Disentangling Latent Representations in Non-Stationary Reinforcement Learning

Rody Haket

Disentangling Latent Representations in Non-Stationary Reinforcement Learning

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Rody Haket
born in Voorburg, the Netherlands

Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

Disentangling Latent Representations in Non-Stationary Reinforcement Learning

Author: Rody Haket
Student id: 4720849

Abstract

Model-free deep reinforcement learning has shown remarkable promise in solving highly complex sequential decision-making problems. However, the widespread adoption of reinforcement learning algorithms has not materialized in real-world applications such as robotics. A primary challenge is the general assumption that environments remain stationary at deployment. This problem is exacerbated when agents rely on pixel-based observations, dramatically increasing task complexity. As a result, these algorithms often fail when environments change over time. The perspective that agents should learn a disentangled representation has already been shown to be effective in improving generalization to domain shifts. We extend previous work by introducing Generalized Disentanglement (GED), an auxiliary contrastive learning task that encourages pixel-based deep reinforcement learning algorithms to isolate factors of variation in the data by leveraging temporal structure. We show that our methodology can improve generalization to unseen domains in several environments.

Thesis Committee:

Chair:	Prof. dr. M.T.J. Spaan, Sequential Decision Making, TU Delft
Daily supervisor:	MSc. O. Azizi, Sequential Decision Making, TU Delft
Committee Member:	Dr. ir. J.H. Krijthe, Pattern Recognition and Bioinformatics, TU Delft

Preface

I would like to express gratitude to my supervisor Matthijs for his guidance and support throughout the project. A special thanks goes to my daily supervisor, Oussama, for our weekly discussions and your insightful feedback. I would like to thank Jesse for his role on the thesis committee. Lastly, I am deeply grateful for my family and friends who supported me throughout my academic journey.

Rody Haket
Delft, the Netherlands
September 17, 2025

Contents

Chapter 1

Introduction

Humans, in general, are incredibly capable of dealing with the ever-changing, uncertain world around us. For example, a person driving a vehicle can adapt to all types of circumstances, such as erratic driving behavior, oncoming traffic, or fluctuating weather conditions. We readily adjust to all types of situations during which we abstract away all kinds of irrelevant details, such as the color and the model of the surrounding vehicles. Autonomous systems, too, should understand which elements of the environment can be ignored and which require attention to ensure safe decision-making. This is especially critical in robotics and full self-driving, where spurious correlations can lead to incorrect decisions, with potentially catastrophic consequences.

Hence, much research has taken the perspective that a human-like structural understanding of the world would also improve deep learning systems (Bengio et al., 2014; Lake et al., 2017; Schölkopf et al., 2021). One of the promising mechanisms to encourage deep neural networks to learn such a structure of the environment is by disentangling the independent factors of variation, also referred to as factors or variables, in the data. In the example above, the factors of variation include the color, shape, and size of the vehicles, but also driving behavior. Learning to represent these factors separately allows the model to adapt to changes in one factor while maintaining its knowledge about others. This improves generalization and transferability to unseen domains (Higgins et al., 2018; Schölkopf et al., 2021).

Previous work by Dunion et al. (2023b) proposed Temporal Disentanglement (TED), an auxiliary learning task to disentangle latent representations in pixel-based model-free deep reinforcement learning (RL). TED encourages the isolation of independent factors of variation by adapting Permutation-Contrastive Learning (PCL) (Hyvärinen and Morioka, 2017) to the RL setting. PCL discriminates between temporal and non-temporal pairs of latent representations using a regression function that models temporally dependent stationary sources. Their work showed that disentangled representations effectively increased robustness and improved generalization to unseen variations of environments.

Related to PCL is the work on time-contrastive learning (TCL) (Hyvärinen and Morioka, 2016) that showed that non-stationary factors of variation can be recovered by learning a classifier to discriminate between time windows. Subsequent research unifies PCL and TCL into a single general framework called Nonlinear Independent Component Analysis (Nonlinear ICA) with Auxiliary Variables (Hyvärinen et al., 2019). This framework assumes that factors of variation are modulated by auxiliary variables that can be leveraged to identify the original signals, holding the potential to disentangle a wider range of sources.

This thesis investigates how temporal structure can be exploited to disentangle pixel-based representations in non-stationary deep reinforcement learning. We introduce Generalized Disentanglement, an auxiliary contrastive task that encourages the agent to separate factors of variation in its learned representation. We evaluate our approach in a wider range of settings to gain more insight into the strengths and limitations of disentangled representation learning in non-stationary RL. We consider using various choices of negative contrastive pairs. Not only can this simplify implementation, our experiments also show that this can greatly minimize the generalization gap for some tasks. Additionally, our methodology demonstrates stable training, improving performance and generalization in non-stationary RL.

1.1 Scope

Non-stationary RL is a broad area of study. Non-stationarity can be demarcated into passive non-stationarity and active non-stationarity (Khetarpal et al., 2022). Passive non-stationarity refers to changes that happen regardless of the agent, whereas active non-stationary changes happen as a consequence of the behavior of the agent. Although reinforcement learning algorithms naturally induce active non-stationary changes in the environment, our experimental settings focus on additional passive non-stationarity. First, we consider the general problem of domain shift. Specifically, the agent encounters a shift in the observations, but the fundamental relationship between the state and the optimal policy is preserved. We refer to this setting as generalization to task-irrelevant factors. Secondly, we investigate the problem of adapting to intra-episodic shifts in the state dynamics. We characterize this as generalization to task-relevant factors. In the two aforementioned settings, we additionally focus on how modulating the factors using non-stationary functions affects performance and generalization. We focus on two types of non-stationarity that are commonly studied in non-stationary RL (Dulac-Arnold et al., 2021; Khetarpal et al., 2022): (i) abrupt change: the factor of variation is modulated according to a step function during an episode with uniformly distributed initial values at the start of each episode and (ii) gradual change: the factor varies according to a cyclic function during an episode with uniformly distributed starting values between episodes. Lastly, we study generalization to unseen variations of an environment. We refer to this as generalization to task-irrelevant and task-relevant factors.

1.2 Research Objective

This work concerns itself with recovering latent variables from the agent's observations by learning a disentangled representation. We specifically focus on applying the Nonlinear ICA with Auxiliary Variables (Hyvärinen et al., 2019) framework to this problem setting. We hypothesize that disentangling the latent representation allows the agent to learn a policy that can adapt to changes in the environment faster. This should improve performance in non-stationary environments as well as generalization to novel domains.

1.3 Research Questions

To achieve our research objective, we ask the following main research question:

How can disentangled representations improve generalization in non-stationary reinforcement learning?

To address our central research question we present Generalized Disentanglement, an auxiliary contrastive learning task to disentangle latent representations in pixel-based reinforcement learning. Since the agent is encouraged to learn a disentangled representation, the agent is more robust to domain shifts, increasing performance and generalization. We explore the main question through the following subquestions:

(RQ1): *How do disentangled representations improve robustness to task-irrelevant factors?*

To investigate this research question, we evaluate how well the policy generalizes when the agent suddenly observes inputs from a different domain, characterized by the task-irrelevant color of the environment. We also refer to the color factor as the distractor in the environment. We explore generalization in three different settings. We evaluate the setting where the color (i) is constant throughout an episode, (ii) changes gradually during an episode, and (iii) changes abruptly within an episode.

(RQ2): *How do disentangled representations improve adaptation to task-relevant factors?*

To explore this research question, we evaluate our methodology in an environment that contains a task-relevant factor that influences the dynamics of the environment throughout an episode. Here, too, we investigate whether the policy is able to adapt to gradual and abrupt changes in the dynamics.

(RQ3): *How do disentangled representations enhance generalization to unseen variations of environments?*

To answer this research question, we investigate our methodology in environments that contain visual task-relevant and task-irrelevant variables. After training in the train domain, we switch to the test domain and evaluate the performance of the agent allowing for continued gradient updates to test how quickly the agent recovers.

1.4 Contributions

The contributions of this work are as follows:

- We extend Temporal Disentanglement to Generalized Disentanglement to learn a disentangled representation of the environment. Our methodology is more flexible and robust, concomitantly improving performance and generalization in non-stationary RL.
- We empirically validate our approach in a wide variety of settings, investigating generalization to changes in task-relevant and task-irrelevant variables of the environment. We provide a detailed overview and discussion of the results.

1.5 Thesis Outline

The remainder of this thesis is structured as follows:

- **Chapter 2** provides a summary of the preliminary knowledge that is foundational to the main methodology. First, we discuss the basics of reinforcement learning. Secondly, we cover aspects of representation learning in RL. Lastly, we describe the theoretical framework of Nonlinear Independent Component Analysis.
- **Chapter 3** discusses the methodology of our approach. We provide a description of the problem setting, the architecture, the learning algorithm, and the optimization objective.
- **Chapter 4** describes the experimental setup of our research. We describe environments, reinforcement learning algorithms, high-level experimental settings, evaluation criteria, and hyperparameters.
- **Chapter 5** presents the results of our experiments. First, we present the results and discuss the experiments with task-irrelevant factors of variation. Next, we present and discuss the experimental results with task-relevant factors. Finally, we present and discuss the results of the experiments with both task-relevant and task-irrelevant variables.
- **Chapter 6** provides a more in-depth discussion of the research. Specifically, we address the interpretation of results and limitations.
- **Chapter 7** examines related research in the field of non-stationary reinforcement learning. We focus on specific techniques that address the problem of non-stationary reinforcement learning. We briefly describe their high-level ideas, their significance to our work, and their differences.
- **Chapter 8** concludes this thesis with a summary of our findings and key contributions.
- **Chapter 9** discusses several directions for future research.

Chapter 2

Background

This chapter covers the relevant background necessary to understand the main contribution of this thesis. We start this chapter with an introduction to reinforcement learning in Section 2.1. Next, we discuss representation learning and its relevance to this thesis in Section 2.2. Lastly, we describe the general Nonlinear Independent Component Analysis framework in Section 2.3.

2.1 Reinforcement Learning

Reinforcement learning is a learning paradigm that frames the system as an environment in which an agent perceives an observation, takes an action based on this observation, and receives a reward that corresponds to the transition to the new state of the agent. Through repeated interaction, the agent learns which action to take in each state. Next, we cover the basic aspects of reinforcement learning and describe state-of-the-art actor-critic algorithms.

2.1.1 Partially Observable Markov Decision Process

The Partially Observable Markov Decision Process (POMDP) is a common model for environments where the true state is not directly observed (Wiering and Van Otterlo, 2012). Formally, the POMDP framework is a six-tuple (S, A, Ω, T, O, R). The symbol S represents the finite set of states, A is the finite set of actions and Ω denotes the finite set of observations. The transition function $T : S \times A \times S \to [0,1]$ is the distribution $P(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})$, where $\mathbf{s}'$ is the next state after executing action $\mathbf{a}$ in state $\mathbf{s}$. At each step, the agent perceives an observation $\mathbf{o} \in \Omega$, sampled from the observation function $O : S \times A \times \Omega \to [0,1]$ with probability $O(\mathbf{s}', \mathbf{a}, \mathbf{o})$. The agent receives a reward after a transition from state $\mathbf{s}$ to state $\mathbf{s}'$. The reward is governed by the reward function $R : S \times A \times S \to \mathbb{R}$.

2.1.2 Value Functions

The general framework of reinforcement learning considers systems where an agent is in a certain state of the environment, can execute actions at discrete time steps $t = 1, 2, 3, \ldots$ and receives a reward when transitioning to a new state. The agent's rule set of which actions to execute in which state is called the policy. A policy $\pi : S \to A$ maps states $\mathbf{s} \in S$ to action $\mathbf{a} \in A$ and can be deterministic or stochastic. To compute such a solution, the policy has to estimate the value of each state after following the current policy π. This is formalized in the value function. The value function $V^{\pi}(\mathbf{s})$ is the expected value over the sum of future discounted rewards achieved by executing the policy π. The value function $V^{\pi}(\mathbf{s})$ is formally defined as:

$$V^{\pi}(\mathbf{s}) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^{k} \mathbf{r}_{t+k} \mid \mathbf{s}_t = \mathbf{s}\right], \tag{2.1}$$

where $\gamma \in [0,1]$ is the discount factor for future rewards.

The Q-function $Q : S \times A \to \mathbb{R}$, also called state-action function, is closely related to the value function as it gives the value of executing a specific action $\mathbf{a}$ in the state $\mathbf{s}$ and is denoted as:

$$Q^{\pi}(\mathbf{s},\mathbf{a}) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^{k} \mathbf{r}_{t+k} \mid \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a}\right]. \tag{2.2}$$

Both the value function and the Q-value function have a recursive form and are called the Bellman equations:

$$V^{\pi}(\mathbf{s}) = \mathbb{E}_{\substack{\mathbf{a}\sim\pi \\ \mathbf{s}'\sim P}}[r(\mathbf{s},\mathbf{a}) + \gamma V(\mathbf{s}')] \tag{2.3}$$

$$Q^{\pi}(\mathbf{s},\mathbf{a}) = \mathbb{E}_{s'\sim P}[r(\mathbf{s},\mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}'\sim\pi}[Q(\mathbf{s}',\mathbf{a}')]] \tag{2.4}$$

Another related function is the advantage function $A^{\pi}(\mathbf{s},\mathbf{a})$. The advantage function is simply the difference between the Q-function and the value function:

$$A^{\pi}(\mathbf{s},\mathbf{a}) = Q^{\pi}(\mathbf{s},\mathbf{a}) - V^{\pi}(\mathbf{s}). \tag{2.5}$$

In general, we are interested in finding the optimal policy. The optimal policy is defined as $V^{\pi^*}(\mathbf{s}) \geq V^{\pi}(\mathbf{s})$, for all $\mathbf{s} \in S$ and for all policies π. This means that by following policy π^* the expectation over the sum of discounted future rewards in each state $\mathbf{s}$ is greater than or equal to all other policies starting from state $\mathbf{s}$. The optimal value function V^* of any MDP, expressed in terms of the executed policy, is V^{π^*} and satisfies:

$$V^*(\mathbf{s}) = \max_{\mathbf{a}\in A} \sum_{\mathbf{s}'\in S} T(\mathbf{s},\mathbf{a},\mathbf{s}')(R(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma V^*(\mathbf{s}')) \tag{2.6}$$

This definition of $V^*(\mathbf{s})$ is better known as the Bellman optimality equation and expresses that the value of a state under an optimal policy is the expected reward for the best action in that state. Consequently, the optimal policy takes the $\arg\max$ over actions for each next state $\mathbf{s}' \in S$:

$$\pi^*(\mathbf{s}) = \arg\max_{\mathbf{a}\in A} \sum_{\mathbf{s}'\in S} T(\mathbf{s},\mathbf{a},\mathbf{s}')(R(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma V^*(\mathbf{s}')). \tag{2.7}$$

The methods that are capable of finding the optimal policy are broadly categorized into model-based and model-free approaches. Model-based reinforcement learning has the objective of learning the transition function T of the environment. In contrast, model-free methods learn the value function by interacting with the environment. Model-free approaches are generally value-based or policy-based. Value-based approaches learn the Q-value function and derive a deterministic policy:

$$\pi(\mathbf{s}) = \arg\max_{\mathbf{a} \in A} Q(\mathbf{s}, \mathbf{a}). \tag{2.8}$$

The disadvantage of value-based approaches is their need to estimate the value function for all possible actions, which is often not feasible in high-dimensional or continuous action spaces. Policy-based approaches, on the other hand, optimize the policy $\pi(\mathbf{s})$ directly by learning the optimal parameters θ^*. This makes policy-based algorithms suitable for continuous action spaces or environments that require exploration through randomness.

