applied sciences

MDPI

Article

An Experimental Study on State Representation Extraction for Vision-Based Deep Reinforcement Learning

Junkai Ren *, Yujun Zeng *, Sihang Zhou and Yichuan Zhang

College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China; sihangjoe@gmail.com (S.Z.); zhangyichuan15@nudt.edu.cn (Y.Z.)
* Correspondence: jk.ren@ieee.org (J.R.); yujunzeng@sina.cn (Y.Z.);

Abstract: Scaling end-to-end learning to control robots with vision inputs is a challenging problem in the field of deep reinforcement learning (DRL). While achieving remarkable success in complex sequential tasks, vision-based DRL remains extremely data-inefficient, especially when dealing with high-dimensional pixels inputs. Many recent studies have tried to leverage state representation learning (SRL) to break through such a barrier. Some of them could even help the agent learn from pixels as efficiently as from states. Reproducing existing work, accurately judging the improvements offered by novel methods, and applying these approaches to new tasks are vital for sustaining this progress. However, the demands of these three aspects are seldom straightforward. Without significant criteria and tighter standardization of experimental reporting, it is difficult to determine whether improvements over the previous methods are meaningful. For this reason, we conducted ablation studies on hyperparameters, embedding network architecture, embedded dimension, regularization methods, sample quality and SRL methods to compare and analyze their effects on representation learning and reinforcement learning systematically. Three evaluation metrics are summarized, including five baseline algorithms (including both value-based and policy-based methods) and eight tasks are adopted to avoid the particularity of each experiment setting. We highlight the variability in reported methods and suggest guidelines to make future results in SRL more reproducible and stable based on a wide number of experimental analyses. We aim to spur discussion about how to assure continued progress in the field by minimizing wasted effort stemming from results that are non-reproducible and easily misinterpreted.

Keywords: deep reinforcement learning; representation learning; unsupervised learning; feature embedding; end-to-end learning

check for updates

Citation: Ren, J.; Zeng, Y.; Zhou, S.; Zhang, Y. An Experimental Study on State Representation Extraction for Vision-Based Deep Reinforcement Learning. *Appl. Sci.* **2021**, *11*, 10337. https://doi.org/10.3390/app112110337

Academic Editor: Byung-Gyu Kim

Received: 31 August 2021
Accepted: 13 October 2021
Published: 3 November 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Deep Reinforcement Learning is an emerging subfield of Reinforcement Learning (RL) that relies on deep neural networks as a function approximator, enabling RL algorithms in complex environments. Over the last few years, end-to-end DRL algorithms have become increasingly more stable and efficient. Notable application successes include learning to play a variety of video games from raw pixels [1], continuous control tasks such as controlling a simulated car from a dashboard camera [2], and subsequent algorithmic developments and applications to agents that successfully navigate mazes and solve complex tasks from first-person camera observations [3–5], and robots that successfully grasp objects in the real world [6].

However, despite the impressive ability, learning policy directly from raw images is found to be relatively unstable and data-inefficient (usually needs millions of samples) than operating on coordinate-state based features [7,8]. On the one hand, unlike in classic reinforcement learning where human-crafted representation is used, vision-based DRL has to learn features directly from raw observations, in addition to policy learning; on the other hand, most RL approaches assume a fully observable state space, i.e., fully

observable Markov Decision Processes (MDPs). However, this assumption is unworkable in real-world robotics due to factors including sensor sensitivity limitations and sensor noise and the lack of knowledge about whether the observation design is complete or not. Particularly in the case of learning directly from raw images, partial observability problem arises, which is defined as a partially observable MDP (POMDP), in which the agent can only sees a portion of the world state. Furthermore, there are numerous extrinsic factors (e.g., hyperparameters or codebases) and intrinsic elements (e.g., effects of random seeds or environment properties) that might affect the performance of DRL [9]. As a result, vision-based DRL typically suffers from slow learning speeds and frequently requires an excessive amount of training time and data to attain desired performance, making it unsuitable to real-world situations where data collection is difficlut and expensive.

For this reason, most of the present DRL algorithms are still limited to solving tasks with state spaces of relatively low dimensionality (less than 10 intrinsic dimensions for value-function based methods and less than 100 for policy gradient methods [10]). In addition, when the ground-truth state information cannot be easily accessed, it is inevitable to extract the embedded states from the original observations to learn desirable policies from DRL algorithms. To overcome this issue, state representation learning (SRL) [11] was proposed which aims at learning to extract the state information which can be used for the policy optimization.

As shown in Figure 1. This paper studies the influence factors (listed on the bottom left corner of this figure) of SRL, and their effects on different kinds of RL methods (listed on the bottom left corner of this figure). In the setting of SRL, instead of getting a low-dimensional coordinate state $s_t \in S$ at time t, the agent receives rather a high-dimensional observation $0_t \in O$, which is a rendering of potentially incomplete view of the corresponding state s_t of the environment [12]. In principle, the agent should be able to learn representations that can recover the task-related state information. Suppose the state information is covered by pixels. In that case, it should be possible to learn from pixels as efficient as from ground-truth states given the reasonable representation.

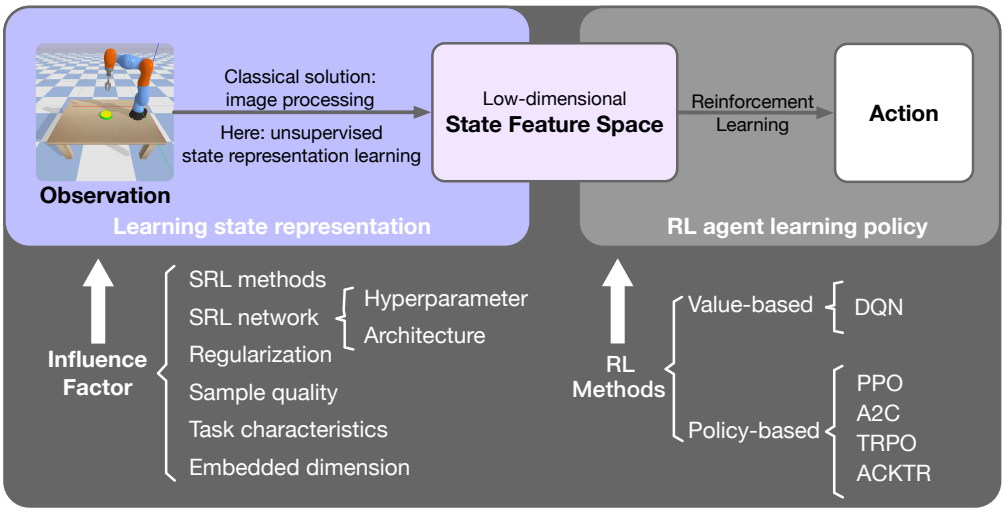


Figure 1. The unsupervised state representation learning (SRL) process and the corresponding influence factors from high-dimensional pixels for DRL.

There are many different SRL methods proposed for this objective, which have made remarkable progress, such as autoencoders, learning forward models, inverse dynamics and learning from robotic priors from the state characteristics [11]. Furthermore, with the tremendous development in label-efficient learning for image classification using contrastive unsupervised representations [13,14] and data augmentation [15,16], Laskin et al. [17,18] combined contrastive learning and data augmentation techniques from computer vision with model-free RL to demonstrate significant sample-efficiency gains on standard RL benchmarks. Besides, transfer learning has improved vision-based RL

performance between different tasks further. PAD [19] performs unsupervised policy adaption with a test-time auxiliary task (e.g., inverse dynamic prediction) to fine-tune the visual encoder of the policy network. By means of unsupervised keypoint detection [20], a clean foreground mask could be reconstructed from noise-augmented observations and keep the RL policy learning between different tasks stable and fast [21].

So far, however, there has been little discussion about the influence factors (as shown in Figure 1) and evaluation metrics of state representation learning. Most previous studies in this field only focus on improving the effectiveness of SRL by means of designing new mechanisms and auxiliary tasks. The evaluation metric is also limited to RL's performance, which means the RL agent's learning process is the only criteria to evaluate the SRL approaches in most cases. The diversity in methods and applications makes this field lack statistically relevant results to analyze the influence factors of the SRL methods, which creates the potential for misleading reporting of results and the demands for tighter standardization of experimental reporting. More importantly, reproducing existing work and accurately judging the improvements offered by novel methods is vital to sustaining the progress of SRL.

Therefore, this paper is set out to analyze the effects of SRL influence factors on vision-based DRL (Section 2). We summarized the criteria that could be used for evaluating SRL performance and compared the SRL performance on different kinds of RL algorithms (including value-based and policy-based methods):

- Firstly, in order to give more statistically representative experimental results and quantitative analysis, we adopted 8 different tasks, including mobile robot navigation and robot arm manipulation environments (Section 3). The complexity of the tasks and the observations increases gradually;
- Secondly, three evaluation criteria, including the quantitative evaluation with reinforcement learning performance, are summarized to evaluate the SRL performance based on the extensive investigation of the related works in Section 4.
- Thirdly, a large number of contrastive experiments were conducted using the control variates approach to investigate SRL influence factors such as hyperparameter, embedding network design, embedded dimension, regularization methods, and sample quality (Figure 1). In each part of the experimental analysis, we posed questions about major factors affecting final performance. We discovered that state embedded dimension and sample quality are crucial in the SRL model, and carefully adjusting the other illustrated factors can assist improve the SRL model's representative property to a some extent (Section 5).
- Finally, we investigated the effects of these variables in reported results through a representative set of RL algorithms. Both value-based and policy-based methods with neural network function approximators, such as DQN, A2C, ACKTR, PPO, TRPO are chosen for the RL experiments (Section 5.7).

The experimental results are discussed in Section 6. Based on our experiments, we conclude with possible recommendations and points of discussion for future works to ensure that SRL continues to matter (Section 7). To the best of our knowledge, the experimental work presented in this paper provides one of the first investigations into adjusting the influence factors and designing efficient SRL models.

2. Problem Formulation

In this section, we will introduce the preliminary knowledge of RL and SRL methods which are integrated in our framework. As shown in Figure 2, when we describe the network used in DRL, it could be separated into two parts for the simplicity of explanation: the first part is used for representation learning, while the second part takes the learned representation as input and learns the value function and the target policy. State representation is learnt to improve two kinds of objectives: performance on the RL task and the auxiliary task.

More details on the auxiliary tasks are given in Sections 2.2.2 and 5.6. There are many kinds of auxiliary tasks that can be added to the SRL scheme, such as image reconstruction, contrastive learning and model prediction. More details on the auxiliary tasks are given in Section 2.2.2. The dash green color means some SRL methods such as CURL can backpropagate the gradient from the RL loss function to the SRL network. The notations adopted in this paper and their definitions are list in Table 1.

