another value function. The recursive application of the Bellman operator onto the current approximation of the value function converges to the true value function V^π of policy π since the Bellman operator is a maximum-norm contraction mapping. Therefore, the equation $B^\pi V = V$ has a single solution only. [Sze10, p. 15]

Since the operator can be applied to arbitrary value functions, a similar result can be obtained for the action-value function Q^π.

The Bellman equations are likewise defined for the optimal value functions by

$$V^*(s) = \max_a \sum_{s',r} \mathcal{T}(s', r|s, a) \left[r + \gamma V^*(s')\right] \tag{2.20}$$

and

$$Q^*(s, a) = \sum_{s',r} \mathcal{T}(s', r|s, a) \left[r + \gamma \max_{a'} Q^*(s', a')\right]. \tag{2.21}$$

2.2.3 *Policies*

The policy defines how the agent selects actions. As mentioned before this can be either done in a deterministic way by single values per dimension or in a stochastic way as a distribution over actions.

Definition (Policy)**.** *A stochastic policy π is a distribution over actions given states,*

$$\pi(a|s) = \mathbb{P}(A_t = a|S_t = s) \tag{2.22}$$

A deterministic policy π is a direct mapping from states to actions,

$$a = \pi(s) \tag{2.23}$$

Moreover, policies can be divided into two main directions: Model-based and model-free reinforcement learning. While in the first case the transition function $\mathcal{T}$ of the MDP is learned and a normal planning algorithm can be used, a representation between states and their corresponding actions is learned in the latter case. Model-free policies again can be subdivided into pure value-based methods and policy gradients. [Fra+18, p. 238]

Value-based methods incorporate learned value functions to estimate how good a specific state and/or action is for the agent. The information of these functions can be used to perform the current best action which is expected to maximize the value function.

2.2.3.1 *On- and Off-Policy Methods*

In order to learn the optimal behavior, a policy must be able to collect and evaluate enough data about the environment. Therefore, the policy has to explore the action and state space to learn how to make good decisions. At the same time it shall exploit the gathered data to empower the agent to make better decisions. The trade-off between both is called the exploration and exploitation dilemma. [SB18, p. 3]

In the context of policies this can be addressed in an on- or off-policy manner:

- Off-policy methods incorporate two different policies: A policy π which actually learns and which should be used to make optimal decisions (target policy) and a policy β that is used for exploration to collect data (behavior policy). Those methods are usually more sample efficient but introduce higher variance and require additional concepts to incorporate the sampled data from β to π. [SB18, p. 103]
- On-policy methods use the learning policy π only. Even if the policy is near-optimal already the assumption is that it is still inoptimal and explores due to this inoptimality. In principle on-policy methods are generally simpler and converge faster even though the converged solution could be a local optimum. [SB18, p. 103]

2.2.3.2 *Policy Iteration*

Given all components of a MDP an optimal policy π_* can be obtained recursively through evaluating a value function V^π for the current policy π by applying the Bellman operator from Section 2.2.2.1 and utilizing this value function afterwards to improve the policy π to yield a better policy π' that is greedy with respect to (w.r.t.) the current value function. This procedure is called policy iteration and can be outlined as follows:

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} ... \xrightarrow{I} \pi_* \xrightarrow{E} V^* \tag{2.24}$$

However, in reinforcement learning the algorithm does not have access to all components of the MDP. Therefore, this scheme must be performed in an approximated form by collecting samples from the environment by applying the current policy w.r.t. the underlying value function and using those samples to update the value function. [SB18, p. 80]

2.2.4 *Q-Learning*

An early off-policy reinforcement learning algorithm is Q-learning which approximates the optimal action-value function Q^* and uses this approximation to collect samples. Since the algorithm is off-policy it uses a behavior policy for exploration which is derived from the current action-value function but utilizes a greedy policy to update the action-value function. The latter update is based on the TD error δ_t of the action-value function:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \underbrace{\left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right]}_{\delta_t} \tag{2.25}$$

This value iteration algorithm is guaranteed to converge to the optimal value function Q^* if the state-action pairs are discrete and if all pairs are continued to be updated or visited indefinitely often. [SB18, pp. 130–131; Fra+18, pp. 243–244].

The algorithm struggles with high dimensional or continuous action spaces since in those settings it cannot be guaranteed that the state-action pairs are visited enough thus

generalization is rarely possible [Mni+15; Fra+18, pp. 243–244]. To handle high-dimensional state spaces function approximation can be used [Mni+15].

2.2.4.1 *Double Q-Learning*

The discussed Q-Learning algorithm includes calculating a maximum of the action-value function in the target calculation in Equation 2.25. Problematically, this maximum is calculated from estimated values. As those estimates will in most cases not actually fit but rather will over- and underestimate the true values, the single calculated maximum estimate will be expected to be greater than the true value. This induces a positive bias. [SB18, pp. 134–135]

By double estimation of the action values it is possible to select the best suited value and to estimate an action which results not in an overestimation of Q-values [Fra+18, pp. 246–247].

2.2.4.2 *Deep Q-Network (DQN)*

Mnih et al. [Mni+15] combine the Q-learning algorithm with deep convolutional neural networks and applied this on 49 different Atari 2600 video games directly learning from pixels. The convolutional neural networks parametrized by θ serve as approximators for the Q-value functions $\hat{Q}(S_t, aAt; \theta)$ and use Equation 2.25 to update the weights:

$$\theta_{k+1} \leftarrow \theta_k + \alpha \left[R_{t+1} + \gamma \max_a \hat{Q}(S_{t+1}, a; \theta_k^-) - \hat{Q}(S_t, A_t; \theta_k) \right] \nabla \hat{Q}(S_t, A_t; \theta_k) \tag{2.26}$$

In addition to this, DQN provides three additional changes that are nowadays also used in many subsequent algorithms to stabilize learning.

First, collected experience in form of tuples $(S_t, A_t, R_{t+1}, S_{t+1})$ is stored in a replay memory. During an update step the algorithm samples a random mini-batch from the memory to update the weights according to Equation 2.26. Using replays of experience in form of mini-batches results in smoother gradients compared to the original Q-learning algorithm and reduces the variance between updates since the tuples in the batches are not correlated.

Secondly, the TD target in Equation 2.26 depends on the current parameters θ_k which can lead to instability of the parameters. To circumvent this problem, Mnih et al. [Mni+15] include target networks that are used to calculate the target only. This replacement decouples the current parameters θ_k from the target calculation since the target networks parameters θ_k^- are only updated after a certain number of Q-learning updates, C, by duplicating the current weights θ_k.

Lastly, DQN clips the TD error $\delta_t = R_{t+1} + \gamma \max_a \hat{Q}(S_{t+1}, a; \theta_k^-) - \hat{Q}(S_t, A_t; \theta_k)$ into the interval $[-1, 1]$.

Additional to those algorithmic changes, Mnih et al. [Mni+15] include a preprocessing step of the input frames by resizing them from $210 \times 160 \times 3$ pixels to $84 \times 84 \times 1$ pixels and by stacking multiple frames together to a single observation. The latter is necessary to

observe the full state as this is usually not observable in a single frame [SB18, p. 438]. A similar preprocessing procedure is used for the algorithm proposed later in this thesis.

2.2.5 *Policy Gradient Methods*

Policy gradient methods in contrast to aforementioned value-based methods skip the usage of value functions to choose an action and select actions directly without considering an estimated quality value of a state and/or action. Thus, the probability that an action a in state s at timestep t is taken can be written as $\pi_\theta(a|s) = \mathbb{P}\{A_t = a|S_t = s, \theta_t = \theta\}$ with θ as the parameterization of the policy. More specifically, θ can represent the parameters of a multi-layer perceptron (MLP) i.e. for low-dimensional observations or the parameters of a convolutional neural network (CNN) for pixel-based observations.

A policy π_θ is optimized so that it seeks to maximize the expected episodes return as scalar performance measure

$$J(\theta) = V^{\pi_\theta}(s_0) = \mathbb{E}_{\pi_\theta}[G(\tau)] = \sum_t \mathbb{E}_{\pi_\theta}[R_t] \tag{2.27}$$

To reach this goal the parameters are updated via stochastic gradient ascent:

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t) \tag{2.28}$$

$$\text{with} \quad \nabla J(\theta_t) = \nabla V^{\pi_{\theta_t}}(S_t) \tag{2.29}$$

with α as a beforehand defined learning rate. The *policy gradient theorem* [SB18, pp. 325, 334] proofs that for the derivative of the performance measure J the expression

$$\nabla J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_a \nabla \pi_\theta(a|S_t) Q^{\pi_\theta}(S_t, a)\right] \tag{2.30}$$

holds.

With $\nabla \pi = \pi \nabla \ln \pi$ and $\sum_a \pi_\theta(a|S_t) Q^{\pi_\theta}(S_t, a) = Q^{\pi_\theta}(S_t, A_t) = \mathbb{E}_{\pi_\theta}[G_t|S_t, A_t]$ this equation can be rewritten as

$$\nabla J(\theta) = \mathbb{E}_{\pi_\theta}[G_t \nabla \ln \pi_\theta(A_t|S_t)]. \tag{2.31}$$

With sampled trajectories this expression can be estimated, which changes the update rule Equation 2.28 to

$$\theta_{t+1} = \theta_t + \alpha G_t \nabla \ln \pi_{\theta_t}(A_t|S_t). \tag{2.32}$$

This is known as the REINFORCE [Wil92] update rule or Monte-Carlo policy gradient. Despite it has good convergence properties it is also affected by high variance. This can be addressed by subtracting a baseline $b(S_t)$ from the sampled return G [SB18, p. 329]:

$$\theta_{t+1} = \theta_t + \alpha \left(G_t - b(S_t)\right) \nabla \ln \pi_{\theta_t}(A_t|S_t), \tag{2.33}$$

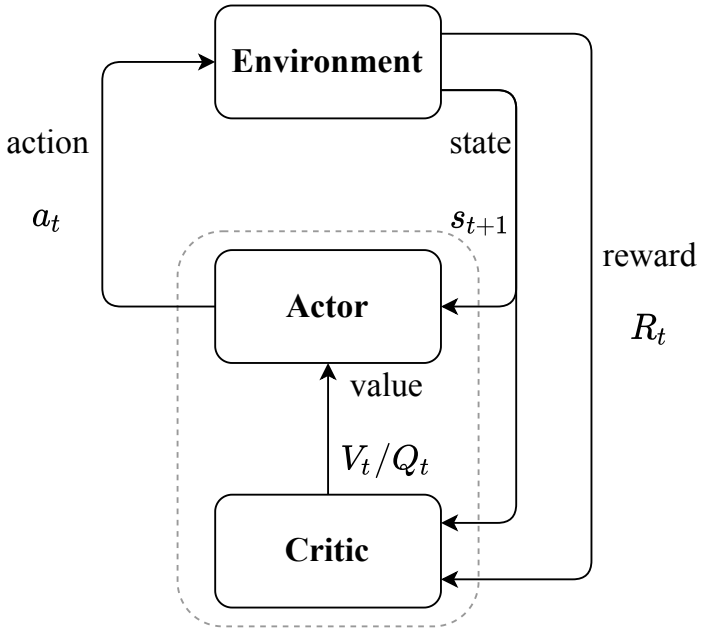


Figure 2.2: The closed-loop interaction with an Actor-Critic architecture. [Sze10, p. 63]

where $b(S_t)$ can be any function which is depended on S_t only. A good choice is an estimated state-value function $V^\pi(S_t)$ or the advantage $A^\pi(S_t, A_t)$ which is defined as the difference between the state-value and the action-value [PS08]:

$$A^\pi(S_t, A_t) = Q^\pi(S_t, A_t) - V^\pi(S_t) \tag{2.34}$$

2.2.5.1 *Actor-Critic Methods*

As the return G_t is the sampled return achieved at the end of a trajectory and because it is rarely known which exact state-action transitions yield this return, G_t is of high variance. On the other hand, TD methods can reduce the variance and use an estimated value function $Q^{\pi_\theta}(S_t, A_t)$ in Equation 2.33 instead. In this case not only a policy must be learned but also a value function. Those methods in which a value function is used as a baseline and in which it is used to update the value estimate for a state from the estimated values of subsequent states refer to *actor-critic* methods. The actor is the policy which predicts actions and the critic represents the approximated value function for bootstrapping. [SB18, p. 331]

2.2.6 *Maximum Entropy Reinforcement Learning*

The entropy $\mathcal{H}$ measures the average uncertainty in a random variable. This information-theoretical concept can be transferred to stochastic policies as an uncertainty measure of

the policy in a specific state. The entropy of a discrete stochastic policy π at timestep t can therefore be defined as

$$\mathcal{H}_t(\pi(\cdot|S_t)) = \mathbb{E}[-\log \pi(\cdot|S_t)] = -\sum_a \pi(a|S_t) \log \pi(a|S_t) \tag{2.35}$$

Such an entropy term can be used to ensure continuous exploration as the entropy of the policy during training could be measured and encourage it to take actions that yield higher entropy values. This regularizes the agent in exploiting current knowledge during learning. By including the entropy formulation as an additional reward, the value of a state changes to

$$V(S_t) = \sum_t \mathbb{E}_\pi \left[R_t - \alpha \log \pi(\cdot|S_t)\right] = \mathbb{E}_\pi \left[Q(S_t, A_t) - \alpha \log \pi(\cdot|S_t)\right] \tag{2.36}$$

As policy gradient methods strive to maximize this expected return (see Equation 2.27), the training objective changes to

$$J(\phi) = \mathbb{E}_{\pi_\phi} \left[Q(S_t, \cdot) - \alpha \log \pi_\phi(\cdot|S_t)\right] = D_{KL} \left[\exp Q(S_t, \cdot) \parallel \pi_\phi(\cdot|s_t)\right] \tag{2.37}$$

with α as a weight-factor for the importance of the entropy term $\mathcal{H}$ against the return. The optimal values of this factor depends upon the underlying MDP and his reward function R. Besides maximizing the reward in each state-action pair and thus exploiting already achieved knowledge, the objective in Equation 2.37 encourages the policy to maximize its own entropy. A goal that can be achieved by exploring more widely. Alternating between evaluation of $Q(S_t, A_t)$ and improvement of $\pi(A_t|S_t)$ results in a policy iteration scheme. [Haa+18b; Haa+18a]

2.2.6.1 *Soft Actor-Critic*

Continuous environments require an approximate form of policy iteration as the exact form can in most cases be calculated for tabular domains only [SB18, p. 73]. To this end, Haarnoja et al. [Haa+18b; Haa+18a] introduces the Soft Actor-Critic (SAC) algorithm, an maximum-entropy reinforcement learning algorithm that learns a parameterized stochastic policy $\pi_\phi(a|s)$ as well as two parameterized Q-functions $Q_{\theta_1}(S_t, A_t)$ and $Q_{\theta_2}(S_t, A_t)$ to estimate the state-action value $Q(S_t, A_t)$. Latter are trained by minimizing the Mean Squared Bellman Error (MSBE):

$$J(\theta_i) = \mathbb{E}\left[\frac{1}{2}\left(Q_{\theta_i}(S_t, A_t) - \left(R_t + \gamma\left(\min_{j=1,2} Q_{\bar{\theta}_j}(S_{t+1}, A_{t+1}) - \alpha \log \pi_\phi(A_{t+1}|S_{t+1})\right)\right)\right)^2\right] \tag{2.38}$$

with parameters $\bar{\theta}_1$ and $\bar{\theta}_2$ of the target Q-functions calculated from a moving average of the respective Q-function weights θ_1 and θ_2.

Using the resulting Q-functions from Equation 2.38, the state-action value $Q(S_t, A_t)$ can be estimated in objective in Equation 2.37 by modifying the original objective to:

$$J(\phi) = \mathbb{E}_{\pi_\phi}\left[\min_{i=1,2} Q_{\theta_i}(S_t, \cdot) - \alpha \log \pi_\phi(\cdot|S_t)\right] \tag{2.39}$$

This objective can be maximized via stochastic gradient ascent to calculate an update for the policy parameters ϕ. Since backpropagation of a stochastic policy is usually due to the sampling process mathematical infeasible, the algorithm makes use of the reparametrization trick: If the policy is defined as a normal probability density $\pi_\phi(a|s) = \mathcal{N}(\mu, \sigma)$ over actions[1], those actions can be calculated by

$$\begin{aligned} a &= \mu + \sigma \odot \varepsilon \\ \text{where} \quad & \varepsilon \sim \mathcal{N}(0, 1) \end{aligned} \tag{2.40}$$

This shifts the actual sampling process into the random variable ε and makes μ as well as σ deterministic allowing to backpropagate through the function approximator.

In addition, SAC learns the already addressed entropy weighting factor α as the entropy varies "[...] unpredictably both across tasks and during training as the policy becomes better." [Haa+18a]. Learning the factor results in a constrained optimization problem of the original maximum entropy objective in Equation 2.39:

$$\max_{\pi_{0:T}} \mathbb{E}_\pi \left[\sum_{t=0}^{T} R_t \right] \quad \text{s.t.} \quad \mathbb{E}_\pi \left[-\log \pi_t(A_t|S_t) \right] \geq \mathcal{H} \quad \forall t \tag{2.41}$$

with $\mathcal{H}$ as a desired minimum expected entropy. Haarnoja et al. [Haa+18a] shows through dual formulation of the problem that the optimal factor $\alpha_t^\star$ can be calculated with

$$\alpha_t^\star = \arg\min_{\alpha_t} \mathbb{E}_{\pi_t^\star} \left[-\alpha_t \log \pi_t^\star(A_t|S_t; \alpha_t) - \alpha_t \bar{\mathcal{H}} \right] \tag{2.42}$$

resulting in an additional objective function for the factor:

$$J(\alpha) = \mathbb{E}_{\pi_t} \left[-\alpha_t \log \pi_t(A_t|S_t) - \alpha_t \bar{\mathcal{H}} \right] \tag{2.43}$$

The method can be entirely trained off-policy alternating between experience collection with the current policy into a buffer and updating the Q-functions as well as the policy parameters with batch samples from the replay buffer. SAC is used in Chapter 5 as the underlying learning algorithm. Algorithm 1 describes SAC in pseudocode. [Haa+18b; Haa+18a]

2.3 SIMULATION TO REALITY TRANSFER (SIM-TO-REAL)

After specifying how general policies can be learned through the framework of RL, the question is tackled how the learned behavior can be used in the real-world if the algorithm learns in simulation only.

This section is supposed to introduce challenges of RL for robotics especially the transfer of learned behavior from simulation to a real robot. Therefore, the *sim-to-real gap* is defined. Furthermore, the theoretical foundations of promising approaches to bridge the sim-to-real gap like domain adaptation (DA) and DR are described.

1 more precisely, the underlying function approximator calculates mean μ and variance σ^2 of the distribution

Algorithm 1 Soft Actor-Critic [Haa+18a]

procedure SAC($\phi, \theta_1, \theta_2, \alpha$) ▷ Initial parameters
  $\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$ ▷ Initialize target network weights
  $\mathcal{D} \leftarrow \emptyset$ ▷ Initialize an empty replay buffer
  for each iteration **do**
    for each environment step **do**
      $A_t \sim \pi_\phi(A_t|S_t)$ ▷ Sample action from the policy
      $S_{t+1} \sim \mathcal{T}(S_{t+1}, R_t|S_t, A_t)$ ▷ Sample transition from the environment
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(S_t, A_t, R_t, S_{t+1})\}$ ▷ Store the transition in the replay pool
    end for
    for each gradient step **do**
      $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$ ▷ Update Q-function parameters
      $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$ ▷ Update policy weights
      $\alpha \leftarrow \alpha - \lambda_\alpha \hat{\nabla}_\alpha J_\alpha(\alpha)$ ▷ Adjust temperature
      $\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i$ for $i \in \{1, 2\}$ ▷ Update target network weights
    end for
  end for
  return Policy π_θ with $\phi, \theta_1, \theta_2, \alpha$
end procedure

2.3.1 *Challenges of Real-World Reinforcement Learning*

A fundamental challenge of reinforcement learning in the context of robotics is data availability and efficiency. Most RL algorithms need immense amounts of data to work properly and need to explore the action space sufficiently enough. Applications in the real-world are restricted to work safely and agents cannot perform safety-critical actions resulting in a limited action space. Besides, the data availability, the data collection process requires an equivalent amount of time. Because of those costs and the slow learning capabilities, the learned behaviors are limited in real-world applications. Dulac-Arnold et al. [DMH19] present a more general overview of the challenges of real-world reinforcement learning including but not limited to robotics:

1. Offline learning from pre-collected data of another control policy
2. Partially observable decision processes
3. Learning from limited data
4. High-dimensional continuous state and action spaces
5. Safety constraints that are not allowed to be violated
6. Unspecified, multi-objective or risk-sensitive reward functions
7. Explainable policies and actions
8. Real-time capabilities meaning that the agent is able to run at the frequency of the control system

9. Delays in the control system especially the delay in sensors, actuators and the reward signal

Training agents in a simulation seems promising for some of these challenges: Safety constraints can be violated in a simulator without consequences while at the same time the simulator can provide an infinite amount of data by-passing the necessity to learn from limited or pre-collected data. Furthermore, many different algorithmic design choices can be tested in a simulator, i.e. making it easier to find appropriate reward functions and policy architectures. But even with high fidelity simulators, the transfer from simulation to the real-world is challenging due to discrepancies between simulation and the real-world [Tob+17].