2.1.3 Policy-Based Reinforcement Learning

Policy gradient methods are a class of reinforcement learning algorithms that optimize a parameterized policy $\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)$, which maps states to action probabilities, by estimating the gradient of the expected cumulative reward with respect to the policy parameters θ. The objective is to maximize the expected return over trajectories $\tau = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \ldots, \mathbf{s}_T)$, where T is the horizon of the trajectory. Formally, this can be expressed as:

$$\begin{aligned} J(\theta) &= \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] \\ &= \sum_\tau P(\tau \mid \theta) R(\tau), \end{aligned} \tag{2.9}$$

where $R(\tau)$ is the discounted cumulative reward for trajectory τ. The probability of a trajectory τ is given by:

$$P(\tau \mid \theta) = \rho_0(\mathbf{s}_0) \prod_{t=0}^{T-1} \pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t) P(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t), \tag{2.10}$$

where $\rho_0(\mathbf{s}_0)$ is the initial state distribution and $P(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$ is the probability of transitioning to $\mathbf{s}_{t+1}$ after executing action $\mathbf{a}_t$ in state $\mathbf{s}_t$.

We can derive the gradient of $J(\theta)$ by taking the derivative:

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta \sum_\tau P(\tau \mid \theta) R(\tau) \\ &= \sum_\tau \nabla_\theta P(\tau \mid \theta) R(\tau) \\ &= \sum_\tau P(\tau \mid \theta) \frac{\nabla_\theta P(\tau \mid \theta)}{P(\tau \mid \theta)} R(\tau) \\ &= \sum_\tau P(\tau \mid \theta) \nabla_\theta \log P(\tau \mid \theta) R(\tau) \\ &= \mathbb{E}_{\tau \sim P(\tau \mid \theta)} [\nabla_\theta \log P(\tau \mid \theta) R(\tau)] \end{aligned} \tag{2.11}$$

Taking the logarithm over $P(\tau \mid \theta)$, we get:

$$\log P(\tau \mid \theta) = \log \rho_0(\mathbf{s}_0) + \sum_{t=0}^{T-1} \left[\log \pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t) + \log P(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)\right]. \tag{2.12}$$

This gives us the gradient:

$$\nabla_\theta \log P(\tau \mid \theta) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t), \tag{2.13}$$

since $\rho_0(\mathbf{s}_0)$ and $P(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$ are independent of θ. We can substitute this back into the gradient expression of Equation 2.11:

$$\nabla_\theta J(\theta) = E_{\tau \sim P(\tau|\theta)} \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t) R(\tau)\right]. \tag{2.14}$$

This shows that we can maximize the expected reward by adjusting the policy parameters in the direction of the log-probability of the actions, weighted by the trajectory's discounted cumulative return. In practice, the expectation is approximated over multiple trajectories using Monte Carlo sampling, which can lead to high-variance gradients. This destabilizes training, slowing down convergence. Actor-critic algorithms overcome the problems of value-based and policy-based approaches by combining direct policy optimization (actor) with value function estimation (critic). Next, we describe two actor-critic algorithms called Soft Actor-Critic (Section 2.1.4) and Proximal Policy Optimization (Section 2.1.5).

2.1.4 Soft Actor-Critic

Soft Actor-Critic (SAC) is an off-policy entropy-maximization reinforcement learning algorithm (Haarnoja et al., 2018). SAC regularizes the policy with an additional entropy term, which promotes exploration and prevents convergence to suboptimal policies. The entropy-regularized objective is:

$$\pi^* = \arg\max_\pi \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t \left(R(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) + \alpha H\left(\pi(\cdot \mid \mathbf{s}_t)\right)\right)\right],$$

where $\alpha > 0$ is the entropy regularization coefficient, $\gamma \in [0,1)$ is the discount factor and $H(\pi(\cdot \mid \mathbf{s}_t))$ is the entropy of the policy at state $\mathbf{s}_t$, defined as:

$$H(P) = \mathbb{E}_{x \sim P}[-\log P(x)].$$

In the entropy-regularized setting, the value functions are modified to include entropy bonuses. The modified value function is defined as:

$$V^\pi(\mathbf{s}) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t \left(R(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) + \alpha H\left(\pi(\cdot \mid \mathbf{s}_t)\right)\right) \mid \mathbf{s}_0 = \mathbf{s}\right], \tag{2.15}$$

and the entropy-regularized Q-function is defined as:

$$Q^{\pi}(\mathbf{s},\mathbf{a}) = \mathbb{E}_{\tau\sim\pi}\left[R(\mathbf{s}_0,\mathbf{a}_0,\mathbf{s}_1) + \sum_{t=1}^{\infty}\gamma^t\left(R(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+1}) + \alpha H\left(\pi(\cdot \mid \mathbf{s}_t)\right)\right) \mid \mathbf{s}_0 = \mathbf{s}, \mathbf{a}_0 = \mathbf{a}\right]. \quad (2.16)$$

The Bellman equation for Q^{π} is:

$$Q^{\pi}(\mathbf{s},\mathbf{a}) = \mathbb{E}_{\mathbf{s}'\sim P,\mathbf{a}'\sim\pi}\left[R(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma\left(Q^{\pi}(\mathbf{s}',\mathbf{a}') + \alpha H\left(\pi(\cdot \mid \mathbf{s}')\right)\right)\right].$$

2.1.5 Proximal Policy Optimization

Proximal Policy Optimization (PPO) is an on-policy reinforcement learning algorithm designed to improve the policy based on current data while avoiding performance collapse due to large updates (Schulman et al., 2017). PPO updates the policy by maximizing a clipped objective function, which limits the new policy from significantly deviating from the old policy. The update rule is as follows:

$$\theta_{k+1} = \arg\max_{\theta}\mathbb{E}_{\mathbf{s},\mathbf{a}\sim\pi_{\theta_k}}\left[L(\mathbf{s},\mathbf{a},\theta_k,\theta)\right], \quad (2.17)$$

where multiple steps of mini-batch stochastic gradient descent (SGD) are typically used. The clipped objective is:

$$L(\mathbf{s},\mathbf{a},\theta_k,\theta) = \min\left(\frac{\pi_\theta(\mathbf{a}\mid\mathbf{s})}{\pi_{\theta_k}(\mathbf{a}\mid\mathbf{s})}A^{\pi_{\theta_k}}(\mathbf{s},\mathbf{a}), \operatorname{clip}\left(\frac{\pi_\theta(\mathbf{a}\mid\mathbf{s})}{\pi_{\theta_k}(\mathbf{a}\mid\mathbf{s})}, 1-\varepsilon, 1+\varepsilon\right)A^{\pi_{\theta_k}}(\mathbf{s},\mathbf{a})\right), \quad (2.18)$$

with ε being a hyperparameter controlling the maximum allowed policy change and $A^{\pi_{\theta_k}}(\mathbf{s},\mathbf{a})$ as the advantage function.

2.2 Representation Learning in RL

As outlined above, reinforcement learning algorithms depend on the state $\mathbf{s}$ to learn a policy in the environment. However, in many real-world situations, the true state of the environment is not directly observed (see Section 2.1.1). Consider robotics applications, where control often depends on computer vision (Hämäläinen et al., 2019; Chebotar et al., 2019; Vecerik et al., 2021). Although initial end-to-end approaches demonstrated remarkable success learning directly from raw pixel observations (Mnih et al., 2013), this approach fails when the environment is more complex or changes over time. Pixel observations, especially, introduce significant complications as the visual input space is large, high-dimensional, sparse, and prone to noise. Naturally, follow-up research leveraged insights from the work in representation learning to address these issues. Representation learning is the process of extracting informational features from raw observations (Bengio et al., 2014). In our work, we adopt the definition by Echchahed and Castro (2025):

Definition. *State representation learning learns a representation function $h_\theta^k : \mathcal{O}_0 \times \mathcal{O}_1 \times \cdots \times \mathcal{O}_k \to \mathcal{Z}$, parameterized by θ, which maps k-step observation sequences to a representation space*

$\mathcal{Z}$, thereby allowing us to define policies $\Pi_{\mathcal{Z}}$ over this reduced space. The encoder enables a policy network π_θ to compute actions $\boldsymbol{a}_t = \pi_\theta(z_t)$, and a value network V_θ to compute values $\boldsymbol{v}_t = V_\theta(z_t)$, based on the representation $z_t = h_\theta(\boldsymbol{o}_t)$, instead of directly using high-dimensional observations. The representation z_t is a vector in $\mathbb{R}^d$, where d is the dimensionality of $\mathcal{Z}$.

Learning a representation function that extracts relevant information from the high-dimensional input mitigates the state-space explosion problem and enhances policy optimization.

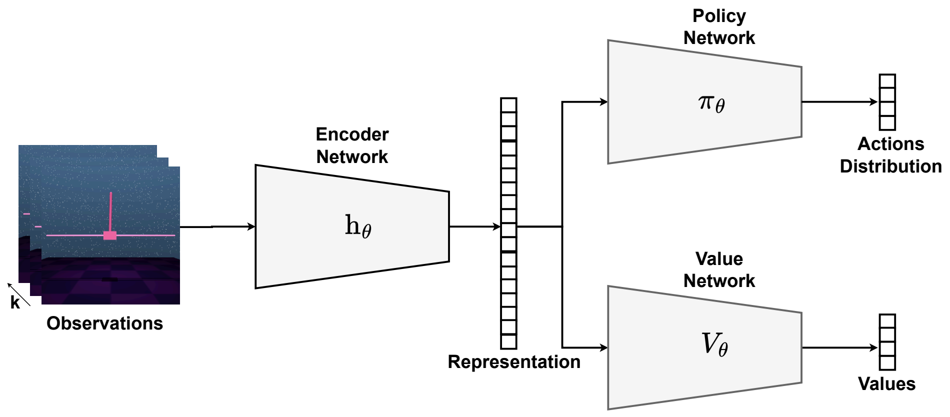


Figure 2.1: High-level overview of state representation learning in Actor-Critic RL algorithms. The k observations are encoded into a representation using encoder h_θ. The representation is passed to the policy network π_θ and value network V_θ that output the actions distribution and values, respectively.

2.2.1 Desirable Properties of State Representations

A key question in state representation learning is what constitutes an optimal state representation (Böhmer et al., 2015; Lesort et al., 2018; Botteghi et al., 2025). Ideally, an optimal representation should support the policy to learn the control task(s) in the environment. This can only be achieved when the representation sufficiently captures all the task-relevant information content while ignoring irrelevant information (e.g., noise, distractors). Furthermore, an optimal representation not only contains sufficient information, but is also well-structured. According to Le Lan et al. (2021), an effective latent structure guarantees a high degree of Lipschitz continuity for the value function of neighboring representations. In essence, this means that neighboring representations should be mapped to similar Q-value predictions.

2.2.2 Disentangled Representations

To address the need for an optimally structured representation, a key direction in representation learning is the concept of disentanglement. In this framework, a dataset **X** is described as a generative process in which each observed value $\mathbf{x} \in \mathbf{X}$ is the realization of a set of underlying factors of variation (Bengio et al., 2014; Schölkopf et al., 2021). The factors of variation, denoted as $\mathbf{y}$, are assumed to be independent and nonlinearly mixed into a high-dimensional space according to a conditional distribution $p(\mathbf{x} \mid \mathbf{y})$. When a representation captures the independent factors of variation in the dataset, the representation is said to disentangle the latent factors of variation. As Bengio et al. (2014) intuitively describe it:

Definition. *A disentangled representation separates the distinct, independent, and informative generative factors of variation in the data. Single latent variables are sensitive to changes in single underlying generative factors while being relatively invariant to changes in other factors.*

Thus, disentangled representation learning is concerned with estimating the generative model of the true factors of variation using $\mathbf{z} = \mathbf{f}(\mathbf{x})$, with $\mathbf{x} \sim p(\mathbf{x} \mid \mathbf{y})$. From a theoretical perspective, the successful identification of true latent sources is challenging, requiring at minimum weakly labeled data or appropriate inductive biases (Locatello et al., 2019).

2.2.3 Self-Supervised Contrastive Learning

Self-supervised contrastive learning is a representation learning technique that learns a representation of data objects by juxtaposing similar data objects against dissimilar data objects without having access to the ground-truth labels of each data object. Consider the problem of learning a class representation of a binary image dataset without having access to the ground-truth label of each image. As a first step, self-supervised contrastive learning artificially creates structure in the dataset by splitting the dataset into positive pairs and negative pairs, often applying data augmentations (e.g., cropping, patching, or rotating) to each element in the pair (Dosovitskiy et al., 2015; Doersch et al., 2016; Gidaris et al., 2018). Positive pairs are instances that belong to the same data object. Negative pairs are instances that do not belong to the same data object. This allows us to train a generative model by reformulating the problem as a supervised classification task. The next step is to train a classifier to learn to discriminate between positive and negative pairs. The prediction loss is used to update the gradients of the classifier and the generative model. Intuitively, if the generative model maps similar data points closely to each other in latent space, while mapping negative data points farther away in latent space, the classifier will be better at discriminating between the classes. This encourages the representation network to capture meaningful features and similarities of both classes. A commonly used objective function in the binary discrimination task is the probabilistic objective function called binary cross-entropy loss. The binary cross-entropy loss is defined as

$$L(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{N} \sum_{n}^{N} [\mathbf{y}_i \times \log(\hat{\mathbf{y}}_i) + (1 - \mathbf{y}_i) \times \log(1 - \hat{\mathbf{y}}_i)]. \tag{2.19}$$

Self-supervised learning is especially attractive for learning a state representation, as it is often assumed that agents do not have access to the true state. Moreover, contrastive learning also has a relationship to disentangled representation learning through the Nonlinear Independent Component Analysis framework, which will be discussed in the next section.

2.3 Nonlinear Independent Component Analysis

Independent Component Analysis (ICA) is one of the successful approaches to blind source separation (Jutten and Herault, 1991; Schmidhuber, 1992; Comon, 1994). Blind source separation is the problem of recovering true signals from mixed signals without supervision. The general model assumes that the sources are mixed by a linear mixing function, mathematically expressed as:

$$\mathbf{x} = \mathbf{W}\mathbf{y} = \sum_{i=1}^{n} y_i w_i. \tag{2.20}$$

Here, $w_i \in \mathbf{W}$ are the weights of the mixing matrix $\mathbf{W}$ and $y_i \in \mathbf{y}$ are the true values of each component $i \in \{1, ..., n\}$ of $\mathbf{y}$. This simple model performs remarkably well, but is often insufficient to describe natural data. Hence, the work on ICA was shortly followed by considering the extension to Nonlinear ICA (Burel, 1992; Deco and Brauer, 1995; Deco and Obradovic, 1996; Pajunen, 1996; Pajunen and Karhunen, 1997; Lee et al., 1997; Hyvärinen and Pajunen, 1999). The Nonlinear ICA model is defined as:

$$\mathbf{x} = \mathbf{f}(\mathbf{y}), \tag{2.21}$$

where $\mathbf{f}$ is the unknown, real-valued, nonlinear, mixing function.

Assume that $\mathbf{x} \in \mathbb{R}^d, \mathbf{y} \in \mathbb{R}^d$, then the problem of Nonlinear ICA is to find the mapping $\mathbf{f}^{-1} : \mathbb{R}^d \to \mathbb{R}^d$ that gives us the original statistically independent components:

$$\hat{\mathbf{y}} = \mathbf{f}^{-1}(\mathbf{x}). \tag{2.22}$$

Recent work established the connection between types of contrastive learning and Nonlinear ICA (Hyvärinen and Morioka, 2016, 2017; Hyvärinen et al., 2019; Morioka et al., 2021). Specifically, Hyvärinen and Morioka (2016) showed that TCL is able to reconstruct non-stationary sources by leveraging temporal structure, recoverable through segment indices as auxiliary information. This work was closely followed by Hyvärinen and Morioka (2017) which proposed PCL to identify temporally dependent stationary sources. These methods were combined into a general framework in (Hyvärinen et al., 2019). This work proposed a generalized contrastive method that utilizes auxiliary variables related to the source signal to recover the underlying independent components. In this framework, it is assumed that we observe $\mathbf{x}_t = \mathbf{f}(\mathbf{y}_t)$, where $\mathbf{y}_t$ is the source signal at time step t and $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ is a smooth, invertible and nonlinear mixing function. Additionally, each component y_i is assumed to depend on an observed auxiliary variable $\mathbf{u}$, but is conditionally independent of other components given $\mathbf{u}$:

$$p(\mathbf{y} \mid \mathbf{u}) = \Pi_i q_i(y_i \mid \mathbf{u}), \tag{2.23}$$

for some conditional log-probability density function (log-pdf) q_i.