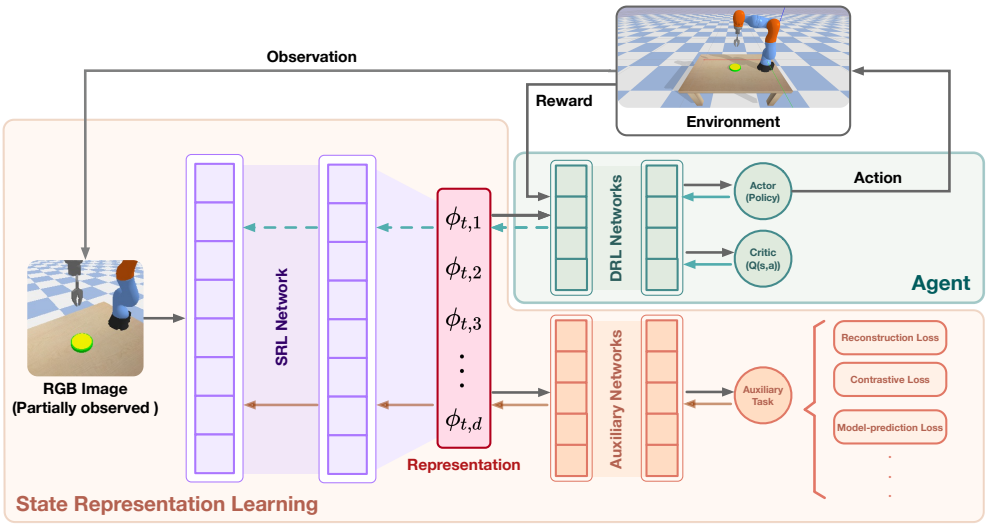


Figure 2. The unsupervised SRL scheme for DRL. The SRL network learns a mapping from original input observations to the embedded features.

Table 1. Summary of the notations in this paper.

Symbol	Definition
π	policy
r_t	reward at time t
o_t	observation at time t
$\mathcal{O}$	observation space
s_t	ground-truth state at time t
$\mathcal{S}$	ground-truth state space
$\phi(s)$	feature vector representing state s
Φ	state representation space
$V_\pi(s)$	value of state s under policy π
$Q_\pi(s,a)$	value of taking action a in state s under policy π
J	The objective function (expected total reward)
G_t	return (accumulated rewards) following time t
θ	parameter vector of policy π_θ

2.1. Reinforcement Learning

2.1.1. Preliminary Knowledge

In RL, an agent learns an optimized policy through interacting with the environment. This process can be formulated as Markov Decision Processes (MDPs). An MDP can be defined as a 4-tuple (S, A, T, R), where S represents the state space, A denotes the action space, transition probability $T : S \times A \mapsto S$ denotes the state transition probability, and R represents the reward function. Specifically, at time step t, the agent faces a state $s_t \in S$, and it follows the policy $\pi(a_t \mid s_t) : s_t \mapsto a_t$ to choose an action $a_t \in A$. Then the agent receives a reward r_t, which could evaluate the performed action. After that, the agent reaches the next state s_{t+1}. The aforementioned process will break when the terminal condition is satisfied.

The return $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ is the total accumulated rewards at state s_t with discount factor $\gamma \in (0, 1]$, which is a hyperparemeter to adjust the impact of actions in the future.

The goal of RL algorithms is to maximize the expected return R_t by updating the policy π. The value function of state s_t under policy π is usually defined as:

$$V^{\pi}(s_t) = \mathbb{E}[G_t|s_t = s]. \tag{1}$$

Similarly, the value of selected action a_t in state s_t under policy π is defined as:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}[G_t|s_t = s, a], \tag{2}$$

which is called action-value function. The advantage function estimates the average advantage value of selected action a_t in state s_t under policy π and it is defined as:

$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t). \tag{3}$$

2.1.2. Value-Based and Policy-Based RL Methods

There are two important categories of RL algorithms: value-based RL methods and policy-based RL methods. Policy-based RL methods explicitly build a representation of the policy $\pi(a_t \mid s_t; \theta)$ and update the parameters θ. In contrast to policy-based RL methods, value-based RL methods don't build any explicit policy, but a value function $V(s; \theta)$ or an action-value function $Q(s, a; \theta)$ with parameters θ. The policy is implicit and can be derived directly from the value function (pick the action with the best value). Actor-critic methods mix policy-based methods and value-based methods, which often achieve better results than both policy-based methods and value-based methods.

In actor-critic methods, the actor is a parameterized policy that determines how actions are selected. Policy gradient methods are usually employed to learn the parameters. We consider a stochastic policy gradient with respect to the policy parameters θ. The following equation gives the updating rule for θ:

$$\theta = \theta + \alpha \nabla J(\theta), \tag{4}$$

where $J(\theta)$ is the expected total reward and $\nabla J(\theta)$ can be calculated by:

$$\nabla J(\theta) = \mathbb{E}_t[\nabla_\theta \log \pi_\theta(a_t|s_t) A(s_t, a_t)] \tag{5}$$

The advantage function captures how better an action is than the others at a given state to evaluate the current policy. It is found that using the advantage function has a better performance than other metrics like the Q values [22].

With sufficient samples, the expectation of $A(s_t, a_t)$ can be evaluated accurately. Unfortunately, in many real-world tasks, it is impractical to collect such a great amount of samples, leading to the high variance of $A(s_t, a_t)$. Therefore, it is necessary to use the value-based critic to estimate the $A(s_t, a_t)$. If two critic networks are employed to estimate the value function $Q^{\pi}(s_t, a_t)$ and $V^{\pi}(s_t)$ respectively, it will lead to more bias. Thus, Time-difference method is often used. In fact, $Q^{\pi}(s_t, a_t)$ and $V^{\pi}(s_t)$ satisfy the following equation:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}[r_t + V^{\pi}(s_{t+1})] \tag{6}$$

Though r_t is a random variable, the variance of r_t is much smaller than that of $A^{\pi}(s_t, a_t)$ when $A^{\pi}(s_t, a_t)$ is estimated directly. So the Equation (6) can be rewritten as:

$$Q^{\pi}(s_t, a_t) \approx r_t + V^{\pi}(s_{t+1}) \tag{7}$$

Therefore, it is possible to use only one critic network to estimate $A(s_t, a_t)$:

$$A^{\pi}(s_t, a_t) = r_t + V^{\pi}(s_{t+1}) - V^{\pi}(s_t) \tag{8}$$

2.2. State Representation Learning

2.2.1. Preliminary Knowledge

State representation learning (SRL) aims at learning compact representations from raw observations without explicit supervision. Many different SRL approaches have been proposed [11], which plays an important role in deep reinforcement learning. SRL is utilized to learn a transformation Π from the observation space $\mathcal{O}$ to the latent state space Φ. Then, a policy π, which takes state representation $\phi \in \Phi$ as input and outputs action $a \in \mathcal{A}$, is learned to solve the task:

$$\mathcal{O} \xrightarrow[SRL]{\Pi} \Phi \xrightarrow[RL]{\pi} \mathcal{A} \tag{9}$$

The state representation contains information extracted from states—the description of the current situation given by the environment. Therefore, a high-quality representation should only keep the valuable information for the task and reduce the search space, thus contributing to address two main challenges of RL: instability and sample inefficiency. It is not only essential to build a robust reinforcement learning agent but also can help improving learning efficiency. Moreover, a state representation learned for a particular task may be transferred to related tasks, speeding up learning in multiple tasks' settings.

Many of the previous works in representation learning for reinforcement learning focused on designing fixed-basis architectures to achieve desirable properties, such as orthogonality. Most of these properties are common to the general machine learning setting. Many approaches either use or search for orthogonal or decorrelated features, such as orthogonal matching pursuit [23], Bellman-error basis functions [24], Fourier basis [25] and tile coding [26,27]. Prototypical input matching methods have been extensively explored, as in kernel methods [28], radial basis functions [26].

In contrast, the idea behind DRL is that the agent designer should not encode representational properties, but rather that the data stream should determine the properties of the representation—excellent representations will arise via appropriate training schemes, where a good representation is defined by success on some task. Recent developments in representation learning explore a different perspective: we should let the training data dictate the properties of the representation through gradient descent. This view is widely held and is reflected in specifying training regimes, including multi-task training [29], auxiliary loss constructing (i.e., autoencoding, next observation prediction and pixel control) [3,30], and training on distribution of problems like meta-learning [31–33].

2.2.2. Auxiliary Tasks

Auxiliary tasks can be added to alter the complexity and the structure of the problem settings. By means of dividing the last layer of the neural network into several heads, it is possible to give various tasks to separate heads and then solve them collaboratively. The purpose of including auxiliary tasks into SRL is to help the learner develop a more accurate representation of the proeblem at hand. Several studies have shown that certain auxiliary tasks linked to the primary task are beneficial, such as learning an additional strategy to alter pixels in the observation maximally or maximally activate each bit in the representation [3].

Auxiliary tasks such as predicting the future conditioned on the past observations and actions [3,34–36] are a few representative examples of using auxiliary tasks to improve the sample-efficiency of model-free RL algorithms. The future prediction is either done in a pixel space [3] or latent space [35]. The sample-efficiency gains from reconstruction-based auxiliary losses have been benchmarked in [3,29,37]. Recently, contrastive learning has been used to extract reward signals in the latent space [38–40]; and study representation learning on Atari games [41].

In the next parts, we present approaches of SRL that are implemented in this paper. Each technique is not mutually exclusive and may be used in conjunction with others to generate new models.

2.2.3. Forward Model and Inverse Model

One critical element to encode for RL is the controlled agent's state. It is equivalent to the robot position in the context of goal-based robotics activities. A straightforward approach is to use the forward model and inverse model.

A forward model that predicts state s_{t+1} given state s_t and action a_t, can be interpreted as learning dynamics of the world. Constraints may be introduced to the state representation by limiting the forward model, for example, by requiring the system to follow linear dynamics [42].

Another approach is to learn an inverse model [43], which predicts the taken action a_t given two successive states s_t and s_{t+1}. This requires embedded states to encode information about the dynamics in order to retrieve the action a_t, which can make such a transition. However, the features extracted by an inverse model are not always sufficient: in our goal-based experiments, they cannot embed the target's position because the agent is unable to act on it. For this reason, additional objective functions are required to encode the target object's position. Two of them are discussed in the new two sections: minimizing image reconstruction errors (autoencoder model) and minimizing reward prediction errors (reward prediction model).

