State Representation Learning for Control: An Overview

Timothée Lesort, Natalia Díaz-Rodríguez, Jean-François Goudou, David Filliat

State Representation Learning for Control: An Overview

Timothée Lesort[1, 2], Natalia Díaz-Rodríguez[1], Jean-François Goudou[2], and David Filliat[1]

[1]U2IS, ENSTA ParisTech, Inria FLOWERS team, Université Paris Saclay, Palaiseau, France.,
{timothee.lesort, natalia.diaz, david.filliat}@ensta-paristech.fr
[2]Vision Lab, Thales, Theresis, Palaiseau, France.,
jean-francois.goudou@thalesgroup.com

Abstract

Representation learning algorithms are designed to learn abstract features that characterize data. State representation learning (SRL) focuses on a particular kind of representation learning where learned features are in low dimension, evolve through time, and are influenced by actions of an agent. The representation is learned to capture the variation in the environment generated by the agent's actions; this kind of representation is particularly suitable for robotics and control scenarios. In particular, the low dimension characteristic of the representation helps to overcome the curse of dimensionality, provides easier interpretation and utilization by humans and can help improve performance and speed in policy learning algorithms such as reinforcement learning.

This survey aims at covering the state-of-the-art on state representation learning in the most recent years. It reviews different SRL methods that involve interaction with the environment, their implementations and their applications in robotics control tasks (simulated or real). In particular, it highlights how generic learning objectives are differently exploited in the reviewed algorithms. Finally, it discusses evaluation methods to assess the representation learned and summarizes current and future lines of research.

Keywords: State Representation Learning, Low Dimensional Embedding Learning, Learning Disentangled Representations, Disentanglement of control factors, Robotics, Reinforcement Learning

1 Introduction

Robotics control and artificial intelligence (AI) in a broader perspective heavily rely on the availability of compact and expressive representations of the sensor data. Designing such representations has long been performed manually by the designer, but deep learning now provides a general framework to learn such representations from data. This is particularly interesting for robotics where multiple sensors (such as cameras) can provide very high dimensional data, while the robot objective can often be expressed in a much lower dimensional space (such as the 3D position of an object in a manipulation task). This low dimensional representation, frequently called the *state* of the system, has the crucial role of encoding essential information (for a given task) while discarding the many irrelevant aspects of the original data. By *Low dimensional*, we mean that the learned state dimension is significantly smaller than the dimensionality of the observation space.

Such state representation is at the basis of the classical reinforcement learning (RL) framework [Sutton, 1998] in which an agent interacts with its environment by choosing an action as a function of the environment state in order to maximize an expected (discounted) reward. Following this framework, we call *observation* the raw information provided by one or several of the robot sensors, and we call *state* a compact depiction of this observation that retains the information necessary for the robot to choose its actions.

While deep reinforcement learning algorithms have shown that it is possible to learn controllers directly from observations [Mnih et al., 2015], reinforcement learning (or other control algorithms) can take advantage of low dimensional and informative representations, instead of raw data, to solve tasks more efficiently [Munk et al., 2016]. Such efficiency is critical in robotic applications where experimenting an action is a costly operation. In robotics, as well as in machine learning, finding and defining interesting states (or features) for control tasks usually requires a considerable amount of manual engineering. It is therefore interesting to learn these features with as little supervision as possible. The goal is thus to avoid direct supervision using a *true* state, but instead use information about the actions performed by the agent, their consequences in the observation space, and rewards (even if sparse, and when available). Along with this information, one can also set generic constraints on what a good state representation should be [Jonschkowski and Brock, 2015, Lesort et al., 2017].

Feature learning in general is a wide domain which aims at decomposing data into different features that can faithfully characterize it. It has been a particular motivation for deep learning to automatically learn a large range of specific feature detectors in high dimensional problems. State representation learning (SRL) is a particular case of feature learning in which the features to learn are low dimensional, evolve through time, and are influenced by actions or interactions. SRL is generally framed in a control setup constrained to favor small dimensions to characterize an instance of an environment or an object, often with a semantic meaning that correlates with some physical feature. The physical feature can be, for instance, a position, distance, angle or an orientation. The objective of SRL is to take advantage of time steps, actions, and optionally rewards, to transform observations into states: a vector of a reduced set of the most representative features that is sufficient for efficient policy learning. It is also worth distinguishing between feature learning on a process that is only observed, and learning the state representation of a process in which the learning agent possesses embodiment and acts. The first one considers learning directly from observations, e.g., pixels, and leaves no room for the agent to act. The latter gives opportunity to exploit more possibilities to learn better representations by balancing between exploration and exploitation, e.g., active learning, or artificial curiosity [Péré et al., 2018, Pathak et al., 2017].