The method for learning this generative model uses an algorithm called Generalized Contrastive Learning (GCL). GCL learns the demixing function using a self-supervised binary discrimination task. By discriminating between actual samples $\tilde{\mathbf{x}} = (\mathbf{x}, \mathbf{u})$ and shuffled samples $\tilde{\mathbf{x}}^* = (\mathbf{x}, \mathbf{u}^*)$, where $\mathbf{u}^*$ is randomly drawn from the distribution of $\mathbf{u}$, independently of $\mathbf{x}$. The discriminator is learned using a regression function of the form:

$$g(\mathbf{x}, \mathbf{u}) = \sum_{i=1}^{n} \psi_i(h_i(\mathbf{x}), \mathbf{u}), \tag{2.24}$$

where h_i is the i^{th} component of the demixed signal $\mathbf{x}$ and ψ_i is a function approximator with universal approximation capacity (Hornik et al., 1989). The theoretical framework in (Hyvärinen et al., 2019) guarantees that independent components can be recovered up to a component-wise transformation, assuming that: (i) the observations follow the model in Equation 2.23, (ii) each conditional log-pdf q_i is sufficiently smooth in the source variable given the auxiliary information, (iii) the auxiliary variable $\mathbf{u}$ sufficiently influences the distribution of independent components, (iv) a nonlinear logistic regression system accurately discriminates between true and permuted samples $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}^*$, (v) the feature mapping is bijective, smooth and has a smooth inverse and (vi) we have access to infinite data.

Chapter 3

Methodology

This chapter presents our framework to disentangle latent representations in model-free deep reinforcement learning. We begin by describing the problem setting that our method aims to solve in Section 3.1. Secondly, we thoroughly describe our methodology in Section 3.2.

3.1 Problem Setting

We build on the standard POMDP formalism as defined in Section 2.1.1. Our setting retains the tuple (S, A, Ω, T, O, R) but extends the transition function and observation function with additional latent factors of variation. Formally, we introduce the task-relevant factor $\zeta_t \in \mathcal{F}_\zeta$, which influences the state dynamics, and the task-irrelevant factor $\xi_t \in \mathcal{F}_\xi$, which affects the observation process. Both factors can change over time following the conditional distribution $P(\zeta_t \mid \zeta_{t-1})$ or $P(\xi_t \mid \xi_{t-1})$, respectively. We extend the transition function $T : S \times A \times \mathcal{F}_\zeta \times S \to [0,1]$ to $T(\mathbf{s}, \mathbf{a}, \zeta, \mathbf{s}') = P(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}; \zeta)$. Correspondingly, the observation function is redefined as $O : S \times A \times \mathcal{F}_\xi \times \Omega \to [0,1]$, where $O(\mathbf{o} \mid \mathbf{s}, \mathbf{a}; \xi)$ denotes the probability of perceiving observation $\mathbf{o}$ after executing action $\mathbf{a}$ and transitioning to state $\mathbf{s}$ given that the latent factor is ξ_t. An optimal agent identifies when a factor is task-irrelevant and minimizes the impact of the factor on the policy accordingly. By contrast, an agent that perfectly isolates task-relevant factors of variation (e.g., state information) improves the agent's ability to adapt to specific changes in the environment, similarly resulting in better generalization to domain shifts. Figure 3.1 depicts the graphical model of the problem setting. The reward is omitted from the graph for visualization purposes.

3.2 Generalized Disentanglement

The objective of Generalized Disentanglement (GED) is to disentangle latent factors of variation through an additional auxiliary contrastive learning task in pixel-based reinforcement learning training. Conceptually, GED follows the framework proposed for TED (Dunion et al., 2023b). Our methodology improves on TED by allowing for greater choice in the non-temporal contrastive pairs. In applicable situations, this eliminates the need to maintain a record of episode

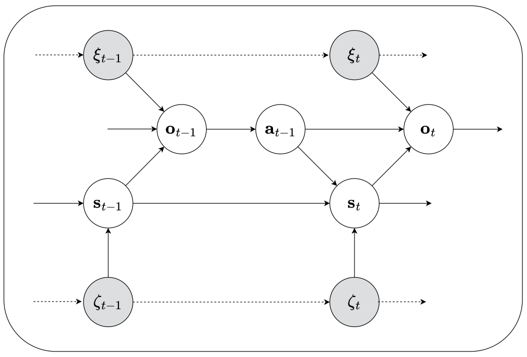


Figure 3.1: The graphical model of our problem setting. We build on the standard POMDP model. Rewards are omitted for visualization purposes. The task-irrelevant factor ξ affects the observation directly. The task-relevant factor ζ affects the agent's observations through the influence on the true state of the environment.

indices, simplifying implementation. Moreover, our generalized optimization objective demonstrates greater stability, extending applicability to a wider range of domains. A visualization of the architecture is shown in Figure 3.2.

3.2.1 Architecture

The architecture of GED can be easily integrated with existing approaches that use an encoder module $h_\theta : O \to Z$ to compress high-dimensional image observations $\mathbf{o} \in O$ into lower-dimensional latent representations $\mathbf{z} \in Z$. Our method guides the encoder h_θ to learn a disentangled representation of image observations through self-supervised contrastive learning. We introduce a classifier $g_\phi : Z \times Z \times U \to \mathbb{R}$ to discriminate between temporal samples and non-temporal samples. The variable U represents an optional conditional variable, or auxiliary variable, that could act as a stronger signal to disentangle the latent variables. Note that we consider conditioning on the history as a necessity, redefining the meaning of $\mathbf{u}$ compared to (Hyvärinen et al., 2019).

The temporal samples $(\mathbf{z}_t, \mathbf{z}_{t+1}, \mathbf{u}_t)$ contain two consecutive latent representations and, optionally, the auxiliary variable. In contrast, non-temporal samples $(\mathbf{z}_t, \mathbf{z}_{t'}, \mathbf{u}_{t'})$ contain two non-temporal latent representations, where the next representation and the auxiliary variable are drawn from the buffer, with $t' \notin \{t-1, t, t+1\}$.

During training, we sample a batch $B = \{(\mathbf{o}_t, \mathbf{o}_{t+1}, \mathbf{u}_t)_i\}_{i=1}^N$ of size N that contains observation pairs and optional auxiliary variables. The observations are encoded using the encoder h_θ. Similarly to TED, $\mathbf{o}_t$ is passed through the encoder connected to the RL modules, and $\mathbf{o}_{t+1}$ is passed through a target encoder $h_{\theta'} : O \to Z$. To improve training stability, the target encoder uses soft updates $\theta' = \eta\theta + (1-\eta)\theta'$, where η is a hyperparameter that controls the update step.

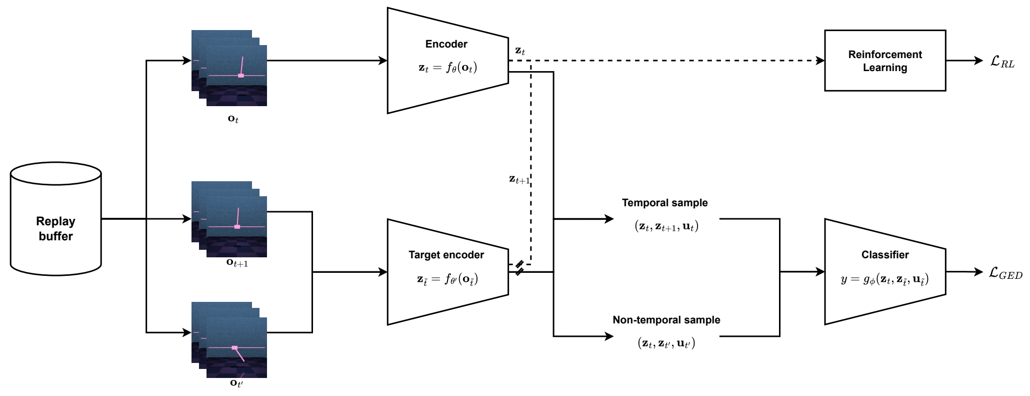

Encoder
$\mathbf{z}_t = f_\theta(\mathbf{o}_t)$
$\mathbf{z}_t$
Reinforcement Learning
$\mathcal{L}_{RL}$
$\mathbf{o}_t$
Replay buffer
$\mathbf{z}_{t+1}$
Temporal sample
$(\mathbf{z}_t, \mathbf{z}_{t+1}, \mathbf{u}_t)$
Target encoder
$\mathbf{z}_{\tilde{t}} = f_{\theta'}(\mathbf{o}_{\tilde{t}})$
Classifier
$y = g_\phi(\mathbf{z}_t, \mathbf{z}_{\tilde{t}}, \mathbf{u}_{\tilde{t}})$
$\mathcal{L}_{GED}$
$\mathbf{o}_{t+1}$
Non-temporal sample
$(\mathbf{z}_t, \mathbf{z}_{t'}, \mathbf{u}_{t'})$
$\mathbf{o}_{t'}$

Figure 3.2: Generalized Disentanglement Architecture: The latent representations of the encoder are encouraged to isolate independent factors of variation by the concurrent training of the classifier which discriminates between temporal and non-temporal samples. The '//' indicates that the gradient flow is stopped. Time steps $\tilde{t} \in \{t+1, t'\}$ depend on the observation being processed.

The batch B of N transitions is augmented to create two types of observation-pair batches: (i) temporal samples X and (ii) non-temporal samples X'. Concretely, this gives us two types of samples:

i $(\mathbf{z}_t, \mathbf{z}_{t+1}, \mathbf{u}_t)$ for temporal pairs.

ii $(\mathbf{z}_t, \mathbf{z}_{t'}, \mathbf{u}_{t'})$ for non-temporal pairs.

3.2.2 Non-Temporal Pairs

Dunion et al. (2023b) showed that TED requires a balanced ratio of same episode, non-temporal samples and different episode, non-temporal samples for optimal performance. We consider more flexible options in the choice of non-temporal contrastive pairs. The optimal sampling scheme is task-dependent. We consider three possible sampling variants for non-temporal pairs:

i **Balanced Ratio**: A balanced ratio between same episode, non-temporal pairs and different episode, non-temporal pairs. The optimization objective corrects for the imbalance between positive and negative pairs.

ii **Same Episode**: All non-temporal pairs belong to the same episode.

iii **Non-Temporal**: Sampling negative pairs without regard for the episode.

The possible sampling schemes are evaluated in a variety of control tasks. The results are presented in Chapter 5.

3.2.3 Optimization Objective

The classifier optimizes the GED loss $\mathcal{L}_{GED}$ using the regression function $g_{comb}(.)$ from (Hyvärinen et al., 2019):

$$g_{comb}(\mathbf{z}_t, \mathbf{z}_{\tilde{t}}, \mathbf{u}_{\tilde{t}}) = \sum_{i=1}^{d} \psi_i(z_t^i, z_{\tilde{t}}^i, \mathbf{u}_{\tilde{t}}), \tag{3.1}$$

where z^i is the i^{th} component of the latent representation $\mathbf{z}_t$, $\mathbf{u}_{\tilde{t}}$ is the optional auxiliary variable and $\psi_i : \mathbb{R}^{2+|u|} \to \mathbb{R}$ are general function approximators, implemented as feedforward neural networks.

At each update, the classifier receives a batch of samples $\{X, X'\}$. The output of the classifier is compared with the synthetically generated ground-truth labels. The label $l = 1$ corresponds to all temporal pairs of samples X and $l = 0$ corresponds to non-temporal samples X'. The classifier is optimized by minimizing the binary cross-entropy loss for all $\mathbf{x}_t \in \{X, X'\}$:

$$\mathcal{L}_{GED}(\mathbf{x}_t, l) = -\alpha\left(l \log \sigma(g_{\text{comb}}(\mathbf{x}_t)) + (1-l)\log(1-\sigma(g_{comb}(\mathbf{x}_t)))\right), \tag{3.2}$$

where σ is the sigmoid function. The parameter α is the same hyperparameter as in TED that scales the magnitude of the term $\mathcal{L}_{GED}$ to balance the policy optimization objective with the disentanglement objective. Since the combined regression function $g_{comb}(.)$ contains a neural network ψ_i for each component of the latent representation, the encoder is encouraged to separate distinct factors of variation into isolated latent components.

The modified pseudocode from (Dunion et al., 2023b) for an update step is provided in Algorithm 1.

Algorithm 1 GED update step

```
Require: Batch of transitions $B = \{..., (\mathbf{o}_t, \mathbf{o}_{t+1}, \mathbf{u}_t), ...\} \sim D$
 1: Create batch $Z_{obs}$ of representations for each observation in $B$: $\mathbf{z}_t = h_\theta(\mathbf{o}_t)$
 2: Create batch $Z_{next\_obs}$ of representations for each next observation in $B$: $\mathbf{z}_{t+1} = h_{\theta'}(\mathbf{o}_{t+1})$
 3: for each transition in $B$ do
 4:     Create temporal sample $\mathbf{x}_t \leftarrow (\mathbf{z}_t, \mathbf{z}_{t+1}, \mathbf{u}_t)$, $X \leftarrow X \cup \mathbf{x}_t$
 5:     Sample $\mathbf{o}_{t'} \sim D$ such that $t' \notin \{t-1, t, t+1\}$ and get representation $\mathbf{z}_{t'} = h_{\theta'}(\mathbf{o}_{t'})$
 6:     Create non-temporal sample $\mathbf{x}'_t \leftarrow (\mathbf{z}_t, \mathbf{z}_{t'}, \mathbf{u}_{t'})$, $X' \leftarrow X' \cup \mathbf{x}'_t$
 7: for each sample $\mathbf{x} \in \{X, X'\}$ do
 8:     Classifier prediction $y = g_\phi(\mathbf{x})$ (see Equation 3.1)
 9:     Calculate binary cross-entropy loss $L_{GED}(\mathbf{x}, l)$ (see Equation 3.2)
10: Calculate average loss for the batch $\mathcal{L}_{GED} \leftarrow \text{mean}(\mathcal{L}_{GED}(\mathbf{x}))$
11: Backpropagate loss to update encoder parameters $\theta$ and classifier parameters $\phi$
12: Update target encoder parameters $\theta' = \eta\theta + (1-\eta)\theta'$
Ensure: Loss $\mathcal{L}_{GED}$ and updated parameters $\phi$, $\theta$, and $\theta'$
```

Chapter 4

Experimental Setup

This chapter discusses the experimental setup to evaluate our approach. We start by describing the environments in Section 4.1. Secondly, we discuss the baseline RL agents in Section 4.2. Next, we define the non-stationary settings in Section 4.3. Lastly, we discuss the evaluation criteria in Section 4.4 and the choice of hyperparameters in Section 4.5.

4.1 Environments

The experiments are conducted in a variety of control tasks. We first describe the DM Control Suite environments Cartpole Swingup and Finger Spin, followed by the Panda environment Reach Dense. Next, we describe Gymnasium's Car Racing environment. Finally, we describe Procgen environments Coinrun and Leaper. See Appendix A for visualizations of each environment.

4.1.1 DM Control Suite

The DM Control (DMC) Suite (Tassa et al., 2018) is a collection of reinforcement learning environments built on the MuJoCo Physics Engine.

Cartpole Swingup

The Cartpole Swingup (`cartpole`) task is a modification of the classic cartpole balancing task. In this task, the agent starts out with the pole in a hanging position. The goal is to swing the pole in an upright position by controlling the cart horizontally along the linear track. Once the pole is in an upright position, the goal is to balance the pole. The one-dimensional continuous action space represents the force on the cart. The agent receives a reward proportional to the pole's orientation, the cart's velocity, and the cart's proximity to the track's center.

Finger Spin

In the Finger Spin (`finger`) task, the agent controls a 'finger' to spin an object around its axis as fast as possible. The finger is fixed in space but can rotate around its hinge and the object can be manipulated through contact. The two-dimensional continuous action space represents the torque applied to the actuated joints of the finger. The reward is based on the angular velocity of the object.