2.2.4. Autoencoder

One intuitive step to get a state representation is to condense the observation into a low-dimensional embedded state that can be reconstructed. Because this method excludes potential actions, it does not make use of the robotic environment. We compared three kinds of autoencoders in this paper, including original Autoencoders [44], Variational Autoencoders (VAE) [45]—autoencoders that enforce the latent variables to follow a given distribution, and Denoising Autoencoder (DAE) [46] which could enhance the flexibility of data stream method in exploiting unlabeled samples.

Autoencoders tend to encode only aspects of the environment that are salient in the input image. This means they are not task-specific: relevant elements can be ignored, and distractors (unnecessary information) can be encoded into the state representation. They usually demand more dimensions in order to encode a scene (e.g., in our experiments, it requires more than 10 dimensions to encode a 2D position of the mobile robot correctly).

2.2.5. Reward Prediction

The objective of a reward prediction model [34] leads to state representations that are specially appointed by the goal of the task. This, however, does not imply that the state space must be disentangled or have any certain structure.

2.2.6. Robotic Prior Knowledge

Robotic prior knowledge about the dynamics or physics of the world may be used to construct an appropriate representation of states. This kind of knowledge may account for temporal continuity or causality principles that reflect an agent's interactions with its surroundings [47,48].

2.3. Combination SRL Model and Splits SRL Model

There are generally two types of SRL model for learning state representations: the combination srl model and the splits srl model, which will be introduced briefly:

2.3.1. Combination SRL Model

In SRL Combination model, the combination of different objective functions is done by averaging the corresponding extracted features on a single embedding.

Combining objectives makes it possible to share the strengths of each model. In our application example, the previous sections suggest that we should mix objectives to encode both robot and target positions. The simplest way to combine objectives is to minimize a

weighted sum of the different loss functions, i.e., reconstruction loss $\mathcal{L}_{\text{reconstruction}}$, inverse dynamics prediction loss $\mathcal{L}_{\text{inverse}}$ and reward prediction losses $\mathcal{L}_{\text{reward}}$:

$$\begin{aligned}\mathcal{L}_{\text{combination}} =& w_{\text{reconstruction}} \cdot \mathcal{L}_{\text{reconstruction}} + w_{\text{inverse}} \cdot \mathcal{L}_{\text{inverse}} \\ &+ w_{\text{reward}} \cdot \mathcal{L}_{\text{reward}}\end{aligned} \tag{10}$$

Each weight reflects the proportional significance we place on certain objectives. We selected the weights such that the gradients are comparably important.

2.3.2. Splits SRL Model

Combining objectives into a single embedding is not the only option to have features that are sufficient to solve the tasks. Stacking representations, which also favors disentanglement, is another way of solving the problem. In SRL Splits model, the model described that combines different objective functions using splits of the state representation. The intrinsic idea of the SRL Splits model is that the state representations are divided into several parts, where each part optimizes a fraction of the objectives.

This stacked representation learning model prevents objectives that can be opposed from cancelling out and allows a more stable optimization. This process is similar to training several models but with a shared feature extractor, that projects the observations into the state representation.

3. Environment Description

In this section, we will describe the task environments with incremental difficulty. The mobile robot navigation environment and the Kuka robot arm environment with OpenAI Gym [49] interface are adopted, making integration with DRL algorithms easy.

To assure fairness, we execute five experiment trials for each evaluation, each with a distinct random seed (all experiments use the same set of random seeds). In all cases, we highlight important results, with complete descriptions of experimental setups, additional learning curves and statistical tables included in the supplemental materials. Unless otherwise mentioned, we use default settings whenever possible.

As shown in Figure 3, the robotic tasks proposed in this paper are variations of two environments: a 2D environment with a mobile robot and a 3D environment with a robotic arm. Each environment may have a continuous or discrete action space, with sparse or shaped rewards, enabling us to cover a wide range of scenarios. The ground-truth state is defined in each scenario: the absolute robot position in static scenarios or the relative position in moving goal scenarios. The tasks have incremental difficulty: the minimal number of variables for describing each task (minimal state dimension for solving the task with RL) is increasing from 2 (mobile robot 1D with random target) to 5 (robotic arm with random target). The detail description for each task will be introduced in Section 3.

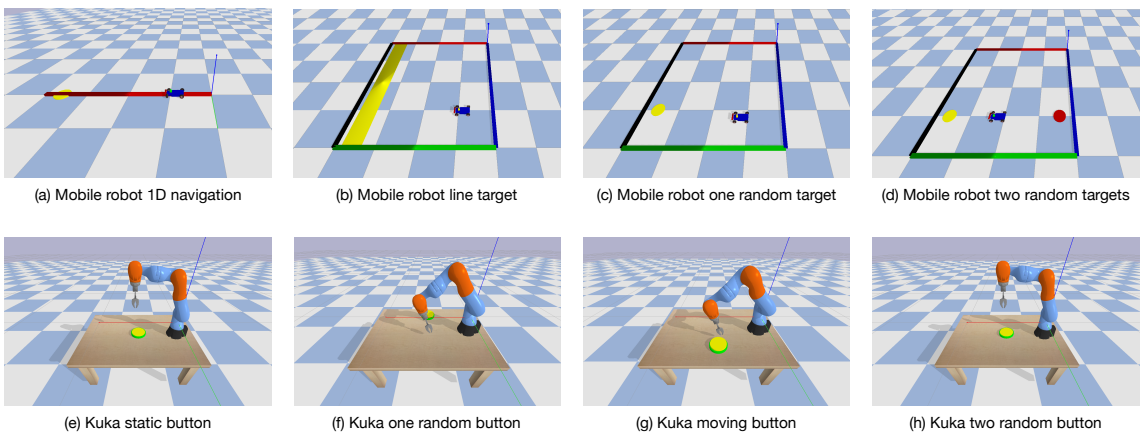


Figure 3. The Kuka robot arm tasks and mobile robot tasks adopted in this paper.

3.1. Mobile Robot Navigation Environment

This setting simulates a navigation task using a small car resembling the task of [47], with either a cylinder or a horizontal band on the ground as a goal, which can be fixed or moving from episode to episode.

As it shows in Figure 3, in the mobile robot 1D navigation task, the robot moves along a straight line, and the target is a yellow cylinder; In the other three tasks of this environment, the robot moves on a 2D plane. The target in the mobile robot line-target task is a yellow band. The targets in the mobile robot 1-target task and 2-target task are yellow cylinder and yellow/red cylinders, respectively.

The action space of the mobile robot in 1D navigation task and 2D navigation tasks has 2 dimensions (left, right) and 4 dimensions (forward, backward, left, right), respectively (Table 2). The ground truth states in the 1D task and 2D task are the absolute coordinates of the robot and the target in static scenarios or the relative position in moving target scenarios. The reward function is defined as follows:

$$r = \begin{cases} +1 & \textit{when the car reaches the target} \\ -1 & \textit{when the car hits the wall} \\ 0 & \textit{otherwise} \end{cases} \tag{11}$$

Table 2. The property descriptions of two simulation environments .

Dataset	Reward	Action Space
Mobile Robot	Sparse	4 (2 *), Discrete (left, right, forward *, backward *)
Robotic Arm	Sparse	5, Discrete, (forward, backward, left, right, down)

* is for mobile robot 1D target task.

3.2. Kuka Robot Arm Environment

This setting simulates a Kuka robotic arm fixed on a table, with the task of pushing a button that may move or not in between episodes. The arm can be controlled either in the x, y and z position using inverse kinematics, or directly controlling the joints.The action space of the Kuka robot arm has 5 dimensions (Table 2). The ground truth state is the absolute coordinates of the robot and the target(s) in static scenarios or the relative position in moving target scenarios. The episode will terminate when the arm hits the table. The reward function is defined as follows:

$$r = \begin{cases} +1 & \textit{when the arm pushes the button} \\ -1 & \textit{when the arm hits the table} \\ 0 & \textit{otherwise} \end{cases} \tag{12}$$

3.3. State Representation Learning Pipeline

The complete pipeline consisted of (1) training the state representation networks with different settings; (2) RL performance evaluation on tasks with fixed representations; (3) SRL influence factors' property evaluation. We save the learned representation by saving parameters ϕ, and then evaluate these fixed representations in terms of properties for learning performance in SRL and RL.

In each scenario, RGB images from a fixed position are defined as observations for state representation learning. Note that apart from providing all described environments, mobile robot 2D navigation datasets use 4 discrete actions (right, left, forward, backward); robot arms use one more action (down).

4. Evaluation Criteria

In this section, we summarized three evaluation criteria to analyze the representation learning influence factors and compare their effects on the final performance of reinforce-

ment learning. The first two kinds of criteria come from the field of pattern recognition. They do not rely on specific RL algorithms or the policy training process, but only focus on the intrinsic structures of the learned representations and their relationship with the ground-truth states. The final criterion is the RL performance applied to the task, which should be the most convincing evaluation criterion of the extracted state representations. However, this approach is relatively computational expensive because it can only be obtained after training the SRL network and RL network.

4.1. K-Nearest Neighbors Evaluation Criterion (KNN)

The K-Nearest Neighbors (KNN) Evaluation Criterion is a criterion for evaluating embedded representations' quality based on the Nearest-Neighbours method [50], it can help us get the nearest neighbors of an image in the latent representation space because a good representation should exhibit local coherence, which implies that the ground truth states associated with it should be near. Lesort et al. [51] derives a more typical metric named $KNN\text{-}MSE$ on the basis of KNN. A low $KNN\text{-}MSE$ value indicates that a neighbor in the ground truth state space is still a neighbor in the learned representation space, preserving local coherence.

For a given state observation o, we begin by finding its associated state's $\phi(s)$ closet neighbors $\phi(s)'$ in the learned state space Φ by means of KNN. Then we project them into the ground-truth state space $\mathcal{S}$. After that, in the latter space $\mathcal{S}$, we compute the average distance to its neighbors:

$$KNN\text{-}MSE\ (\phi(s)) = \frac{1}{k} \sum_{\phi(s)' \in KNN(\phi(s),k)} \left\| s - s' \right\|^2, \tag{13}$$

where $KNN\text{-}MSE\ (\phi(s), k)$ returns the k nearest neighbors $\phi(s)'$ of $\phi(s)$ in the learned state space Φ, s is the ground-truth state associated to $\phi(s)$, and s' is the one associated to $\phi(s)'$.