2.3.1.1 *Sim-to-Real gap*

The *sim-to-real* or *reality gap* describes inconsistencies between physical parameters of a simulation and the real-world (including but not limited to mass, density, light intensity, illuminance, friction, damping, glossiness, etc.). Furthermore, this gap is further widened by incorrect physical modeling since simulators are limited by physical assumptions causing that it is not able to compute the correct physical behavior. Thus, the environment of the simulation is only able to approximate the real transition model $\hat{\mathcal{T}}(s', r|s, a) \approx \mathcal{T}^{\star}(s', r|s, a)$. [Pen+18]

2.3.2 *Domain Adaptation*

Domain adaptation (DA) is one method to bridge the sim-to-real gap and related to transfer learning of deep neural networks (DNNs) [PC14]. In transfer learning a trained DNN is adapted such that it can perform on a different domain where data is less available. In general, a concept called fine-tuning is necessary to adapt the parameters of the DNN so that they can be used in the new domain. Similar, in reinforcement learning and specifically sim-to-real transfer, the reinforcement learning agent is trained in simulation (source domain) and gets data of the real robot (target domain) for adaptation. The principle assumes that the representations in the source domain share common characteristics in the target domain [Pen+18].

Tobin et al. [Tob+17] points out that DA for robotics is also related to *iterative learning control* where a update scheme is used to improve the dynamics model for the controller with real-world data. In DA however the controller/agent itself is improved since the underlying algorithm usually does not rely on a known dynamics model if it is model-free.

2.3.3 *Domain Randomization*

On the other hand, domain randomization (DR) can be seen as the zero-shot transfer learning version of sim-to-real transfer (even though DR and DA are not mutually exclusive). DR trains an agent solely with synthetic data and is able to transfer it to the real-world

Algorithm 2 Meta-Algorithm for DR in combination with a policy gradient algorithm

Require: learning rate α, DR hyperparameters ψ, policy gradient objective J

procedure DR(ψ)
  Randomly initialize parameters θ of policy π_θ
  while not done **do**
    Sample domain parameters $\xi \sim P_\psi$
    Collect trajectories τ using policy π_θ in domain e_ξ
    Compute adapted parameters θ with gradient ascent: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta, \tau)$
  end while
  return Policy π_θ
end procedure

without fine-tuning by adapting physical and visual parameters of the data during the learning process in the source domain. In the context of simulated environments, this shall introduce a higher variability in the experienced dynamics in the source domain which might be prominent in the target domain.

More formally, a simulation environment e_ξ parameterized by $\xi \in \mathbb{R}^D$ is defined, where D is the number of changeable parameters. In the context of DR, ξ represents a configuration of different physical parameters that can be changed in the simulation environment and is sampled from a distribution $\xi \sim P_\psi$. The probability distribution P_ψ can again be parameterized by a set of variables ψ, e.g. the mean and the variance of a normal distribution. The configuration ξ changes the system dynamics and/or appearance causing the approximate transition probabilities $\hat{\mathcal{T}}(s', r|s, a, \xi)$ of the environment to be depended on the configuration. [Pen+18]

Because of this parameterization the introduced policy gradient learning objective in Equation 2.27 is modified under the DR setting to

$$J(\theta) = \mathbb{E}_{\xi \sim P_\psi} \left[\mathbb{E}_{\pi_\theta, \tau \sim e_\xi} \left[G(\tau) \right] \right] \tag{2.44}$$

so that the policy π_θ seeks to maximize the expected return across the distribution P_ψ of dynamics models. This policy gradient optimization shall result in a generalization of the policy across the discrepancies between the different dynamics models. The assumption behind DR is that this generalization includes the real-world transition model $\mathcal{T}^\star$. [Pen+18]

Algorithm 2 presents the general scheme of learning a policy with DR.

2.3.3.1 *Automatic Domain Randomization*

Automatic Domain Randomization (ADR) [Akk+19] is an recent algorithm which combines DR with curriculum learning. The algorithm automatically generates new and harder environments by increasing the parameters of the distribution P_ψ for domain randomization based on the average performance of the agent in the environment so far. Thus it creates an endless curriculum.

In each training iteration one parameter ξ_i is randomly chosen to take one boundary value of its distribution. The residual parameters are sampled from the distribution P_ψ.

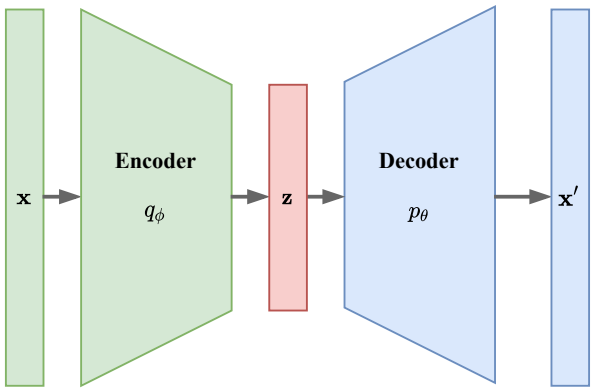


Figure 2.3: The general structure of an AE: The parametrized encoder q_ϕ maps the input x to a lower dimensional feature vector z whereas the decoder p_θ maps this vector back to the input space. Both encoder and decoder are trained jointly by minimizing the error between input x and reconstruction x'. Since the decoder p_θ tries to reconstruct the input x using the lower dimensional feature vector z only, this vector must be a compressed representation of x.

The agent is then evaluated on the environment. If the agent's average performance is sufficient enough after a number of timesteps, the distribution parameters ψ can be adapted to include harder or simpler scenarios. These parameters are simply updated with a step size hyperparameter Δ. If the performance exceeds predefined thresholds t_H or t_L, the parameters are increased if t_H was reached or decreased if t_L was reached. No parameters of the agent are updated during the execution of the algorithm. Instead, policy training and ADR evaluation alternate.

2.4 REPRESENTATION LEARNING

In machine learning it is often of interest to train a model that learns a representation of the observed inputs as appropriate representations make learning subsequent tasks easier [GBC16; BCV13]. Particularly, it might be useful to learn representations without the necessity of using many labels i.e. in an semi-, weakly- or unsupervised manner. Therefore, this section is about representation learning. Especially, the first part introduces the unsupervised representation learning method named autoencoder (AE) while the second part describes the probabilistic version, the variational autoencoder (VAE). How enhanced versions of aforementioned methods can be used to learn representations that disentangle the underlying FoV in the data is described at last.

2.4.1 *Autoencoders*

A family of methods that learn representations unsupervised is named autoencoder (AE). The basic task of an AE is to reconstruct a given input $x \in \mathcal{X}$ using an encoder q_ϕ and a decoder p_θ parametrized by ϕ and ψ. The encoder maps from the input space $\mathcal{X}$ to a

(perhaps lower dimensional) feature space $\mathcal{Z}$ whereas the decoder maps from this feature space back to the input space $\mathcal{X}$. Specifically, given an input x the encoder computes the output $z = q_\phi(x)$. Using this feature vector the decoder maps z back to the input space with $x' = p_\theta(z) = p_\theta(q_\phi(x))$. Both encoder and decoder are trained jointly by minimizing the error between input x and reconstruction x'. Usually this is done by minimizing the mean squared error (MSE) between inputs and reconstructions of a given dataset $x_k \in \mathcal{D}$:

$$\min J(\mathcal{D}) = \min \sum_k \|x_k - p_\theta(q_\phi(x_k))\|^2 \tag{2.45}$$

This optimization is usually performed using stochastic gradient descent (SGD) by training a multilayer perceptron (MLP) or convolutional neural network (CNN). The structure of an AE is illustrated in Figure 2.3. [GBC16, pp. 493–494]

By using fewer dimensions in the feature space $\mathcal{Z}$ than the dimensions of the input space $\mathcal{X}$ an AE can be used as dimensionality reduction method [GBC16, p. 515].

2.4.2 *Variational Autoencoders*

A variational autoencoder (VAE) [KW14] is a deep latent variable model that is a probabilistic version of the deterministic AE as the feature vector z is sampled from a parametrized distribution $q_\phi(z|x)$ instead of being a result of a deterministic calculation. Using a VAE instead of a deterministic AE allows to generate new data.

A latent variable model includes model variables z that are part of the model i.e. a DNN but that are not observed. Given observations x to be modeled, a latent variable model is described by a joint distribution $p_\theta(x, z)$ in the unconditional case and as a conditional joint distribution $p_\theta(x, z|y)$ in the case if the model condition observation x and latent variable z on a context y. The distributions are parametrized by θ. [KW+19]

Marginalizing the distribution over the latent variables yields the *marginal likelihood*:

$$\begin{aligned} p_\theta(x) &= \int p_\theta(x, z) dz \\ &= \int p_\theta(x|z) p_\theta(z) dz \\ &= \int p_\theta(z|x) p_\theta(x) dx \end{aligned} \tag{2.46}$$

Problematically, using DNNs makes the calculation of the marginal likelihood intractable as $p(x|z)$ and $p(z|x)$ are intractable due to the complicated, nonlinear structures of neural networks [KW14].

Nevertheless, approximations are possible by incorporating a *parametric inference model* $q_\phi(z|x)$ with variational parameters ϕ. In traditional variational inference (VI) methods, parameters are typically not shared over datapoints. Instead those methods optimize parameters for each datapoint separately. However, in VAEs each datapoint shares the variational parameters resulting in *amortized variational inference* [GG14]. The optimization

of these parameters ensures $q_\phi(z|x) \approx p_\theta(z|x)$ by maximizing the log-likelihood of observing sample x:

$$\begin{aligned}
\log p_\theta(x) &= \log \int p_\theta(x,z) dz \\
&= \log \int p_\theta(x,z) \frac{q_\phi(z|x)}{q_\phi(z|x)} dz \\
&= \log \mathbb{E}_{q_\phi(z|x)} \left[\frac{p_\theta(x,z)}{q_\phi(z|x)} \right] \\
&\geq \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)} \right] ; \text{ Jensen's inequality} \\
&= \underbrace{\mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x,z) \right] + \mathcal{H}(q_\phi(z|x))}_{ELBO} \\
&= \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z) + \log p_\theta(z) \right] + \mathcal{H}(q_\phi(z|x)) \\
&= \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z) \right] + \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p_\theta(z)}{q_\phi(z|x)} \right] \\
&= \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z) \right] - D_{KL} \left(q_\phi(z|x) \parallel p_\theta(z) \right)
\end{aligned} \tag{2.47}$$

The resulting functional is known as evidence lower bound (ELBO) since it is a lower bound on the log-likelihood of the data $\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z) \right] - D_{KL} \left(q_\phi(z|x) \parallel p_\theta(z) \right)$. The difference between the left and the right hand side of the inequality Equation 2.47 is the KL divergence $D_{KL} \left(q_\phi(z|x) \parallel p_\theta(z|x) \right)$ as can be shown by directly reformulating the original objective $q_\phi(z|x) \approx p_\theta(z|x)$:

$$\begin{aligned}
D_{KL} \left(q_\phi(z|x) \parallel p_\theta(z|x) \right) &= \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)} \right] \\
&= \mathbb{E}_{q_\phi(z|x)} \left[\log q_\phi(z|x) \right] - \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x,z) \right] + \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x) \right] \\
&= \underbrace{-\mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x,z) \right] - \mathcal{H}(q_\phi(z|x))}_{-ELBO} + \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x) \right]
\end{aligned} \tag{2.48}$$

Since the log-probability of the right hand side does not depend on the posterior q, the result indicates that maximizing the ELBO is equal to minimizing the KL divergence. Latter is minimal if the posteriors q and p are equal. [Jor+99; BKM17; KW+19]

Maximizing the ELBO yields a better generative model as the optimization approximately maximizes the marginal likelihood $p_\theta(x)$ while at the same time it minimizes the KL-divergence between the approximate posterior distribution $q_\phi(z|x)$ and the true posterior $p_\theta(z|x)$. [KW+19]

Given a dataset $\mathcal{D}$ with independently and identically distributed (i.i.d.) observations $x \in \mathcal{D}$ it is now possible to jointly optimize the parameters ϕ and θ using stochastic gradient descent (SGD). The optimization minimizes the negative ELBO iteratively using minibatches from the dataset. Calculating an estimate of the ELBO includes calculating the

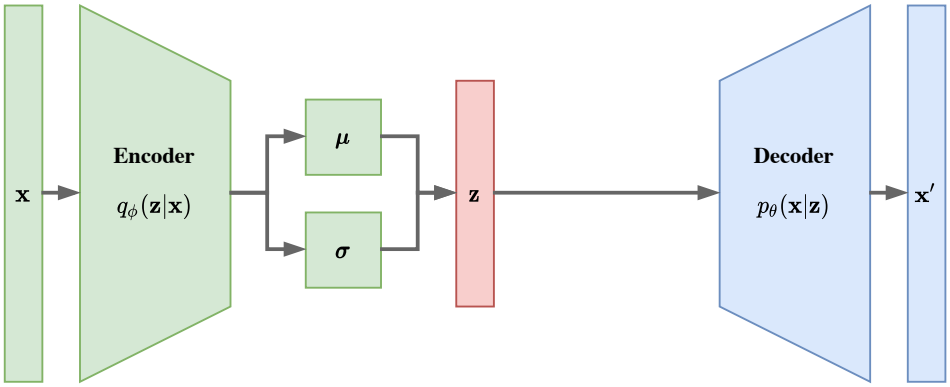


Figure 2.4: The general structure of a VAE: Similar to an AE, a VAE maps the input x from the input space $\mathcal{X}$ to a lower dimensional feature vector z that is sampled from a distribution $q_\phi(z|x)$ with parameters ϕ. The decoder $p_\theta(x|z)$ then maps the sample z back to the feature space $\mathcal{X}$. The posterior distribution q_ϕ usually takes the form of an isotropic Gaussian with mean μ and standard deviation σ.

density $q_\phi(z|x)$ in order to sample $z \sim q_\phi(z|x)$. A common choice is an isotropic Gaussian posterior:

$$q_\phi(z|x) = \mathcal{N}(z; \mu, \text{diag}(\sigma^2)) \tag{2.49}$$

The distribution parameters μ and σ are the outputs of the DNN with parameters ϕ. This choice also determines the prior $p(z)$ to take the form of an isotropic unit Gaussian:

$$p(z) = \mathcal{N}(0, I) \tag{2.50}$$

Using SGD the procedure computes a Monte Carlo (MC) estimate w.r.t. parameters ϕ. The gradient of this estimator is intractable due to the expectation of the ELBO over the distribution q_ϕ. By reparametrizing the ELBO, a gradient of the MC expectation estimate w.r.t. ϕ can be calculated. This is similar to the reparametrization of the policy output in Equation 2.40 if a isotropic Gaussian posterior is used: [KW14]

$$\begin{aligned} z &= \mu + \sigma \odot \varepsilon \\ \text{where} \quad \varepsilon &\sim \mathcal{N}(0, 1) \end{aligned} \tag{2.51}$$

A VAE can learn representations of given inputs in an unsupervised manner and can be used for downstream tasks like reinforcement learning [Nai+18; HS18; Haf+19; Haf+20; Yar+21].

2.4.3 *Disentangled Representations*

This section serves as an overview of disentangled representations and how generative factors of data can be learned in an unsupervised manner using the framework of VAEs. Therefore, the concept of disentangled representation learning is defined and criteria that

those representations shall obey are established based on recent literature. In order to train generative models to learn disentangled representations, different methods based on VAEs are introduced.

2.4.3.1 *Disentangle Explanatory Factors of Variation*

For an AI to act truly intelligent, it shall understand, recognize and keep track of the world around [RN09, p. 1045]. Bengio et al. [BCV13] assumes that this can only be accomplished if it can discover and disentangled the underlying explanatory factors hidden in the observations. Those explanatory factors are known as factors of variation (FoV). Therefore, it is assumed that observations are generated by a two-step generative process: A set of factors of variation $S \in \mathbb{R}^n$ are sampled from a distribution $P(S)$ with probability mass function $p(s)$. With these factors the actual observation $X \in \mathbb{R}^m$ can be sampled from the conditional distribution $p(x|s)$. [Loc20]

According to Bengio et al. [BCV13] the goal is to learn a representation $z \in \mathbb{R}^n$ of which distinct dimensions are sensitive to changes of specific FoV while being invariant to others. This is the intuition behind *disentangled representations*. Although no clear definition of disentangled representations has yet been agreed upon, previous approaches have in common that they align to intuitive characteristics of disentangled representations [Hig+18]. Those can be described by three criteria [EW18; RM18]: *modularity*, *compactness* and *explicitness*.

- Modularity denotes that each dimension of the representation should encode information about one FoV only.
- In a compact representation a FoV is encoded by a single dimension or by a small number of dimensions.
- Explicitness expresses whether there is a simple transformation (e.g. linear) between the dimension and the encoded FoV.

However, it is not generally agreed upon if the latter criterion is required for a representation to be disentangled [Hig+18].

To ensure disentanglement of the FoV in the learned representations, disentanglement should be measurable. Problematically, due to different definitions which share a common intuition only, there exist several evaluation metrics inconsistent to each other. Furthermore, those metrics assume access to ground-truth factors of the evaluation data.

A method to inspect the representation visually without requiring labels are latent traversals. Given a sample x, the fully trained VAE can encode x into a representation z. This representation can now be evaluated by freezing all dimensions of it except one, i.e. z_i. It is now possible to traverse the dimension z_i and to generate reconstructions x' based on the current value of z_i and the residual frozen dimensions of z. If z_i disentangles a single FoV this should be visible in the reconstructions as only a distinct factor should change between the reconstructions. [Hig+17a]

2.4.3.2 *Unsupervised Disentangled Representation Learning*

To learn disentangled representations, variants of variational autoencoders [KW14] were proposed as the parameters of a VAE learns to mimic hidden random processes that are assumed to be the FoV [KW14].

Higgins et al. [Hig+17a] adapts the original VAE objective by introducing an additional hyperparameter β in front of the KL divergence term of the ELBO:

$$\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - \beta D_{KL}\left(q_\phi(z|x) \parallel p_\theta(z)\right) \tag{2.52}$$

Setting $\beta > 1$ results in a stronger pressure on the posterior q_ϕ as the encoder is forced to match the factorized Gaussian prior p_θ. This constrains the capacity of the latent bottleneck and encourages the encoder to learn disentangled representations. The intuition behind this is that a disentangled representation is the most efficient representation of the data. By constraining the capacity of the bottleneck the objective enforces the encoder to adopt this most efficient encoding. [Hig+17a]

Burgess et al. [Bur+18] compares the β-VAE loss in Equation 2.52 to the information bottleneck method (see Section 2.1.1):

$$I(\tilde{X}, X) - \beta I(\tilde{X}, Y) \tag{2.53}$$

Considering this principle, the KL divergence of the β-VAE objective can be seen as an upper bound on the amount of information which can be transmitted through the latent representation. The capacity to transmit additional information through the representation is 0 if the KL divergence is 0. The ELBO encourages the encoder to embed points nearby in the latent space. The posterior $q_\phi(z|x)$ shall match the factorized Gaussian prior and can only reduce the mean values or broaden the variances to do so. But both results in a greater degree of overlap between the approximate posterior distributions. This will go on the costs of the log-likelihood of the objective as samples from different true posteriors are not discriminable in this case. Therefore, a strong bottleneck pressure (as it is the case for the β-VAE) forces the posterior to map similar points as much as possible nearby while preventing too much overlap of the Gaussians to preserve discriminability. Burgess et al. [Bur+18] key hypothesis is that this is only feasible if the posterior encodes components that contributes differently to the log-likelihood of the objective. Therefore, the capacity-constrained posterior only encodes information that contribute to the mentioned log-likelihood sufficient enough. For instance, encoding information about the color of a large object in an image contributes more to the objective than the color of a small object. [Bur+18]

The authors propose to improve disentanglement by increasing the capacity of the bottleneck with a target KL term C:

$$\mathcal{L}(\theta, \phi; x, z, C) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \gamma \left| D_{KL}(q_\phi(z|x) \| \; p(z)) - C \right| \tag{2.54}$$

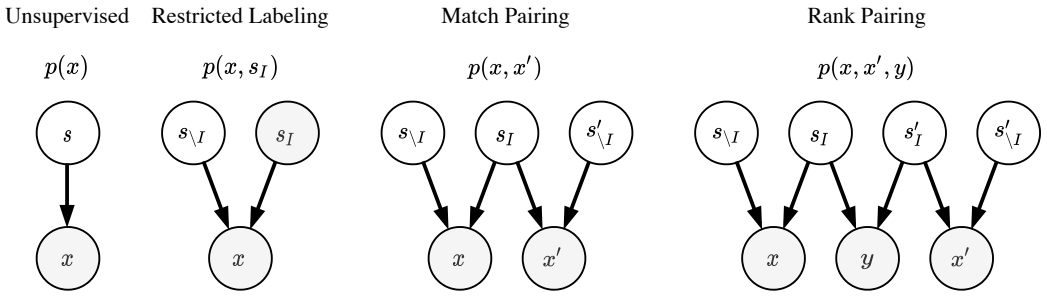


Figure 2.5: Types of weak supervision: In restricted labeling the algorithm observes samples x and labeled factors s_I for a distinct subset of factors S_I. In match pairing samples x and x' share a number of known factors I. In rank pairing on the other hand the samples do not have to share factors as only the relation y between the factor instances s_I and $s_{\setminus I}$ is observed by the agent. [Shu+20]

Increasing this target KL term during training reduces the pressure on the posterior $q_\phi(z|x)$ to match the unit Gaussian prior $p(z)$ and releases additional capacity in the representation. This should encourage the algorithm to encode additional information into the representation that did not improve the log-likelihood $\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$ significant enough so far. [Bur+18]

2.4.3.3 *Weakly-Supervised Disentangled Representation Learning*

Recently, Locatello et al. [Loc+19] showed that it is impossible to learn disentangled representations in an unsupervised manner without inductive biases on the model or data. For this reason, weakly-supervised algorithms for learning disentangled representations were proposed [BTN18; Hos19; Loc+20]. Shu et al. [Shu+20] categorizes the main different approaches for weakly-supervised disentanglement and describes additional criteria for disentangled representations similar to those of Eastwood et al. [EW18] and Ridgeway et al. [RM18]. The main methods using weak supervision partition the FoV in two subsets $S = (S_I, S_{\setminus I})$ with $|S| = n$. These methods either observe the subset of factors S_I as additional input to samples x from X (*restricted labeling*) or the factors are shared over multiple samples. The latter case can be divided into *match pairing* and *rank pairing*. While in match pairing the samples x and x' share a known subset of factors I (i.e. $s_i = s'_i \; \forall i \in I$), in rank pairing only the ratio between observations x and x' is known via an indicator variable y (i.e. $s_i \geq s'_i \; \forall i \in I$). An overview is depicted in Figure 2.5. [Shu+20]

The Group-based Variational Autoencoder (GVAE) of Hosoya [Hos19] and the Adaptive Group-based Variational Autoencoder (Ada-GVAE) of Locatello et al. [Loc+20] use match pairing as weak supervision signal. Furthermore, both methods assume that the number of factors $k = |\setminus I|$ in which the samples differ is constant during training.