4.1.2 Panda Gym

The library Panda Gym (Gallouédec et al., 2021) contains a set of robotic environments based on PyBullet (Coumans and Bai, 2016) physics engine and Gym (Brockman et al., 2016).

Reach Dense

The Reach Dense (`reach`) environment simulates the Franka Emika Panda[1] robotic arm. The arm has 7 degrees of freedom and a parallel finger gripper. In the Reach Dense task the agent controls the end-effector using a three-dimensional continuous action space, representing the x, y, and z coordinates. The objective is to move the end-effector to a randomly generated target position in a 3D space. The "Dense" refers to the agent receiving a continuous reward based on the negative Euclidean distance between the end-effector and the target.

4.1.3 Gymnasium

Gymnasium (Towers et al., 2024) is a popular open-source Python library that provides a wide range of reinforcement learning environments.

Car Racing

The Car Racing (`carracing`) environment is a continuous control task in which the agent has to control a car in a 2D top-down racing simulation. The car navigates a randomly generated track. The goal is to drive around the track as quickly and accurately as possible. To do so, the agent has control over a three-dimensional continuous action space representing the angle of the car's wheels, acceleration and braking. The reward is based on the car's progress along the track. The track is divided into tiles, and the agent receives a reward of +1000/N for visiting each new tile, where N is the total number of tiles on the track. Additionally, a small penalty of -0.1 is applied for each step to encourage faster completion. An episode in the `carracing` environment terminates if: (i) the car visits at least 95% of all track tiles, (ii) the car is off track for too long, or (iii) the maximum number of steps is reached.

4.1.4 Procgen

Procgen (Cobbe et al., 2020) is a collection of 16 unique procedurally generated environments. The wide variety of environments acts as a benchmark for generalization and sample efficiency of reinforcement learning algorithms. Each environment offers a wide diversity of challenges and tunable difficulty settings. The observation space of each environment consists of images. These characteristics make these environments highly suitable to test our methodology.

[1]https://franka.de/

Coinrun

The Coinrun (`coinrun`) environment is a platformer, where an agent starts out on the left and has to navigate to the far right end of the environment to collect a coin. The agent is able to walk and navigate the environment using a discrete action space of 15 actions, representing movement and action inputs. Levels contain different obstacles such as pacing enemies, spinning saw blades, and pits. Contact with any of these obstacles ends the episode. An episode also ends when the coin is collected. The agent receives a sparse reward of +10 for collecting the coin and zero reward otherwise.

Leaper

The Leaper (`leaper`) environment is a 'Frogger'-like environment where the objective of the agent is to cross a river by jumping on moving surfaces. In `leaper`, the agent has to first cross four roads with cars moving in both horizontal directions. The agent then has to cross a river by jumping on moving logs. Similarly, it has a discrete action space of 15 actions. An episode ends when the agent collides with a vehicle, falls in water, or has successfully crossed the finish line. The agent receives a sparse reward of +10 for successfully traversing the environment and zero reward otherwise.

4.2 Reinforcement Learning Algorithms

We evaluated our extension with two different state-of-the-art deep-RL algorithms to show that GED can be applied to any pixel-based deep-RL algorithm. Specifically, we integrated GED with Reinforcement Learning with Augmented Data (RAD) and Proximal Policy Optimization (PPO) to enhance the learned representations. In this section, we describe the implementation details of both agents.

4.2.1 Reinforcement Learning with Augmented Data

The implementation of RAD is based on the Soft Actor-Critic (SAC) algorithm. We augment image observations using padding and random cropping. The actor models a diagonal Gaussian policy implemented as a deep neural network. The actor outputs the mean and standard deviation of the action distribution, with actions sampled using the reparameterization trick and normalized to $[-1,1]$ using `tanh`. The two critics are implemented using independent neural networks. The critics estimate the expected return for state-action pairs, and target networks are maintained for stable Bellman updates. The weights of the convolutional encoder are tied between the actor and the critic. We apply entropy regularization via the temperature parameter, which is learned via gradient descent to match the target entropy, encouraging exploration.

4.2.2 Proximal Policy Optimization

The implementation of PPO follows the standard actor-critic architecture with a shared encoder. The encoder converts (stacked) image observations into a compact latent representation. The shared latent representation is passed to the policy head that outputs a distribution over actions and a value head that estimates the state value. We implemented additional techniques discussed by Andrychowicz et al. (2021) to improve learning, such as Generalized Advantage Estimation (GAE), normalization of advantages, entropy bonus, global gradient clipping, and mini-batch updates.

4.3 Non-Stationary Settings

In the real world, elements of the environment can vary over time, identifiable according to a certain temporal structure. We limit our research to investigate two types of non-stationary functions. These can be categorized as abrupt change and gradual change (Dulac-Arnold et al., 2021; Khetarpal et al., 2022).

4.3.1 Abrupt Change

Abrupt change refers to the setting where a time series contains N segments of length L generated independently using a uniform distribution. The value at any time step t can be described as:

$$X_t = \mu_i + \varepsilon_t$$

with

$$\mu_i \sim U(l,h),$$

where μ_i is fixed for each segment $i \in \{1,...,N\}$, l,h are the respective minimum and maximum, and ε_t is i.i.d. noise. We refer to this function as a piecewise uniform function.

4.3.2 Gradual Change

Gradual change refers to the setting where a time series is generated using a sinusoidal function with amplitude β and frequency ν. The value at any time step t can be expressed as:

$$X_t = \beta \sin(\nu t) + \varepsilon_t,$$

where ε_t is i.i.d. noise. We refer to this function as a cyclic function.

4.4 Evaluation Criteria

In this section, we describe the evaluation metrics that we use to assess the performance of the proposed methodology.

4.4.1 Cumulative Episodic Reward

We use the standard evaluation framework for finite-horizon episodic problems to evaluate the performance of the RL agent. In these problems, the agent's goal is to maximize the total reward collected within an episode. The cumulative reward R is defined as the sum of rewards for each time step t over the total number of steps T. In formal notation, that is:

$$R = \sum_{t=1}^{T} r_t. \tag{4.1}$$

4.4.2 Uniform Manifold Approximation and Projection

To analyze the learned latent representations, we employ Uniform Manifold Approximation and Projection (UMAP) (McInnes et al., 2020), a dimensionality reduction technique designed for visualizing high-dimensional data. UMAP is a manifold learning method that projects high-dimensional data into a lower-dimensional space while preserving both local and global topological structures. UMAP assumes that high-dimensional data lies on a low-dimensional manifold embedded in the ambient space. The algorithm constructs a high-dimensional graph representation of the data, where points are connected based on their local neighborhood structure, and then optimizes a low-dimensional embedding to preserve this topological structure. Specifically, UMAP models the data using a fuzzy simplicial complex, which captures the local connectivity of points via a weighted graph. The edge weights are determined by a similarity function that measures the proximity of points in the high-dimensional space. Since UMAP retains local connectivity, UMAP provides insight into how close identical state observations are in latent space.

4.4.3 Mean Absolute Error

To validate whether the agent assigns lower value to task-irrelevant factors, we use the MAE metric. The MAE quantifies the average magnitude of errors between predicted and target values, providing a straightforward measure of prediction accuracy in the same units as the data. Formally, let $\mathbf{Y} \in \mathbb{R}^n$ be a vector of n target values, and $\hat{\mathbf{Y}} \in \mathbb{R}^n$ be the corresponding vector of n predicted values. The MAE is defined as the average of the absolute differences between the target and predicted values:

$$\text{MAE} = \frac{1}{n} \left\|\mathbf{Y} - \hat{\mathbf{Y}}\right\|_1. \tag{4.2}$$

A lower MAE indicates that the predicted values are closer to the target values, reflecting better model performance.

4.5 Hyperparameters

Since hyperparameter tuning is computationally expensive and time-consuming, we based most of our hyperparameters on the existing literature. The hyperparameters for the environments `cartpole`, `finger` and `reach` are the same as in the experiments with task-irrelevant variables mentioned in Dunion et al. (2023b). The hyperparameters for the Procgen environments were also taken from Dunion et al. (2023b), excluding the α hyperparameter. For `carracing` we took the general hyperparameters mentioned in (Petrazzini and Antonelo, 2021). We optimized α

using the two-stage approach described in Patterson et al. (2024). We selected the value of α that corresponds to the highest cumulative episodic reward over 5 seeds obtained by the benchmark algorithm to reduce hyperparameter selection bias. A detailed overview of the hyperparameters is given in Appendix B.

Chapter 5

Experimental Results

This chapter covers the experimental results that assess the impact of disentangled representation learning on generalization. We begin by presenting the results of the experiments with task-irrelevant color distractors in Section 5.1. Section 5.2 examines results from experiments with task-relevant factors. Lastly, Section 5.3 details the experimental results with task-relevant and task-irrelevant factors of variation.

5.1 Generalization to Task-Irrelevant Factors

Our first experiments evaluate generalization to unseen task-irrelevant factors of variation in three different settings: (i) constant color distractors, (ii) abruptly changing color distractors, and (iii) gradually changing color distractors.

5.1.1 Implementation Details

For the environments `cartpole` and `finger` we used the `distracting-control` library (Stone et al., 2021) to modify the task-irrelevant color of the agent. We extended the library to modulate the color according to the functions defined in Section 4.3. For `reach` we implemented a custom wrapper that applies the same color distractors in a similar manner as in the `distracting-control` library using the same disjoint train and test sets. Given a distractor function Δ, we update the color value at any time step t using:

$$X_t = X_0 + \Delta_t,$$

where X_0 is the initial value at $t = 0$ sampled from a uniform distribution.
The experiments with constant color distractors follow the settings of Dunion et al. (2023b). The experiments with abruptly and gradually changing distractors within an episode are designed to be more challenging. In these experiments we sample the initial value from a uniform distribution equal to the sampling distribution with constant color distractors. Differently, the initial sampled value acts as the midpoint of the interval from which all subsequent color values can be sampled. Thus, these settings cover a wider range of color values, increasing difficulty. For an overview of the details see Appendix C.

5.1.2 Baselines

The base RL algorithm in each environment is RAD. We additionally compare to the base RL algorithm enhanced with TED. Differently from TED, we present the performance results of GED trained with same-episode non-temporal pairs during representation learning.

5.1.3 Results

Figure 5.1 presents the results of each environment and setting, with the left, center, and right columns showing `cartpole`, `finger`, and `reach`, respectively. The top row displays the setting where the color is fixed throughout an episode. The middle and bottom rows display the abruptly and gradually changing color distractor settings, respectively. Most notably, the baseline algorithm is not robust to the three types of color distractors across environments. Although the results show that RAD is able to adapt and recover in the setting with constant color distractors in `cartpole` and `finger`, RAD fails to learn a meaningful policy when color distractors are more challenging. In `reach`, RAD is highly variable, significantly underperforming TED and GED. In comparison, TED and GED show comparable performance in the train domain in `cartpole`, with no significant deviations in terms of generalization to the unseen domain. However, GED and TED, too, are much less robust to the test domain when the color distractor exhibits greater variance during an episode. Nevertheless, the two agents are able to recover to previous performance as in the train domain. The results are noticeably different in `finger`. GED is consistent in terms of training performance and generalization to the unseen domain, TED is also more robust to the test domain in this environment. However, the sudden large drop in performance after switching to the test domain in the setting with gradually changing color distractors suggests that TED suffers from catastrophic forgetting. In `reach` the performance of both models is comparable across settings. The results in terms of training performance, generalization and recovery are mixed, showing no significant difference between the two algorithms. Overall, these results indicate that GED can improve robustness to task-irrelevant color distractors, even when the color distractors display temporal patterns. The mixed results, however, suggest that generalization is highly dependent on the environment.

5.1.4 Analysis of Learned Latent Representations on Policy Generalization

To obtain a better understanding of how the inductive bias on the representation to disentangle latent factors affects policy generalization, we analyze the structure of the learned latent representation. In addition, we validate the results by analyzing generalization of the learned Q-network.

Structure of Learned Latent Representations

We analyze the structure of the latent representations by projecting the high-dimensional representations into a low-dimensional space using UMAP. We collected 10k observations from both the train and test domains. We ensured identical initial true states by fixing the seed at the start of the episode in both environments and executed the same action trajectories. The dataset thus contains observation pairs that are identical in true state but differ in color. We encoded

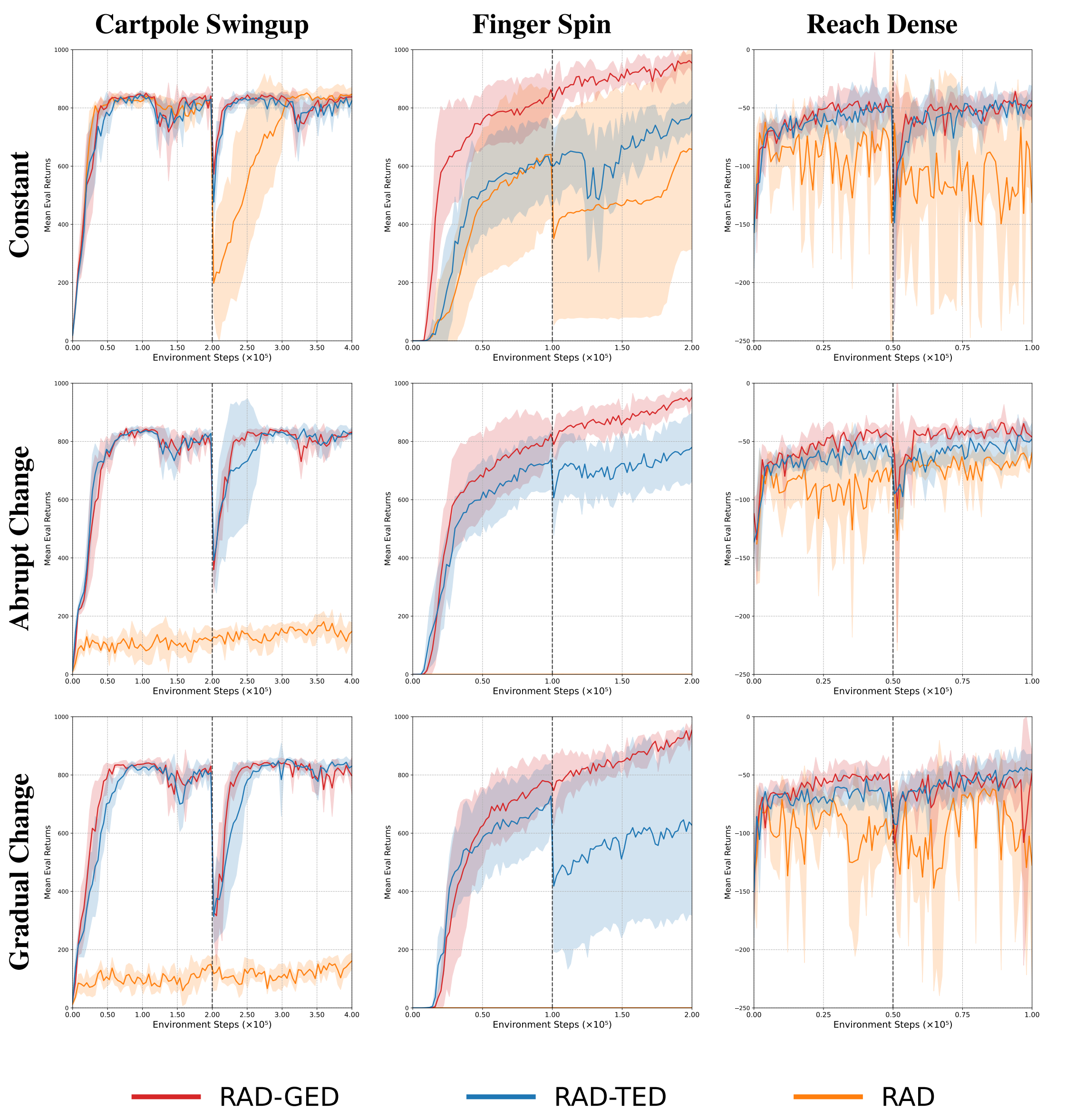


Figure 5.1: Generalization to unseen color distractors in `cartpole` (left column), `finger` (center column) and `reach` (right column). The vertical lines indicate the switch to the test domain. The top row plots the results with constant color distractors. The middle row visualizes the plots with abruptly changing color distractors. The bottom row presents the plots with gradually changing color distractors. Returns are the average of 10 evaluation episodes over 5 seeds. The shaded region shows 95% confidence bounds.

the observations into latent representations using the learned representation network in the train domain. We compare the UMAPs of the RAD encoder h_θ^{RAD} against the GED encoder h_θ^{GED} to contrast between the structure in latent space. Figure 5.2 shows RAD's UMAP on the left and GED's UMAP on the right. Each point in the plot is colored according to the underlying color value of the image observation. The UMAP of GED shows high overlap between observation pairs from the train and test domain. The extracted samples indicate that the encoder h_θ^{GED} preserves local structure based on the true underlying state. Conversely, the latent representation generated by the baseline model fails to retain local similarity between closely related observations. This suggests that GED learned a representation that is relatively invariant to the color distractor.