As stated above, learning in this context should be performed without explicit supervision. In this article we therefore focus on SRL where *learning* does not have the *pattern recognition* regression or classification sense, but rather the sense of the process of model building [Lake et al., 2016]. Building such models can then exploit a large set of objectives or constraints, possibly taking inspiration from human learning. As an example, infants expect inertial objects to follow principles of persistence, continuity, cohesion and solidity before appearance-based elements such as color, texture and perceptual goodness. At the same time, these principles help guide later learnings such as object' rigidity, softness and liquids properties. Later, adults will reconstruct perceptual scenes using internal representations of the objects and their physically relevant properties (mass, elasticity, friction, gravity, collision, etc.) [Lake et al., 2016]. In the same way, the SRL literature may make use of knowledge about the physics of the world, interactions and rewards whenever possible as a semi-supervision or self-supervision that aids the challenge of learning state representations

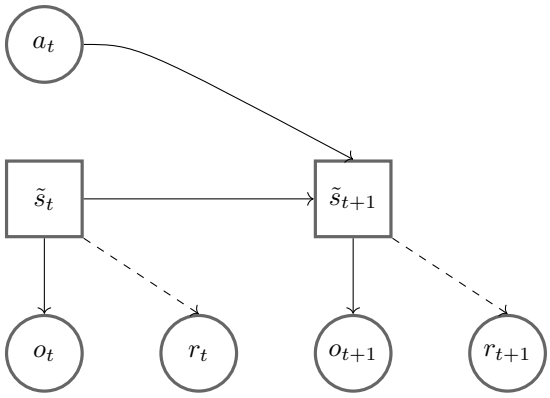

Figure 1: General model : circle are observable and square are the latent state variables.

without explicit supervision.

Recently, several different approaches have been proposed to learn such state representation. In this review paper, our objective is to present and analyze those different approaches, highlight their commonalities and differences, and to propose further research directions. We extend a previously published review [Böhmer et al., 2015] with the most recent and rapidly evolving literature of the past years and focus on approaches that learn low dimensional Markovian representations without direct supervision, i.e., exploiting sequences of observations, actions, rewards and generic learning objectives. The works selected in this survey mostly evaluate their algorithms in simulations where agents interact with an environment. More marginally, some SRL algorithms are tested on real settings such as robotics tasks, e.g., manipulation or exploration as detailed in Section 4.4.

In the remainder of the paper, we first introduce the formal framework and notation, then present the objectives that can be used to learn state representations, and discuss the implementation aspects of these approaches before summarizing some current and future lines of research.

2 Formalism and definitions

2.1 SRL Formalism

The nomenclature we use is very close to the one used in reinforcement learning literature [Sutton, 1998] and is illustrated in Fig. 1. We define an environment $\mathcal{E}$ where an agent performs actions $a_t \in \mathcal{A}$ at time step t and where $\mathcal{A}$ is the action space (continuous or discrete). Each action makes the agent transition from a true state $\tilde{s}_t$ to $\tilde{s}_{t+1}$ which is unknown but assumed to exist. We call the true state space $\tilde{\mathcal{S}}$. The agent obtains an observation of $\mathcal{E}$ from its sensors, denoted $o_t \in \mathcal{O}$ where $\mathcal{O}$ is the observation space.

Optionally, the agent may receive a reward r_t. The reward is given at $\tilde{s}_t$ by a reward function designed to lead the agent to a certain behavior that solves a task. The reward is optional as learning a state representation don't aim to solve a task, but is often present as one of the goal of SRL may be to improve task learning performance.

The SRL task is to learn a representation $s_t \in \mathcal{S}$ of dimension K with characteristics similar to those of $\tilde{s}_t$ without using $\tilde{s}_t$. More formally, SRL learns a mapping ϕ of the history of observation to the current state $s_t = \phi(o_{1:t})$. Note that actions $a_{1:t}$ and rewards $r_{1:t}$ can also be added to the parameters of ϕ [Jonschkowski and Brock, 2015]. In this paper, we are specifically interested in the particular setting in which this mapping is learned through proxy objectives without access to the true state $\tilde{s}_t$. This family of approaches is called unsupervised or self-supervised.

Finally, we note $\hat{o}_t$ the reconstruction of o_t (similarly for $\hat{a}_t$ and $\hat{r}_t$), that will be used in various SRL approaches.

2.2 SRL approaches

Based on the previously defined notations, we can briefly summarize the common strategies used in state representation learning that are detailed in the next sections. In the following, θ represents the parameters optimized by minimizing the model's loss function. In most of the approaches we present, this model is implemented with a neural network.