Given samples x and x' that share a known number of factors $n - k = |I|$ it is possible to enforce structure in the approximated posterior $q_\phi(z|x)$ as the true posterior follows:

$$p(z_i|x) = p(z_i|x') \quad \forall i \in I, \tag{2.55}$$

$$p(z_i|x) \neq p(z_i|x') \quad else. \tag{2.56}$$

If the indices I are known it is possible to enforce the constraint in Equation 2.55 during training, the approximate posteriors can be averaged for each of both samples:

$$\tilde{q}_\phi(z_i|x) = \text{avg}\left(q_\phi(z_i|x), q_\phi(z_i|x')\right) \quad \forall i \in I \tag{2.57}$$

$$\tilde{q}_\phi(z_i|x) = q_\phi(z_i|x) \quad else. \tag{2.58}$$

The same applies for sample x' whereby the averaging of the posterior distribution in Equation 2.57 is defined by:

$$\hat{\mu} = \frac{1}{2}(\mu + \mu'), \quad \hat{\sigma} = \frac{1}{2}(\sigma + \sigma') \tag{2.59}$$

This results in an combined β-VAE objective:

$$\begin{aligned}\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] + \mathbb{E}_{q_\phi(z|x')}\left[\log p_\theta(x'|z)\right] \\ - \beta D_{KL}\left(\tilde{q}_\phi(z|x) \parallel p_\theta(z)\right) - \beta D_{KL}\left(\tilde{q}_\phi(z|x') \parallel p_\theta(z)\right)\end{aligned} \tag{2.60}$$

In contrast to the work of Hosoya [Hos19], the algorithm of Locatello et al. [Loc+20] does not observe the indices of shared factors I but estimates them for each set of samples using the KL divergence of the posterior distributions of the samples and choosing those $n - k$ indices whose KL divergence is smallest. Locatello et al. [Loc+20] proofed also that a learned posterior $q(z|x)$ is disentangled given unlimited samples x and x' that comply to aforementioned assumptions. Therefore, weakly-supervised methods that use data which align to the concept of match pairing can be used to learn disentangled representations reliably. [Hos19; Loc+20]

3

RELATED WORK

In the last years, the topic of vision-based reinforcement learning was very prominent but mostly in simulation only. Due to the reality gap in simulation and the large amount of training data needed in the real-world, most training is performed on low-dimensional states. On the other hand, disentangled representation learning is currently a task which is evaluated on toy data only and mostly not used for downstream task, especially RL.

In this chapter, related work with respect to two similar but not mutually exclusive directions is described: General vision-based RL for robotics and simulation to reality transfer which includes but is not limited to domain randomization. Furthermore, recent methods combining disentangled representations and RL are reviewed.

3.1 VISION-BASED REINFORCEMENT LEARNING

A first major attempt on vision-based RL and a catalyst for RL research is the work of Mnih et al. [Mni+13]. They show that it is possible to train agents with Q-Learning solely on high dimensional pixel-based observations using CNNs, replay buffers for off-policy training to prevent catastrophic forgetting and careful tuning of the image preprocessing. During the collection of the four frames, the most recent action is repeated in a row.

Since this first success on images, vision-based methods significantly underperformed compared to state-based methods in terms of efficiency and overall performance. Regarding performance, the original deep Q-learning algorithm was updated with improvements including double Q-learning [Wan+16; HGS16; Hes+18]. In recent time, the efficiency of vision-based RL methods gained interest. Yarats et al. [Yar+21] combines SAC with autoencoders and observed that using VAEs prevent the agent from learning more efficient. The authors depict that using deterministic AEs is a better solution. Laskin et al. [LSA20] combine contrastive learning with RL to jointly train a vision-module and a policy by using a contrastive auxiliary task. For the contrastive loss calculation, image pairs are generated by applying random shifts on an original, larger image. Similar, Stooke et al. [Sto+21] use a contrastive loss between observations over multiple timesteps and random shifts. Yarats et al. [Yar+21] use simple data augmentation methods on top of standard RL policies by regularizing the action-value functions of the critic with those image augmentations.

3.2 VISION-BASED ROBOT LEARNING

In addition to describing methods on vision-based RL with evaluation in simulation only, this section focuses on learning-based control for robots based on high-dimensional pixel inputs. For instance, there exist numerous different approaches trying to learn grasping policies and therefore focusing on relatively slow dynamical systems. One of the first successful attempts of vision-based RL on robots is the work of Levine et al. [Lev+16]. The authors train a policy based on a convolutional neural network end-to-end via a guided policy search method. The network takes $240 \times 240 \times 3$ raw image observations as well as the current robot configuration as input and outputs the needed motor torques. Unlike in the work of Mnih et al. [Mni+13] no frame stacking is used. The authors conclude that the method's generalization ability is moderate only so that the policy cannot perform in settings with disturbances.

Instead of using one robot, multiple robots can be used simultaneously to train a single policy on raw images only by applying a scalable distributed update scheme as seen in the work of Kalashnikov et al. [Kal+18] and Levine et al. [Lev+18]. Such a learned policy is robust to unseen objects and can respond dynamically to upcoming disturbances. But even if the algorithm does not need much human intervention, it still needs much real-world data to accomplish this behavior which is expensive, especially if using multiple robots. In contrast, Zhu et al. [Zhu+20] show how to efficiently train a reinforcement learning agent in real-world scenarios on a single robot without human intervention. They propose a learning pipeline in which the user provides goal images of successful scenes that the robot should achieve so that no reward function must be specified. Zhu et al. [Zhu+20] combine learning of a task controller and learning of a disturbance policy that should ensure robustness against arbitrary initial states. However, the environment must be designed in such a way that safe exploration is possible without failure states. Therefore, training in simulation might be more promising.

Imitation learning [Zhu+18; Fin+17] can also be used to train real-world policies from vision by using demonstrations. Those needed demonstrations can be safely collected by a human or another controller. But while collection by humans is expensive, it is possible that another controller may not even exist. Furthermore, it is not clear if the learned behavior is robust or generalizes at all, or if it overfits the demonstration data heavily.

The integration of standard control methods (i.e. model predictive control) in combination with vision-based learning is also possible. Finn et al. [FL17] use a video-prediction model for a model predictive controller with a goal-specifying image as its first input. Unfortunately, the authors do not examine how the method generalizes and if it is robust to unseen situations except there is much room for improvement. In the work of Zeng et al. [Zen+19] a controller learns to grasp and to throw objects in different positioned boxes by planning with an analytical model and additionally learns systems dynamics which cannot be modeled easily (i.e. aerodynamics).

3.3 SIM-TO-REAL & DOMAIN RANDOMIZATION

The reality gap or also called sim-to-real gap was already introduced in Section 2.3 of Chapter 2. With growing interest in robotics and real-world reinforcement learning in the past years, the problem of closing the sim-to-real gap has gained momentum too. In classical control, the sim-to-real gap is commonly bridged by system identification [Lju99]. But for more complex problems, system identification is not a general solution due to uncertainties, errors in modeling, the necessity of prior knowledge, etc. [Lju91].

Similar to system identification, other methods propose to utilize real-world data to assist a simulation of the system during the training pipeline for a successful sim-to-real transfer. For instance, Hwangbo et al. [Hwa+19] learn an actuator network on a minimal set of real-world data to map joint position errors to joint torques. The network is then used to assist a rigid body simulation during training of a robust TRPO policy. The complete method achieves robustness to different situations (i.e. high-speed locomotion, recovery from a fall, etc.) by learning multiple control policies for different behaviors in a curriculum. The authors argue that since the policies are trained on simulated robots with randomized inertial properties and noisy velocity measurements, it is robust against unmodeled effects. This method is one of few examples we know which controls a fast dynamical system at high frequency (> 100 Hz) without repeating actions or aggregating multiple observations. However, it is not trained with visual data but with low-dimensional states only and needs real-world data.

Other methods incorporate domain adaptation to bridge the reality gap, i.e. by including additional layers during the fine-tuning process [Rus+17], by using efficient model-based reinforcement learning where the model acts as a prior which is adapted to the unseen dynamics [FLA16] or by using generative networks to transform synthetic images into more realistic ones to reduce the number of required real-world images [Bou+18]. Many other approaches exist for sim-to-real transfer using domain adaptation (i.e. [Hof+18; Chr+16; Tze+20; CH15]). However, they all have in common that they require additional data of the target domain making them unfeasible for zero-shot sim-to-real transfer.

3.3.1 *Domain Randomization*

Domain randomization as a method for sim-to-real transfer and acquiring robustness to uncertainty, has been gaining momentum in recent time.

DR for visual aspects was introduced by Tobin et al. [Tob+17] by sampling visual conditions uniformly while learning to detect and to pick-up target objects between distractor objects. The authors show that with randomization a realistically looking simulation is not necessary for a successful sim-to-real transfer. Furthermore, the learned behavior can be robust against unseen situations. A more robust version of visual domain randomization is used in in the work of Slaoui et al. [Sla+19] by combining DR with regularization to achieve invariance across different domains.

In contrast to the randomization of visual aspects, Peng et al. [Pen+18] use dynamics randomization in conjunction with memory-based policies and a sparse reward function without carefully calibrating the simulation beforehand to transfer policies from simulation to the real world. The authors argue that system identification is not necessary in their case because of the memory-based property of the policy which should infer the system dynamics by the internal memory. They show that policies which are parameterized with a LSTM layer and trained with domain/dynamics randomization are indeed better than policies with simple feed-forward layers only. Furthermore, they demonstrate that recurrent policies are robust against perturbations in the real environment. But it is unclear if this applies to vision-based tasks equally as the authors only use low-dimensional states as input.

Tan et al. [Tan+18] train quadruped robots in simulation to run forwards as fast as possible and transfer these learned policies to the real world without fine-tuning them. They determine system identification as an important part to narrow the reality gap. In addition to improving the fidelity of the simulation, they also try to achieve more robust policies with the following extensions: Domain randomization, perturbation forces, and a compact design of the observation space. Tan et al. compare the achieved trajectories with those of handtuned controllers and discovered that the learned policies are as fast as the handtuned ones but consume significantly less power. Their results show that a good simulation is crucial since controllers without the extensions perform poorly in the real world. Seperating the three methods, they conclude that although domain randomization leads to more robust controllers the method also leads to supoptimal ones. Similar applies to perturbation forces[1] as this method is seen by the authors as some type of domain randomization. Furthermore, their results reveal that for sim-to-real transfer a large observation space is not beneficial although it is good in simulation. However, a small observation space might not be available in every task and visual information might be essential for solving specific tasks. Therefore, further research on this is still necessary.

Apart from examining the benefit of DR as it is done by Peng et al. [Pen+18] and Tan et al. [Tan+18], a better parameterization of the randomization distributions has been receiving much attention in recent time. I.e. real data can be exploited to improve the parameterization of the randomization distributions. In the work of Ruiz et al. [RSC18], the algorithm generates a synthetic dataset using a randomized simulation. This dataset is then used to train a main task model (MTM) such that it achieves maximum accuracy on the validation set of real-world data. The reward of the MTM on the validation set is then used to update a policy that sets the parameters of the randomized simulation. Muratore et al. [Mur+20] combine DR and Bayesian optimization to tune the parameters of a simulated environment by sampling data from the real-world target environment. Their goal is to learn a policy that maximizes the return in the real-world by searching over the space of source domain distribution parameters using Bayesian optimization. But instead of using high-dimensional images they use the state-space of the system only.

1 In the original paper these are called adversarial attacks [Pin+17].

Mehta et al. [Meh+20] and Chebotar et al. [Che+19] use a discrepancy measure between trajectories to teach a model how to adjust the DR parameters. Mehta et al. [Meh+20] measures the discrepancy between randomized trajectories and a simulated baseline trajectory whereas Chebotar et al. [Che+19] use a weighted discrepancy measure between randomized trajectories and real-world trajectories for which the current policy is deployed on the real robot. Those discrepancies are then used to optimize the parameters of the DR parameter model to minimize the discrepancy in the next step. Even if training with integrated real-data show robust behaviors, those methods still leave the possibility to unsafe actions during the real-world deployment. Therefore, more methods for training in simulation only are still needed.

Learning in-hand manipulation of a cube in simulation only is accomplished in the work of Andrychowicz et al. [And+19] by randomizing most physical and visual parameters of the system uniformly during training. The authors conduct an ablation study of important randomizations (including physics, vision, and noise) and discovere that, although a policy trained on full randomization converges much slower, it also leads to the best performance in the real world. Furthermore, the majority of training time is spent making the policy robust to different physical dynamics whereas other randomizations are less time consuming. Akkaya et al. [Akk+19] continued the previous work accomplished by Andrychowicz et al. [And+19]. Instead of manipulating a cube, the task is changed to learn the dexterous part of solving a Rubik's cube. They apply the algorithm described in Chapter 2 to generate a curriculum of growing randomizations. Similar to Peng et al. [Pen+18], Akkaya et al. [Akk+19] and Andrychowicz et al. [And+19] use memory-based policies to yield robust and adapting behaviors to different disturbances and occlusions which are not used during training. Problematically, both methods require a lot of computing power making them unfeasible for mass usage.

One advantage of simulation is being able to use information that is not available or difficult to obtain in the real-world. Pinto et al. [Pin+18] use this advantage to train an actor-critic policy on synthetic depth images which are generated using DR. While the actor uses images only, the critic additionally exploits the full low-dimensional configuration of the simulated robot during training in simulation. This is possible as the critic is not needed anymore after training.

Another approach by Jeong et al. [Jeo+19], which exploit the advantage of information in simulation, use a state-based agent to train a vision-based agent via behavior cloning and DR. But instead of using DR alone for the sim-to-real transfer, an additional self-supervised domain adaptation step is included which needs additional real-world data.

However, most of the described methods lack interpretability. Furthermore, none of those methods can verify that the learned internal representations transfer to the target domain successfully beforehand and share no insight during evaluation on why the method might fail.

3.4 DISENTANGLED REPRESENTATIONS FOR REINFORCEMENT LEARNING

So far, disentangled representations have mostly been used on toy datasets only and especially without further downstream tasks like RL. In connection with reinforcement learning there is only a small number of publications.

Higgins et al. [Hig+17b] combine β-VAE with policy learning but split the training phases. Based on a pre-collected dataset, by using a pre-determined policy, a β-VAE model is trained on the dataset in a *Learn to See* phase. In the following *Learn to Act* phase, the policy learns by interacting with the environment using the concatenated latent representations of stacks of frames provided by the freezed β-VAE. The authors evaluate the proposed agent named DisentAngled Representation Learning Agent (DARLA) on a simulated benchmark and on a sim-to-real reaching task. In the latter setting, the method is able to successfully transfer to the real robot. Problematically, the method requires separated training procedures. Especially the training of the vision modules require already collected observations from an existing policy that might not be available.

Curious Object-Based seaRch Agent (COBRA) is a model-based RL algorithm utilizing a vision module, a transition model, an exploration policy and a reward predictor. In an unsupervised exploration phase the first three components are trained without learning a task while the reward predictor is learned in the task phase. The vision module uses MoNet [Bur+19] a disentanglement architecture based on an autoregressive VAE which decomposes the objects visible in the input frames into disentangled slots. Each slot describes meaningful properties like color, position, etc. of the object. The union z of those slots represent the full scene. The transition model predicts the next union z_{t+1} on the basis of the current slots z_t and action a_t. The final agent is robust to task-irrelevant perturbations but since the evaluation only takes place in simulation on simple 2-dimensional environments, it is not clear if the agent can be used for much more complicated task in a sim-to-real setting.

An attempt to combine policy learning with disentangled representation learning is the work in Yang et al. [Yan+19]. They extend standard policy learning losses with additional auxiliary losses to learn an action-conditional VAE jointly with a policy. The VAE encodes the current observation as normal while the decoder predicts the next observation conditioned on the action sampled from the policy. The policy predicts the action on the basis of the mean μ and the standard deviation σ of the encoded Gaussian. Due to the dependence of the decoder on the current action, the latent traversals shown indicate that it learns to disentangle environment factors and to predict changes due to the input action. The method does not utilize synthetic environment changes as incorporated by DR. Therefore, using this method with DR for a sim-to-real transfer would be interesting.

Wulfmeier et al. [Wul+20] study different representations for simulated robots on the questions if and how such representations support exploration and overall behavior in (multi-task) reinforcement learning. The architectures being investigated include Transporters, MoNet, β-VAE and a standard VAE. They are pretrained similar as DARLA but use additional proprioceptive robot states as policy input. The authors show that disentangled models manage to include all relevant information to solve the task quickly while entangled models fail to achieve this. Furthermore, they argue disentanglement is especially helpful for

auxiliary tasks since those allow the agent to explore the state space more efficiently. They reveal that representations with a medium and small number of dimensions may shrink the observability of the task since information get lost during encoding. For representations with only a few dimensions, entangled models can perform better since those can incorporate more information into smaller representations.

In another large-scale study, Träuble et al. [Trä+21] investigates if disentangled representations are helpful for out-of-distribution (OOD) generalization, meaning the generalization to unknown FoV. All models in the study are pretrained in a completely decoupled manner like DARLA. They highlight reward is strongly correlated with OOD performance but disentanglement is not beneficial for sim-to-real transfers. Instead, it would be more useful to apply input noise.

Combining weakly-supervised disentanglement with demonstration learning was shown to be useful by Hristov et al. [HR21]. The authors train a robot on a real-world manipulation task and incorporate weak supervision signals from user inputs in a restricted labeling setting. For this purpose, the user defines the groups the current trajectory corresponds to.

4

DISENTANGLING VISUAL REINFORCEMENT LEARNING WITH DOMAIN RANDOMIZATION

In this chapter the main idea is presented for training disentangled representations jointly with a RL policy using DR in a curriculum learning setting. First, the general problem formulation is outlined. Afterwards it is described how little additional information from the simulation can be used to incorporate weakly-supervised disentanglement which then can be used to guide the DR process. Due to the curriculum learning setting of DR, the algorithm is constantly confronted with new randomizations. Without further additions the algorithm would not be able to learn about these new observations. Therefore, an approach to handle this is described. Finally, the combination of the individual methods is presented.

4.1 PROBLEM FORMULATION

The main goal of this thesis is to introduce a method that is able to learn a disentangled representation jointly with a reinforcement learning policy. A policy able to disentangle important FoV should transfer from the source domain to a different target domain from which we know it differs in an unknown set of FoV from the source domain. Therefore, it might be appealing to train the agent on a set of source domains with changing FoV and to let it learn to represent such factors (equal to DR).

To learn such policies the agent faces environments that share the properties already described in Section 2.2.1, Section 2.3.3 and Section 2.4.3. As defined in Section 2.3.3, the environment e_ξ is parametrized by a set of changeable parameters $\xi \in \mathbb{R}^D$ where these parameters are sampled from a distribution $\xi \sim P_\psi$ with ψ as the parametrizing variables of the distribution. In standard disentangled representation learning settings this distribution is generally uniform due to the structure of the possessing dataset. In DR the distribution is often a combination of fixed Gaussian, uniform or log-uniform distributions. However, in the setting of this framework it is assumed that the distributions are of fixed type for each factor but continuously changing variables ψ. Therefore, randomizing the parameters of an environment has in this setting the identical meaning as changing the factors of variation of it wherefore DR and changing FoV can be seen as similar transformations.

Except from this parametrization, the environment e_ξ is defined as normal MDP which yields observations $s \in \mathcal{S}$ from a high-dimensional image observation space $\mathcal{S} \in \mathbb{R}^{W \times H \times C}$ that describes C-channel images of width W and height H. Rather than learning to solve the environment directly in this observation space, the policy should utilize the lower-

dimensional representation space $\mathcal{Z} \in \mathbb{R}^K$ by jointly learning a mapping $g_\phi : \mathcal{S} \to \mathcal{Z}$ parametrized by ϕ. For this to be possible it is assumed that the associated representation $z \in \mathcal{Z}$ is fully observable from an observation s. Furthermore, the representation space should include a transformation of the factors ξ and additional information about the environment needed to solve the underlying task. Therefore, the number of dimensions of the representation K should be larger than the number of overall factors D.

In accordance with the match-pairing setting of Section 2.4.3, two environments e_{ξ_1} and e_{ξ_2} share a known subset $\xi_I \subset \xi$ of FoV so that the factors can be splitted into this shared part I and a differing part $\setminus I$. This is equivalent to $\xi_I \cap \xi_{\setminus I} = \xi$. Observations s_1 and s_2 which are now sampled from e_{ξ_1} and e_{ξ_2} share the same factors ξ_I. Similar to Equation 2.55, this alignment constraint imply for the true posterior

$$p(z_i|s_1) = p(z_i|s_2) \quad \forall i \in I, \tag{4.1}$$

$$p(z_i|s_1) \neq p(z_i|s_2) \quad else. \tag{4.2}$$

Since the factors of ξ are independent dimensions, each of these dimensions only changes the corresponding transition probability of the environment so that two environments with an equal factor $\xi_{1,i} = \xi_{2,i}$ share the same transition probabilities:

$$\begin{aligned} \mathcal{T}(s', r|s, a, \xi_i) &= \mathcal{T}(s'_1, r|s_1, a, \xi_{1,i}) \\ &= \mathcal{T}(s'_2, r|s_2, a, \xi_{2,i}) \end{aligned} \tag{4.3}$$

Therefore, the disentanglement of those factors is supposed to help the agent to perceive and handle the dynamics of the environment. The agent now shall disentangle the factors ξ using mapping g_ϕ while it learns to maximize the expected return

$$J(\theta) = \mathbb{E}_{\xi \sim P_\psi} \left[\mathbb{E}_{\pi_\theta, g_\phi, \tau \sim e_\xi} \left[G(\tau) \right] \right] \tag{4.4}$$

with policy π_θ using inputs from the learned mapping g_ϕ.