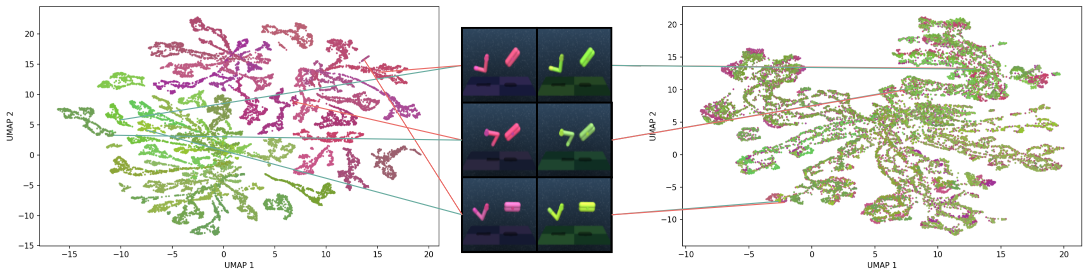


Figure 5.2: UMAP visualizations of the latent space generated by the RAD encoder (left) and RAD-GED encoder (right) over a dataset of 10k observation pairs. The color-coded lines indicate the position of the observation in the two-dimensional UMAP plot. The RAD-GED encoder preserves local distances between observations with identical states. The RAD encoder does not retain local similarity between identical state observations.

Q-Network Generalization

The results in Figure 5.1 indicate how well the policy generalizes to the unseen domain. To validate these results we analyze generalization of the Q-network to identical state-pairs. Similarly to above, we collected 10k observation-pairs from the train and test domain for each environment and color distractor setting pair. We computed the MAE between the learned Q-network $Q_\theta(\mathbf{z}^{train}, \mathbf{a}^{train})$ and $Q_\theta(\mathbf{z}^{test}, \mathbf{a}^{train})$, where the actions $\mathbf{a}^{train}$ are the optimal actions according to the learned policy of the train domain. The MAE should be close to zero when the Q-network assigns similar values for identical true states regardless of color.

Table 5.1 shows the metrics for each environment and color distractor pair. The MAE for RAD with non-stationary color distractors is not reported since the model did not learn a meaningful policy. The average MAE over estimated Q-values in `finger` is consistently below 1.0 for GED. This corresponds with the consistent generalization results shown in Figure 5.1. TED's Q-network generalizes slightly worse when color distractors are present. This aligns with worse policy generalization to the unseen domain. Comparatively, the MAE in `cartpole` and `reach`

		Cartpole Swingup	Finger Spin	Reach Dense
Distractor	**Model**	**MAE**	**MAE**	**MAE**
Constant	RAD	47.474 ± 41.05	11.387 ± 19.545	-
	RAD-TED	10.556 ± 2.929	0.307 ± 0.140	4.303 ± 1.51
	RAD-GED	15.924 ± 11.379	0.4814 ± 0.182	5.213 ± 5.220
Abrupt	RAD	-	-	-
	RAD-TED	14.344 ± 2.584	1.115 ± 0.583	4.138 ± 3.306
	RAD-GED	15.880 ± 1.514	0.701 ± 0.157	3.119 ± 1.801
Gradual	RAD	-	-	-
	RAD-TED	16.717 ± 4.604	1.406 ± 1.782	5.109 ± 2.995
	RAD-GED	19.224 ± 3.980	0.894 ± 0.419	2.822 ± 0.895

Table 5.1: MAE between Q-values estimates evaluated on train domain and test domain observations. MAE is not reported for RAD in abruptly and gradually changing color distractor settings as the model consistently failed to learn a meaningful policy. The MAE is computed using 10k observation pairs, averaged over 5 seeds.

is significantly higher for both models across settings. This aligns with the results presented in Figure 5.1 that show that both agents do not generalize well to the unseen domain in these environments.

5.2 Generalization to Task-Relevant Factors

The second experiment evaluates generalization when task-relevant factors influence the dynamics of the environment.

5.2.1 Implementation Details

We modified the `carracing` environment using the open-source toolkit NS-Gym (Keplinger et al., 2025). This toolkit contains an easy-to-use interface to setup schedulers and update functions. Specifically, we extended the codebase to be able to modify the friction coefficient of the track tiles at each time step in the `carracing` environment. We implemented piecewise uniform and cyclic functions as defined in Chapter 4.3. We clipped the friction coefficient to never exceed the default value in positive direction. We use the Nature CNN (Mnih et al., 2015) with an additional linear projection layer to encode the image observations into latent representations. The encoder takes grayscaled, normalized frame-stacks as input. The actor is conditioned on the encoded observations and outputs parameters (α, β) of a Beta-distribution (not to be confused with the GED loss coefficient α) through separate linear layers with Softplus activations, ensuring positive values (Petrazzini and Antonelo, 2021). Furthermore, we also use the trick of combining throttle/brake for the policy network. We map the actions generated by the policy network of dimension 2 to a valid 3 dimensional action using:

$$\mathbf{a}' = \begin{bmatrix} 2a_0 - 1 \\ 2\max(0, a_1 - 0.5) \\ 2\max(0, 0.5 - a_1) \end{bmatrix}$$

We train the policy on 100 levels. We test generalization on a disjoint set of 100 unseen levels.

5.2.2 Baselines

The base RL algorithm in `carracing` is PPO. The base RL algorithm enhanced with TED is called TED. Our version, GED, is trained with a balanced ratio of same episode and different episode non-temporal pairs. We trained three GED variants. The first variant GED is the default variant. The second variant, denoted as GED-T, defines the auxiliary variable $\mathbf{u}_t = \mathbf{t}$, where $\mathbf{t}$ is the normalized time step in the environment. The third variant defines $\mathbf{u}_t = \mathbf{a}_t$, where $\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{z}_t)$ and is denoted as GED-A.

5.2.3 Results

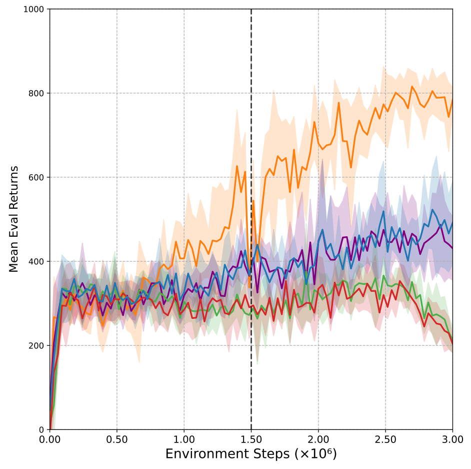
Figure 5.3 illustrates the results of GED variants, TED and PPO in the `carracing` environment with abruptly changing friction (left) and gradually changing friction (right). Notably, we observe that the baseline algorithm PPO significantly outperforms GED and TED in both settings. All models have similar performance in the early stages of training. However, in the second half of training PPO starts to outperform GED and TED considerably, which continues in the test domain. In contrast, GED and GED-A show no signs of improvement in the test domain and fall to lower performance at the final stage of training in both settings. TED and GED-T show mild performance improvements in the setting with abruptly changing friction, but no signs of improvement when the friction changes gradually. This suggests that the disentangled representation objective hurts policy learning in this environment.

5.3 Generalization to Both Task-Relevant and Task-Irrelevant Factors

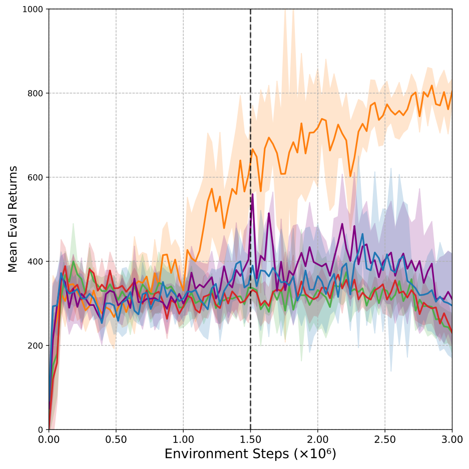
Finally, we evaluate how GED aids generalization to unseen variations of environments with both task-relevant and task-irrelevant factors of variation.

5.3.1 Implementation Details

We show the results in Procgen environments `coinrun` and `leaper`. We train on 100 levels on the easy difficulty and test generalization on 100 unseen easy levels. We trained GED with non-temporal sample pairs.



(**a**) Abruptly changing friction



(**b**) Gradually changing friction

Figure 5.3: Generalization to unseen tracks in `carracing` with intra-episodic (**a**) abruptly changing friction and (**b**) gradually changing friction. PPO outperforms both PPO-GED and PPO-TED, showing minimal difficulty adapting to changes in the friction coefficient. Returns are the average of 10 evaluation episodes over 5 seeds. The shaded region shows 95% confidence bounds.

5.3.2 Baselines

The base RL algorithm for both `procgen` environments is PPO. The second benchmark algorithm is TED.

5.3.3 Results

The results are shown in Figure 5.4. In `coinrun` the base RL algorithm PPO is clearly the quickest to learn. Moreover, we do not observe a noticeable drop in performance when switching to unseen levels. GED takes longer to learn, but obtains similar returns at the end of training. GED generalizes to the test levels, but also exhibits relatively larger variance in the test domain. In comparison, TED underperforms both algorithms during training. At test time, TED does not improve over the results obtained in the train domain. Its large variance indicates that this is due to poor initialization. In `leaper`, PPO overfits on the training levels, after which returns degenerate. Although its performance does not drop immediately after switching to the test levels, PPO continues to slowly degrade in performance as training updates continue. GED has slightly lower maximum returns and also experiences a decline in the train domain, but is more robust. At test time, it continues to improve its performance. TED underperforms both algorithms during training, but does obtain higher returns at the end of testing compared to PPO.

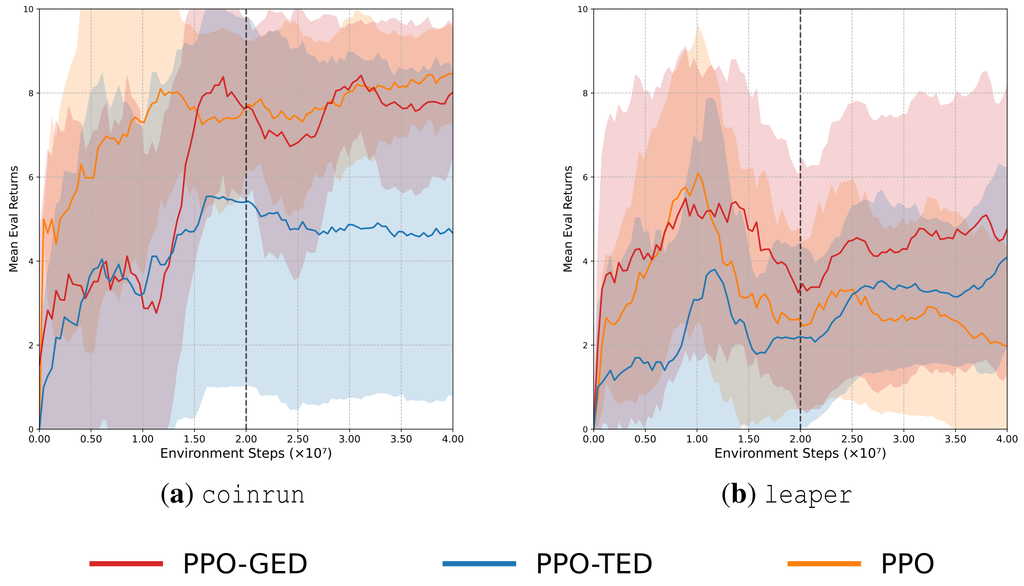


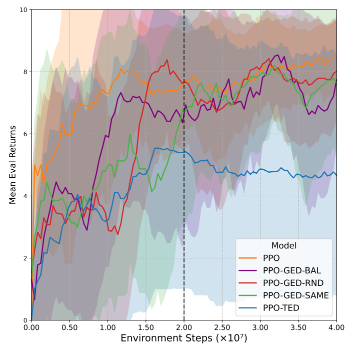
Figure 5.4: Results of generalization to task-relevant and task-irrelevant factors of variation. The vertical dotted line indicates the switch to unseen levels. PPO-GED shows consistent performance in both environments. PPO does not generalize well in `leaper`. PPO-TED is relatively unstable, decreasing average performance. Returns are the average of 8 evaluation episodes over 5 seeds. The shaded region shows 95% confidence bounds. The graphs plot the 10-point rolling average for readability.

5.4 Ablations

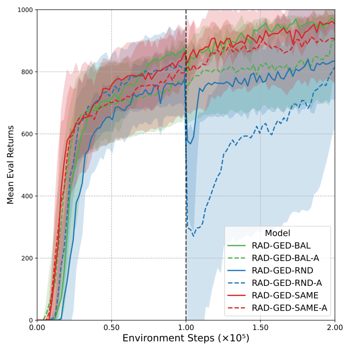
We present the results of the ablations to show how different parts of the methodology influence the results. First, we compare different non-temporal sampling schemes in Subsection 5.4.1. Next, we investigate the effect of the action as auxiliary variable in Subsection 5.4.2. Lastly, in Subsection 5.4.3 we investigate the robustness of GED with different loss coefficients.

5.4.1 Choice of Non-Temporal Samples

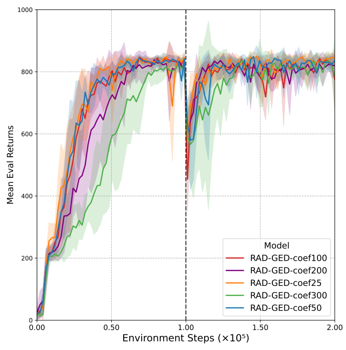
In Section 3.2 we defined three non-temporal samples: (i) balanced ratio non-temporal samples, (ii) same episode non-temporal samples, and (iii) non-temporal samples. Figure 5.5a shows generalization to unseen variations of environments in `coinrun` with each non-temporal sampling scheme. We denote the balanced ratio, same-episode non-temporal samples and generally non-temporal sampling schemes as '-BAL', '-SAME' and '-RND', respectively. Note that the balanced ratio sampling scheme can be viewed as TED with its optimization objective $\mathcal{L}_{TED}$ replaced with our optimization objective $\mathcal{L}_{GED}$.



(**a**) `coinrun`



(**b**) `finger`



(**c**) `cartpole`

Figure 5.5: Figure 5.5a shows GED in `coinrun` with different settings for non-temporal samples. Figure 5.5b presents results in `finger` with actions as the auxiliary variable. Figure 5.5c shows GED with different values for the loss coefficient α.

All three variations are slower to reach performance levels obtained by PPO, with GED-BAL and GED-SAME reaching lower mean returns at the end of training. In the test domain all the variants obtain similar performance, but GED-BAL and GED-SAME show more variance. This shows that GED is relatively robust to different non-temporal pairs during contrastive learning.