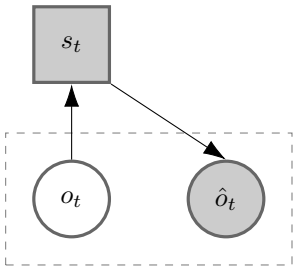

Figure 2: Auto-Encoder: reconstructing the observation. The error is computed between observation o_t and its reconstruction $\hat{o}_t$. White components are input data and gray ones are output data.

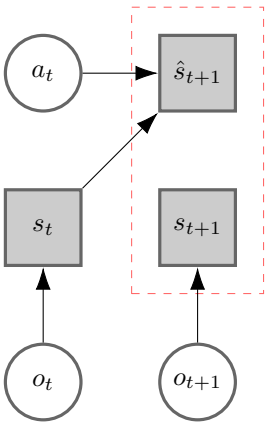

Figure 3: Forward Model: predicting next state s_{t+1} from s_t and a_t. The error is computed between predicted state $\hat{s}_{t+1}$ and the actual next state s_{t+1}

- **Reconstructing the observation**: learning the function ϕ (Eq. 1) so that it is possible to reconstruct the observation with a decoder ϕ^{-1} (Eq. 2) by minimizing the reconstruction error between the original observation and its predicted reconstruction. The reconstruction is learned under different constraints that give to s_t specific characteristics (e.g., dimensionality constraints, local denoising criterion [Vincent et al., 2010], sparse encoding constraints [Vincent et al., 2008], etc.) (Fig. 2).

$$s_t = \phi(o_t; \theta_\phi) \tag{1}$$

$$\hat{o}_t = \phi^{-1}(s_t; \theta_{\phi^{-1}}) \tag{2}$$

  where θ_ϕ and $\theta_{\phi^{-1}}$ are the parameters learned for the encoder and decoder, respectively.

- **Learning a forward model**:

  A forward model predicts s_{t+1} from o_t or s_t and a_t (Fig. 3). In this context, we want to learn the mapping ϕ from o_t to s_t using the model that predicts s_{t+1} from o_t. Hence the prediction work in two steps, first encoding from o_t to s_t then transition from s_t to $\hat{s}_{t+1}$. We can not compute any error on s_t, however at $t+1$ the model can learn from the error between $\hat{s}_{t+1}$ and s_{t+1}. The error is back propagated through the transition model and the encoding model. Consequently the methods allows to learn a model ϕ.

$$\hat{s}_{t+1} = f(s_t, a_t; \theta_{fwd}) \tag{3}$$

  Learning such a model makes it possible to impose structural constraints on the model for state representation learning. For example, the forward model can be constrained to be linear between s_t and s_{t+1}, imposing that the system in the learned state space follows simple linear dynamics.

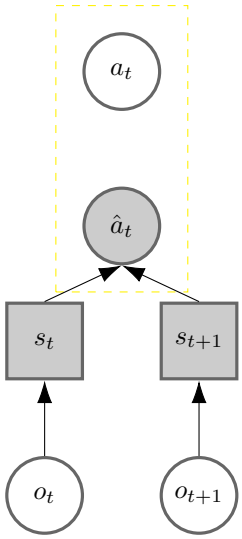

Figure 4: Inverse Model: predicting action a_t from s_t and s_{t+1}. The error is computed between predicted action $\hat{a}_t$ and the actual action a_t

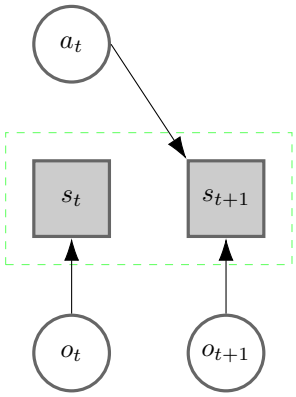

Figure 5: Model with prior: The error is computed by applying loss functions on several states

- **Learning an inverse model**:

  An inverse model predicts action a_t given observations o_t and o_{t+1} or states s_t and s_{t+1}. Like for forward model, the goal here is to learn the mapping ϕ from o_t to s_t through two steps, encoding for o_t and o_{t+1} and action prediction. The error is computed for action prediction between a_t and $\hat{a}_t$ and then back-propagated to learn the encoding

  $$\hat{a}_t = g(s_t, s_{t+1}; \theta_{inv}) \tag{4}$$

  Learning such model enforces that the state encodes enough information to recover the action that modified the state (Fig. 4).