4.2 GUIDING DOMAIN RANDOMIZATION WITH WEAK-SUPERVISION

Section 2.4.3 of Chapter 2 described that training generative models with weak-supervision by match-pairing is superior to training those models unsupervised. By using simulations as source domains, information about factor pairs $\xi_{1,i}$ and $\xi_{2,i}$ between simulation instances is readily available and thus match-pairing or other forms of weak supervision can be used.

As Akkaya et al. [Akk+19] describe, training with a gradually increasing curriculum simplifies the training process and eliminates the necessity to tune the DR variables by hand. It can be assumed that similar statement also holds for disentangled representations. Furthermore, Akkaya et al. [Akk+19] introduce ADR as a method to guide the domain randomization parameters based on the performance of the agent so that the training on randomized environments create a curriculum.

The main focus of this section is to combine these two ideas. As mentioned in the last section, the agent should disentangle the domain randomization variables into distinct dimensions of a representation z from input states s using mapping g_ϕ. Since the goal is to learn disentangled representations, this mapping is the encoder q_ϕ of a VAE. Given shared factor indices I between two observations s_1 and s_2, this mapping should be learned with weak supervision by enforcing Equation 4.1 using an averaging strategy $\text{avg}(\cdot)$ between two encodings:

$$\tilde{q}_\phi(z_i|s_1) = \text{avg}\left(q_\phi(z_i|s_1), q_\phi(z_i|s_2)\right) \quad \forall i \in I \tag{4.5}$$

$$\tilde{q}_\phi(z_i|s_1) = q_\phi(z_i|s_1) \quad else. \tag{4.6}$$

As before, this applies to s_1 and s_2 equally. The new encoding $\tilde{q}$ can be used to optimize the combined β-VAE loss of Equation 2.60. In general, the averaging strategy of [Hos19] is used in this thesis.

Considering a different perspective, Equation 4.1 can also be used to evaluate the current disentanglement by evaluating that shared factors between both observations are encoded identically

$$q_\phi(z_i|s_1) = q_\phi(z_i|s_2) \quad \forall i \in I \tag{4.7}$$

$$q_\phi(z_i|s_1) \neq q_\phi(z_i|s_2) \quad else. \tag{4.8}$$

This allows to measure disentanglement during training and to guide ADR to adapt its parameters using a performance metric for each factor ξ_i of the environment. For a number of evaluation steps M in which a batch of N paired $(s_1^{(1)}, s_2^{(1)}), \ldots, (s_1^{(N)}, s_2^{(N)})$ observations is sampled from a replay buffer, this metric is given by

$$\text{performance}_i = \frac{1}{M} \sum_{l=1}^{M} \exp\left(-\frac{1}{N} \sum_{k=1}^{N} \left| q_\phi(z_i|s_1^{(k)}) - q_\phi(z_i|s_2^{(k)}) \right| \right) \in (0, 1]. \tag{4.9}$$

In this thesis, the difference between the encoders is implemented as the difference between the means of the distributions. But ither distances are also possible. Based on this metric, ADR can change its parameters given two performance thresholds t_L and t_H:

1. $\text{performance}_i \geq t_H$: The encoder q_ϕ learned to disentangle the current factors sufficient enough, hence the parameters ψ_i of the DR distribution for the measured dimension i can be increased.

2. $t_H > \text{performance}_i > t_L$: The disentanglement performance of the encoder q_ϕ is acceptable but must be trained further, hence the parameters ψ_i are neither increased nor decreased.

3. $\text{performance}_i \leq t_L$: The disentanglement performance of the encoder q_ϕ is not sufficient enough anymore, hence the parameters ψ_i are decreased.

In each training step, this adapted ADR algorithm samples two observations s_1 and s_2 by generating these with k different factors and $n - k$ unchanged factors between those. Similar

to the original ADR algorithm, in each episode the procedure selects a factor dimension ξ_i and bounds its value with equal probability to one of its current boundary values (i.e. the lower value ψ_i^L of the associated uniform distribution $\mathcal{U}(\psi_i^L, \psi_i^H)$). After generation, those samples can be used to train the policy and the VAE by adding them to a replay buffer. Training of the VAE proceeds as described above in a weakly-supervised way. The evaluation is executed after every T training steps using batches of random samples from the replay buffer and adapts the current parameters ψ_j based on the current performance on disentanglement with predefined update step sizes Δ_i.

4.3 CAPACITY-BASED DOMAIN RANDOMIZATION

As described in Section 2.4.3, Burgess et al. [Bur+18] claims that due to the induces bottleneck pressure of β-VAE, the encoder q_ϕ learns to encode only this information that contributes to the log-likelihood sufficient enough. Unfortunately, if the algorithm is continually confronted with new information, the encoder uses most of its capacity on information that occurs in earlier training steps while having not enough capacity left for later information. Therefore, if the DR distributions are changed during training, distinct factor values that are newly introduced might not be learned at all as the capacity of the VAE is not sufficient enough anymore. Using an additional increasing target KL as proposed by Burgess et al. [Bur+18] should help to improve disentanglement during sequential training as demonstrated in [Ach+18].

Instead of linearly increasing the target KL term C during training it is possible to condition this term on the current parameterization of the DR distributions as the distributions provide a quantity on how much information has been added as a result of the expansion. It is possible to compare the initial DR distribution P_{ψ_0} with the current DR distributions P_{ψ_k} using the KL divergence

$$C_j = D_{KL}(P_{\psi_j} \parallel P_{\psi_0}). \tag{4.10}$$

Both distributions are identical at the beginning of the training, so that the capacity starts at 0. As training progresses, the distribution P_ψ deviates more and more from the initial distribution, increasing the capacity. Since the KL divergence is always non-negative, the capacity will never reach a negative value. This ensures that the representation capacity can always reach this target value. For the standard unsupervised disentangled representation learning setting, the loss from Equation 2.54 is thereby adapted to

$$\mathcal{L}(\theta, \phi; x, z, C) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \gamma|D_{KL}(q_\phi(z|x) \parallel p(z)) - \underbrace{D_{KL}(P_{\psi_j} \parallel P_{\psi_0})}_{C}|. \tag{4.11}$$

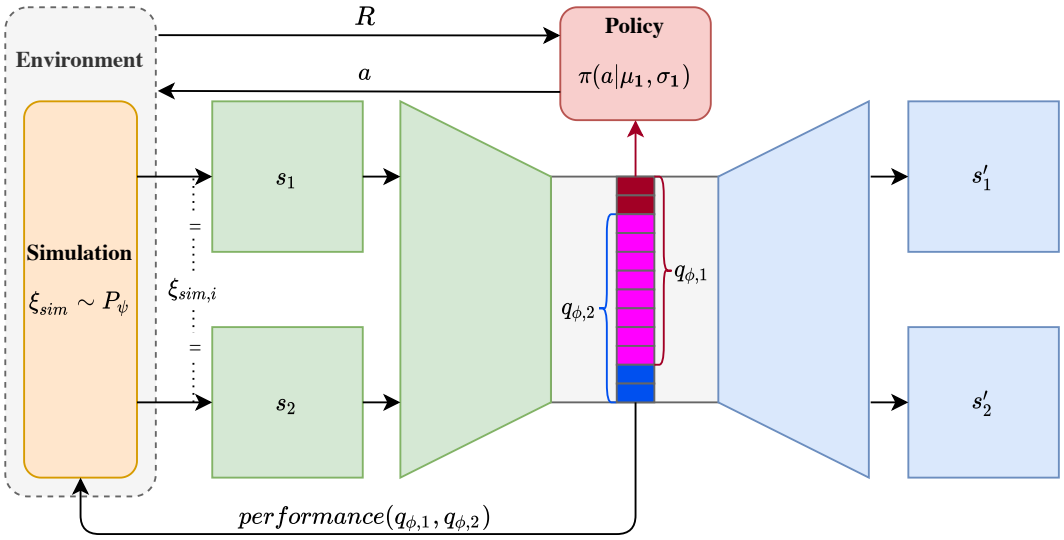


Figure 4.1: Combining weakly-supervised disentanglement with DR for RL.

4.4 CAPACITY-BASED WEAKLY-SUPERVISED DOMAIN RANDOMIZATION

Combining the ideas from the previous Section 4.2 and Section 4.3 yields a new algorithm that learns distinct randomizations in predefined dimensions of a latent representation jointly with a RL policy. In general, every algorithm can be used as underlying RL framework since the main contributions of this thesis affect the extraction of the representation only. In the on-policy case a few adaptions have to be made as those methods usually do not utilize a reply buffer which is necessary for the VAE to prevent catastrophic forgetting. During evaluation SAC is used as RL algorithm to learn the policy.

Using SAC, the framework can be divided into three main stages:

1. In the first stage, the algorithm interacts with the environment generating augmented transitions (s_t, a_t, R_t, s_{t+1}) with $s_t = (s'_t, s''_t, I_t)$ using the current policy π_θ and the encoder q_ϕ of the VAE. During this process the augmented state s''_{t+1} which differs in k factors from the primary state s'_{t+1} and the indices I_t of the differing factors are generated. Those transitions are stored in a replay buffer $\mathcal{D}$. After a complete episode of the environment, new factor values ξ for all factors are sampled from the DR distribution to parameterize the environment.

2. After a specific number of environment steps (usually 1), policy and VAE parameters are updated using sampled batches of transitions x from the replay buffer. The policy is updated as described in Section 2.2.6.1 whereas the VAE is updated using the weak supervision updates from the previous Section 4.2 and Section 4.3 utilizing the shared factor indices I between the primary state s' and the augmented version s''. The loss functions for the weakly-supervised training from Equation 2.60 and for the

Algorithm 3 Automatic Domain Randomization with Weakly-Supervised Disentangled Representations

procedure SAC-WACCI
  $\mathcal{D} \leftarrow \emptyset$ ▷ Initialize empty replay buffer
  for each iteration **do**
    $\xi \sim P_\psi$
    for each environment step **do**
      $\mu'_t, \sigma'^2_t \leftarrow q_\phi(s'_t)$
      $a_t \sim \pi_\theta(a_t|\mu'_t, \sigma'^2_t)$
      $s'_{t+1}, R_t \leftarrow \textsc{Env}(a_t, \xi)$
      $s''_{t+1}, I_{t+1} \leftarrow \textsc{GenerateDifference}(s'_{t+1}, k)$
      $s_{t+1} = (s'_{t+1}, s''_{t+1}, I_{t+1})$
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, R_t, s_{t+1})\}$
    end for
    for each gradient step **do**
      $x \sim \mathcal{D}$
      $\pi_\theta \leftarrow \textsc{UpdatePolicy}(\pi_\theta, x)$
      $p_\phi, q_\phi \leftarrow \textsc{UpdateVAE}(p_\phi, q_\phi, x)$
    end for
    if it is time to evaluate **then**
      $p \leftarrow \textsc{EvaluatePerformance}(q_\phi, \psi, \mathcal{D})$
      $\psi \leftarrow \textsc{UpdateDomainRandomizationParameters}(p, \psi, t_H, t_L, \Delta)$
      $\mathcal{C} \leftarrow \textsc{UpdateCapacity}(\mathcal{C}, \psi)$
    end if
  end for
  return Policy π_θ, VAE g_θ
end procedure

unsupervised target capacity training with known distributions from Equation 4.11 can be combined to

$$\begin{aligned}\mathbb{E}_{q_\phi(z|s)}\left[\log p_\theta(s'|z)\right] + \mathbb{E}_{q_\phi(z|s)}\left[\log p_\theta(s''|z)\right] \\ - \gamma \left| D_{KL}(q_\phi(z|s') \| \, p(z)) - D_{KL}(P_{\psi_j} \, \| \, P_{\psi_0}) \right| \\ - \gamma \left| D_{KL}(q_\phi(z|s'') \| \, p(z)) - D_{KL}(P_{\psi_j} \, \| \, P_{\psi_0}) \right| \end{aligned} \tag{4.12}$$

3. If it is time to evaluate after every T training steps, the VAE is evaluated according to the performance metric described in Section 4.2 by exploiting knowledge from weak supervision. Based on the defined thresholds t_H and t_L the parameters of the DR distributions ψ are adapted using the predefined step size hyperparameter Δ to define a simpler, identical or more difficult task through the environment. Due to the update of those parameters, the capacity target term C of the combined loss Equation 4.12 is adapted as well.

The procedure is listed in Algorithm 3 and visualized in Figure 4.1. In the following chapters this method is denoted as Automatic Controlled Capacity Increase (ACCI) VAE. In combination with a SAC agent it is referred to as SAC-ACCI.

5

EXPERIMENTAL RESULTS

In this chapter, the proposed method SAC-ACCI is evaluated on a simulated control task. For this propose, the overall experimental setup including the description of specific implementation choices is presented. The results of the training as well as the overall evaluation of the method on the original task (source domain) is discussed. In an additional experiment, the method is evaluated in a sim-to-sim setup (target domain). Lastly, the interpretability of the latent representation of the method is examined with three smaller experiments which investigate different parts of the VAE.

5.1 EXPERIMENTAL SETUP

Before the overall results are described, the training and evaluation task is outlined and the different network architectures, hyperparameters and preprocessing steps are described. Specifically, SAC algorithm as RL objective was used coupled with the proposed ACCI framework. The implementation is build on top of the publicly released implementation from [Yar+21]. The training of the agents and all experiments were performed on a compute cluster with 96 CPU cores and 8 NVIDIA Tesla V100 GPUs.

5.1.1 *Environment*

As experimental and evaluation platform, the cart-pole swing-up (CPSU) scenario, a standard continuous control task, is used. It is based on the MuJoCo [TET12] physics engine integrated in the OpenAI Gym [Bro+16] framework. The task objective is to control the position of a cart on a rail to swing-up and balance a hinged pole in an upright position while this pole starts downwards at the beginning of each episode. The cart is controlled by setting the single-valued torque of an attached direct-drive motor. The action-space of the environment is therefore $\mathcal{A} \in \mathbb{R}$ but limited to values inside $[-3, 3]$. The standard state-space consists of the position and velocity of the cart and the angle between the horizontal and the pole as well as the angular-velocity of the pole. In this setting however, each state is a high-dimensional pixel-observation $\mathcal{S} \in \mathbb{R}^{W \times H \times C}$ of the scene neglecting the original state. As the true state is not fully-observable with a single image (i.e. the velocities), the algorithm observes a small history in each step consisting of the last K observations by stacking a set of K frames together resulting in $\mathcal{S} \in \mathbb{R}^{W \times H \times C \cdot K}$. The

reward is obtained by taking the cosine from the angle of the pole to the horizontal. During training, each episode consists of 250 timesteps ensuring that the balancing part does not consume the majority of the time later in the training as the swing-up part is usually short in time.

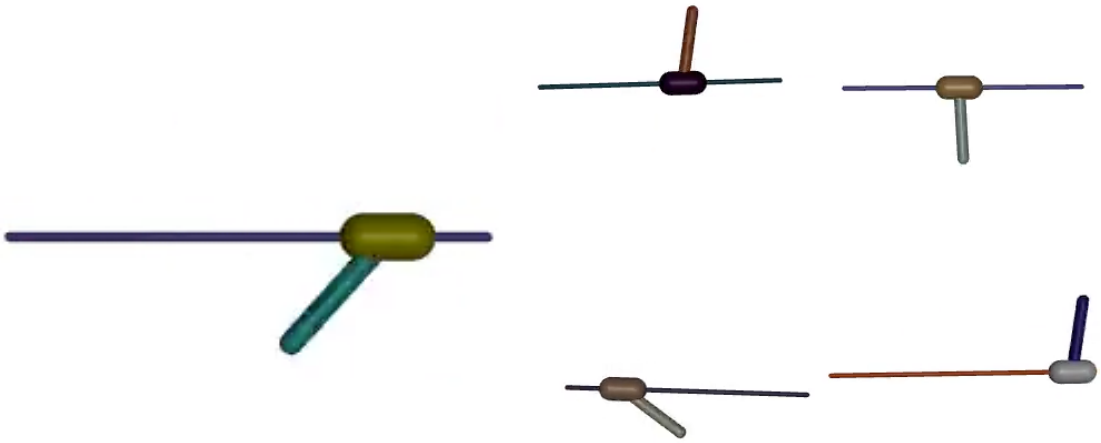


Figure 5.1: Visualization of the original environment (left hand side) and comparison of different randomizations (right hand side).

5.1.2 *Training Setup & Hyperparameters*

Using the simulation and a framework that allows to change specific simulation parameters, a DR study is performed by randomizing the five parameters in Table 5.1 during training and evaluation (similar to the evaluation of DARLA). Six policies are trained on different random seeds using SAC-ACCI for $1,000,000$ training steps.

For the domain randomization, all lower and upper bounds were selected so that a policy with all randomizations is able to converge to a good performance level. The lower and the upper bound of the camera properties were chosen to be physical reasonable and also not too difficult during training.

To be able to compare the proposed methods to other methods, additional baseline policies were trained. For the quantitative comparison, the SAC-AE algorithm proposed by Yarats et al. [Yar+21] is used. All methods build upon this implementation. Furthermore, an adapted version of the same method using a VAE instead of an AE is also used. As in Yarats et al. [Yar+21], the convolutional layers of the encoder of these methods receive gradients from the representation learning and the critic. The fully-connected layers of the encoder additionally receive gradients from the actor. For additional qualitative experiments, DARLA was utilized by training a VAE on a pre-collected dataset in which states are similar distributed as in a joint training run in order to make the results comparable. The dataset for pre-training the vision module of DARLA was collected by a state-based SAC. The trained VAE was then used to train a SAC algorithm on the actual task with the same

number of steps as the other methods. This implementation from the DARLA algorithm deviates from the original, as the original uses the sampled latent state z as policy input and uses other RL algorithms as SAC. In this thesis, the outputs μ and σ^2 of the posterior itself are used instead, as using those values does stabilize the policy learning. This is reasonable since the latent state z is always a result of randomness due to the sampling process. Identically to Yarats et al. [Yar+21], each algorithm first collects 1000 observations using a random policy at the beginning of the training and afterwards collects observations by sampling actions from the current policy. The encoder of DARLA and SAC-ACCI only receive gradients during the representation learning phase and does not share gradients with the actor and critic during policy learning.

For all methods, two architectures are used:

- The VAE architecture for the main method consists of an encoder of four convolutional layers identically to the architecture used by Higgins et al. [Hig+17b], each with kernel size 4, and stride 2 in the height and width dimensions. The number of filters for the first two layers is 32 and 64 for the following two, respectively. The convolutional layers are followed by a fully connected layer of size 256 units. In between

Category	**Parameter**	**Dist.**	**Type**	**Values**		
				Nominal	**Mean**	**Std. Dev.**
Camera	Position	$\mathcal{N}$	Additive	0	0	0.01
				-3	0	0.01
				0	0	0.01
	Angle	$\mathcal{N}$	Additive	90°	0	0.573°
				0°	0	0.573°
				0°	0	0.573°
				Nominal	**Lower Bound**	**Upper Bound**
RGB	Rail	$\mathcal{U}$	Replace	77	0	255
				77	0	255
				179	0	255
	Cart	$\mathcal{U}$	Replace	179	0	255
				179	0	255
				0	0	255
	Pole	$\mathcal{U}$	Replace	0	0	255
				179	0	255
				179	0	255

Table 5.1: Parameters for DR in the MuJoCo simulation environment. $\mathcal{N}$ describes a Gaussian distribution with mean and standard deviation whereas $\mathcal{U}$ describes uniform distributed values with lower and upper bound. The nominal parameter values of the original environment can be completely replaced by the new randomized values (type *Replace*) or are added on top of them (type *Additive*).

Method	Architecture	Shared Gradients	Framestack	Decoder	β/γ
SAC-AE [Yar+21]	[Yar+21]	Yes	`Collective`	MSE	–
SAC-VAE [Yar+21]	[Yar+21]	Yes	`Collective`	MSE	1
SAC-VAE [Yar+21]	[Yar+21]	Yes	`Collective`	MSE	4
SAC-VAE [Yar+21]	[Yar+21]	Yes	`Collective`	MSE	16
DARLA [Hig+17b]	[Hig+17b]	No	`Separate`	Bernoulli	4
SAC-ACCI	[Yar+21]	No	`Collective`	MSE	100
SAC-ACCI	[Yar+21]	No	`Collective`	Bernoulli	100
SAC-ACCI	[Hig+17b]	No	`Separate`	Bernoulli	100
SAC-ACCI	[Hig+17b]	No	`Separate`	Bernoulli	500

Table 5.2: All trained agents used for evaluation. The first five rows show the agents that are used as baselines. The last four rows show the settings of different SAC-ACCI agents. The column *Architecture* indicates the VAE architecture used while *Shared Gradients* lines out if gradients of the critic are used to update the encoder as proposed by Yarats et al. [Yar+21]. A stack of frames can be processed in a single pass by the encoder (`Collective`) whereas in the `Separate` case each frame is computed individually and the result is stacked afterwards. To optimize the reconstruction accuracy either a MSE-Loss or a Bernoulli-Loss (BCE-Loss) can be used. The last column shows the value used for the hyperparameters β or γ.