5.4.2 Action as Auxiliary Variable

We investigated how defining the auxiliary variable $\mathbf{u}_t$ as the action between $\mathbf{s}_t$ and $\mathbf{s}_{t+1}$ affects generalization. We show the results with the three sampling schemes '-SAME', '-BAL', and '-RND'. The models without the auxiliary variable are presented with solid lines. The models with auxiliary variables are denoted as '-A' and presented with dotted lines. The results are presented in Figure 5.5b. The results indicate that the action as the auxiliary variable can significantly hurt generalization. Although this is reasonable given that the color distractor does not depend on the executed action, it highlights that the auxiliary variable can negatively affect the learned representations.

5.4.3 Robustness to GED Loss Coefficient

Next, we evaluate GED with different values for the loss coefficient α to obtain insight into how much it affects robustness. The results presented in Figure 5.5c depict the performance in `cartpole` with constant color distractors. Interestingly, both lower and higher values of the coefficient α show improved zero-shot generalization. Lower α values show similar performance in the train domain, but the coefficient $\alpha = 25$ has worse training stability. In contrast, higher α values of $\alpha = 200$ and $\alpha = 300$ take longer to train. Moreover, $\alpha = 300$ is relatively unstable in the test domain, indicated by the large confidence interval. Overall, this shows that the GED is relatively robust to different hyperparameters of α but that tuning is required for optimal performance.

Chapter 6

Discussion

This chapter begins by interpreting the results of Chapter 5 in Section 6.1. Next, in Section 6.2, we discuss limitations of our research.

6.1 Interpretation of Results

This work concerned itself with leveraging temporal structure to disentangle latent representations in pixel-based deep reinforcement learning. We introduced Generalized Disentanglement (GED), an auxiliary contrastive task that synthesizes Temporal Disentanglement with the Nonlinear Independent Component Analysis with Auxiliary Variables framework to isolate factors of variation in the environment. We focused our experiments on settings where variables exhibit temporal structure. Our main hypothesis was that disentangling the latent representations improves generalization to domain shifts.

Our experiments demonstrate that agents can benefit from disentangled representation learning with visual task-relevant and task-irrelevant variables. Mainly, the results show that GED can achieve better generalization to unseen domains with task-irrelevant distractors, even when these factors are modulated according to a non-stationary function. Our analysis suggests that in these cases the value of the color is assigned less weight, increasing invariance to shifts in the task-irrelevant factor. However, we stress that we did not observe consistent generalization across environments. We hypothesize that this is due to differences in environmental settings. To illustrate, the initial position in the `cartpole` environment is always in the middle of the frame, with relatively small deviations. In contrast, `finger` has a broader range of initial positions, which naturally helps to learn a more robust policy. This highlights that self-supervised representation learning does not guarantee that the agent minimizes the influence of task-irrelevant factors on the policy. Secondly, our methodology reduces the risk of overfitting to environmental variations. However, with few such variations, it does not consistently outperform the baseline algorithm. Although these results align with earlier research that finds that the optimal self-supervised learning method depends on the RL task (Li et al., 2023), this limits the practical applicability of our method.

We also aimed to study whether a disentangled representation can improve policy generalization in settings where the dynamics of the environment changes gradually or abruptly due to an implicit factor of variation. We did not find that our methodology improved the policy compared to the baseline algorithm. Given that we evaluated our methodology in a single environment, the findings from this experiment should be interpreted with caution. We can only suggest that in this environment, a disentangled representation negatively affects policy learning. A more rigorous investigation in a diverse set of environments and settings would be necessary to evaluate whether our auxiliary task can improve policy generalization under dynamics shifts.

Our methodology considers different types of optimal negative pairs to enhance generalization to unseen domains. Our results show that exclusively defining same episode non-temporal pairs as negative samples can remarkably improve generalization to changes in task-irrelevant factors, even when the task-irrelevant factor changes throughout an episode. Ablation studies confirmed that different types of non-temporal samples have different results in terms of stability and generalization to unseen domains. This implies that the type of non-temporal samples should be carefully considered.

We compared GED against TED to investigate whether our methodology has practical benefits to improve representation learning and generalization. We demonstrated that our extensions improve upon the benchmark algorithm, showing improved policy learning and increased generalization. We stress, however, that our results deviate from the results presented by Dunion et al. (2023b) in the same environments. This is despite our best efforts to ensure a fair comparison by selecting identical algorithms, environmental settings and hyperparameters in all experiments. Specifically, in both Procgen environments, the underperformance of TED can be attributed to poor initialization. We hypothesize that the relatively few classifier parameters may not capture the environmental structure as well when environmental variations are sparse.

While our main results suggest that GED is relatively stable across experiments using loss-coefficients α tuned for TED, the ablations show that other choices for the loss-coefficient α improve generalization to the unseen domain. This suggests that our results could have been different if the hyperparameters were optimized separately. We recommend future research to conduct independent hyperparameter optimization of both methodologies.

To conclude, our empirical research highlights the importance of structured state representation learning as a promising avenue to improve the robustness of RL algorithms, broadening the applicability of RL algorithms for real-world deployment.

6.2 Limitations

This research has potential limitations. In the following section, we discuss key limitations related to benchmark availability and measuring disentanglement.

6.2.1 Available Benchmarks

We restricted our research to compare GED against the baseline algorithms and to the benchmark methodology TED (Dunion et al., 2023b). We evaluated the models in different environments to show how GED compares to TED in a variety of settings. Our research leveraged existing libraries such as `distracting-control` and `ns-gym` to assess both algorithms in more challenging settings that exhibit temporal structure. Preferably, our research would have also compared GED against other methods in non-stationary reinforcement learning. However, there is no single benchmark that is widely adopted for evaluation. This complicates the process of evaluating GED against existing approaches. Future research could validate our methodology against a wider array of methodologies.

6.2.2 Measuring Disentanglement

Previous work (Dunion et al., 2023b) measured disentanglement in the environments with task-irrelevant color distractors by computing the Z-diff score. The Z-diff score is a supervised disentanglement metric that uses a simple linear classifier to identify which component is held fixed in a batch of latent representations. Initially, we mirrored this approach to evaluate the level of disentanglement of the learned representation. However, we found that the Z-diff score is not a reliable metric for quantifying disentanglement in RL settings. Specifically, the score is highly dependent on sample diversity, the number of factors and the label values. Moreover, the description provided by previous work does not allow for reproduction of the results. This brings us to the more important observation that interpretation of learned representations in deep reinforcement learning is challenging. We underscore the necessity to develop interpretability tools specifically designed for unsupervised datasets, natural in pixel-based deep RL.

Chapter 7

Related Work

This chapter reviews existing approaches related to non-stationary RL. First, we discuss research on the topic of experience replay in Section 7.1. Secondly, we examine knowledge distillation as an approach to cope with non-stationarity in Section 7.2. Next, we discuss various approaches using context detection in Section 7.3. In Section 7.4, we review several methods that leverage domain randomization to improve generalization to domain shifts. Following, Section 7.5 discusses approaches that use meta-learning to learn robust policies in non-stationary RL. Finally, Section 7.6 surveys existing approaches in state representation learning.

7.1 Experience Replay

One of the problems with distribution shift is that the agent forgets previously learned tasks (Rolnick et al., 2019; Kaplanis et al., 2020). This phenomenon, also called catastrophic forgetting, can be prevented by using an experience buffer (Isele and Cosgun, 2018; Liu et al., 2021; Queeney et al., 2021; Venuto et al., 2021). The experience buffer stores a record of previous states, actions, and rewards at each time step. During training, these samples are drawn from the buffer. Backpropagation over the past samples supports the agent to remember how to handle previous experiences, mitigating the sudden failure of previously learned tasks. An obvious drawback of this approach is the additional storage requirements to maintain a large buffer. Pseudo-rehearsal (Atkinson et al., 2021) addresses this by generating previously seen data using a deep-generative model. This has the benefit that only the experiences of the current task need to be retained while maintaining the ability to train on previously relevant samples. Experience replay has been demonstrated to be an effective method for reducing the risk of catastrophic forgetting, but its usage is deeply limited by the diversity of samples it stores. The experience buffer is a vital component of our approach. Its use, in addition to preventing catastrophic forgetting, is to keep track of which records belong to which time step, which is a crucial detail in the sampling strategy of our methodology.

7.2 Knowledge Distillation

A second popular approach is knowledge distillation (Buciluă et al., 2006; Hinton et al., 2015). Knowledge distillation is the process of transferring information from one neural network, called the target network, to another neural network. A benefit of knowledge distillation in RL is stabilization of gradient updates, which reduces the risk of catastrophic forgetting Rusu et al. (2016); Li and Hoiem (2018); Espeholt et al. (2018); Igl et al. (2021); Lan et al. (2023); Zhang et al. (2023). Through knowledge distillation, a neural network gradually matches the target network. Our methodology employs a target network to stabilize the training of the representation function.

7.3 Context Detection

Another commonly studied approach to non-stationary RL is context detection (Padakandla et al., 2020; Alegre et al., 2021; Chen et al., 2022). These approaches take the perspective that the environment can be described by a possibly infinite number of different MDPs. Each MDP is characterized by its context, which refers to the specific settings of the MDP. Change-point detection algorithms seek to find the point at which the MDP changes from one MDP to another. Successful identification of the current MDP enables the policy to shift its actions such that they are optimal for the current context. Lifelong Latent Actor Critic (LILAC) (Xie et al., 2020) is an off-policy SAC-based algorithm proposed to effectively deal with lifelong episodic non-stationarity by identifying the hidden parameters $\mathbf{z}$, or context, at each episode using an LSTM network. The policy network is conditioned on $\mathbf{s}$ and inferred latents $\mathbf{z}$ to execute actions $\mathbf{a}$ that are optimal for the current environment. ZeUS (Sodhani et al., 2021) learns a Lipschitz continuous context space that captures the environment's change in dynamics or rewards. The high-level idea of learning this context space is that similar tasks are close in latent space. Consequently, small changes in the context space lead to small changes in the policy. During evaluation, the agent infers the contexts from a few interactions with the environment and adapts accordingly. Reactive Exploration (Steinparz et al., 2022) utilizes context detection to improve its exploration strategy. Reactive Exploration keeps track of the expected future reward at every time step. When the predicted reward and the actual reward deviate too strongly, the agent adapts by prioritizing exploration of the environment. This has the benefit that the policy recovers faster. Our methodology does not consider active change point detection techniques. That is, there is no active component in our system that notifies the agent that the environment has (partially) shifted. Instead, we leverage temporal structure within the environment to identify original factors of variation. It is completely up to policy optimization whether changes in disentangled latent factors are leveraged to improve the policy.

7.4 Domain Randomization

Domain randomization subjects the agent to small random changes in the simulated environment, such as randomization of textures, colors, lighting, and camera poses (Sadeghi and Levine, 2017; Tobin et al., 2017). This, in turn, encourages the agent to learn a more robust policy that can cope with changes in the environment. A specific application of domain randomization is Sim2Real: training a policy in a simulated environment and deploying it in the real world. This influential work by Tobin et al. (2017) showcased that training in a low-cost simulated environment is a viable approach to training robots. One of the meaningful extensions to this work addressed the problem of the simulated-to-reality gap by converting simulated images into photorealistic images using a generative model (Rao et al., 2020). The drawback of this approach is that it requires some real image data. SimGAN (Jiang et al., 2021) considers the problem of tuning the simulation such that the parameter distributions better match the dynamics of the real world. SimGAN uses adversarial reinforcement learning to train an agent to change the simulated environment's parameters such that they mimic the real world more accurately. Domain randomization typically samples independently and identically distributed environment noise. In contrast, our work focuses on disentangling the factors of variation by exploiting the temporal structure in the data. We do note that the main theoretical framework underlying our methodology does require noise/variation in the environment, which would naturally aid generalization. Nevertheless, our experiments are set up such that the isolated noise component during training does not explain the generalization performance in our test domain.

7.5 Meta-Learning in RL

Meta-learning takes the perspective that learning can also be learned (Schmidhuber, 1987; Duan et al., 2016; Wang et al., 2017; Xu et al., 2018; Nagabandi et al., 2019). In this setting, an agent first has to learn to generalize on many different tasks simultaneously, after which it is forced to learn a new set of tasks. Meta-RL (Wang et al., 2017) approaches meta-learning by using a recurrent neural network. In short, the Long-Short Term Memory (LSTM) enables the agent to maintain a memory of past actions and rewards. This effectively supports the agent to infer shifts in the environment, allowing the agent to adapt to different tasks. Alternatively, there are also gradient-based approaches to meta-learning (Finn et al., 2017). Specifically, Model Agnostic Meta-Learning (MAML) is one of the influential algorithms in meta-learning-based reinforcement learning (Finn et al., 2017). MAML optimizes the parameters θ by updating the gradients with respect to a batch of K examples of T sampled tasks. This high-level framework makes MAML highly attractive for the application of domain adaptation in non-stationary RL (Javed and White, 2019; Arndt et al., 2019; Caccia et al., 2021). Other works use model-based approaches to meta-reinforcement learning (Nagabandi et al., 2019; Lin et al., 2021). Chandak et al. (2020) propose to forecast future performance to specifically deal with smooth non-stationary changes. Achieving the meta-learning objective of effectively and efficiently learning how to learn new tasks should result in sample-efficient retraining for novel tasks. Our work also evaluates the proposed methodology in terms of efficient retraining. However, we do not consider the objective of learning novel tasks. Instead, we evaluate whether our methodology is capable of doing the same task well when the environment transitions to an unseen domain.

7.6 State Representation Learning

State representation learning has been the subject of much research in pixel-based deep RL. Various work (Laskin et al., 2020; Yarats et al., 2020; Hansen et al., 2021) focuses on image augmentations to aid generalization. Reinforcement learning with Augmented Data (RAD) (Laskin et al., 2020) addresses the problem of sample efficiency and generalization in image-based deep reinforcement learning by applying data augmentations to images during training. This work demonstrates that this technique effectively increases returns in fewer environmental steps and improves generalization to novel domains. We use RAD as our baseline algorithm, as it has the advantage of increased generalization and sample efficiency. However, data augmentation techniques do not necessarily learn a structural representation. Moreover, this technique does not reliably generalize to stronger types of variation (Kirk et al., 2023). GED can be used alongside data augmentation techniques to improve generalization and improve robustness to domain shifts.

Comparatively, invariance techniques have been shown to be able to effectively cope with visual domain shifts. Zhang et al. (2020) deals with visual changes and distractors by learning invariant causal representations that generalize across environments with different observation functions but shared underlying dynamics. The method isolates task-relevant latent features while filtering out spurious, environment-specific visual variations such as background color or camera angle changes. This enables robust policy learning that maintains performance despite visual perturbations. Deep Bisimulation for Control (DBC) by Zhang et al. (2021) learns a representation that disregards task-irrelevant information by minimizing an auxiliary bisimulation metric. These methods are effective in dealing with visual task-irrelevant factors of variation, but have several limitations, including embedding explosion (Kemertas and Aumentado-Armstrong, 2021) and latent instability. Moreover, these methods do not address the problem of dealing with task-relevant visual distractors.

Other lines of work on state representation learning encourage the representation to capture relevant structure in the environment by incorporating inductive biases. For example, some methods encourage the learned representation to produce similar representations for related inputs: Srinivas et al. (2020) match representations of different augmentations of the same frame, Mazoure et al. (2020) match representations from consecutive time steps, and Agarwal et al. (2021) align representations based on policy similarity metrics. Van den Oord et al. (2019), instead, predicts future representations. Schwarzer et al. (2021) pretrain an encoder for later fine-tuning. In contrast to GED, these approaches do not encourage the encoder to learn a disentangled representation.