4.2. Ground Truth Correlation (GTC)

A Pearson correlation coefficient's matrix is computed for each dimension pair $(s, \phi(s))$:

$$\rho_{\phi(s),s} = \frac{\mathbb{E}\left[\left(\phi(s) - \mu_{\phi(s)}\right)(s - \mu_s)\right]}{\sigma_{\phi(s)}\sigma_s} \tag{14}$$

where s is the ground truth state, $\phi(s)$ is the learned state. μ_s and σ_s are the mean and standard deviation of state s respectively.

We can visualize the correlation matrix to quantitatively assess the ability of a model to encode relevant information in the states learned. However, the visualization idea is impractical when meeting high-dimensional spaces such as those in our experiments. As a result, Raffin et al. proposed another two metrics: Ground Truth Correlation (GTC) and the mean of that—GTC_{mean} [52]. They can be used to assess a model's capacity to encode important information, measuring the similarity between the state representation $\phi(s)$ and the ground-truth state s. The only difference between GTC and GTC_{mean} is that that GTC_{mean} just need one scalar value to evaluate the learned state. More specifically, for each component s_i of s, GTC_i gives the maximum absolute correlation value between s_i and any component of the predicted states $\phi(s)$:

$$GTC_i = \max_j \left| \rho_{\phi(s),s}(i,j) \right| \in [0,1] \tag{15}$$

$$GTC_{\text{mean}} = \mathbb{E}[GTC] \tag{16}$$

with $i \in [0, |s|]$, $j \in [0, |\phi(s)|]$, $s = [s_1; \ldots; s_n]$, and s_k being the k^{th} dimension of the ground truth state vector. Taking the Mobile Robot 2D navigation with one random target task as an example, the the ground-truth state have a dimension of 4 ($|s| = 4$), which is made up of the two-dimensional robot position and the two-dimensional target position.

4.3. Quantitative Evaluation with Reinforcement Learning

The reinforcement learning performance in the representation space is the most essential criterion for evaluating SRL approaches. In the experiments of this paper, five DRL algorithms (DQN, A2C, ACKTR, PPO, TRPO) including both value-based and policy-based methods are implemented and compared. The detail training configurations for each algorithm can be found in Appendix A.

5. Experimental Analysis

In this section, we pose several questions about the influence factors of SRL. Then we performed a set of experiments designed to provide insights into these questions. In particular, we investigate the influence of hyperparameters, embedding network architecture, embedded state dimension, sample quality, regularization approaches and SRL methods. In most of these experiments, PPO was chosen as the baseline RL algorithm because it works well across environments without much hyperparameter tuning.

To evaluate the properties of the embedded state representations, in the experiments except that in Section 5.4, we collect pixel data offline by doing a 50 episodes exploration, which generate more than 20,000-step transitions. The datasets that are composed of state observations (224×224@3 pixels) are collected from a random policy. In Section 5.4, the random policy was substituted by pre-trained PPO policies to analyze the influence of sample quality.

Each state representation is learned using 20,000 samples. This dataset is separated into the training set (80%) and validation set (20%). We kept for each method the model with the lowest validation loss during the 30 training epochs. We used the same network architecture from [52] for the most of models.

5.1. Hyperparameters

What is the magnitude of the influence that the hyperparameter settings can have on baseline performance?

Hyperparameters tuning plays an essential role in eliciting the best results from many algorithms. The choice of hyperparameters is crucial for reproducible end-to-end training [17]. However, the optimal hyperparameter configuration is often not consistent in related literature, and the range of values considered is often not reported. Sometimes, a high level of stochasticity appears due to random seeds [9]. It is dificult to assess the performance of high-level algorithmic ideas without understanding lower-level choices, as performance can be heavily influenced by hyperparameters tuning and implementation-level details, making it hard to attribute progress in representation learning and deep reinforcement learning. Thus, it slows down further research [53–55]. Furthermore, poor hyperparameter selection can be detrimental to a fair comparison against baseline algorithms.

In two mobile robot navigation challenges, we investigate three sets of hyperparameter choices for the kernel number in each convolutional layer and the corresponding pooling layer of the classic autoencoder neural network. The specific settings are shown in Figure 4, the total number of neural network hyperparameters in these three configurations are 1.8k+, 87.9k and 789.0k, respectively.

Results: The evaluation criteria such as GTC, GTC_{mean}, $KNN\text{-}MSE$ and average total reward can been seen in Table 3. The related learning performance of DQN and PPO methods is shown in Figure 5. The experiment results show that the effects of neural network hyperparameters are not consistent across algorithms or environments. However, the neural network hyperparameters do significantly affect learning performance, especially in the early stage of training. The performance of learnt latent feature representations is also related to the selected DRL algorithm. Usually, neural networks with more hyperparameters and deeper layers can show better function approximation and feature extraction ability. However, the vanishing gradients problem, exploding gradients problem and overfitting problem are easier to be encountered [56].

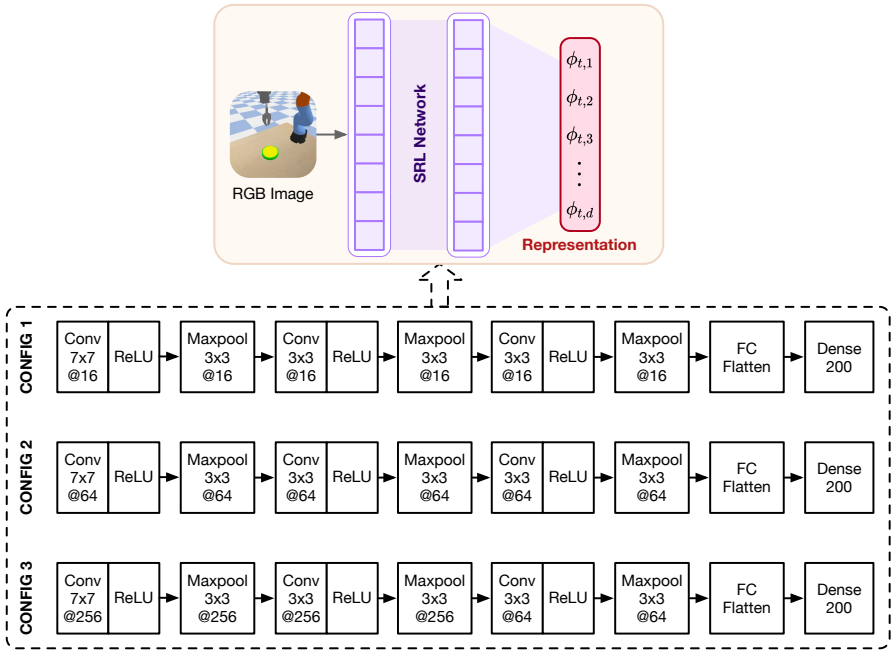
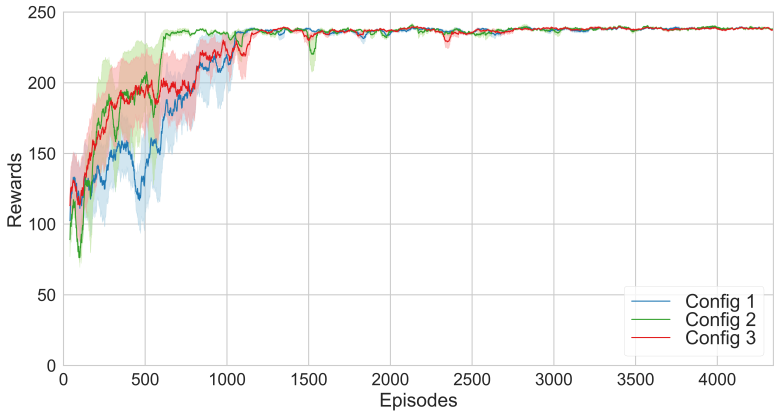
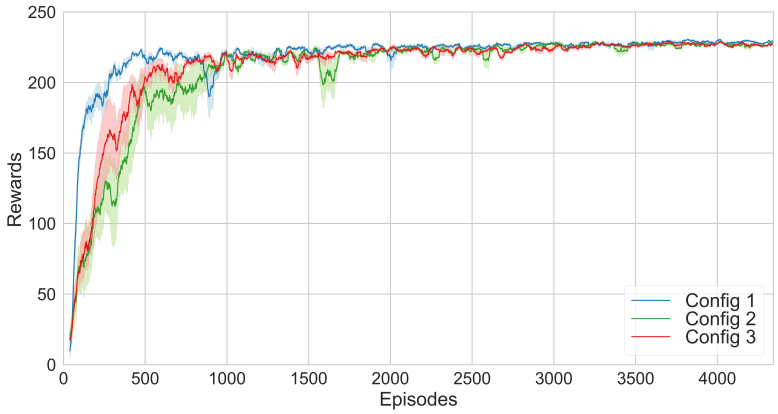


Figure 4. Three sets of hyperparameter configurations of the encoding network with increasing trainable parameters. The input image is encoded using the pre-trained encoder to obtain the latent representation, which the RL agent then uses to output the actions.



(a) RL performance based on DQN algorithm..



(b) RL performance based on PPO algorithm.

Figure 5. The influence of the SRL network configurations in mobile robot 2D navigation task.

Table 3. The influence of the network configuration in mobile robot navigation task.

Task	Network Configuration	Correlation					*KNN-MSE*	Average Total Reward
		Max Correlation Matrix				Mean		PPO (1e6 Timesteps)
1D-Nav.	Config 1 (1.8 k*)	0.7821	2.9816e-15	None	None	0.3910	0	238.41 ± 3.72
	Config 2 (87.9 k)	0.8563	3.2195e-15	None	None	0.4282	0	238.35 ± 4.68
	Config 3 (789 k)	0.5612	2.3240e-15	None	None	0.2806	0	238.21 ± 3.76
2D-Nav.	Config 1 (1.8 k*)	0.4797	0.5046	0.9999	0.9999	0.7461	0.00063	229.04 ± 4.90
	Config 2 (87.9 k)	0.3303	0.4587	0.9999	0.9999	0.6973	0.00066	227.30 ± 5.38
	Config 3 (789 k)	0.2176	0.3834	0.9999	0.9999	0.6503	0.00064	227.05 ± 6.09

5.2. Network Architecture

How can the choice of network architecture for the SRL encoders affect results?

In this section, we utilized three different autoencoder methods to compare the influence of encoding network architecture, including original autoencoder, variational autoencoder (VAE) [45] and denoising autoencoder (DAE) [46]. Traditional autoencoder accepts and compresses input, then recreates the original input. This is an unsupervised technique because all you need is the original data, without any labels of known, correct results; A variational autoencoder assumes that the source data has some underlying probability distribution (such as Gaussian) and then attempts to find the parameters of the distribution; Denoising autoencoders are an example of deep architectures that are designed to recover noisy data, trying to achieve a good representation by changing the reconstruction criterion.