ReLU activations are used. The latent layer comprises 128 units parametrizing 64 independent Gaussian distributions. Smaller sizes like 64 units as used in the DARLA algorithm or 10 units as used in standard disentanglement settings were empirically found not to be sufficient enough meaning that some agents were not able to learn to solve the task. Therefore, to make the different methods comparable, the standard number of units is 128 in the latent layer.

- The other architecture is identical to the one of Yarats et al. [Yar+21]. It also consists of an encoder with four convolutional layers, each with kernel size 3 but without stride except for the first layer which has stride 2. The number of filters for each layer is constantly 32. The output of the last convolutional layer directly serves as input for the last fully-connected layer. Latter compromises 128 units as in the architecture described first.

In both cases, the decoder architecture is simply the reverse of the encoder, utilizing deconvolutional layers. The adjustable hyperparameters β and γ where chosen according to recent literature on disentangled representations [Hig+17b; Bur+18; Ach+18; Dit+20; Trä+21].

Furthermore, two processing procedures are used for the input stack of frames: In the first procedure the stack of K frames is passed to the encoder without additional modifications forming a single input for the policy network (`Collective`). In the second case, each observation frame is passed to the encoder individually forming an single encoder output for each frame. These individual outputs are concatenated together again afterwards to form a K-dimensional input to the policy network (`Seperate`).

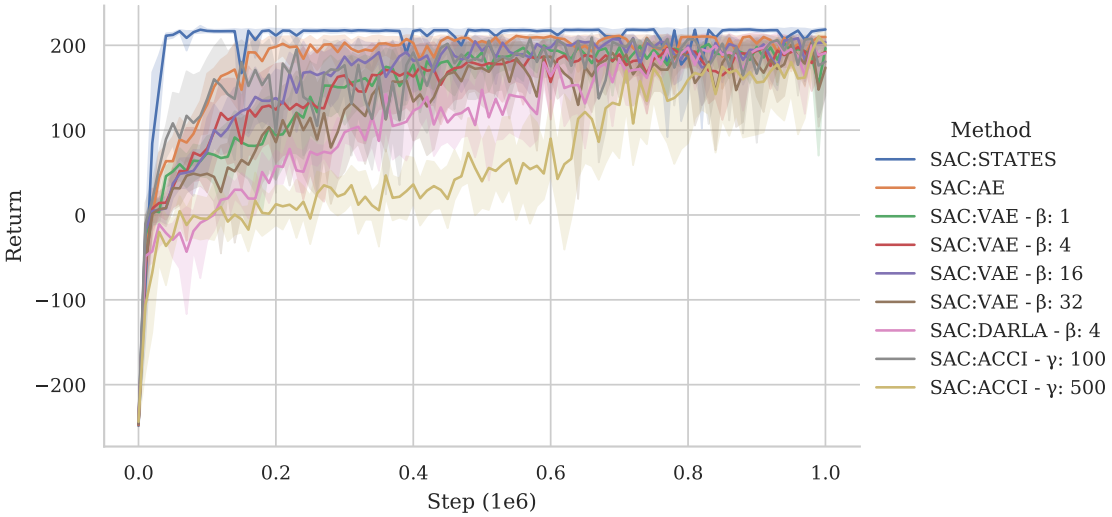


Figure 5.2: Evaluation of average episode return over 10 evaluation steps after every 10000 environment steps. The SAC-ACCI agents are the last two of Table Table 5.2.

Both actor and critic of the SAC agent receive the output from the encoder network as input and pre-process it with layer normalization [BKH16]. The result is then further processed by three fully-connected layers with 1024 hidden units and ReLU activations in between. The critic uses double Q-learning as described in Section 2.2. The output of the actor parameterizes a Gaussian distribution with mean and variance. Further hyperparameters of SAC can be found in Table 5.3 and are taken from Yarats et al. [Yar+21]. These hyperparameters were not optimized as this is out of the scope of this thesis. As described in Chapter 4, the ACCI framework needs additional hyperparameters to update the capacity term C and the DR distributions. The used values are shown in Table 5.4.

During training it was noticed that a mistake was made in the implementation. The original implementation by Yarats et al. [Yar+21] uses individual AE encoders for the actor as well as for the critic. However, both encoders share the weights for the convolutional layers but not for the latter fully-connected layers. The encoder of the critic is used for the reconstructions. Since the encoder of the actor is never updated by gradients from the actor, the fully-connected layers are never updated at all. However, all agents were still able to train successfully, since these layers only represent a non-linear transformation which is compensated by the subsequent layers in the actor.

5.2 SOURCE DOMAIN EVALUATION

During training, each agent is evaluated after every $10,000$ environment steps by computing the average episode return over 10 evaluation episodes. Instead of sampling from the Gaussian policy of SAC the mean of the Gaussian is taken during evaluation. This evaluation is shown in Figure 5.2. For comparison, a SAC agent based on the original states was included to show an upper bound for the pixel-based methods. The SAC-AE version of Yarats et al. [Yar+21] performs best on the high-dimensional pixel observation space

Parameter	Value
Replay buffer capacity	$500,000$
Batch size	128
Discount γ	0.99
Optimizer	Adam
Actor learning rate	10^{-3}
Actor update frequency	2
Actor log std.-dev. bounds	$[-10, 2]$
Critic learning rate	10^{-3}
Critic target update frequency	2
Critic Q-function soft-update rate τ_{Q}	0.01
Critic encoder soft-update rate τ_{enc}	0.05
Autoencoder learning rate	10^{-3}
Temperature learning rate	10^{-4}
Temperature Adam's β_1	0.5
Init temperature	0.1

Table 5.3: A complete overview of used hyperparameters.

Parameter	Value
Number of differences k	1
Evaluation Offset T	$10,000$
Evaluation Steps M	$1,500$
Discount γ	0.99
Thresholds t_L, t_H	$(0.8, 0.975)$
Step size Δ_{Cam}	0.0003
Step size Δ_{RGB}	0.0075
Init. distribution $P_{\psi_0}^{(Cam)}$	$\mathcal{N}(0.00, 0.001)$
Init. distribution $P_{\psi_0}^{(RGB)}$	$\mathcal{U}(115, 166)$

Table 5.4: A complete overview of used hyperparameters for the ACCI algorithm.

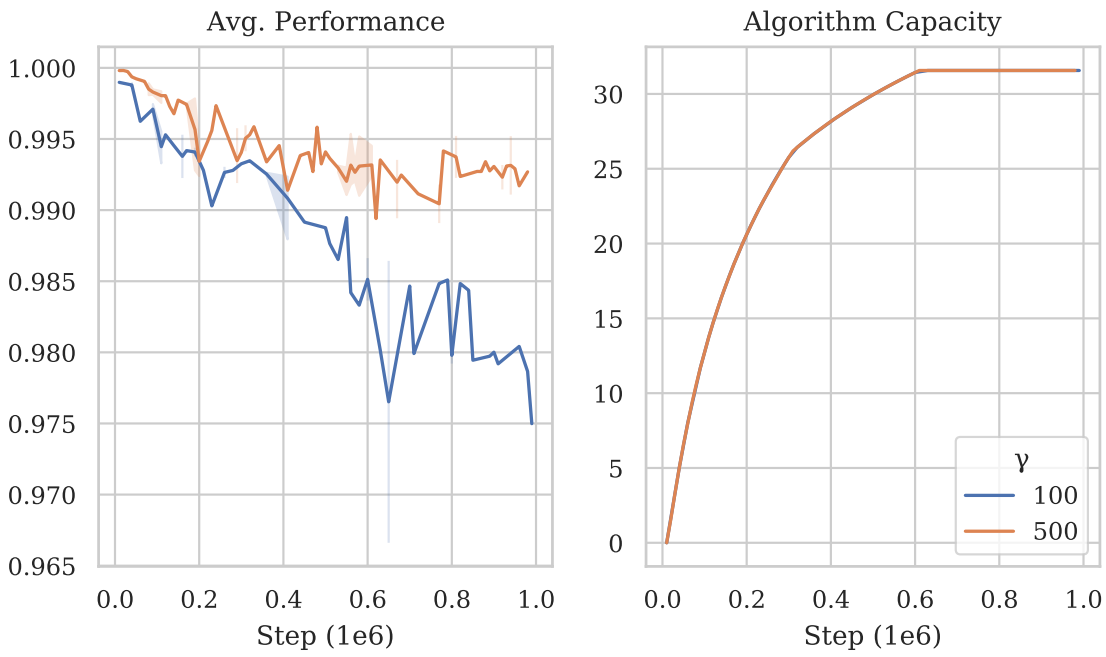


Figure 5.3: Average performance of the VAE and capacity change during training for $\gamma \in \{100, 500\}$.

while all other methods using VAEs perform worse. Nevertheless, all methods manage to successfully perform the task but with varying degrees of stability. As Yarats et al. [Yar+21] already concluded, methods using VAEs perform worse due to the stochastic nature of generative models. Additionally, a stronger pressure on the bottleneck destabilizes and slows down the overall learning process. Due to the higher pressure, a greater adaptation in the structure of the latent bottleneck takes place during training. As a result, actor and critic have to adjust to these adapted inputs. This is especially the case for SAC-ACCI with high bottleneck pressure. The high pressure slows down learning of a sufficient policy.

5.2.1 *Encoding Performance*

Additional to the performance on the control task, the encoding performance measured by Equation 4.9 and capacity of the SAC-ACCI algorithm was tracked and is both shown in Figure 5.3 for $\gamma \in \{100, 500\}$. During the course of training the average performance of the weakly-supervised VAE degrades over time but remains at an acceptable level. The hypothesis why the performance is significantly better at the beginning than at later stages of the process is that the encoder network is initialized with small weights causing similar representations at the beginning. Over the course of training with an increasing range of possible factor values, it gets more difficult to distinguish between those factors causing a worse performance. But a stronger pressure on the bottleneck (greater γ) helps to keep performance at a very high level, resulting in an expectation for better disentanglement of factors. As the performance never falls below the upper threshold t_H in both cases, the capacity increases continuously until it reaches it maximum value of this environment.

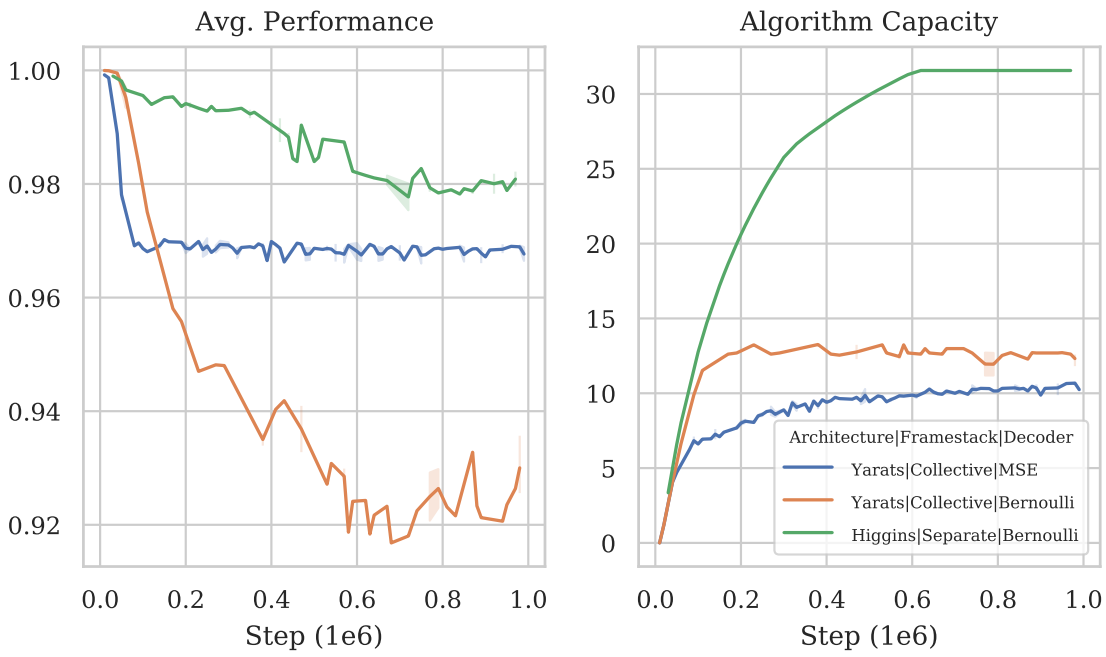


Figure 5.4: Average performance of the VAE and capacity change during training for different parameter settings but all with hyperparameter $\gamma = 100$.

In Figure 5.4 the same evaluation but for different algorithm settings is shown. Using a different network architecture than Higgins et al. [Hig+17b] while additionally processing the stack of frames in a single pass does significantly hurt the encoding performance. In both of these settings the encoder is not able to keep the performance above the threshold t_H, resulting in randomized training environments without increasing difficulty as the distributions are not updated anymore until the threshold t_H is passed again. In both cases this is not the case, as the performance settles below the threshold value. Since the distribution parameters are not updated anymore, the capacity also settles to an inferior value significantly below the maximum value. This may have several reasons: Separating the framestack into individual frames for the VAE helps especially in the weakly-supervised case as the provided indices of the stack are splitted likewise. Before, the encoder and decoder have to process a stack of frames in a single pass in which each state has an individual factor change. In this case, it is more challenging to learn to distinguish the changes based on the stack of multiple indices as not each of them are applicable to each state of the stack. If the stack is splitted into individual observations each with corresponding individual indices, it is naturally simpler to infer the change in the observations as the optimization is carried out for each frame individually. Furthermore, the network architecture proposed by Yarats et al. [Yar+21] is smaller, especially in the convolutional structure in comparison to the structure of Higgins et al. [Hig+17b]. This might lead to an insufficient learning capacity to learn the environment factors. Additionally, this architecture utilizes non-linearities on the latent, restricting the possible structure in the latent imposed by the adapted ELBO.

Figure 5.5: Visualization of the original environment in MuJoCo (left) and the corresponding environment in PyBullet (right).

5.3 SIMULATION-TO-SIMULATION EXPERIMENTS

In order to investigate the transfer performance of the agents without having access to a real world counterpart, an additional environment based on the PyBullet [CB21] physics engine is used similar to the evaluation of Yu et al. [YLT19]. Such an transfer experiment

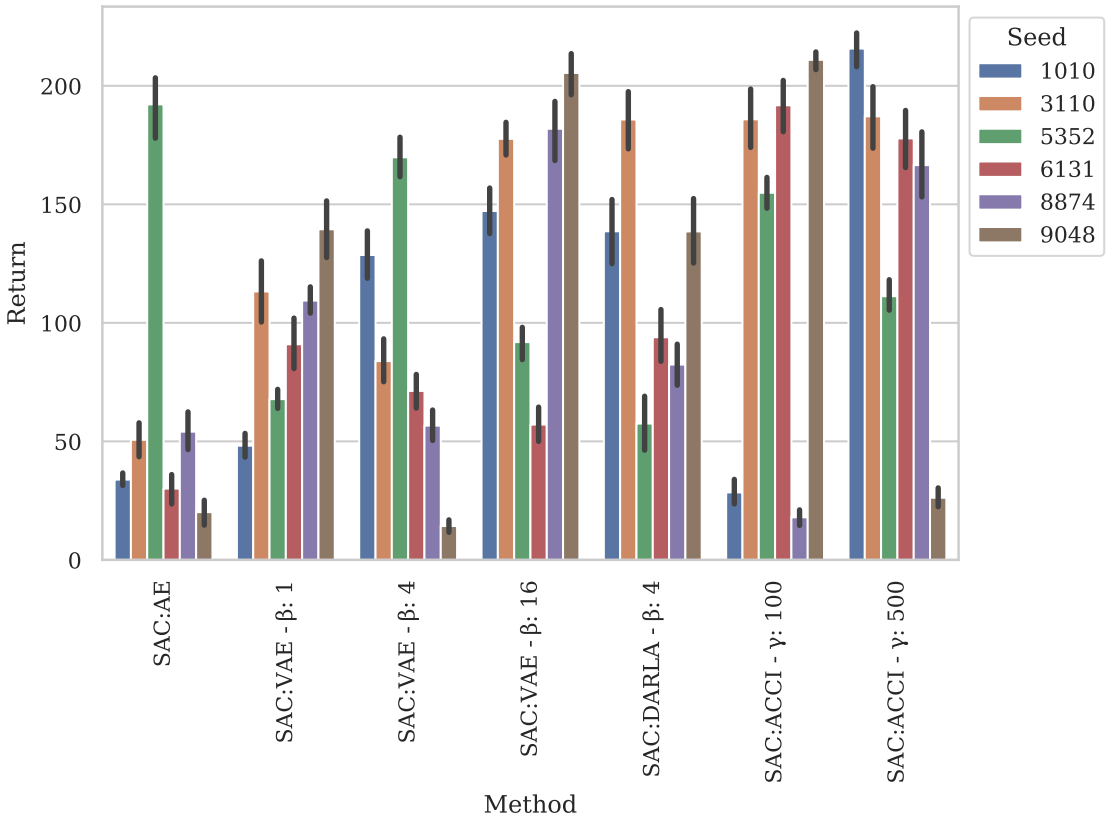


Figure 5.6: Average return of all trained agents on a dynamically equivalent PyBullet environment of the CPSU task. The agents are sorted according to their random seed. The SAC-ACCI agents outperforms most of the other environments especially those without disentanglement.

should simulate a resembling distributional shift as a simulation to reality transfer. This additional target environment shares the same differential equations as the original MuJoCo task but due to different implementation assumptions of the engine both simulations might behave different. In addition, both simulations differ especially in their rendering making this PyBullet environment a suitable transfer objective in this setting (see Figure 5.5). In recent literature this type of performance is referred to as OOD generalization [Dit+20; Trä+21] as the observed frames and their internal characteristics are not seen before during training. Thus, the generated data is distributionally different from the source training environment.

Each trained agent is deployed in this environment for 150 episodes. The episode return is measured and averaged across all episodes of each agent. As can be seen in Figure 5.6, the SAC-ACCI outperforms most of the other baselines with regard to robustness to the choice of random seed. Most baselines are able to find a sufficient policy that can be transferred to the target domain but with specific random seeds only. This observation leads to the hypothesis that these algorithms rely more on the choice of random seed than on the actual implementation. SAC-ACCI transfers to the target domain reliably without being heavily dependent on the random seed. The results of Figure 5.6 indicate that a higher pressure on the bottleneck leads to more robust agents as the average return correlates with disentanglement hyperparameters β and γ. Despite different results regarding OOD generalization by Dittadi et al. [Dit+20] and Träuble et al. [Trä+21], a higher pressure correlates with OOD generalization in this case. The main difference between the results of this thesis and Träuble et al. [Trä+21] is the joint training of VAE and SAC. Additionally, Träuble et al. [Trä+21] utilize proprioceptive robot states as policy input in addition to the generated representation.

5.4 LATENT STATE INTERPRETABILITY

Disentangled representations shall encode the hidden FoV of the environment in distinct latent units. As those hidden factors are usually defined as concrete human-readable concepts, the learned representations should be similarly interpretable. To observe what each part of the generative model actually learns, three experiments are conducted on a small number of agents mainly of the proposed SAC-ACCI method. Especially, the SAC-ACCI agents with $\gamma = 500$ are used for the investigations as it is expected that this agents should be able to disentangle the factors best. First latent traversals as described in Section 2.4.3 are created using the trained agents. In the following experiment, embeddings from the latent representations are generated using the principal component analysis (PCA) method. The last experiment investigates the attribution of regions of the input image to the disentangling latents of the representation.

Each experiment uses agents able to transfer to the target environment from the previous section thus ensuring that the representation is also able to generalize to distributional shifts. Images are created with a modified version of *disentanglement_lib* [Loc+19].

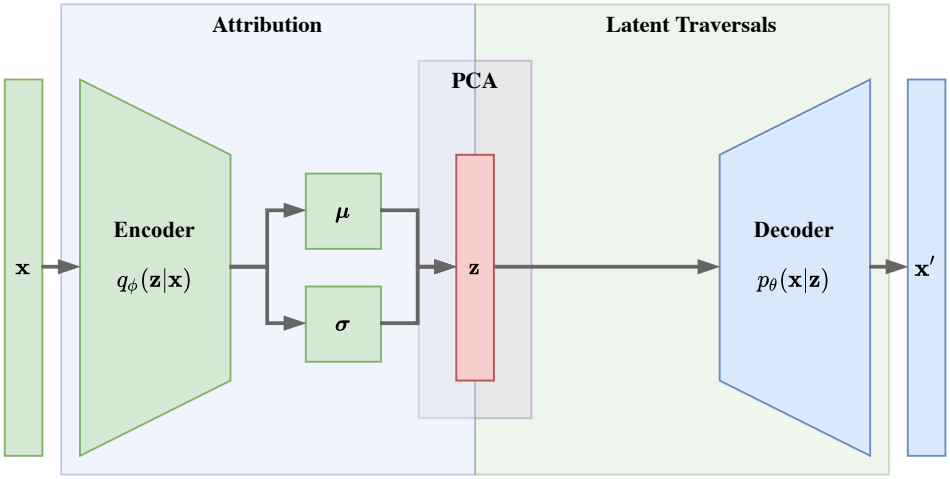


Figure 5.7: Experiments to measure interpretability of the generative model.

5.4.1 *Latent Traversals*

Creating traversals of individual latents is twofold: By changing individual latents, they reveal the structure of the learned distribution. Moreover, they also show what the decoder has learned, based on the latent representation. Therefore, latent traversals allow to analyze the process from the latent representation to the actual reconstruction. Each traversal is generated by traversing the latent representation in range $[-3, 3]$ in steps of size 0.5. Therefore, each traversal yields 7 reconstructions.

Traversals for SAC-ACCI are shown in Figure 5.8 and Figure 5.9. The first five latents correspond to those that shall disentangled the factors imposed by weak supervision, respectively the camera settings in the first two latents and the RGB values of the objects in the third to fifth latent. As can be seen in the figures, both camera factors are instead encoded in the second latent only. This indicates that the factor representation is compact but not modular as two factor values are encoded in a single dimension of the representation. Moving the camera or changing its angle produces similar output images, making it difficult for the encoder to distinguish between both factors. If additional pressure is applied to the representation, it seams to be easier for the encoder to encode both factors into a single latent.