Closest to our methodology is research specifically focused on learning a disentangled representation to improve generalization in RL. DARLA (Higgins et al., 2018) addresses visual domain shifts by learning a disentangled state representation in RL using a pretrained β-VAE. DARLA achieves zero-shot domain transfer by effectively isolating the different factors of variation (object shape, color) in the training domain. DARLA demonstrated that learning a disentangled,

domain-invariant representation enables RL policies to generalize across visually changing environments. The downside of DARLA is that policy transfer is not guaranteed, since the policy could still overfit to domain-specific features. Latent Unified State Representation (Xing et al., 2021) solves this using a two-stage approach where the learned embedding is guided to disentangle domain-specific and domain-general factors of variation. This perspective of learning a representation that divides domain-specific and domain-general factors of variation is foundational for AdaRL (Huang et al., 2022). AdaRL learns a Dynamic Bayesian Network (DBN) over observations, latent states, rewards, actions, and domain-specific change factors that are shared across domains. This graphical model is leveraged to identify minimal sufficient representations of underlying domain-shared latent states and domain-specific change factors. After transfer to the target domain, the DBN estimates the target domain's change factors, allowing reuse of the source policy conditioned on the minimal sufficient representation without further training. FANS-RL (Feng et al., 2022) extends AdaRL by considering non-stationary change factors. This enables the method to learn a representation that captures environmental shifts both within and between episodes. Resultingly, both methods demonstrate improved adaptation to domain shifts. AdaRL/FANS-RL use causal DBNs to explicitly model changes via factored representations and minimal sufficient representations for efficient adaptation. Our method employs self-supervised contrastive learning between temporal and non-temporal pairs, focusing on robust generalization to unseen domains without graphs or explicit change factors.
As introduced earlier, our work extends TED (Dunion et al., 2023b), an auxiliary learning task that can be applied to existing pixel-based RL algorithms. TED implements a variation of PCL to improve generalization to target domains by effectively disentangling relevant and/or irrelevant factors of variation. The follow-up research in Nonlinear Independent Component Analysis proposed a general framework to recover temporally dependent stationary sources and non-stationary sources (Hyvärinen et al., 2019). The presented work applies these ideas to construct a more flexible and general method. Our methodology obtains results that are on par with or exceed TED in non-stationary reinforcement learning settings. Subsequently, Dunion et al. propose two additional disentangled representation learning methods. First, Conditional Mutual Information for Disentanglement (Dunion et al., 2023a) disentangles representations by minimizing conditional mutual information between latent components to address generalization when factors of variation are correlated. The presented results show high robustness to correlation shift. Secondly, Multi-View Disentanglement (Dunion and Albrecht, 2024) (MVD) learns a disentangled representation from multiple camera views, increasing robustness to failure of cameras.

Chapter 8

Conclusion

This thesis presented Generalized Disentanglement, an auxiliary contrastive learning task that adopts the Nonlinear Independent Component Analysis with Auxiliary Variables framework to disentangle latent representations in pixel-based reinforcement learning. In comparison to earlier work, GED is more flexible in the choice of non-temporal pairs, considerably simplifying the sampling scheme during representation learning. Moreover, our results show that our optimization objective results in more stable learning, increasing the applicability of the auxiliary task to a wider variety of settings.

We evaluated our methodology in a broad range of environments with both continuous and discrete control algorithms. First, we considered generalization to task-irrelevant factors of variation that change over time, both between episodes and within episodes. Our results showed that our methodology can significantly improve generalization to unseen domains, though this is highly dependent on the environment. Secondly, in our experiment concerning task-relevant factors, we did not observe an improvement over the baseline algorithm. This finding indicates that a disentangled representation does not always benefit policy learning. Finally, we considered environments with task-relevant and task-irrelevant variables. GED reduced overfitting and achieved higher average cumulative rewards with better generalization to unseen levels.

The primary objective of our proposed auxiliary task is to make RL algorithms more robust to changes in the environment. GED accomplishes this by leveraging temporal structure to disentangle latent representations, improving generalization to unseen domains. These findings highlight the importance of state representation learning in pixel-based deep RL, motivating continued research in this area.

Chapter 9

Future Work

Our research demonstrates that the proposed auxiliary self-supervised representation learning task enhances generalization to unseen environments. There are several promising research directions related to self-supervised representation learning in pixel-based deep RL that could extend our methodology and address limitations of our approach.

Extend Auxiliary Variables

Our approach allows for an optional auxiliary variable that could improve the learned representations. Although we briefly consider different choices of auxiliary variables, our empirical results do not indicate that our chosen auxiliary variables meaningfully improve policy generalization in our investigated settings. Future work could investigate how different types of auxiliary variables can improve the learned representation. As an example, there is an option to define the auxiliary variable as the domain index when we have prior knowledge of the domains. Alternatively to discriminating between temporal and non-temporal pairs, this allows us to contrast between actual pairs $\tilde{\mathbf{x}} = (\mathbf{z}_t, D_i)$ and shuffled samples $\tilde{\mathbf{x}}^* = (\mathbf{z}_t, D_i^*)$, where D_i is the index of the ith domain and D_i^* is randomly sampled, independently from $\mathbf{z}_t$. Through contrastive learning, this could induce the latent representation to disentangle domain-specific features from domain-shared features. This can potentially support policy generalization in settings where the optimal policy shifts between domains.

Independent Innovation Analysis

The subsequent work on Nonlinear ICA (Hyvärinen and Morioka, 2016, 2017; Hyvärinen et al., 2019) introduced Independent Innovation Analysis (IIA) (Morioka et al., 2021). IIA can be viewed as an alternative formulation to the problem of recovering true variables from mixed latent variables:

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{s}_t),$$

where $\mathbf{f}: \mathbb{R}^{2n} \rightarrow \mathbb{R}^n$ represents a Nonlinear Vector Autoregressive (NVAR) mixing model, $\mathbf{x}_t = [\mathbf{x}_1, \cdots, \mathbf{x}_t]^T$ are the observations and $\mathbf{s}_t = [s_1, ..., s_t]^T$ are the innovations for every time step t. This formulation essentially states that the current observation is a nonlinear mixing between the previous observation and a temporally independent innovation, or error. We can take the general ideas proposed by Morioka et al. to create a novel method that is analogous to GED. Concretely, this formulation lends itself well to pixel-based RL algorithms that implement a sequential network as the encoder h_θ, such as a Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) or Transformer (Vaswani et al., 2017), to encode a sequence of image observations into a latent representation. The main advantage of this extension would be a stronger inductive bias to isolate the factors of variation in separate latent components of the representation. The obvious downside is that this framework can only be applied to sequential encoders, limiting the applicability of this extension in pixel-based RL.

Connectivity-Contrastive Learning

A limitation of Nonlinear ICA is its assumption that there are no causal relations between source signals. Morioka and Hyvärinen (2023) propose Connectivity-Contrastive Learning (CCL) as a framework to aid causal discovery during representation learning. Concretely, CCL encourages spatial structure in learned representations. The model assumes that we have n two-dimensional matrix observations $\mathbf{X}^{(n)} \in \mathbb{R}^{p \times d}$. The elements of each matrix $x_{ai}^{(n)}$ are collected from multiple nodes $a \in \mathcal{V}$ ($|\mathcal{V}| = p$) and across multiple observational modalities $i \in \{1, ..., d\}$. The observed data is assumed to be a combination of latent variables, mixed using an unknown mixing function for each node a, component j and sample n:

$$x_a^{(n)} = \mathbf{f}(\mathbf{s}_a^{(n)}),$$

where $\mathbf{f}: \mathbb{R}^d \rightarrow \mathbb{R}^d$ is the observational mixing function and $\mathbf{s}_a^{(n)} = (s_{aj})_j^{(n)} \in \mathbb{R}^d$ is the a-th row of the latent matrix $\mathbf{S}^{(n)} = (s_{aj})^{(n)} \in \mathbb{R}^{p \times d}$. The learning algorithm trains a feature extractor $h: \mathbb{R}^d \rightarrow \mathbb{R}^d$ by optimizing a multinomial logistic regression to predict pair labels (a, b) from paired observations $(\mathbf{x}_a, \mathbf{x}_b)$ for every pair and sample. Similarly to GED, this algorithm can be integrated in pixel-based deep RL as a self-supervised auxiliary task. A potential benefit of CCL is that its focus on spatial structure can improve generalization to changes in task-irrelevant factors. However, a potential downside of this algorithm is scalability as the pairwise classification objective grows with the number of nodes. Future work could investigate how the number of nodes scales with performance or combine CCL with Slot Attention (Locatello et al., 2020).

Bibliography

Rishabh Agarwal, Marlos C. Machado, Pablo Samuel Castro, and Marc G. Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning, 2021. URL `https://arxiv.org/abs/2101.05265`.

Lucas N. Alegre, Ana L. C. Bazzan, and Bruno C. da Silva. Minimum-delay adaptation in non-stationary reinforcement learning via online high-confidence change-point detection, 2021. URL `https://arxiv.org/abs/2105.09452`.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, et al. What matters for on-policy deep actor-critic methods? a large-scale study. In *International conference on learning representations*, 2021.

Karol Arndt, Murtaza Hazara, Ali Ghadirzadeh, and Ville Kyrki. Meta reinforcement learning for sim-to-real domain adaptation, 2019. URL `https://arxiv.org/abs/1909.12906`.

Craig Atkinson, Brendan McCane, Lech Szymanski, and Anthony Robins. Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting. *Neurocomputing*, 428:291–307, March 2021. ISSN 0925-2312. doi: 10.1016/j.neucom.2020.11.050. URL `http://dx.doi.org/10.1016/j.neucom.2020.11.050`.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives, 2014. URL `https://arxiv.org/abs/1206.5538`.

Wendelin Böhmer, Jost Tobias Springenberg, Joschka Boedecker, Martin Riedmiller, and Klaus Obermayer. Autonomous learning of state representations for control: An emerging field aims to autonomously learn state representations for reinforcement learning agents from their real-world sensor observations. *KI-Künstliche Intelligenz*, 29(4):353–362, 2015.

Nicolò Botteghi, Mannes Poel, and Christoph Brune. Unsupervised representation learning in deep reinforcement learning: A review. *IEEE Control Systems*, 45(2):26–68, 2025.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. URL `https://arxiv.org/abs/1606.01540`.

Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

Gilles Burel. Blind separation of sources: A nonlinear neural algorithm. *Neural Networks*, 5(6): 937–947, Nov 1992. doi: 10.1016/S0893-6080(05)80090-5. URL `https://hal.science/hal-03223331`.

Massimo Caccia, Pau Rodriguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Caccia, Issam Laradji, Irina Rish, Alexandre Lacoste, David Vazquez, and Laurent Charlin. Online fast adaptation and knowledge accumulation: a new approach to continual learning, 2021. URL `https://arxiv.org/abs/2003.05856`.

Yash Chandak, Georgios Theocharous, Shiv Shankar, Martha White, Sridhar Mahadevan, and Philip S. Thomas. Optimizing for the future in non-stationary mdps, 2020. URL `https://arxiv.org/abs/2005.08158`.

Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979, 2019. doi: 10.1109/ICRA.2019.8793789.

Xiaoyu Chen, Xiangming Zhu, Yufeng Zheng, Pushi Zhang, Li Zhao, Wenxue Cheng, Peng Cheng, Yongqiang Xiong, Tao Qin, Jianyu Chen, et al. An adaptive deep rl method for non-stationary environments with piecewise stable context. *Advances in Neural Information Processing Systems*, 35:35449–35461, 2022. URL `https://arxiv.org/abs/2212.12735`.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning, 2020. URL `https://arxiv.org/abs/1912.01588`.

Pierre Comon. Independent component analysis, a new concept? *Signal processing*, 36(3): 287–314, 1994.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning, 2016.

Gustavo Deco and Wilfried Brauer. Nonlinear higher-order statistical decorrelation by volume-conserving neural architectures. *Neural Networks*, 8(4):525–535, 1995. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(94)00108-X. URL `https://www.sciencedirect.com/science/article/pii/089360809400108X`.

Gustavo Deco and Dragan Obradovic. *An Information-Theoretic Approach to Neural Computing*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 1996. ISBN 0387946667.

Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction, 2016. URL `https://arxiv.org/abs/1505.05192`.

Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks, 2015. URL https://arxiv.org/abs/1406.6909.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning, 2016. URL https://arxiv.org/abs/1611.02779.

Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. An empirical investigation of the challenges of real-world reinforcement learning, 2021. URL https://arxiv.org/abs/2003.11881.

Mhairi Dunion and Stefano V. Albrecht. Multi-view disentanglement for reinforcement learning with multiple cameras, 2024. URL https://arxiv.org/abs/2404.14064.

Mhairi Dunion, Trevor McInroe, Kevin Sebastian Luck, Josiah P. Hanna, and Stefano V. Albrecht. Conditional mutual information for disentangled representations in reinforcement learning, 2023a. URL https://arxiv.org/abs/2305.14133.

Mhairi Dunion, Trevor McInroe, Kevin Sebastian Luck, Josiah P. Hanna, and Stefano V. Albrecht. Temporal disentanglement of representations for improved generalisation in reinforcement learning, 2023b. URL https://arxiv.org/abs/2207.05480.

Ayoub Echchahed and Pablo Samuel Castro. A survey of state representation learning for deep reinforcement learning. *arXiv preprint arXiv:2506.17518*, 2025.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018. URL https://arxiv.org/abs/1802.01561.

Fan Feng, Biwei Huang, Kun Zhang, and Sara Magliacane. Factored adaptation for non-stationary reinforcement learning, 2022. URL https://arxiv.org/abs/2203.16582.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017. URL https://arxiv.org/abs/1703.03400.

Quentin Gallouédec, Nicolas Cazin, Emmanuel Dellandréa, and Liming Chen. panda-gym: Open-source goal-conditioned environments for robotic learning, 2021. URL https://arxiv.org/abs/2106.13687.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations, 2018. URL https://arxiv.org/abs/1803.07728.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://arxiv.org/abs/1801.01290.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Stabilizing deep q-learning with convnets and vision transformers under data augmentation, 2021. URL `https://arxiv.org/abs/2107.00644`.

Irina Higgins, Arka Pal, Andrei A. Rusu, Loic Matthey, Christopher P Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning, 2018. URL `https://arxiv.org/abs/1707.08475`.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL `https://arxiv.org/abs/1503.02531`.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Biwei Huang, Fan Feng, Chaochao Lu, Sara Magliacane, and Kun Zhang. Adarl: What, where, and how to adapt in transfer reinforcement learning, 2022. URL `https://arxiv.org/abs/2107.02729`.

Aapo Hyvärinen and Hiroshi Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ica, 2016. URL `https://arxiv.org/abs/1605.06336`.

Aapo Hyvärinen and Hiroshi Morioka. Nonlinear ica of temporally dependent stationary sources. In *Artificial intelligence and statistics*, pages 460–469. PMLR, 2017.

Aapo Hyvärinen and Petteri Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12(3):429–439, 1999. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(98)00140-3. URL `https://www.sciencedirect.com/science/article/pii/S0893608098001403`.

Aapo Hyvärinen, Hiroaki Sasaki, and Richard E. Turner. Nonlinear ica using auxiliary variables and generalized contrastive learning, 2019. URL `https://arxiv.org/abs/1805.08651`.

Aleksi Hämäläinen, Karol Arndt, Ali Ghadirzadeh, and Ville Kyrki. Affordance learning for end-to-end visuomotor robot control. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1781–1788, 2019. doi: 10.1109/IROS40897.2019.8968596.

Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. Transient non-stationarity and generalisation in deep reinforcement learning, 2021. URL `https://arxiv.org/abs/2006.05826`.

David Isele and Akansel Cosgun. Selective experience replay for lifelong learning, 2018. URL `https://arxiv.org/abs/1802.10269`.

Khurram Javed and Martha White. Meta-learning representations for continual learning. *Advances in neural information processing systems*, 32, 2019.

Yifeng Jiang, Tingnan Zhang, Daniel Ho, Yunfei Bai, C. Karen Liu, Sergey Levine, and Jie Tan. Simgan: Hybrid simulator identification for domain adaptation via adversarial reinforcement learning, 2021. URL `https://arxiv.org/abs/2101.06005`.

Christian Jutten and Jeanny Herault. Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24(1):1–10, 1991. ISSN 0165-1684. doi: https://doi.org/10.1016/0165-1684(91)90079-X. URL `https://www.sciencedirect.com/science/article/pii/016516849190079X`.

Christos Kaplanis, Claudia Clopath, and Murray Shanahan. Continual reinforcement learning with multi-timescale replay, 2020. URL `https://arxiv.org/abs/2004.07530`.