Results: Figure 6 shows how significantly learning performance can be affected by simple changes to the embedding network architecture. The autoencoder has mixed results: it solves all environments, yet it sometimes under-performs navigation tasks. When we explored the latent space using the S-RL Toolbox visualization tools [52], we noticed that one state space dimension can act on both robot and target positions in the reconstructed image. In addition, this approach does not use additional information that the environment provides, such as actions and rewards, leading to a latent space that may lack informative structure.

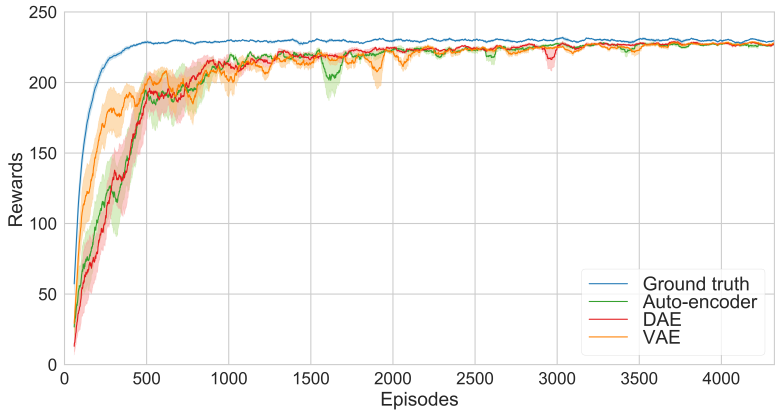


Figure 6. Performance (mean and standard error) on mobile robot 2D navigation task with different autoencoder methods using PPO algorithm.

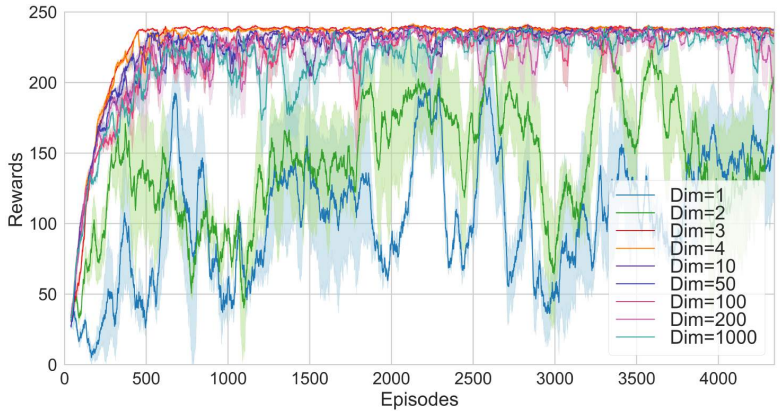
5.3. Embedded State Dimension

How to choose the embedded dimension of the SRL model?

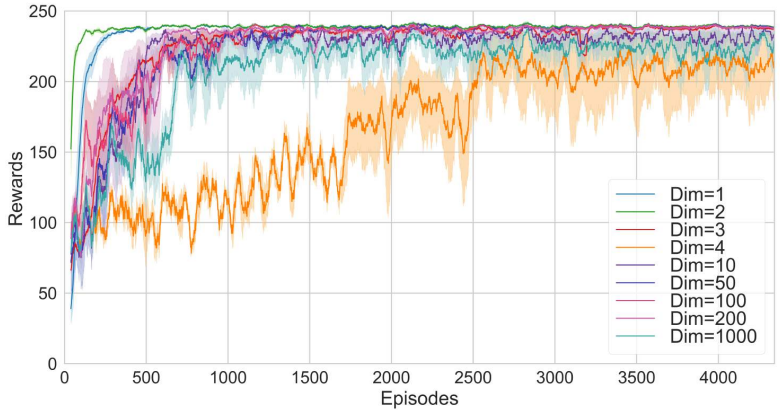
To assess how the choice of embedded dimension can affect the RL algorithm performance, we use our aforementioned default set of hyperparameters except of the embedded dimension across an extended suite of discrete robotic tasks and study how well the RL algorithms perform across the different embedded states.

The following two tasks with OpenAI Gym interface are utilized for the experiments: mobile robot 1D navigation task and mobile robot 2D navigation with random-target task. The dimensions of the ground-truth state space (i.e., robot and target positions) in these two tasks are 2 and 4 separately. The dimensions of the action space are 2 and 4, respectively. The complexity of the image observations also increases, which makes the state representation learning process gradually difficult.

Results: Table 4 shows the GTC, GTC_{mean}, $KNN\text{-}MSE$ and associate mean reward performance of RL methods (DQN and PPO) per episode (average on 100 episodes) after $1e6$ steps for the mobile robot navigation tasks. When the environment and the task are simple, as shown in Figures 7 and 8, a small state dimension could help the RL agent learn faster generally; however, when the agent interacts with a complicated environment in which the task is relatively difficult, the state dimension for the SRL model must be large enough to solve the task efficiently. Increasing the state dimension above a certain threshold, on the other hand, has no effect (positively or negatively) on RL performance.

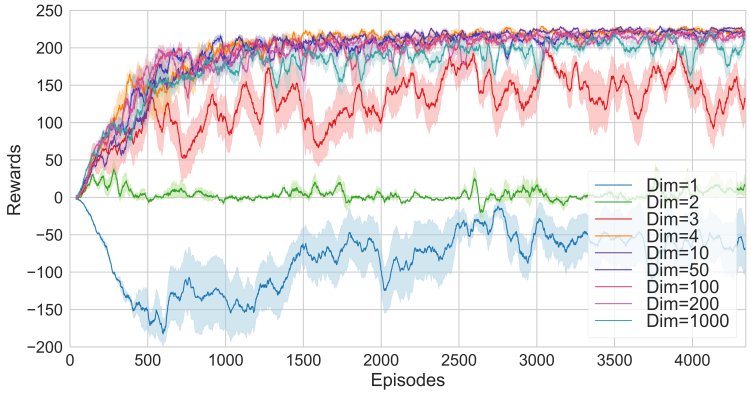


(a) RL performance based on DQN algorithm.

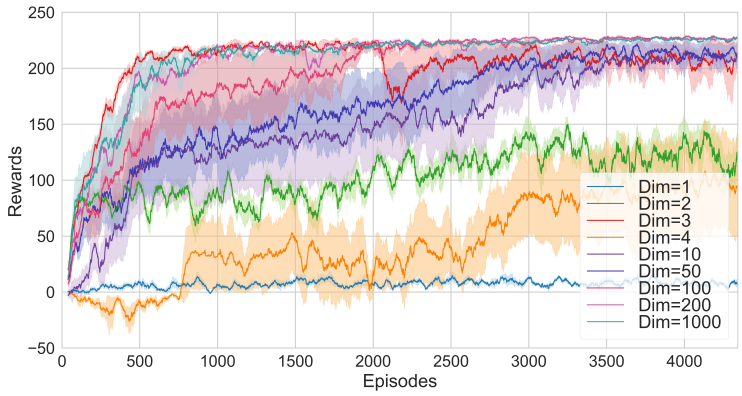


(b) RL performance based on PPO algorithm.

Figure 7. Performance (mean and standard error for 5 runs) of DQN and PPO algorithms with state representations learned from different embedded dimension in mobile robot 1D navigation environment.



(a) RL performance based on DQN algorithm.



(b) RL performance based on PPO algorithm.

Figure 8. Performance (mean and standard error for 5 runs) of DQN and PPO algorithms with state representations learned from different embedded dimension in mobile robot 2D navigation environment.

Table 4. The influence of the embedded dimension in Mobile Robot environment. The average total reward is calculated among the last 100 episodes of 10^6 training timesteps.

Task	Embedded Dimension	Ground Truth Correlation (GTC)					*KNN-MSE*	Average Total Reward	
		Max Correlation Matrix				Mean		DQN	PPO
1D-Nav.	1	0.6304	2.4059e-15	None	None	0.3152	**4e-05**	137.45 ± 75.10	**238.95 ± 4.29**
	2	0.7808	1.7720e-15	None	None	0.3904	0	201.39 ± 50.51	**239.06 ± 4.31**
	3	0.8609	3.6633e-15	None	None	0.4304	0	**238.29 ± 4.83**	237.54 ± 5.02
	4	0.5340	2.8306e-15	None	None	0.2670	0	**237.18 ± 8.39**	214.43 ± 42.96
	10	0.4186	2.5974e-15	None	None	0.2074	0	237.43 ± 5.66	230.76 ± 26.32
	20	0.5624	2.0297e-15	None	None	0.2812	0	235.12 ± 16.97	238.51 ± 4.60
	50	0.6885	2.2200e-15	None	None	0.3443	0	235.85 ± 12.09	238.69 ± 4.62
	100	0.8709	2.3983e-15	None	None	**0.4355**	0	225.80 ± 12.51	237.97 ± 5.23
	200	0.8563	3.2195e-15	None	None	0.4282	0	213.4 ± 40.65	238.35 ± 4.68
	1000	0.7564	4.0314e-15	None	None	0.3782	0	227.10 ± 25.33	222.80 ± 34.69
2D-Nav.	1	0.2332	0.2354	0.9999	0.9999	0.6172	**2.0640**	−68.63 ± 43.05	8.91 ± 15.14
	2	0.0410	0.8035	0.9999	0.9999	0.7111	0.0683	9.68 ± 26.49	121.25 ± 60.75
	3	0.3574	0.7163	0.9999	0.9999	0.7684	0.0096	125.15 ± 50.85	207.74 ± 32.35
	4	0.6639	0.6878	0.9999	0.9999	**0.8379**	0.0007	**221.73 ± 22.02**	207.0 5 ± 54.74
	10	0.1448	0.1485	0.9999	0.9999	0.5733	0.00063	**224.31 ± 13.85**	207.15 ± 26.38
	50	0.2263	0.1990	0.9999	0.9999	0.6063	0.00064	220.05±18.82	213.31 ± 21.43
	100	0.2062	0.2997	0.9999	0.9999	0.6265	0.00062	217.76 ± 20.79	226.95 ± 5.66
	200	0.3303	0.4587	0.9999	0.9999	0.6973	0.00066	217.43 ± 22.76	**227.30 ± 5.38**
	1000	0.4258	0.4311	0.9999	0.9999	0.7142	0.00086	212.72±21.68	**225.87 ± 8.01**

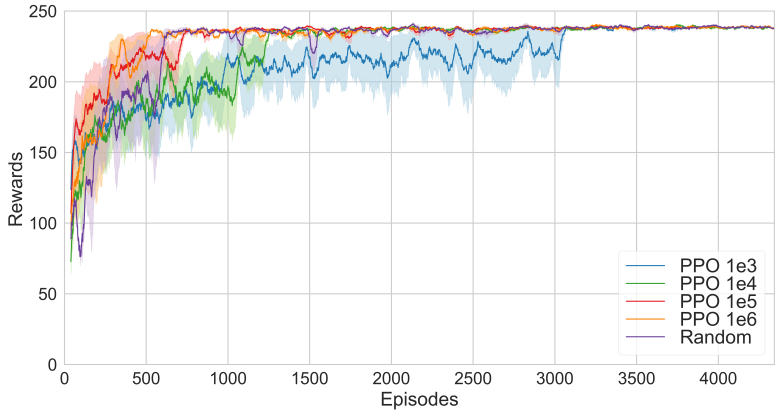
5.4. Sample Quality

How does the sample quality affect RL algorithm performance?