Additionally, all decoders learned to ignore color values by simply predicting grey color values instead of the true value. On the one hand this indicates good generalization across a variety of color changes. If the encoding is invariant to color, the underlying policy does not perceive any information about color at all. This results in a desired behavior if color is irrelevant for the task. However, the main reason the color is not reconstructed might be the small effect of it on the log-likelihood. The cost of reconstructing the colors is higher than simply ignoring them. Tasks that require knowledge about the color can therefore not be learned if the encoder does not encode these factors. With this experiment, however,

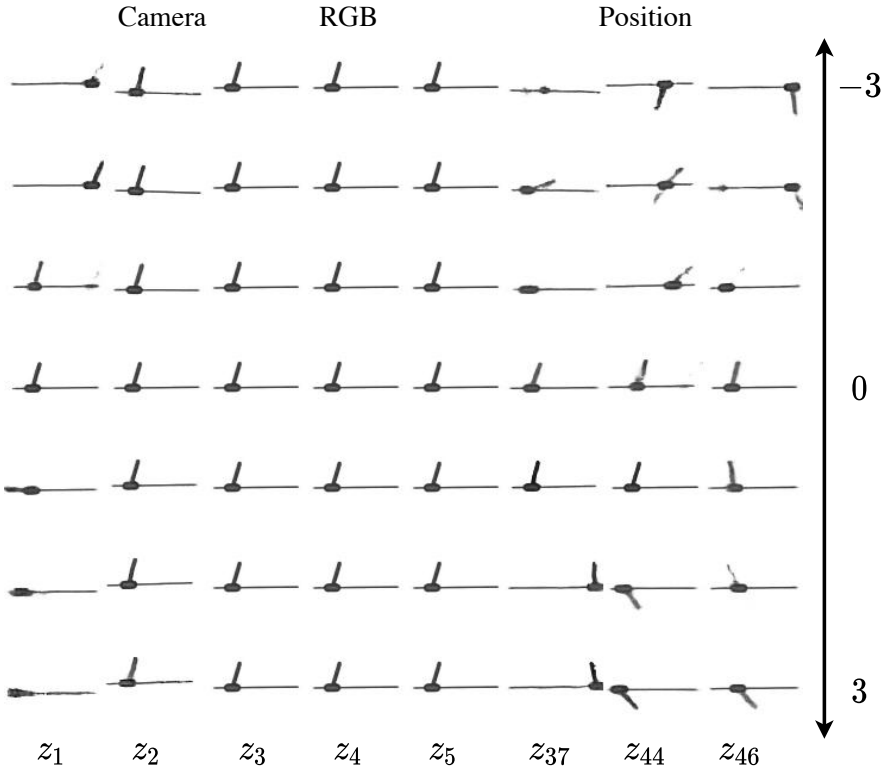


Figure 5.8: Latent traversal of SAC-ACCI with $\gamma = 100$.

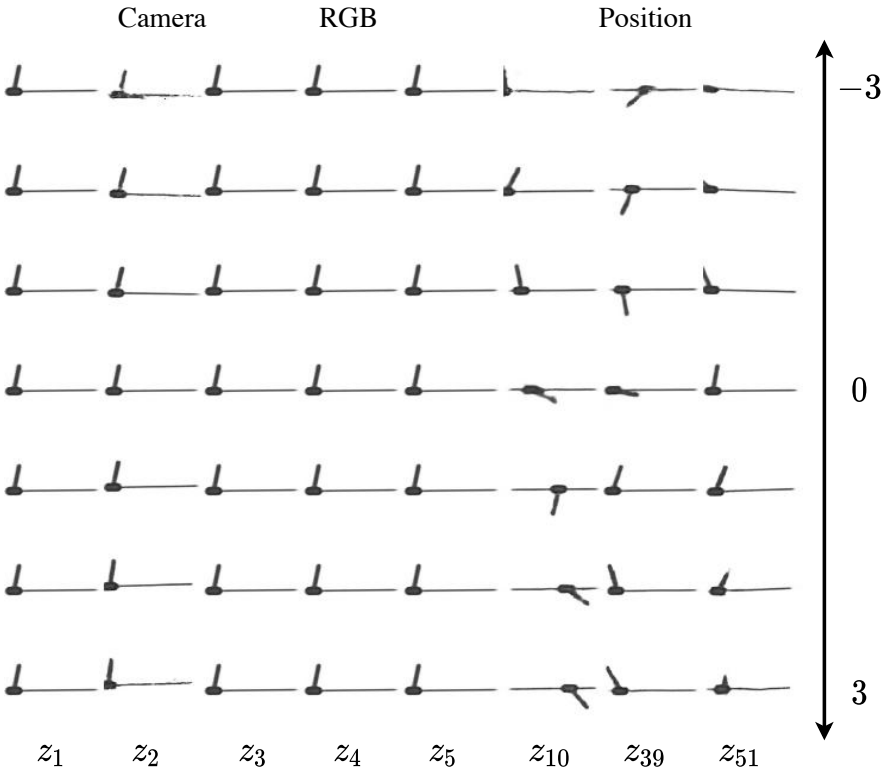


Figure 5.9: Latent traversal of SAC-ACCI with $\gamma = 500$.

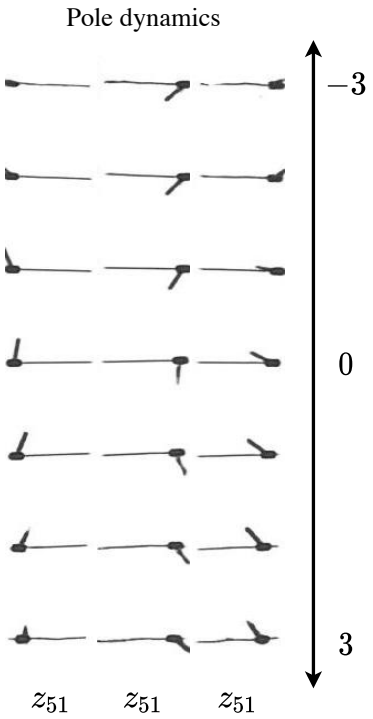


Figure 5.10: Latent traversals of latent variable z_{51} of SAC-ACCI with $\gamma = 500$ for different input images.

it remains unknown whether the encoder does not encode these values or whether the decoder does not consider these factors for a reconstruction.

The residual latents z_{10}, z_{39} and z_{51} of the SAC-ACCI agent with parameter $\gamma = 500$ learn cart position and pole angle mixed in different latents. While latent z_{10} learns about the distribution of the cart and the pole, latent z_{39} learns mostly about the pole itself. However, the representation of the pole of latent z_{39} is non-linear since the reconstruction is not continuous between the fifth and the sixth traversal image. Traversing latent z_{51} clearly shows that the module learned to distinguish the dynamics of the pole while the cart dynamics does not change throughout the traversal, as can be seen in Figure 5.10.

Furthermore, the camera movements and pendulum dynamics are separated in the modules. In the case $\gamma = 100$ (Figure 5.8) the results are similar to the case $\gamma = 500$. In particular, the camera parameters are represented nearly equally. Since these are explicitly learned through weak supervision, high bottleneck pressure parameters like $\gamma \geq 100$ are not needed to disentangle these factors, as Locatello et al. [Loc+20] already observed. Therefore, the weak supervision signal alone is sufficient. However, this is different for the position factors that are not explicitly learned. In the $\gamma = 100$ case, the decoding for these factors is less precise and not as explicit as in the $\gamma = 500$ case.

For comparison purposes, similar traversals for the entangled SAC-VAE agent with $\beta = 1$ and for the DARLA agent with $\beta = 4$ are shown in Figure 5.11 and Figure 5.12[1]. Comparing these traversals with those of SAC-ACCI shows the better disentanglement of the proposed method to prior methods.

1 Additional latent traversals can be found in Appendix A.1

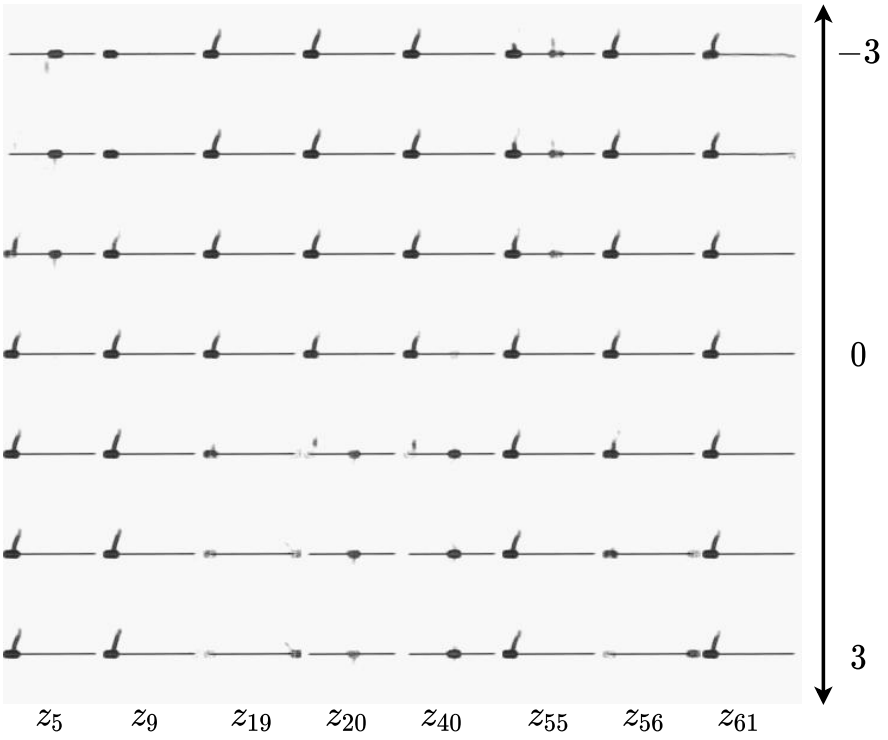


Figure 5.11: Latent traversals of the entangled SAC-VAE with $\beta = 1$.

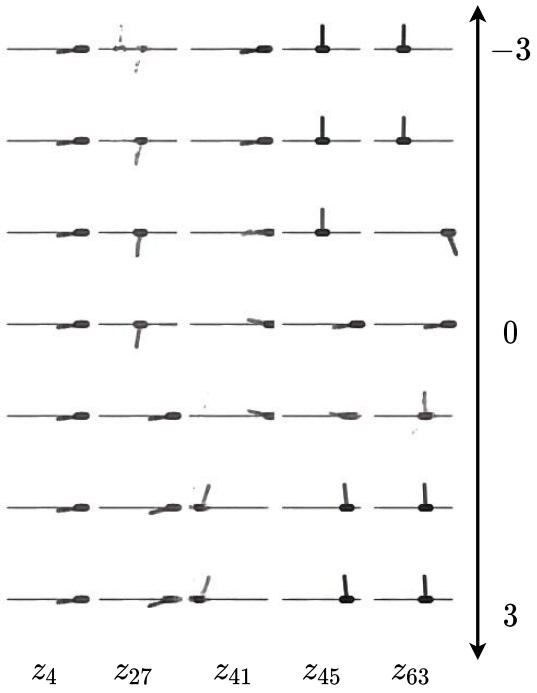


Figure 5.12: Latent traversal of SAC-DARLA with $\beta = 4$.

5.4.2 *Embeddings of Encodings*

Instead of examining the decoder and its results with specific adapted representations, the representation itself can also be examined directly. By reducing the dimensionality of the representation even further it may possible to uncover learned factors in the representations which are not used by the decoder. The goal of this experiment is to find FoV that are encoded in the representation but not processed by the decoder. Therefore, the representation is investigated directly. For this purpose, representations calculated from 30000 input frames are encoded and transformed into embeddings using the dimensionality reduction technique PCA.

5.4.2.1 *PCA*

The principal component analysis (PCA) method is a dimensionality reduction technique which is most often used for data compression, feature extraction and data visualization [Bis06, p. 561]. For a better understanding of the sections results, the following summary of PCA should give an short introduction into this technique: Given data X, PCA tries to find a orthogonal projection M of X onto a lower dimensional linear subspace Y (*principal subspace*) that maximizes the variance of the projected data such that $Y = XM$. This is achieved by maximizing the functional trace $\left(M^T \text{cov}(X) M\right)$. The solution of a PCA is the solution to the eigenproblem

$$\text{cov}(X)M = \lambda M. \tag{5.1}$$

The result is a linear mapping consisting of d principal eigenvectors of the covariance matrix $\text{cov}(X)$ with principal eigenvalues λ. In a probabilistic sense, PCA can also be defined as the maximum likelihood solution of a probabilistic latent variable model

$$X = WH + b + \sigma z \tag{5.2}$$

with d-dimensional Gaussian latent variable H and a noise process $z \sim \mathcal{N}(z; 0, I)$. The underlying FoV are primarily responsible for the variance in the data. Therefore, due to the maximization of the variance in PCA, the orthogonal projection should uncover the FoV. [Bis06; VPV+09; GBC16]

As PCA tries to find explanatory factors (FoV) in the data [GBC16, p. 479] it is similar to VAEs. Under the assumption that the representation of a VAE encodes these explanatory factors fully (as in the disentanglement setting), it can be further assumed that the principal components of a PCA applied on this VAE representation align with the encoded factors. I.e. given n FoV that are encoded by an VAE with a m-dimensional latent representation, the principal components of the encoding should equally align to those factors. In the CPSU setting of this experiment each observation consists of 7 FoV including camera position, camera angle, RGB values of the bodies, cart position and pendulum angle. The learned representation should be able to fully describe these factors. This in turn makes it possible to reduce the representation to 7 principal components.

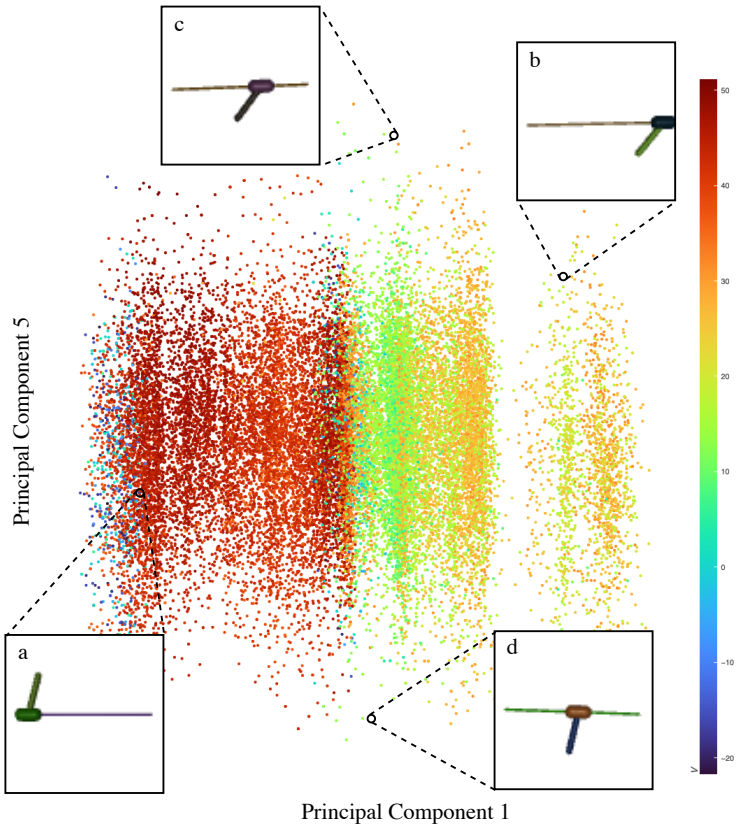


Figure 5.13: Visualization of principal components 1 and 5 of a PCA of the representation of an SAC-ACCI agent.

The PCA technique is applied to the mean values of the latent representation of the input data. Figure 5.13 shows the relationship between all principal components[2]. By defining the principal space to be 7-dimensional, 99.82% of the variance in the data is explained by all 7 principal components. By visually examining the states associated with the embedding points, the characteristics of the principal components can be studied in more detail. I.e. if the corresponding input frames of the representations are linked with the embedding of the PCA, a structured subspace is uncovered for components 1 and 5. As can be seen in Figure 5.13, principal component 1 describes the dynamics of the CPSU task ranging from states with a high reward in which the pendulum is stabilized (a) to states with low rewards in which the pendulum hangs downwards (b). In Figure 5.13 this applies for the whole horizontal axis. The embedding is color coded by the predicted state values of the critic for the corresponding observation. Red colors indicate a high possible return while green and blue colors predict a low return as it is described by the bar at the side of the picture. Principal component 5 on the other hand describes the possible camera changes in the images (c and d). Following the vertical axis, the rotation of the camera gradually changes consistently. The results indicate that PCA reveals the same representation dimensions for these combined FoV that were already revealed by the last experiment. Other FoV were not found to be aligning to principal components. Especially,

2 Additional PCA clusters can be found in Appendix A.2

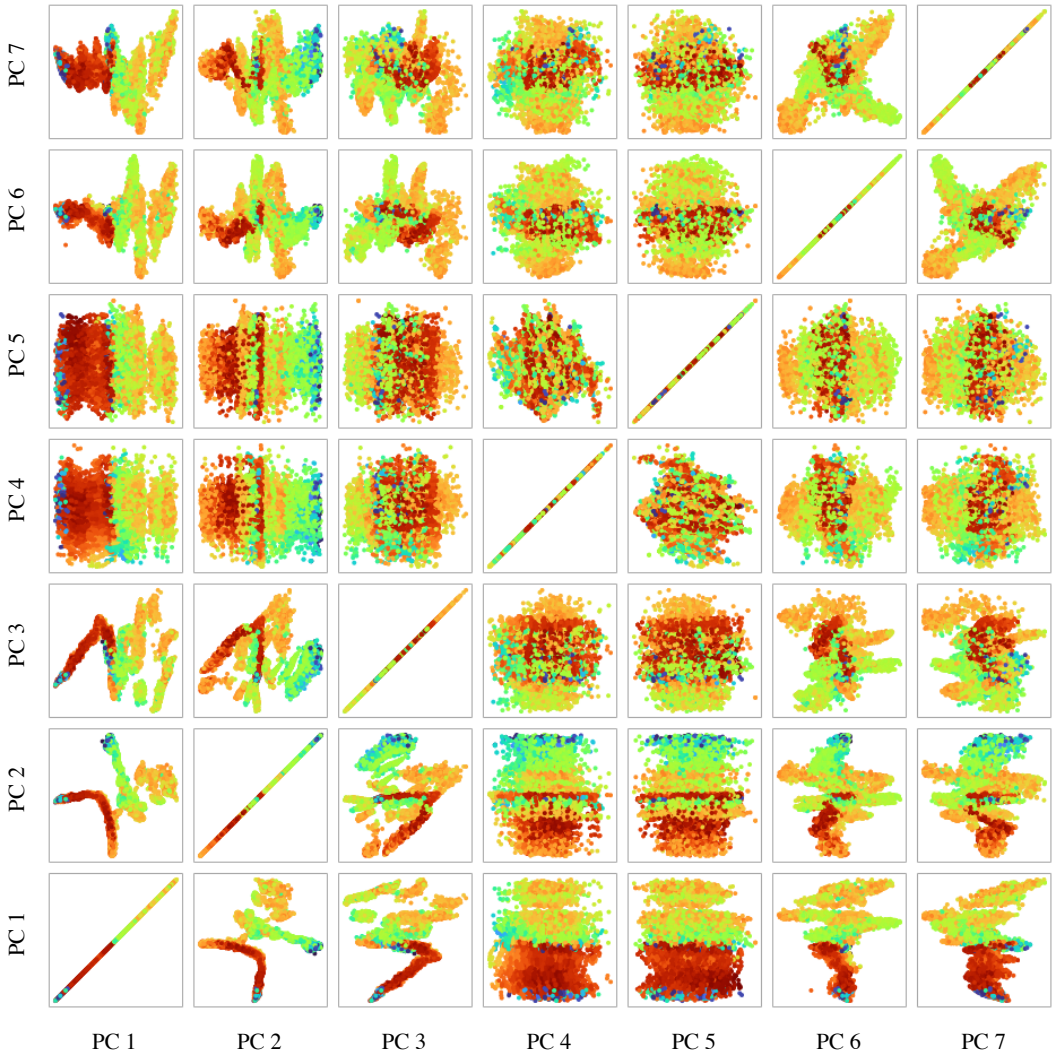


Figure 5.14: Visualization of the relationship between all 7 principal components of the representation of the mainly investigated SAC-ACCI agent. Especially, principal components 1, 4 and 5 span a plane in their shared subspace. While principal components 1 and 5 are meaningful as they describe the dynamics and camera variations, there is no direct evidence of a interpretable connection between principal component 4 and the FoV.

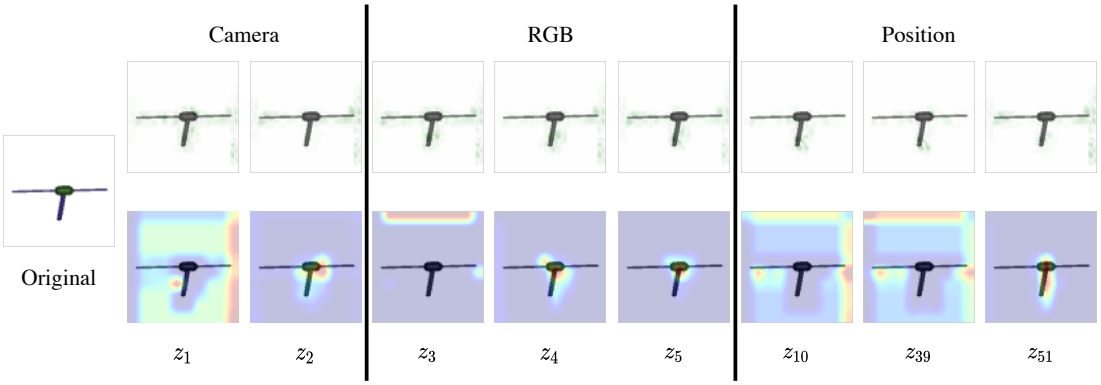


Figure 5.15: Saliency maps (top row) and Grad-Cam images (bottom row) for different latent dimensions of the main SAC-ACCI agent. The investigated latent dimensions correspond to those of the previous experiments. For both methods it can be seen that the agent mainly pays attention to the cart and the pendulum.

no pattern could be found in the subspace that describes the color of the cart, pendulum or rail, reinforcing the assumption of the last section that information about color is not encoded at all in this setting.