Mete Kemertas and Tristan Aumentado-Armstrong. Towards robust bisimulation metric learning. *Advances in Neural Information Processing Systems*, 34:4764–4777, 2021.

Nathaniel S. Keplinger, Baiting Luo, Iliyas Bektas, Yunuo Zhang, Kyle Hollins Wray, Aron Laszka, Abhishek Dubey, and Ayan Mukhopadhyay. Ns-gym: Open-source simulation environments and benchmarks for non-stationary markov decision processes, 2025. URL `https://arxiv.org/abs/2501.09646`.

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives, 2022. URL `https://arxiv.org/abs/2012.13490`.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76: 201–264, 2023.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253, 2017.

Qingfeng Lan, Yangchen Pan, Jun Luo, and A. Rupam Mahmood. Memory-efficient reinforcement learning with value-based knowledge consolidation, 2023. URL `https://arxiv.org/abs/2205.10868`.

Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data, 2020. URL `https://arxiv.org/abs/2004.14990`.

Charline Le Lan, Marc G Bellemare, and Pablo Samuel Castro. Metrics and continuity in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8261–8269, 2021.

Te-Won Lee, B.-U. Koehler, and R. Orglmeister. Blind source separation of nonlinear mixing models. In *Neural Networks for Signal Processing VII. Proceedings of the 1997 IEEE Signal Processing Society Workshop*, pages 406–415, 1997. doi: 10.1109/NNSP.1997.622422.

Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Franois Goudou, and David Filliat. State representation learning for control: An overview. *Neural Networks*, 108:379–392, 2018.

Xiang Li, Jinghuan Shang, Srijan Das, and Michael S. Ryoo. Does self-supervised learning really improve reinforcement learning from pixels?, 2023. URL `https://arxiv.org/abs/2206.05266`.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018. doi: 10.1109/TPAMI.2017.2773081.

Zichuan Lin, Garrett Thomas, Guangwen Yang, and Tengyu Ma. Model-based adversarial meta-reinforcement learning, 2021. URL `https://arxiv.org/abs/2006.08875`.

Xu-Hui Liu, Zhenghai Xue, Jing-Cheng Pang, Shengyi Jiang, Feng Xu, and Yang Yu. Regret minimization experience replay in off-policy reinforcement learning, 2021. URL `https://arxiv.org/abs/2105.07253`.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations, 2019. URL `https://arxiv.org/abs/1811.12359`.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention, 2020. URL `https://arxiv.org/abs/2006.15055`.

Bogdan Mazoure, Remi Tachet des Combes, Thang Long Doan, Philip Bachman, and R Devon Hjelm. Deep reinforcement and infomax learning. *Advances in Neural Information Processing Systems*, 33:3686–3698, 2020.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020. URL `https://arxiv.org/abs/1802.03426`.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013. URL `https://arxiv.org/abs/1312.5602`.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Hiroshi Morioka and Aapo Hyvärinen. Connectivity-contrastive learning: Combining causal discovery and representation learning for multimodal data. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 3399–3426. PMLR, 25–27 Apr 2023. URL https://proceedings.mlr.press/v206/morioka23a.html.

Hiroshi Morioka, Hermanni Hälvä, and Aapo Hyvärinen. Independent innovation analysis for nonlinear vector autoregressive process, 2021. URL https://arxiv.org/abs/2006.10944.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning, 2019. URL https://arxiv.org/abs/1803.11347.

Sindhu Padakandla, Prabuchandran K. J., and Shalabh Bhatnagar. Reinforcement learning algorithm for non-stationary environments. *Applied Intelligence*, 50(11):3590–3606, June 2020. ISSN 1573-7497. doi: 10.1007/s10489-020-01758-5. URL http://dx.doi.org/10.1007/s10489-020-01758-5.

P. Pajunen and J. Karhunen. A maximum likelihood approach to nonlinear blind source separation. In *International Conference on Artificial Neural Networks (ICANN'97), Lausanne, Switzerland, October 8-10, 1997*, pages 541–546, 1997.

Petteri Pajunen. Nonlinear blind source separation by self-organizing maps. *ICONIP 96*, 2: 1207–1210, 1996.

Andrew Patterson, Samuel Neumann, Martha White, and Adam White. Empirical design in reinforcement learning, 2024. URL https://arxiv.org/abs/2304.01315.

Irving GB Petrazzini and Eric A Antonelo. Proximal policy optimization with continuous bounded action space via the beta distribution. In *2021 IEEE symposium series on computational intelligence (SSCI)*, pages 1–8. IEEE, 2021.

James Queeney, Ioannis Ch. Paschalidis, and Christos G. Cassandras. Generalized proximal policy optimization with sample reuse, 2021. URL https://arxiv.org/abs/2111.00072.

Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11157–11166, 2020.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience replay for continual learning, 2019. URL https://arxiv.org/abs/1811.11682.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation, 2016. URL https://arxiv.org/abs/1511.06295.

Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image, 2017. URL https://arxiv.org/abs/1611.04201.

Jurgen Schmidhuber. Evolutionary principles in self-referential learning. *On learning how to learn: The meta-meta-... hook.) Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*, 1 (2):48, 1987.

Jürgen Schmidhuber. Learning factorial codes by predictability minimization. *Neural computation*, 4(6):863–879, 1992.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL `https://arxiv.org/abs/1707.06347`.

Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron C Courville. Pretraining representations for data-efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12686–12699, 2021.

Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Towards causal representation learning, 2021. URL `https://arxiv.org/abs/2102.11107`.

Shagun Sodhani, Franziska Meier, Joelle Pineau, and Amy Zhang. Block contextual mdps for continual learning, 2021. URL `https://arxiv.org/abs/2110.06972`.

Aravind Srinivas, Michael Laskin, and Pieter Abbeel. CURL: contrastive unsupervised representations for reinforcement learning. *CoRR*, abs/2004.04136, 2020. URL `https://arxiv.org/abs/2004.04136`.

Christian Alexander Steinparz, Thomas Schmied, Fabian Paischer, Marius-Constantin Dinu, Vihang Prakash Patil, Angela Bitto-Nemling, Hamid Eghbal-zadeh, and Sepp Hochreiter. Reactive exploration to cope with non-stationarity in lifelong reinforcement learning. In *Conference on Lifelong Learning Agents*, pages 441–469. PMLR, 2022.

Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. The distracting control suite – a challenging benchmark for reinforcement learning from pixels. *arXiv preprint arXiv:2101.02722*, 2021.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite, 2018. URL `https://arxiv.org/abs/1801.00690`.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world, 2017. URL `https://arxiv.org/abs/1703.06907`.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments, 2024. URL `https://arxiv.org/abs/2407.17032`.

Aaron Van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019. URL https://arxiv.org/abs/1807.03748.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Mel Vecerik, Jean-Baptiste Regli, Oleg Sushkov, David Barker, Rugile Pevceviciute, Thomas Rothörl, Raia Hadsell, Lourdes Agapito, and Jonathan Scholz. S3k: Self-supervised semantic keypoints for robotic manipulation via multi-view consistency. In Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 449–460. PMLR, 16–18 Nov 2021. URL https://proceedings.mlr.press/v155/vecerik21a.html.

David Venuto, Elaine Lau, Doina Precup, and Ofir Nachum. Policy gradients incorporating the future, 2021. URL https://arxiv.org/abs/2108.02096.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn, 2017. URL https://arxiv.org/abs/1611.05763.

Marco Wiering and Martijn Van Otterlo. *Reinforcement Learning: State of the Art*. Springer, 2012. ISBN 978-3-642-27644-6. doi: 10.1007/978-3-642-27645-3.

Annie Xie, James Harrison, and Chelsea Finn. Deep reinforcement learning amidst lifelong non-stationarity, 2020. URL https://arxiv.org/abs/2006.10701.

Jinwei Xing, Takashi Nagata, Kexin Chen, Xinyun Zou, Emre Neftci, and Jeffrey L Krichmar. Domain adaptation in reinforcement learning via latent unified state representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10452–10459, 2021.

Zhongwen Xu, Hado van Hasselt, and David Silver. Meta-gradient reinforcement learning, 2018. URL https://arxiv.org/abs/1805.09801.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images, 2020. URL https://arxiv.org/abs/1910.01741.

Amy Zhang, Clare Lyle, Shagun Sodhani, Angelos Filos, Marta Kwiatkowska, Joelle Pineau, Yarin Gal, and Doina Precup. Invariant causal prediction for block MDPs. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11214–11224. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/zhang20t.html.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction, 2021. URL https://arxiv.org/abs/2006.10742.

Tiantian Zhang, Xueqian Wang, Bin Liang, and Bo Yuan. Catastrophic interference in reinforcement learning: A solution based on context division and knowledge distillation. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12):9925–9939, 2023. doi: 10.1109/TNNLS.2022.3162241.

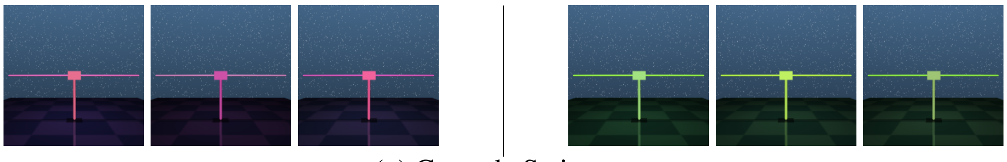
List of Figures

Appendix A

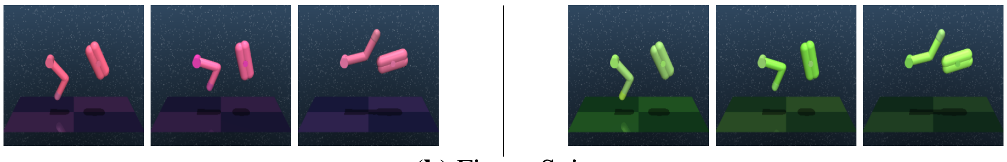
Environments

In this appendix, we visualize the environments used in our experiments.

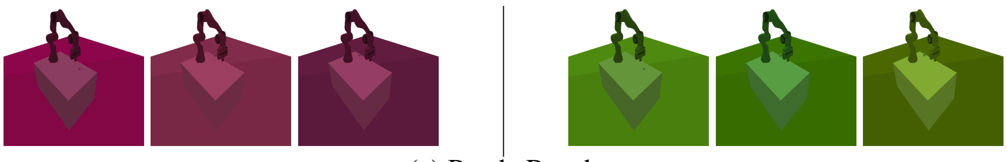
Environments with Task-Irrelevant Factors



(**a**) Cartpole Swingup



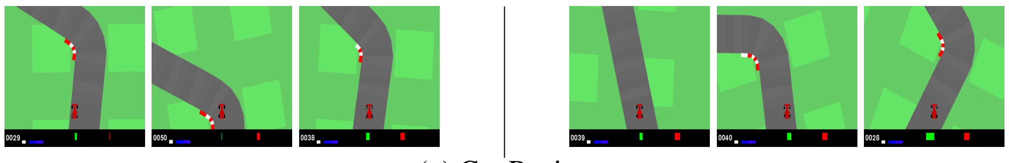
(**b**) Finger Spin



(**c**) Panda Reach

Figure A.1: Example frames from each environment with color distractors used in our experiments. The images on the left are from the train domain and the images on the right are from the test domain. All frames are shown before any image preprocessing was applied.

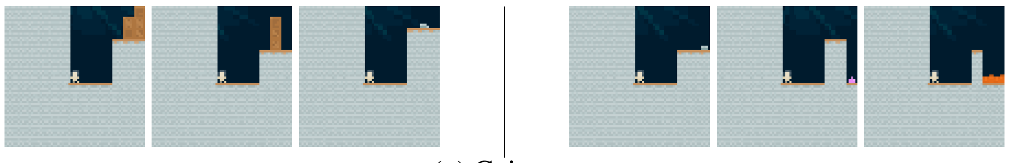
Environment with Task-Relevant Factors



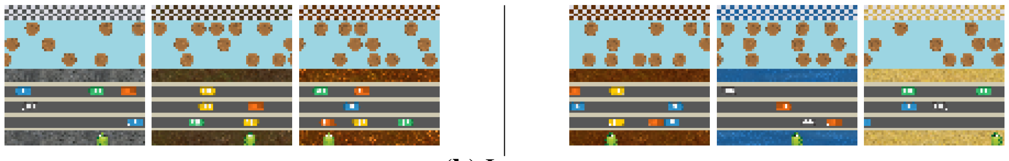
(**a**) Car Racing

Figure A.2: Example frames of the Car Racing environment used in our experiments. The images on the left are from the training levels and the images on the right are from the testing levels. All frames are shown before any image preprocessing was applied.

Environments with Task-Relevant and Task-Irrelevant Factors



(**a**) Coinrun



(**b**) Leaper

Figure A.3: Example frames for each Procgen environment used in our experiments. The images on the left are from the training levels and the images on the right are from the testing levels.

Appendix B

Hyperparameters

In this appendix we give an overview of the hyperparameters.

Experiments with Task-Irrelevant Factors

Environment	**Base algorithm**	**Value of α**
Cartpole Swingup	RAD	100
Finger Spin	RAD	25
Panda Reach	RAD	200

Table B.1: GED loss coefficient α across environments.

Hyperparameter name	**Value**
Replay buffer capacity	1000000
Initial steps	1000
Stacked frames	3
Action repeat	2 for Finger Spin, 4 otherwise
Batch size	128
Discount factor	0.99
Optimizer	Adam
Learning rate (actor, critic, encoder)	1e-3
Target soft-update rate η	0.01
Actor update frequency	2
Actor log stddev bounds	$[-10, 2]$
Latent representation dimension	50
Image size	(84, 84)
Image pad	4
Initial temperature	0.1

Table B.2: Hyperparameter settings shared for RAD-GED.

Experiments with Task-Relevant Factors

Hyperparameter name	Value
Image size	(96, 96)
Discount factor γ	0.99
GAE λ	0.95
# Timesteps per rollout	250
Epochs per rollout	10
# Minibatches per rollout	16
Entropy bonus	0.01
PPO clip range	0.2
Learning rate	3e-4
# Workers	1
# Environments per worker	1
LSTM?	No
Frame stack	4
Loss coefficient α	0.01

Table B.3: Hyperparameter values for PPO-GED.

Experiments with Task-Relevant and Task-Irrelevant Factors

Hyperparameter name	**Value**
Image size	(64, 64)
Discount factor γ	0.999
GAE λ	0.95
# Timesteps per rollout	250
Epochs per rollout	3
# Minibatches per rollout	8
Entropy bonus	0.01
PPO clip range	0.2
Learning rate	5e-4
# Workers	1
# Environments per worker	8
LSTM?	No
Frame stack?	No
Loss coefficient α	0.25 for coinrun, 0.25 for leaper

Table B.4: Hyperparameter values for PPO-GED.

Appendix C

Environment Settings

In this appendix we provide a detailed overview of the parameters of our experiments.

Distractor Settings in Experiments with Task-Irrelevant Factors

Distractor Settings	
Constant Color Distractor	
$\delta_{\text{min}}, \delta_{max}$	-0.1, 0.1
ε_t	0.00
Abruptly Changing Distractor	
$\delta_{\text{min}}, \delta_{max}$	-0.2, 0.2
l, h	-0.1 , 0.1
ε_t	0.03
L	40
Gradually Changing Distractor	
$\delta_{\text{min}}, \delta_{max}$	-0.2, 0.2
β	0.1
ν	0.251
ε_t	0.03

Table C.1: Color distractor settings using `distracting-control` library. $\delta_{min}, \delta_{max}$ refer to the maximum change in color channel compared to the original color value.

Non-Stationary Settings in Experiments with Task-Relevant Factors

Non-Stationary Settings	
Abrupt Non-Stationarity	
l, h	-0.5, 0.0
ε_t	0.05
L	50
Gradual Non-Stationarity	
β	$\frac{0.05t}{100}$
ν	0.02
ε_t	0.05

Table C.2: Non-stationary settings in Car Racing using the `ns-gym` library.