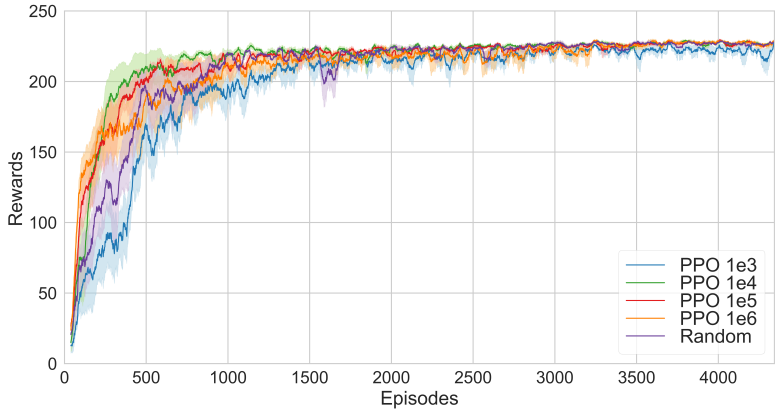
The quality of samples used for SRL plays an important role in state representation especially in the visual-based RL. We usually need to do a trade-off between the computation complexity and the state representation accuracy, because visual-based RL demands more computation resource to process pixel-based data in order to learn effective state representation and policy, and a limited numbers of samples cannot cover the whole state space.

In this part, we generated five pixel datasets for mobile robot 1D navigation task and mobile robot one-random-target task separately, which were created by random policy, pre-trained PPO policy after 1000 (1e3) steps, pre-trained PPO policy after 10,000 (1e4) steps, pre-trained PPO policy after 100,000 (1e5) steps and pre-trained PPO policy after 1,000,000 (1e6) steps respectively. We used 10,000 samples from each kind of datasets to learn state representations. Reward is sparse (as illustrated in Table 2) and actions are discrete (encoded as integers) for all datasets.

Results: Table 5 gives the *GTC* metrics, KNN-MSE metrics for several approaches and the associated RL performance using PPO algorithm. The corresponding RL performance curves are shown in Figure 9. We find that in the two mobile robot navigation tasks, the effects of sample quality are not consistent across environments. Datasets generated from random policy, on the other hand shows a relatively stable performance because the random policy allows the agent to explore more state and action space, effectively covering the huge policy search space.



(a) Learning performance in mobile robot 1D navigation task.



(b) Learning performance in mobile robot 2D navigation task.

Figure 9. The influence of sampling methods in mobile robot navigation tasks based on PPO algorithm.

Table 5. The influence of sample quality in mobile robot environment.

Task	Sampling Method	Correlation					*KNN-MSE*	Average Total Reward
		Max Correlation Matrix				Mean		PPO (1e6 Timesteps)
1D-Nav.	Random	0.8563	3.2195e-15	None	None	0.4282	0	238.35 ± 4.68
	PPO 1e3	0.8635	3.9844e-15	None	None	0.4318	0	238.45 ± 3.72
	PPO 1e4	0.6170	4.2631e-15	None	None	0.3085	1e-05	238.60 ± 4.11
	PPO 1e5	0.6170	4.2631e-15	None	None	0.3085	0	238.39 ± 3.66
	PPO 1e6	0.8110	2.5734e-15	None	None	0.4055	0	238.41 ± 4.06
2D-Nav.	Random	0.3303	0.4587	0.9999	0.9999	0.6973	0.00066	227.30 ± 5.38
	PPO 1e3	0.7826	0.3982	0.9999	0.9999	0.7952	0.00066	222.52 ± 13.86
	PPO 1e4	0.7804	0.3324	0.9999	0.9999	0.7782	0.00058	227.56 ± 5.25
	PPO 1e5	0.7804	0.3324	0.9999	0.9999	0.7782	0.00058	227.15 ± 5.96
	PPO 1e6	0.7784	0.3052	0.9999	0.9999	0.7704	0.00055	227.27 ± 8.48

5.5. Regularization Method

Is regularization item useful for learning SRL model?

One of the most common problems data science professionals face is to avoid overfitting. Regularization is a technique that makes slight modifications to the learning algorithm such that the model generalizes better. Thus, it improves the model's performance on the unseen data as well.

In this section, we employ the regularized state representation (*ell*1 norm and *ell*2 norm regularization terms on SRL neural networks weight) to learn navigation strategies in the mobile robot navigation with random target task. We contrast the performance of different regularization methods and analyze the results.

ℓ_1 norm and ℓ_2 norm are the most common types of regularization. ℓ_2 norm is also known as weight decay as it forces the weights to decay towards zero (but not exactly zero); In ℓ_1 norm, we penalize the absolute value of the weights. Unlike ℓ_2, the weights may be reduced to zero here. Hence, it is very useful when we are trying to compress our model. Otherwise, we usually prefer ℓ_2 over it. These update the general cost function by adding another term known as the regularization term.

$$\mathcal{J} = L(x, \hat{x}) + \lambda R(w), \tag{17}$$

where L is the regular loss function, R is the regularization term, and λ is the scaling parameter in front of the regularization term to adjust the trade-off between the two objectives.

Due to the addition of this regularization item, the values of weight matrices decrease because it assumes that a neural network with smaller weight matrices leads to simpler models. Therefore, it will also reduce overfitting to quite an extent.

Results: In the mobile robot one-random-target task, we test several regularization scaling parameter λ for ℓ_1 norm and ℓ_2 norm from 10^{-1} to 10^{-6} and choose the best one, which is 10^{-6} for ℓ_1 norm and 10^{-3} for ℓ_2 norm. As it shows in Figure 10, ℓ_2 regularized SRL is second to ground truth, while ℓ_1 is the worst. ℓ_1 regularization in SRL can generate adequate sparsity. In comparison, ℓ_2 regularization, which is easily solved by the ridge regression, is utilized to achieve faster and robust learning. ℓ_1 and ℓ_2 regularizations have their contributions for robust sparse representation. ℓ_1 regularization is the preferred choice when having a high number of features as it provides sparse solutions. Although ℓ_1 regularization technique has a computational benefit since zero-coefficient features can be avoided, it is discovered that ℓ_1-norm based SRL is not necessary to produce positive results as expected.

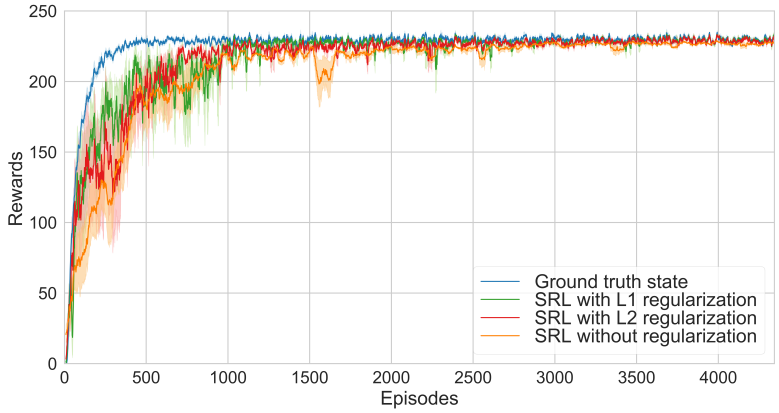


Figure 10. Influence of regularization methods on RL performance in mobile robot one-random-target task (mean and standard error for five runs with PPO algorithm). The coefficients of ℓ_1 regularization and ℓ_2 regularization are set as 10^{-6} and 10^{-3} respectively.

5.6. State Representation Learning Methods

How can the SRL methods affect the RL algorithm performance?

As illustrated in Section 2.2, here we compare the performance of several SRL methods on the mobile robot one-random-target task, including forward model, inverse model, autoencoders, robotic priors, reward prediction, combination model and supervised model (state representations trained with GT states model).

Results: Figure 11 shows the average reward during the learning of PPO algorithm based on different SRL models. The effect of the forward model is the worst. The second last one is the autoencoder and reward prediction. As illustrated in Section 2.3, combining objectives make it possible to share the strengths of each model. In contrast it should be noted that the combination SRL model is not the most effective model. This is because that combing different objectives into a single embedding could sometimes result in the collision between themself, thus weakening the contribution of each component SRL model.

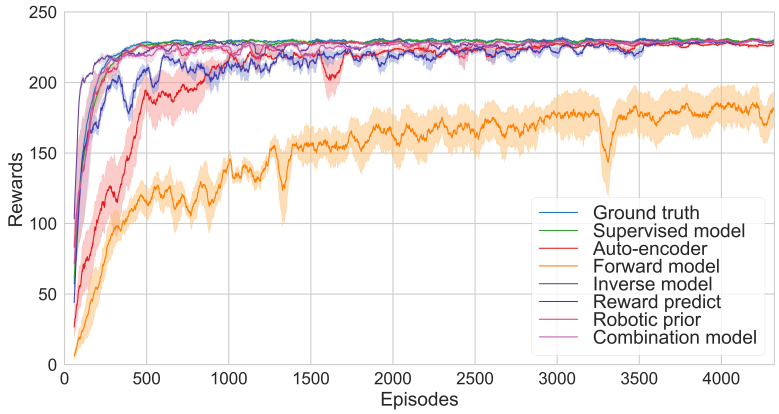


Figure 11. Performance (mean and standard error for 5 runs) on mobile robot one-random-target task with PPO algorithm for different SRL mehtods.