5.4.3 *Attribution*

So far, the experiments mainly focused on the decoder and the representation itself, but less on the encoder. However, since the encoder generates the representation and thus contributes to a large extent to the disentanglement, it should be examined as well. To analyze the importance of the input, methods like Saliency [SVZ14], Integrated Gradients [STY17] or Grad-Cam [Sel+17] can be used. Based on exemplary inputs, these methods seek to quantify the extent to which regions of a neural network are responsible for the output. In other words they are looking for parts which contribute to the output. [Anc+18]

For this experiment Saliency as well as Grad-Cam are utilized. The error between the total approximated and the true integrated gradients was too large in the Integrated Gradients methods. Therfore, this method was excluded for the experiment. These methods can be summarized as follows: Saliency maps show attribution values which are simply the absolute value of the partial derivative of the specified target output with respect to the input. Grad-Cam on the other hand is specialized on convolutional neural networks. Similar to Saliency, the gradients of the target outputs are calculated but with respect to a specified target convolutional layer. The gradients are then averaged over the channels of the layer. Afterwards, the result is multiplied with the layer activations computed in the forward pass beforehand. Both methods require a specified target output that is usually a class of interest in the classification layer of the network. In the context of this work, the values of interest are the latent dimensions of the bottleneck representation. Therefore, the algorithms are provided with specific latent dimensions as target outputs in the experiment. All visualizations are generated using *Captum* [Kok+20].

Figure 5.15 shows Grad-Cam results and Saliency maps for different latent dimensions that were investigated in the previous experiments already[3]. For Grad-Cam the penultimate convolutional layer is used for calculation of the gradients. Each of the individual investigations uses the same input frame, describing a state shortly after the beginning of an episode. For both methods it can be seen that the encoder mainly pays attention to the cart and the pendulum. This is plausible since the movement of both causes the most variation in the data. It is challenging to distinguish between the different Saliency maps as they remain mostly unchanged between the latent dimensions. However, the Grad-Cam images additionally show that for other latent dimensions, the encoder apparently looks at the residual area. Since this region remains mostly unchanged but still forms a large part of the image, it is likely that the encoder focuses on this area to reduce the log-likelihood objective. Due to the constant values in this area it might be possible for the VAE to encode this information in a small number of latent dimensions only. Focusing on this large area could also help to reconstruct the area of the pendulum and the rail more easily. The information provided by the saliency maps and Grad-Cam images does not assist in the interpretation of the disentanglement, as no dimension of the Grad-Cam images indicates that special attention is paid to any particular factor. Due to the similarity of the Saliency maps, no direct attention to specific factors is visible either. This conclusion is also the case for the weakly-supervised dimensions.

3 Additional visualizations of attribution can be found in Appendix A.3

6

CONCLUSION & OUTLOOK

DR has received much attention in recent years as a general method for transferring RL policies from a source domain to a different target domain, especially from simulations to the real-world. Many algorithmic advances have been claimed and DR is often used besides other extensions in robotic literature. On the other hand, the representations of agents trained with DR are barely interpretable. However, interpretability would help robotic agents in the transfer from the source domain to a different target domain, especially if the agent fails. The interpretable components would indicate the reason for the agent's failure.

Disentangled representations, on the other hand, attempt to capture the FoV in the data and are therefore readily interpretable. Previously, there have been few attempts only which combine RL and disentangled representations. However, these attempts mostly consider the two learning tasks in isolation from each other. Beyond that, a reference to DR is not established.

This thesis tackled the problem of learning control policies and disentangled representations jointly in the context of DR. Due to the similarities in the generative factors between domain randomization and disentangled representation learning, both methods were combined to obtain policies that generalize to different domains. Furthermore, recent literature on DR suggest to train policies on a curriculum instead of static DR distributions. In this case, the performance of the policy is used to adapt the parameters of the distributions. This idea was transferred to disentangled representation learning by building a framework that is able to train RL policies jointly with disentangling VAEs purely on images. Instead of using the policy performance on the task, the disentanglement performance of the encoder was used to adapt the parameters of the distributions. However, measuring the disentanglement performance of encoders is usually not trivial since the real-world FoV are not measurable without further supervision. In this case, the additional information provided by the simulation was used to incorporate weak supervision. On the one hand, weakly-supervised disentanglement enforces stronger disentanglement and on the other hand makes it also measurable. However, the encoder utilizes its capacity early in the training, which means that no capacity is left for novel factor values later during training. To this end, a controlled increase of the capacity based on the current DR distribution was used to ensure learning of factors also later during training.

The performance of the proposed method was benchmarked on a simulated CPSU task against different baselines. It was shown that the proposed method learns robust policies while maintaining a high level in disentanglement performance. But this depended largely on

the preprocessing strategy and the chosen network architecture. An additional simulation-to-simulation experiment for the same task in another physics engine demonstrated the power of the proposed method. The policies were able to solve the task even under this distributional shift in the observation space. Experiments on the interpretability of the learned representation showed that even with weak supervision not all factors can be learned and further methods have to consider especially those factors which are not as relevant as other factors for the reconstruction loss of the VAE. However, some of the factors can be learned much more disentangled using weak-supervision.

6.1 FUTURE INVESTIGATIONS & IDEAS

This section presents additional ideas that could not be investigated in this thesis and remain open problems which could be interesting for future work:

- The evaluation was performed on the source domain and an target domain consisting of an additional simulation environment based on the PyBullet physics engine. As the ultimate goal would be to transfer the method from the source domain to the real-world correspondence, as it is usually the case in the DR setting. Applying the method on real-world observations and collecting reconstructions would help identifying real-world factors that are not learned sufficient enough making the method especially interpretable in this setting.
- To ensure that the state is fully observable, the agent is provided with a stack of states in each step. The main method splits this stack for the VAE so that it processes the single frames individually. The resulting representations are concatenated together again to be provided to the policy which requires fully observable states. Another approach would be to include a recurrent network architecture either already in the encoder or in the policy. To this end, frame stacking would be unnecessary as the short-term history is provided by the hidden recurrent state.
- Currently, only visual parameters of the environment are randomized. However, this reflects only half of the sim-to-real gap. In addition, dynamics parameters should be randomized. In the weak-supervision setting of this thesis this imposes a problem for the generation of the differing image pairs because this would require two synced simulations since actions on the individual environments would result in different states due to the different dynamics. In the long term the simulation states would be such different that the assumptions implied by weak supervision would not apply anymore. Furthermore, it must be ensured that changing parameters at any step in time does not change the stability of the simulation. An approach would be to randomized these parameters for each episode and by optimizing the agent based on episodes in contrast to steps.
- The proposed method SAC-ACCI reconstructs the current observation s_t to use the resulting bottleneck representation as input for a model-free policy. In contrast, it would also be possible to adapt the method for a model-based approach by predicting

the next observation s_{t+1}. In this case this would also incorporate predicting future dynamics since future observations s_{t+1} could be different for different dynamics parameters allowing to randomize these parameters additional to the current one.

- The evaluation compares SAC-ACCI with other but also similar methods. Therefore, there is a lack of comparisons with methods that do not use VAEs. Such a larger study could included algorithms already listed in Chapter 3.
- Disentanglement could be advanced by incorporating inductive biases via additional structure in the representation. Similar to Toth et al. [Tot+20] the representation could be structured by differential equations like the equations of motion. Learning such structures would certainly improve the interpretability of the method.
- To investigate the interpretability of the representations, further experiments have to be executed. Ablation studies on the representation similar to Meyes et al. [Mey+19; MSM20] and Morcos et al. [Mor+18] should be performed to investigate the performance and dependency of the RL policy. In particular, the disentanglement of these representations could be tested with such experiments, as ablations of individual dimensions could expose the interdependence of the different dimensions.

BIBLIOGRAPHY

[Ach+18] Alessandro Achille et al. "Life-Long Disentangled Representation Learning with Cross-Domain Latent Homologies." In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/a0afdf1ac166b8652ffe9dee6eac779e-Paper.pdf.

[Akk+19] Ilge Akkaya et al. "Solving rubik's cube with a robot hand." In: *arXiv preprint arXiv:1910.07113* (2019).

[Anc+18] Marco Ancona et al. "Towards better understanding of gradient-based attribution methods for Deep Neural Networks." In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=Sy21R9JAW.

[And+19] OpenAI: Marcin Andrychowicz et al. "Learning dexterous in-hand manipulation." In: *The International Journal of Robotics Research* 39.1 (Nov. 2019), pp. 3–20. DOI: 10.1177/0278364919887447.

[Asi64] Isaac Asimov. *Visit to the World's Fair of 2014*. Accessed: 2010-10-04. Aug. 1964. URL: https://archive.nytimes.com/www.nytimes.com/books/97/03/23/lifetimes/asi-v-fair.html?wptouch_preview_theme=enabled.

[BCV13] Y. Bengio, A. Courville, and P. Vincent. "Representation Learning: A Review and New Perspectives." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (Aug. 2013), pp. 1798–1828. DOI: 10.1109/tpami.2013.50.

[Bea+16] Charles Beattie et al. "Deepmind lab." In: *arXiv preprint arXiv:1612.03801* (2016).

[Bel57] Richard Bellman. "A Markovian decision process." In: *Journal of mathematics and mechanics* (1957), pp. 679–684.

[Ber+19] Christopher Berner et al. "Dota 2 with large scale deep reinforcement learning." In: *arXiv preprint arXiv:1912.06680* (2019).

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization." In: *arXiv preprint arXiv:1607.06450* (2016).

[BKM17] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. "Variational Inference: A Review for Statisticians." In: *Journal of the American Statistical Association* 112.518 (Apr. 2017), pp. 859–877. DOI: 10.1080/01621459.2017.1285773.

[Bou+18] Konstantinos Bousmalis et al. "Using simulation and domain adaptation to improve efficiency of deep robotic grasping." In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 4243–4250.

[Bro+16] Greg Brockman et al. *OpenAI Gym*. 2016. eprint: arXiv:1606.01540.

[BTN18] Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. "Multi-Level Variational Autoencoder: Learning Disentangled Representations From Grouped Observations." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (Apr. 2018). URL: https://ojs.aaai.org/index.php/AAAI/article/view/11867.

[Bur+18] Christopher P Burgess et al. "Understanding disentangling in β-VAE." In: *arXiv preprint arXiv:1804.03599* (2018).

[Bur+19] Christopher P Burgess et al. "Monet: Unsupervised scene decomposition and representation." In: *arXiv preprint arXiv:1901.11390* (2019).

[CB21] Erwin Coumans and Yunfei Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. http://pybullet.org. 2016–2021.

[CH15] Mark Cutler and Jonathan P How. "Efficient reinforcement learning for robots using informative simulated priors." In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 2605–2612.

[Che+19] Yevgen Chebotar et al. "Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience." In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, May 2019. DOI: 10.1109/icra.2019.8793789.

[Chr+16] Paul Christiano et al. "Transfer from simulation to real world through learning deep inverse dynamics model." In: *arXiv preprint arXiv:1610.03518* (2016).

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. 2 ed. Wiley John + Sons, Sept. 2006. 792 pp. ISBN: 0471241954. DOI: https://doi.org/10.1002/047174882X. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/047174882X. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/047174882X.

[Dha+17] Prafulla Dhariwal et al. *OpenAI Baselines*. https://github.com/openai/baselines. 2017.

[Dit+20] Andrea Dittadi et al. "On the Transfer of Disentangled Representations in Realistic Settings." In: *International Conference on Learning Representations*. 2020.

[DMH19] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. "Challenges of real-world reinforcement learning." In: *arXiv preprint arXiv:1904.12901* (2019).

[EW18] Cian Eastwood and Christopher KI Williams. "A framework for the quantitative evaluation of disentangled representations." In: *International Conference on Learning Representations*. 2018.

[Fin+17] Chelsea Finn et al. "One-Shot Visual Imitation Learning via Meta-Learning." In: *CoRL*. 2017.

[FL17] Chelsea Finn and Sergey Levine. "Deep visual foresight for planning robot motion." In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 2786–2793.

[FLA16] Justin Fu, Sergey Levine, and Pieter Abbeel. "One-shot learning of manipulation skills with online dynamics adaptation and neural network priors." In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 4019–4026.

[Fra+18] Vincent François-Lavet et al. "An Introduction to Deep Reinforcement Learning." In: *Foundations and Trends® in Machine Learning* 11.3-4 (2018), pp. 219–354. DOI: 10.1561/2200000071.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[GG14] Samuel Gershman and Noah Goodman. "Amortized inference in probabilistic reasoning." In: *Proceedings of the annual meeting of the cognitive science society*. Vol. 36. 36. 2014.

[Haa+18a] Tuomas Haarnoja et al. *Soft Actor-Critic Algorithms and Applications*. Tech. rep. 2018.

[Haa+18b] Tuomas Haarnoja et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor." In: *International conference on machine learning*. PMLR. 2018, pp. 1861–1870.

[Haf+19] Danijar Hafner et al. "Learning Latent Dynamics for Planning from Pixels." In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 2555–2565. URL: https://proceedings.mlr.press/v97/hafner19a.html.

[Haf+20] Danijar Hafner et al. "Dream to Control: Learning Behaviors by Latent Imagination." In: *International Conference on Learning Representations*. 2020.

[Hes+18] Matteo Hessel et al. "Rainbow: Combining improvements in deep reinforcement learning." In: *Thirty-second AAAI conference on artificial intelligence*. 2018.

[HGS16] Hado van Hasselt, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning." In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI'16. Phoenix, Arizona: AAAI Press, 2016, pp. 2094–2100.

[Hig+17a] I. Higgins et al. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework." In: *International Conference on Learning Representations*. 2017.

[Hig+17b] Irina Higgins et al. "DARLA: Improving Zero-Shot Transfer in Reinforcement Learning." In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1480–1490. URL: https://proceedings.mlr.press/v70/higgins17a.html.

[Hig+18] Irina Higgins et al. "Towards a definition of disentangled representations." In: *arXiv preprint arXiv:1812.02230* (2018).

[Hof+18] Judy Hoffman et al. "Cycada: Cycle-consistent adversarial domain adaptation." In: *International conference on machine learning*. PMLR. 2018, pp. 1989–1998.

[Hos19] Haruo Hosoya. "Group-Based Learning of Disentangled Representations with Generalizability for Novel Contents." In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. IJCAI'19. Macao, China: AAAI Press, 2019, pp. 2506–2513. ISBN: 9780999241141.

[HR21] Yordan Hristov and Subramanian Ramamoorthy. "Learning from Demonstration with Weakly Supervised Disentanglement." In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=LdaugeHU-qO.

[HS18] David Ha and Jürgen Schmidhuber. "Recurrent World Models Facilitate Policy Evolution." In: *Advances in Neural Information Processing Systems 31*. https://worldmodels.github.io. Curran Associates, Inc., 2018, pp. 2451–2463. URL: https://papers.nips.cc/paper/7512-recurrent-world-models-facilitate-policy-evolution.

[Hwa+19] Jemin Hwangbo et al. "Learning agile and dynamic motor skills for legged robots." In: *Science Robotics* 4.26 (Jan. 2019), eaau5872. DOI: 10.1126/scirobotics.aau5872.

[Jeo+19] Rae Jeong et al. "Self-Supervised Sim-to-Real Adaptation for Visual Robotic Manipulation." In: *arXiv preprint arXiv:1910.09470* (2019).

[Joh+16] Matthew Johnson et al. "The Malmo Platform for Artificial Intelligence Experimentation." In: *25th International Joint Conference on Artificial Intelligence (IJCAI-16)*. AAAI - Association for the Advancement of Artificial Intelligence, July 2016. URL: https://www.microsoft.com/en-us/research/publication/malmo-platform-artificial-intelligence-experimentation/.

[Jor+99] Michael I Jordan et al. "An introduction to variational methods for graphical models." In: *Machine learning* 37.2 (1999), pp. 183–233.

[Kal+18] Dmitry Kalashnikov et al. "QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation." In: *Proceedings of The 2nd Conference on Robot Learning*. 2018.

[Kok+20] Narine Kokhlikyan et al. *Captum: A unified and generic model interpretability library for PyTorch*. 2020. arXiv: 2009.07896 [cs.LG].

[KW+19] Diederik P Kingma, Max Welling, et al. "An Introduction to Variational Autoencoders." In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392.

[KW14] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes." In: *2nd International Conference on Learning Representations, (ICLR) 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2014. URL: http://arxiv.org/abs/1312.6114.

[Lev+16] Sergey Levine et al. "End-to-end training of deep visuomotor policies." In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.

[Lev+18] Sergey Levine et al. "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection." In: *The International Journal of Robotics Research* 37.4-5 (2018), pp. 421–436.

[Lju91] L. Ljung. "Issues in system identification." In: *IEEE Control Systems Magazine* 11.1 (1991), pp. 25–29. DOI: 10.1109/37.103346.

[Lju99] Lennart Ljung. "System Identification." In: *Wiley Encyclopedia of Electrical and Electronics Engineering*. American Cancer Society, 1999. ISBN: 9780471346081. DOI: https://doi.org/10.1002/047134608X.W1046. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/047134608X.W1046. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/047134608X.W1046.

[Loc+19] Francesco Locatello et al. "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations." In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 4114–4124. URL: https://proceedings.mlr.press/v97/locatello19a.html.

[Loc+20] Francesco Locatello et al. "Weakly-Supervised Disentanglement Without Compromises." In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 6348–6359. URL: https://proceedings.mlr.press/v119/locatello20a.html.

[Loc20] Francesco Locatello. "Enforcing and Discovering Structure in Machine Learning." en. PhD thesis. 2020. DOI: 10.3929/ETHZ-B-000474164.

[LSA20] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. "Curl: Contrastive unsupervised representations for reinforcement learning." In: *International Conference on Machine Learning*. PMLR. 2020, pp. 5639–5650.

[Meh+20] Bhairav Mehta et al. "Active domain randomization." In: *Conference on Robot Learning*. PMLR. 2020, pp. 1162–1176.

[Mey+19] Richard Meyes et al. "Ablation studies in artificial neural networks." In: *arXiv preprint arXiv:1901.08644* (2019).

[Mni+13] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning." In: *arXiv preprint arXiv:1312.5602* (2013).

[Mni+15] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning." In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. DOI: 10.1038/nature14236.

[Mni+16] Volodymyr Mnih et al. "Asynchronous methods for deep reinforcement learning." In: *International conference on machine learning*. 2016, pp. 1928–1937.

[Mor+18] Ari S. Morcos et al. "On the importance of single directions for generalization." In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=r1iuQjxCZ.

[MSM20] Richard Meyes, Moritz Schneider, and Tobias Meisen. "How Do You Act? An Empirical Study to Understand Behavior of Deep Reinforcement Learning Agents." In: *arXiv preprint arXiv:2004.03237* (2020).

[Mur+20] Fabio Muratore et al. "Bayesian Domain Randomization for Sim-to-Real Transfer." In: *arXiv preprint arXiv:2003.02471* (2020).

[Nai+18] Ashvin V Nair et al. "Visual Reinforcement Learning with Imagined Goals." In: *Advances in Neural Information Processing Systems* 31 (2018), pp. 9191–9200.

[PC14] Novi Patricia and Barbara Caputo. "Learning to Learn, from Transfer Learning to Domain Adaptation: A Unifying Perspective." In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2014. DOI: 10.1109/cvpr.2014.187.

[Pen+18] X. Peng et al. "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018), pp. 1–8.

[Pin+17] Lerrel Pinto et al. "Robust Adversarial Reinforcement Learning." In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, June 2017, pp. 2817–2826. URL: http://proceedings.mlr.press/v70/pinto17a.html.

[Pin+18] Lerrel Pinto et al. "Asymmetric Actor Critic for Image-Based Robot Learning." In: *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, June 2018. DOI: 10.15607/rss.2018.xiv.008.

[PS08] Jan Peters and Stefan Schaal. "Reinforcement learning of motor skills with policy gradients." In: *Neural Networks* 21.4 (May 2008), pp. 682–697. DOI: 10.1016/j.neunet.2008.02.003.

[RM18] Karl Ridgeway and Michael C Mozer. "Learning deep disentangled embeddings with the F-statistic loss." In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 2018, pp. 185–194.

[RN09] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. USA: Prentice Hall Press, 2009. ISBN: 0136042597.

[RSC18] Nataniel Ruiz, Samuel Schulter, and Manmohan Chandraker. "Learning to simulate." In: *arXiv preprint arXiv:1810.02513* (2018).

[Rus+17] Andrei A Rusu et al. "Sim-to-real robot learning from pixels with progressive nets." In: *Conference on Robot Learning*. PMLR. 2017, pp. 262–270.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Sel+17] Ramprasaath R. Selvaraju et al. "Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization." In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[Sha48] C. E. Shannon. "A Mathematical Theory of Communication." In: *Bell System Technical Journal* 27.3 (July 1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[Shu+20] Rui Shu et al. "Weakly Supervised Disentanglement with Guarantees." In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=HJgSwyBKvr.

[Sil+16] David Silver et al. "Mastering the game of Go with deep neural networks and tree search." In: *Nature* 529.7587 (Jan. 2016), pp. 484–489. DOI: 10.1038/nature16961.