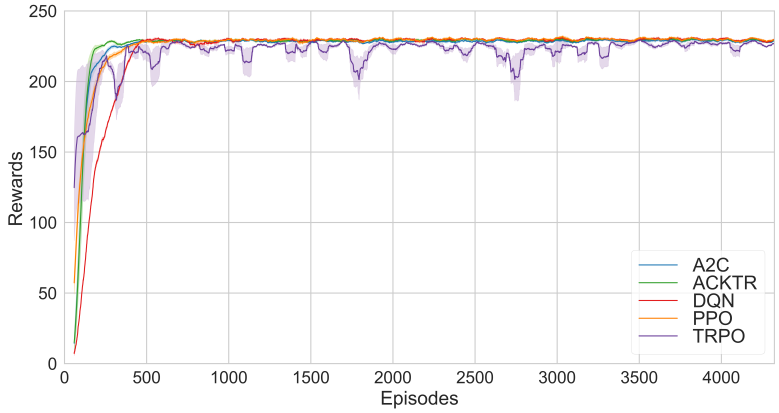
5.7. Reinforcement Learning Method

How do the state representations affect different RL algorithms' performance?

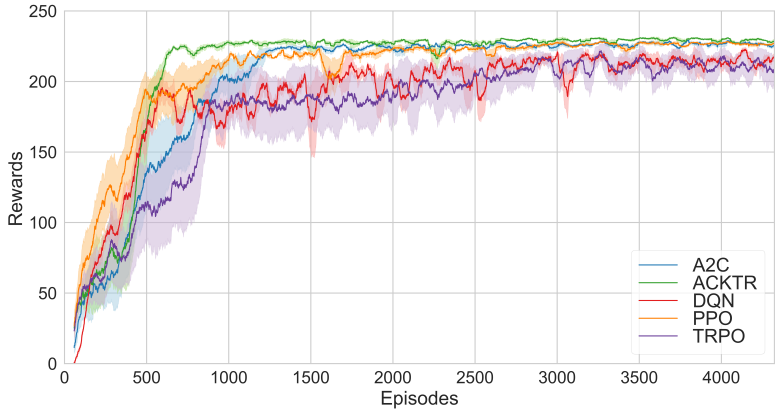
In this section, we contrast several RL algorithms including both value-based algorithms (DQN) and policy-based algorithms (A2C, PPO, TRPO, ACKTR) on both ground truth state and the learned state.

Results: In the mobile robot navigation with random target task, as illustrated in Figure 12, RL algorithms converge faster and perform more consistently when they deal with with ground truth state in general. However, when we replace it with the learned

state, the agent learns much slower and unstably, where DQN and TRPO perform worst. We observed that PPO was one of the best RL algorithm for SRL benchmarking. It allows us to obtain good performance and is consistent without needing to change any hyperparameters.



(a) RL algorithms' performance using ground truth states.



(b) RL algorithms' performance using SRL (auto-encoder) states.

Figure 12. Performance (mean and standard error) on RL algorithms using SRL states with mobile robot one-random-target task.

6. Summary Analysis and Discussion

Given the above results, we try to analyze whether representations show consistent performance in different environments. Regarding the given property measures, some representations have consistent performance while others do not. Some measures that do not have any strong relationship with the RL performance. Below we list major points summarized from the experiments.

- We found that auxiliary tasks do help with reinforcement learning during our experiments, but not all of them. On the contrary, adding a decoder for reconstructing the observation image is consistently bad at predicting expert knowledge such as the coordinate of the agent. Although the input-decoder auxiliary task has been empirically shown effective when it is used to prevent the representation from converging to 0, adding the decoder forces the agent to include every detail in the observation, even part of them is useless. Thus, one guess is that such a constraint requires a larger capacity in the representation than necessary and hurts the representation's ability to extract useful information.
- GTC_{mean} is a good indicator of the performance that can be obtained in RL. When measuring GTC_{mean}, we see that representations with a relatively high GTC_{mean} score always show a good performance in RL (a higher mean reward).

- Auxiliary tasks generally improve the representation in terms of complexity reduction and decorrelation, regardless of whether the RL performance is improved or not. But there does not appear to be a clear relationship between the measures of these properties and the RL performance. In short, auxiliary tasks should be designed according to the task characteristics.
- Learning policies from vision-based DRL is sensitive to hyperparameter changes. For instance, hyperparameters (such as the embedded dimension) tuning of DQN is required in order to have decent results from the pixels (Figure 7 and Figure 8). However, the DRL performance is relatively stable by means of SRL methods. This can be explained by the reduced search space: the task is simpler to solve when features are already extracted.

7. Conclusions and Future Work

This work presents the advantages of decoupling feature extraction from policy learning in vision-based RL, on a set of robotics tasks. This decomposition reduces the search space, accelerates training, does not degrade final performances and gives more easily interpretable representations. Various experiments have been conducted to analyze the effect of several influence factors of the SRL model. We illustrate the variability in the reported methods and suggest guidelines to make future studies in SRL more reproducible and stable.

Although we have made conclusions based on plenty of experimental results, there remain things we would like to investigate further. The first thing is to improve the criterion and look for better definitions for properties listed in this work. Furthermore, there has been a substantial effort to characterize transfer, generalization, and overfitting in DRL, primarily in terms of performance [57–59]. This motivates the need for new domains and evaluation methodologies in SRL. Last but not least, future work should confirm the results in this paper by experimenting with real robots in more complex tasks.

Author Contributions: Conceptualization, J.R.; Project administration, Y.Z. (Yujun Zeng); Software, J.R. and Y.Z. (Yichuan Zhang); Writing, J.R. and S.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by a grant from the National Natural Science Foundation of China No.62106279.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Experimental Settings

In this section, we show detail hyperparameter configurations of the algorithms adopted in our experiments, including DQN, A2C, TRPO, PPO and ACKTR. Our experiments are done using the available codebase from OpenAI rllab and OpenAI Baselines. Each of our experiments are performed over five experimental trials with different random seeds, and results are averaged over all trials. Hyperparameters are as follows unless explicitly specified as otherwise (such as in hyperparameter modifications where we alter a hyperparameter under investigation).

Appendix A.1. DQN Algorithm Configurations

- Batch size: 32
- Learning rate: 10^{-4}
- Discount factor: $\gamma = 0.99$
- Replay buffer size: 5×10^4

- Training frequency: 4
- Start learning step: 500
- Target Q network update frequency: 500
- Exploration fraction: 0.1
- Exploration final epsilon: $\epsilon = 0.02$

Appendix A.2. A2C Algorithm Configurations

- Batch size: 32
- Learning rate: 7×10^{-4}
- Discount factor: $\gamma = 0.99$
- Replay buffer size: 5×10^{4}
- Training frequency: 4
- Start learning step: 500
- Target Q network update frequency: 500
- Exploration fraction: 0.1
- Exploration final epsilon: $\epsilon = 0.02$

Appendix A.3. PPO Algorithm Configurations

- Minibatch size: 32×4
- Learning rate: 2.5×10^{-4}
- Discount factor: $\gamma = 0.99$
- VF coeff. c_1: 0.5
- Entropy coeff. c_2: 0.01
- Num. epochs: 4
- Clip range: 0.2

Appendix A.4. ACKTR Algorithm Configurations

- Learning rate: 7×10^{-4}
- Discount factor: $\gamma = 0.99$
- VF coeff. c_1: 0.5
- Entropy coeff. c_2: 0.01
- VF fisher coeff. c_3: 1

References

1. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.A.; Fidjeland, A.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
2. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. In Proceedings of the 4th International Conference on Learning Representations, ICLR, San Juan, Puerto Rico, 2–4 May 2016.
3. Jaderberg, M.; Mnih, V.; Czarnecki, W.M.; Schaul, T.; Leibo, J.Z.; Silver, D.; Kavukcuoglu, K. Reinforcement Learning with Unsupervised Auxiliary Tasks. In Proceedings of the 5th International Conference on Learning Representations, ICLR, Toulon, France, 24–26 April 2017.
4. Jaderberg, M.; Czarnecki, W.M.; Dunning, I.; Marris, L.; Lever, G.; Castaneda, A.G.; Beattie, C.; Rabinowitz, N.C.; Morcos, A.S.; Ruderman, A.; et al. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science* **2019**, *364*, 859–865. [CrossRef] [PubMed]
5. Espeholt, L.; Soyer, H.; Munos, R.; Simonyan, K.; Mnih, V.; Ward, T.; Doron, Y.; Firoiu, V.; Harley, T.; Dunning, I.; et al. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. In Proceedings of the 35th International Conference on Machine Learning, ICML, Stockholmsmässan, Stockholm, Sweden, 10–15 July 2018; Volume 80, pp. 1406–1415.
6. Kalashnikov, D.; Irpan, A.; Pastor, P.; Ibarz, J.; Herzog, A.; Jang, E.; Quillen, D.; Holly, E.; Kalakrishnan, M.; Vanhoucke, V.; et al. QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation. *arXiv* **2018**, arXiv:1806.10293.
7. Lake, B.M.; Ullman, T.D.; Tenenbaum, J.B.; Gershman, S.J. Building machines that learn and think like people. *Behav. Brain Sci.* **2017**, *40*, e253. [CrossRef] [PubMed]
8. Kaiser, L.; Babaeizadeh, M.; Milos, P.; Osinski, B.; Campbell, R.H.; Czechowski, K.; Erhan, D.; Finn, C.; Kozakowski, P.; Levine, S.; et al. Model Based Reinforcement Learning for Atari. In Proceedings of the 8th International Conference on Learning Representations, ICLR, Addis Ababa, Ethiopia, 26–30 April 2020.

9. Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; Meger, D. Deep Reinforcement Learning That Matters. In Proceedings of the 32th AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, LA, USA, 2–7 February 2018; pp. 3207–3214.
10. Lange, S.; Riedmiller, M.A. Deep auto-encoder neural networks in reinforcement learning. In Proceedings of the International Joint Conference on Neural Networks, IJCNN, Barcelona, Spain, 18–23 July 2010; pp. 1–8.
11. Lesort, T.; Rodríguez, N.D.; Goudou, J.; Filliat, D. State representation learning for control: An overview. *Neural Netw.* **2018**, *108*, 379–392. [CrossRef] [PubMed]
12. Kaelbling, L.P.; Littman, M.L.; Cassandra, A.R. Planning and acting in partially observable stochastic domains. *Artif. Intell.* **1998**, *101*, 99–134. [CrossRef]
13. He, K.; Fan, H.; Wu, Y.; Xie, S.; Girshick, R.B. Momentum Contrast for Unsupervised Visual Representation Learning. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, Seattle, WA, USA, 13–19 June 2020; pp. 9726–9735.
14. Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G.E. A Simple Framework for Contrastive Learning of Visual Representations. In Proceedings of the 37th International Conference on Machine Learning, ICML, Virtual Event, 13–18 July 2020.
15. Cubuk, E.D.; Zoph, B.; Mané, D.; Vasudevan, V.; Le, Q.V. AutoAugment: Learning Augmentation Strategies From Data. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Long Beach, CA, USA, 16–20 June 2019; pp. 113–123.
16. Cubuk, E.D.; Zoph, B.; Shlens, J.; Le, Q. RandAugment: Practical Automated Data Augmentation with a Reduced Search Space. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 14–19 June 2020.
17. Laskin, M.; Srinivas, A.; Abbeel, P. CURL: Contrastive Unsupervised Representations for Reinforcement Learning. In Proceedings of the 37th International Conference on Machine Learning, ICML, Virtual Event, 13–18 July 2020.
18. Laskin, M.; Lee, K.; Stooke, A.; Pinto, L.; Abbeel, P.; Srinivas, A. Reinforcement Learning with Augmented Data. *arXiv* **2020**, arXiv:2004.14990.
19. Hansen, N.; Jangir, R.; Sun, Y.; Alenyà, G.; Abbeel, P.; Efros, A.A.; Pinto, L.; Wang, X. Self-Supervised Policy Adaptation during Deployment. In Proceedings of the 9th International Conference on Learning Representations, ICLR, Virtual Event, 3–7 May 2021.
20. Jakab, T.; Gupta, A.; Bilen, H.; Vedaldi, A. Unsupervised Learning of Object Landmarks through Conditional Image Generation. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018; pp. 4020–4031.
21. Wang, X.; Lian, L.; Yu, S.X. Unsupervised Visual Attention and Invariance for Reinforcement Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 19–25 June 2021.
22. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.P.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. In Proceedings of the 33rd International Conference on Machine Learning, ICML, New York, NY, USA, 19–24 June 2016; Volume 48, pp. 1928–1937.
23. Painter-Wakefield, C.; Parr, R. Greedy Algorithms for Sparse Reinforcement Learning. In Proceedings of the 29th International Conference on Machine Learning, ICML, Edinburgh, UK, 26 June–1 July 2012.
24. Parr, R.; Painter-Wakefield, C.; Li, L.; Littman, M.L. Analyzing feature generation for value-function approximation. In Proceedings of the 29th International Conference on Machine Learning, ICML, Corvallis, QR, USA, 20–24 June 2007.
25. Konidaris, G.D.; Osentoski, S.; Thomas, P.S. Value Function Approximation in Reinforcement Learning Using the Fourier Basis. In Proceedings of the 25th AAAI Conference on Artificial Intelligence, AAAI, San Francisco, CA, USA, 7–11 August 2011.
26. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
27. Sutton, R.S. Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 1996; pp. 1038–1044.
28. Xu, X.; Hu, D.; Lu, X. Kernel-based least squares policy iteration for reinforcement learning. *IEEE Trans. Neural Netw.* **2007**, *18*, 973–992. [CrossRef] [PubMed]
29. Yarats, D.; Zhang, A.; Kostrikov, I.; Amos, B.; Pineau, J.; Fergus, R. Improving Sample Efficiency in Model-Free Reinforcement Learning from Images. *arXiv* **2019**, arXiv:1910.01741.
30. Bellemare, M.G.; Dabney, W.; Dadashi, R.; Taïga, A.A.; Castro, P.S.; Roux, N.L.; Schuurmans, D.; Lattimore, T.; Lyle, C. A Geometric Perspective on Optimal Representations for Reinforcement Learning. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 4358–4369.
31. Finn, C.; Abbeel, P.; Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In Proceedings of the 34th International Conference on Machine Learning, ICML, Sydney, NSW, Australia, 6–11 August 2017; Volume 70, pp. 1126–1135.
32. Javed, K.; White, M. Meta-Learning Representations for Continual Learning. *arXiv* **2019**, arXiv:1905.12588.
33. Silver, D.; van Hasselt, H.; Hessel, M.; Schaul, T.; Guez, A.; Harley, T.; Dulac-Arnold, G.; Reichert, D.P.; Rabinowitz, N.C.; Barreto, A.; et al. The Predictron: End-To-End Learning and Planning. In Proceedings of the 34th International Conference on Machine Learning, ICML, Sydney, NSW, Australia, 6–11 August 2017; Volume 70, pp. 3191–3199.
34. Shelhamer, E.; Mahmoudieh, P.; Argus, M.; Darrell, T. Loss is its own Reward: Self-Supervision for Reinforcement Learning. In Proceedings of the 5th International Conference on Learning Representations, ICLR, Toulon, France, 24–26 April 2017.
35. van den Oord, A.; Li, Y.; Vinyals, O. Representation Learning with Contrastive Predictive Coding. *arXiv* **2018**, arXiv:1807.03748.

36. Schmidhuber, J. *Making the World Differentiable: On Using Fully Recurrent Self-Supervised Neural Networks for Dynamic Reinforcement Learning and Planning in NON-Stationary Environments*; Technical Report FKI-126; Institut für Informatik, Technische Universität München: München, Germany, 1990.
37. Higgins, I.; Pal, A.; Rusu, A.A.; Matthey, L.; Burgess, C.; Pritzel, A.; Botvinick, M.; Blundell, C.; Lerchner, A. DARLA: Improving Zero-Shot Transfer in Reinforcement Learning. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017.
38. Sermanet, P.; Lynch, C.; Chebotar, Y.; Hsu, J.; Jang, E.; Schaal, S.; Levine, S. Time-Contrastive Networks: Self-Supervised Learning from Video. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation, ICRA, Brisbane, Australia, 21–25 May 2018; pp. 1134–1141.
39. Dwibedi, D.; Tompson, J.; Lynch, C.; Sermanet, P. Learning Actionable Representations from Visual Observations. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, Madrid, Spain, 1–5 October 2018; pp. 1577–1584.
40. Warde-Farley, D.; de Wiele, T.V.; Kulkarni, T.D.; Ionescu, C.; Hansen, S.; Mnih, V. Unsupervised Control Through Non-Parametric Discriminative Rewards. In Proceedings of the 7th International Conference on Learning Representations, ICLR, New Orleans, LA, USA, 6–9 May 2019.
41. Anand, A.; Racah, E.; Ozair, S.; Bengio, Y.; Côté, M.; Hjelm, R.D. Unsupervised State Representation Learning in Atari. *arXiv* **2019**, arXiv:1906.08226.
42. Watter, M.; Springenberg, J.T.; Boedecker, J.; Riedmiller, M.A. Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images. *arXiv* **2015**, arXiv:1506.07365.
43. Pathak, D.; Agrawal, P.; Efros, A.A.; Darrell, T. Curiosity-driven Exploration by Self-supervised Prediction. In Proceedings of the 34th International Conference on Machine Learning, ICML, Sydney, NSW, Australia, 6–11 August 2017.
44. Baldi, P. Autoencoders, Unsupervised Learning, and Deep Architectures. In Proceedings of the Unsupervised and Transfer Learning—Workshop held at ICML 2011, Bellevue, DC, USA, 2 July 2012; Volume 27, pp. 37–50.
45. Kingma, D.P.; Welling, M. Auto-Encoding Variational Bayes. In Proceedings of the 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, 14–16 April 2014.
46. Vincent, P.; Larochelle, H.; Bengio, Y.; Manzagol, P. Extracting and composing robust features with denoising autoencoders. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, 5–9 June 2008*; Cohen, W.W., McCallum, A., Roweis, S.T., Eds.; ACM: New York, NY, USA, 2008; Volume 307, pp. 1096–1103.
47. Jonschkowski, R.; Brock, O. State Representation Learning in Robotics: Using Prior Knowledge about Physical Interaction. In *Robotics: Science and Systems X*; University of California: Berkeley, CA, USA, 2014.
48. Jonschkowski, R. Learning Robotic Perception through Prior Knowledge. Ph.D. Thesis, Technical University of Berlin, Berlin, Germany, 2018.
49. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. OpenAI Gym. *arXiv* **2016**, arXiv:1606.01540.
50. Sermanet, P.; Lynch, C.; Hsu, J.; Levine, S. Time-Contrastive Networks: Self-Supervised Learning from Multi-view Observation. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops, Honolulu, HI, USA, 21–26 July 2017; pp. 486–487.
51. Lesort, T.; Seurin, M.; Li, X.; Rodríguez, N.D.; Filliat, D. Unsupervised state representation learning with robotic priors: A robustness benchmark. *arXiv* **2017**, arXiv:1709.05185.
52. Raffin, A.; Hill, A.; Traoré, R.; Lesort, T.; Rodríguez, N.D.; Filliat, D. S-RL Toolbox: Environments, Datasets and Evaluation Metrics for State Representation Learning. *arXiv* **2018**, arXiv:1809.09369.
53. Andrychowicz, M.; Raichuk, A.; Stanczyk, P.; Orsini, M.; Girgin, S.; Marinier, R.; Hussenot, L.; Geist, M.; Pietquin, O.; Michalski, M.; et al. What Matters for On-Policy Deep Actor-Critic Methods? A Large-Scale Study. In Proceedings of the 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, 3–7 May 2021.
54. Islam, R.; Henderson, P.; Gomrokchi, M.; Precup, D. Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control. *arXiv* **2017**, arXiv:1708.04133.
55. Engstrom, L.; Ilyas, A.; Santurkar, S.; Tsipras, D.; Janoos, F.; Rudolph, L.; Madry, A. Implementation Matters in Deep RL: A Case Study on PPO and TRPO. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020.
56. Glorot, X.; Bordes, A.; Bengio, Y. Deep Sparse Rectifier Neural Networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*; Gordon, G., Dunson, D., Dudík, M., Eds.; PMLR: Fort Lauderdale, FL, USA, 2011; Volume 15, pp. 315–323.
57. Cobbe, K.; Klimov, O.; Hesse, C.; Kim, T.; Schulman, J. Quantifying Generalization in Reinforcement Learning. In Proceedings of the 36th International Conference on Machine Learning, ICML, Long Beach, CA, USA, 9–15 June 2019; Volume 97, pp. 1282–1289.
58. Farebrother, J.; Machado, M.C.; Bowling, M. Generalization and Regularization in DQN. *arXiv* **2018**, arXiv:1810.00123.
59. Packer, C.; Gao, K.; Kos, J.; Krähenbühl, P.; Koltun, V.; Song, D. Assessing Generalization in Deep Reinforcement Learning. *arXiv* **2018**, arXiv:1810.12282.