[Sil+17a] David Silver et al. "Mastering chess and shogi by self-play with a general reinforcement learning algorithm." In: *arXiv preprint arXiv:1712.01815* (2017).

[Sil+17b] David Silver et al. "Mastering the game of Go without human knowledge." In: *Nature* 550.7676 (Oct. 2017), pp. 354–359. DOI: 10.1038/nature24270.

[Sla+19] Reda Bahi Slaoui et al. "Robust Visual Domain Randomization for Reinforcement Learning." In: *arXiv preprint arXiv:1910.10537* (2019).

[Sto+21] Adam Stooke et al. "Decoupling representation learning from reinforcement learning." In: *International Conference on Machine Learning*. PMLR. 2021, pp. 9870–9879.

[STY17] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic Attribution for Deep Networks." In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 3319–3328.

[SVZ14] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps." In: *In Workshop at International Conference on Learning Representations*. Citeseer. 2014.

[Sze10] Csaba Szepesvári. "Algorithms for Reinforcement Learning." In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 4.1 (Jan. 2010), pp. 1–103. DOI: 10.2200/s00268ed1v01y201005aim009.

[Tan+18] Jie Tan et al. "Sim-to-real: Learning agile locomotion for quadruped robots." In: *arXiv preprint arXiv:1804.10332* (2018).

[Tas+20] Yuval Tassa et al. *dm-control: Software and Tasks for Continuous Control*. 2020. arXiv: 2006.12983 [cs.RO].

[Tea19] Deepmind Team AlphaStar. "Alphastar: Mastering the real-time strategy game starcraft ii." In: *DeepMind blog* 24 (2019).

[TET12] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control." In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2012. DOI: 10.1109/iros.2012.6386109.

[Tob+17] Josh Tobin et al. "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)*. IEEE, Sept. 2017. DOI: 10.1109/iros.2017.8202133.

[Tot+20] Peter Toth et al. "Hamiltonian Generative Networks." In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=HJenn6VFvB.

[TPB99] Naftali Tishby, Fernando C. Pereira, and William Bialek. "The information bottleneck method." In: *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*. 1999, pp. 368–377. URL: https://arxiv.org/abs/physics/0004057.

[Trä+21] Frederik Träuble et al. "Representation Learning for Out-of-distribution Generalization in Reinforcement Learning." In: *ICML 2021 Workshop on Unsupervised Reinforcement Learning*. 2021.

[Tze+20] Eric Tzeng et al. "Adapting deep visuomotor representations with weak pairwise constraints." In: *Algorithmic Foundations of Robotics XII*. Springer, 2020, pp. 688–703.

[VPV+09] Laurens Van Der Maaten, Eric Postma, Jaap Van den Herik, et al. "Dimensionality reduction: a comparative." In: *J Mach Learn Res* 10.66-71 (2009), p. 13.

[Wan+16] Ziyu Wang et al. "Dueling Network Architectures for Deep Reinforcement Learning." In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1995–2003. URL: https://proceedings.mlr.press/v48/wangf16.html.

[Wil92] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." In: *Machine learning* 8.3-4 (1992), pp. 229–256.

[Wul+20] Markus Wulfmeier et al. "Representation matters: Improving perception and exploration for robotics." In: *arXiv preprint arXiv:2011.01758* (2020).

[Yan+19] John Yang et al. "Towards Governing Agent's Efficacy: Action-Conditional β-VAE for Deep Transparent Reinforcement Learning." In: *Proceedings of The Eleventh Asian Conference on Machine Learning*. Ed. by Wee Sun Lee and Taiji Suzuki. Vol. 101. Proceedings of Machine Learning Research. PMLR, 17–19 Nov 2019, pp. 32–47. URL: https://proceedings.mlr.press/v101/yang19a.html.

[Yar+21] Denis Yarats et al. "Improving Sample Efficiency in Model-Free Reinforcement Learning from Images." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 12. 2021, pp. 10674–10681. URL: https://github.com/denisyarats/pytorch_sac_ae.

[YLT19] Wenhao Yu, C. Karen Liu, and Greg Turk. "Policy Transfer with Strategy Optimization." In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=H1g6osRcFQ.

[Zen+19] Andy Zeng et al. "TossingBot: Learning to Throw Arbitrary Objects with Residual Physics." In: *Robotics: Science and Systems XV*. Robotics: Science and Systems Foundation, June 2019. DOI: 10.15607/rss.2019.xv.004.

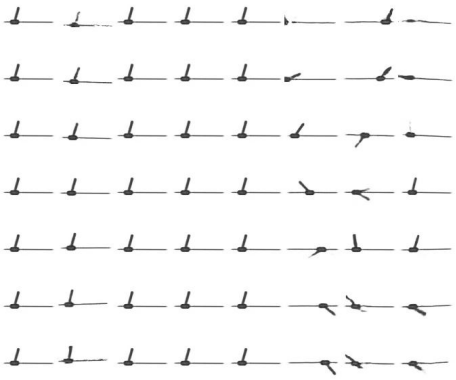
[Zhu+18] Yuke Zhu et al. "Reinforcement and imitation learning for diverse visuomotor skills." In: *arXiv preprint arXiv:1802.09564* (2018).

[Zhu+20] Henry Zhu et al. "The Ingredients of Real World Robotic Reinforcement Learning." In: *International Conference on Learning Representations*. 2020.

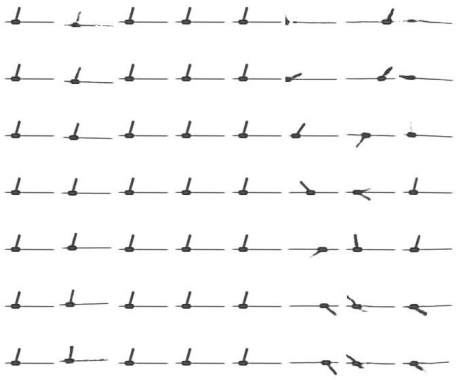
A

APPENDIX

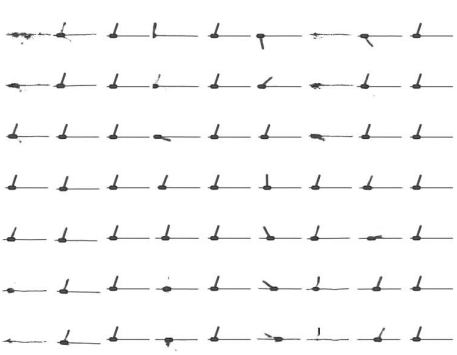
A.1 LATENT TRAVERSAL



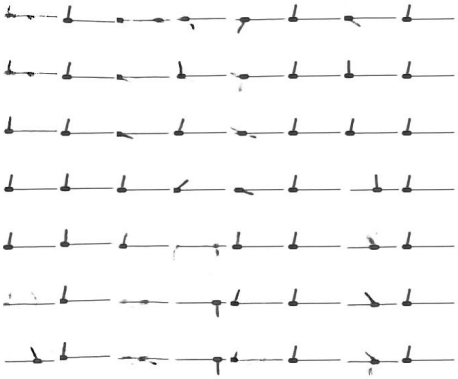
(a) SAC-ACCI - $\gamma = 500$ (Seed 1010)



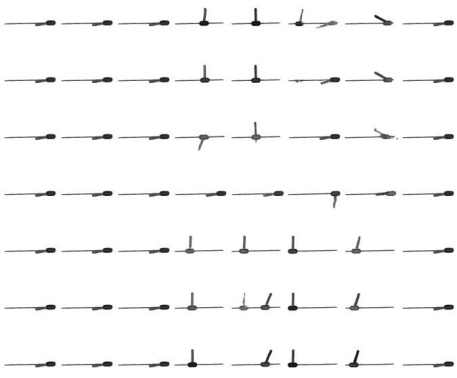
(b) SAC-ACCI - $\gamma = 500$ (Seed 1010)



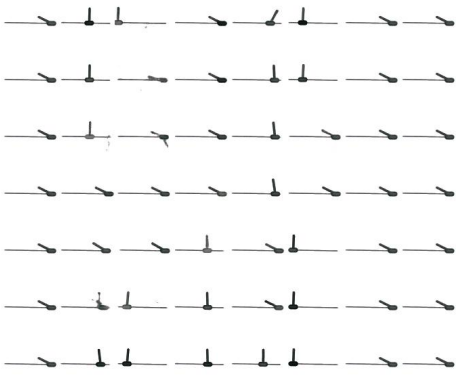
(c) SAC-ACCI - $\gamma = 100$ (Seed 6131)



(d) SAC-ACCI - $\gamma = 100$ (Seed 1010)

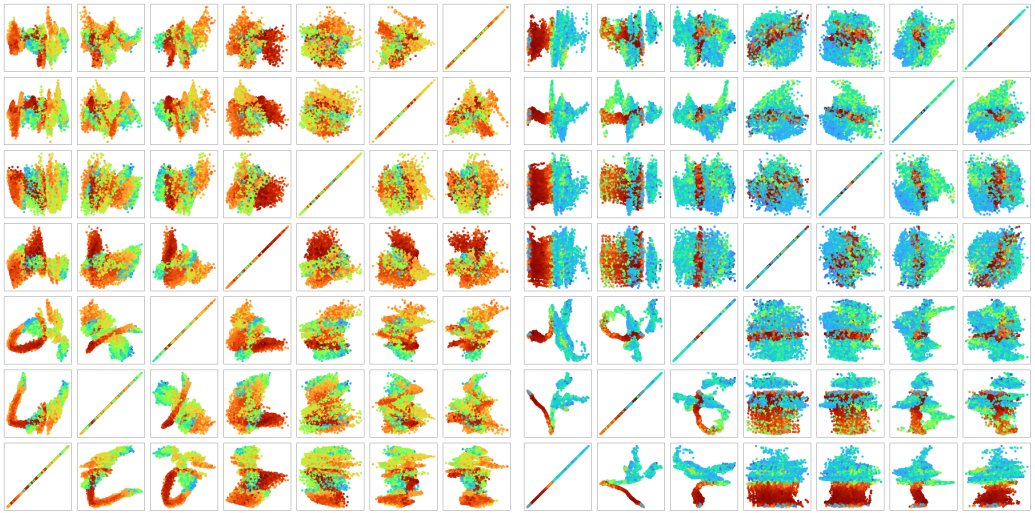


(e) SAC-DARLA - $\beta = 4$ (Seed 5352)



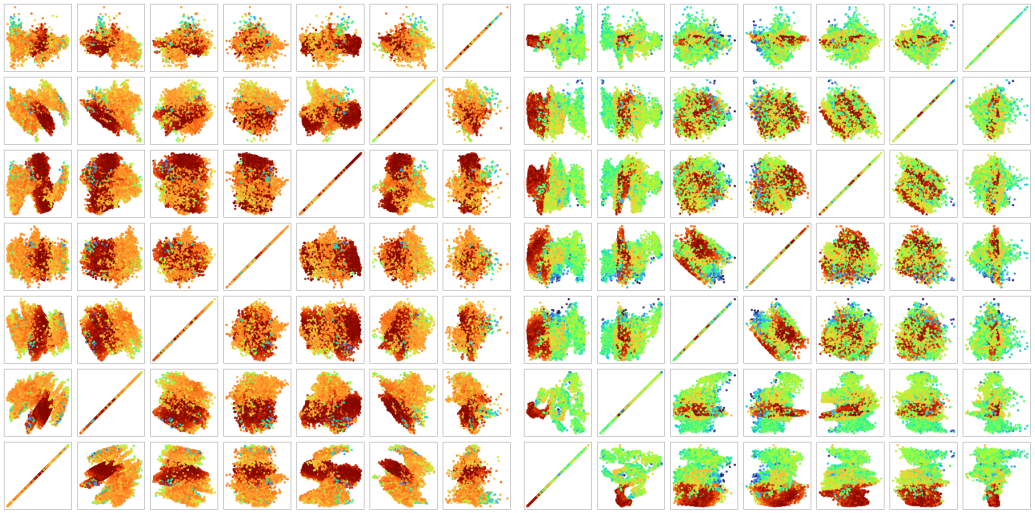
(f) SAC-DARLA - $\beta = 4$ (Seed 9048)

A.2 PCA



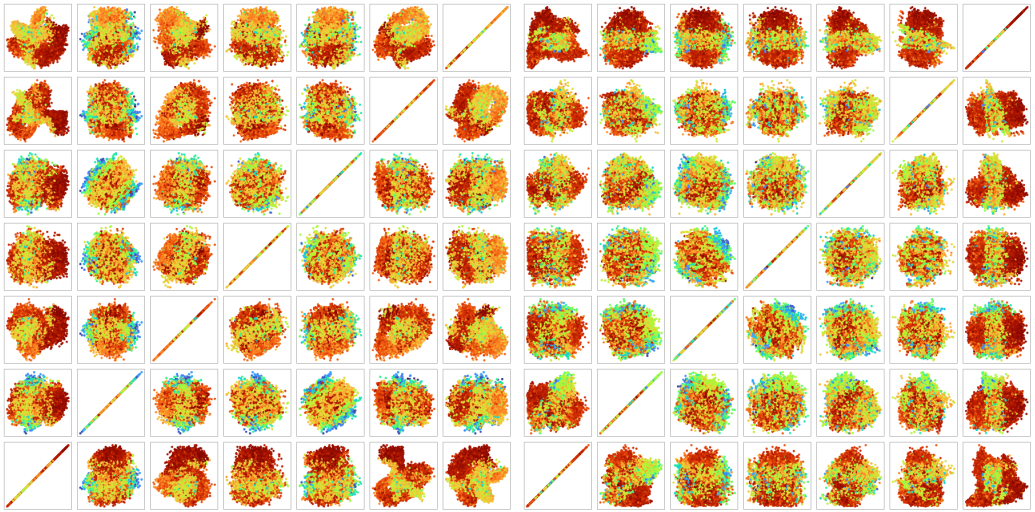
(a) SAC-ACCI - $\gamma = 500$ (Seed 5352)

(b) SAC-ACCI - $\gamma = 500$ (Seed 8874)



(c) SAC-ACCI - $\gamma = 100$ (Seed 3110)

(d) SAC-ACCI - $\gamma = 100$ (Seed 8874)

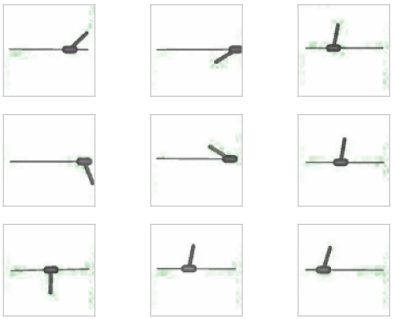


(e) SAC-DARLA - $\beta = 4$ (Seed 5352)

(f) SAC-DARLA - $\beta = 4$ (Seed 6131)

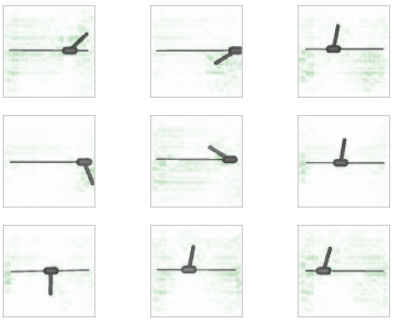
A.3 ATTRIBUTION

A.3.1 *Saliency*

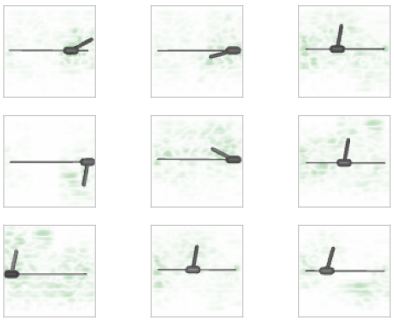


(a) SAC-ACCI - $\gamma = 500$ (Seed 1010)

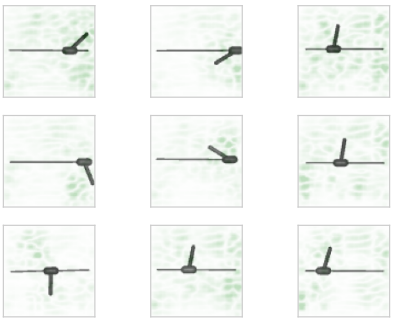
(b) SAC-ACCI - $\gamma = 500$ (Seed 1010)



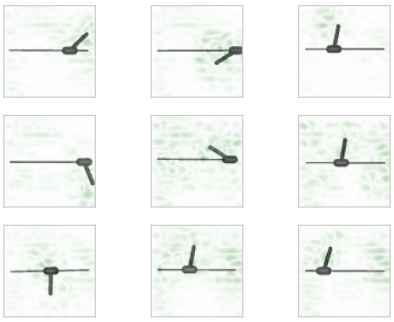
(c) SAC-ACCI - $\gamma = 100$ (Seed 1010)



(d) SAC-ACCI - $\gamma = 100$ (Seed 6131)

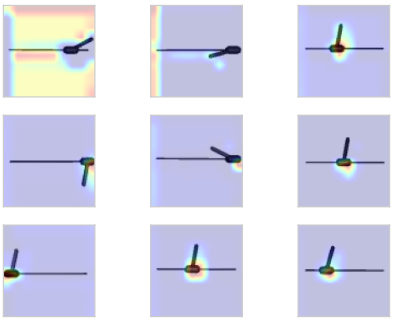


(e) SAC-DARLA - $\beta = 4$ (Seed 3110)

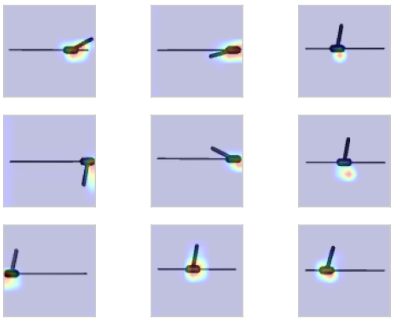


(f) SAC-DARLA - $\beta = 4$ (Seed 5352)

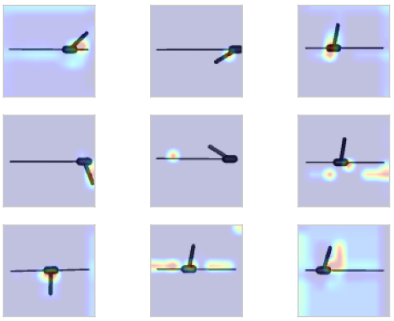
A.3.2 *GradCam*



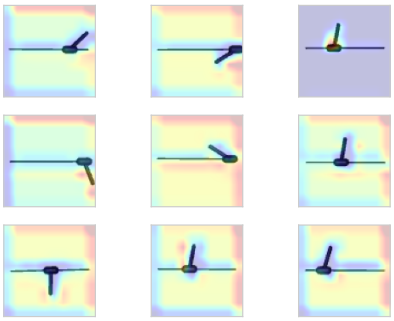
(a) SAC-ACCI - $\gamma = 500$ (Seed 1010)



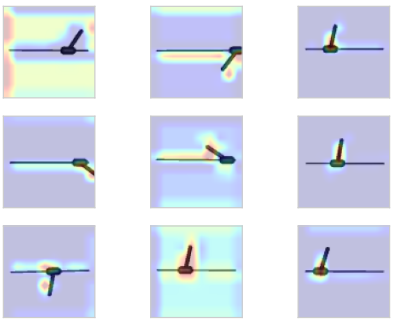
(b) SAC-ACCI - $\gamma = 500$ (Seed 1010)



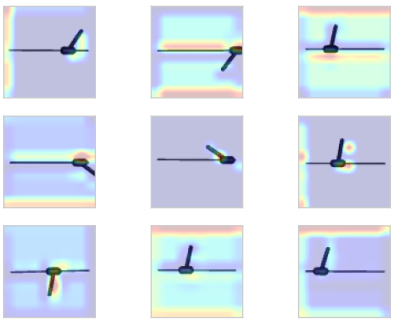
(c) SAC-ACCI - $\gamma = 100$ (Seed 5352)



(d) SAC-ACCI - $\gamma = 100$ (Seed 8874)



(e) SAC-DARLA - $\beta = 4$ (Seed 5352)



(f) SAC-DARLA - $\beta = 4$ (Seed 9048)

Zentrales Prüfungsamt/Central Examination Office

Eidesstattliche Versicherung
Statutory Declaration in Lieu of an Oath

Schneider, Moritz | 345827

Name, Vorname/Last Name, First Name | Matrikelnummer (freiwillige Angabe) Matriculation No. (optional)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel
I hereby declare in lieu of an oath that I have completed the present paper/Bachelor thesis/Master thesis* entitled

Interpretable Domain Randomization for Reinforcement Learning with Disentangled Representations

selbstständig und ohne unzulässige fremde Hilfe (insbes. akademisches Ghostwriting) erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
independently and without illegitimate assistance from third parties (such as academic ghostwriters). I have used no other than the specified sources and aids. In case that the thesis is additionally submitted in an electronic format, I declare that the written and electronic versions are fully identical. The thesis has not been submitted to any examination body in this, or similar, form.

Aachen, 05.10.2021

Ort, Datum/City, Date | Unterschrift/Signature

*Nichtzutreffendes bitte streichen

*Please delete as appropriate

Belehrung:
Official Notification:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.
Para. 156 StGB (German Criminal Code): False Statutory Declarations
Whoever before a public authority competent to administer statutory declarations falsely makes such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment not exceeding three years or a fine.
§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.
Para. 161 StGB (German Criminal Code): False Statutory Declarations Due to Negligence
(1) If a person commits one of the offences listed in sections 154 through 156 negligently the penalty shall be imprisonment not exceeding one year or a fine.
(2) The offender shall be exempt from liability if he or she corrects their false testimony in time. The provisions of section 158 (2) and (3) shall apply accordingly.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:
I have read and understood the above official notification:

Aachen, 05.10.2021

Ort, Datum/City, Date | Unterschrift/Signature