- **Using prior knowledge to constrain the state space**: A last approach is to handle SRL by using specific constraints or prior knowledge about the functioning, dynamics or physics of the world (besides the constraints of forward and inverse models) such as the temporal continuity or the causality principles that generally reflect the interaction of the agent with objects or in the environment [Jonschkowski and Brock, 2015]. *Priors* are defined as objective or loss functions $\mathcal{L}$, applied on a set of states $s_{1:n}$ (Fig. 5), to be minimized (or maximized) under specific condition c. An example of condition can be enforcing locality or time proximity within the set of states.

  $$Loss = \mathcal{L}_{prior}(s_{1:n}; \theta_\phi | c) \tag{5}$$

  All these approaches are detailed in Section 3.

2.3 State representation characteristics

Besides the general idea that the state representation has the role of encoding essential information (for a given task) while discarding irrelevant aspects of the original data, let us detail what the characteristics of a good state representation are.

In a reinforcement learning framework, the authors of [Böhmer et al., 2015] defines a good state representation as a representation that is:

- Markovian, i.e. it summarizes all the necessary information to be able to choose an action within the policy, by looking only at the current state.
- Able to represent the true value of the current state well enough for policy improvement.
- Able to generalize the learned value-function to unseen states with similar futures.
- Low dimensional for efficient estimation.

Note that these are some characteristics expected of the state representation, but they cannot be used for learning this representation. Instead, they can later be verified by assessing the task performance for a controller based on the learned state. Note also that multiple state representations can verify these properties for a given problem and that therefore, there is no unique solution to the state representation learning problem. We detail this problem when discussing the evaluation of the learned state space in Section 4.3.

State representation learning can also be linked with the idea of learning disentangled representations that clearly separate the different factors of variation with different semantics. Following [Achille and Soatto, 2017], a good representation must be sufficient, as efficient as possible (i.e., easy to work with, e.g., factorizing the data-generating factors), and minimal (from all possible representations, take the most efficient one).

The *minimal* assumption is comparable to the simplicity prior [Jonschkowski and Brock, 2015]. It assumes that only a small number of world properties are relevant, and that there exists a low dimensional state representation of a higher level input observation. Related to *Occam's razor*, this prior favors state representations that exclude irrelevant information to encourage a lower dimensionality.

The *efficiency* aspect of the representation means that there should be no overlapping between dimensions of the learned state features. Unfortunately, independence of features alone may not be enough to assure a good quality of representations and guarantee a disentanglement of factors of variation [Thomas et al., 2017]. Higher level abstractions can, however, allow to improve this disentanglement and permit easier generalization and transfer. Cues to disentangle the underlying factors can include spatial and temporal scales, marginal independence of variables, and controllable factors [Thomas et al., 2017].

2.4 State representation learning applications

The main interest of SRL is to produce a low dimensional state space in which learning a control policy will be more efficient. Indeed, deep reinforcement learning in the observation space has shown spectacular results in control policy learning [Mnih et al., 2015, Lillicrap et al., 2015, Mnih et al., 2016] but is known to be computationally difficult and requires a large amount of data [Rusu et al., 2016]. Separation of representation learning and policy learning is a way to lighten the complete process. As described in most of the reviewed papers [Mattner et al., 2012, Watter et al., 2015, van Hoof et al., 2016, Munk et al., 2016, Curran et al., 2016, Wahlström et al., 2015, Shelhamer et al., 2017, Oh et al., 2017], this approach is used to make reinforcement learning faster in time and/or lighter in computation.

SRL can be particularly relevant with multimodal observations that are produced by several complementary sensors with high dimensionality as is, for example, the case of autonomous vehicles. Low dimensional representations are then key to make an algorithm able to take decisions from hidden factors extracted from these complementary sensors. This is for instance the case of representation learning from different temporal signals in [Duan, 2017, Bohg et al., 2017]. Audio and images are blended in [Yang et al., 2017] while RGB and depth are combined in [Duan, 2017]. SRL can also be used in a transfer learning setting by taking advantage of a state space learned on a given task to rapidly learn a related task. This is for example the case in [Jonschkowski and Brock, 2015] where a state space related to a robot position is learned in a given navigation task

and reused to quickly learn another navigation task. SRL is also used as pretraining for transfer to other applications afterwards such as reinforcement learning [Munk et al., 2016, Oh et al., 2017]. For concrete examples on SRL application scenarios see Section 4.4.

Another case where SRL could be useful is in the application of Evolution Strategies (ES) for robot control learning [Stulp and Sigaud, 2013]. Evolution strategies are a family of black box optimization algorithms that do not rely on gradient descent and can be an alternative to RL techniques (such as Q-learning and policy gradients) but are less adapted to high-dimensional problems. Indeed, the convergence time of ES algorithm depends on the dimension of the input: the larger the dimension is, the larger amount of solutions ES has to explore [Stulp and Sigaud, 2013]. ES optimization methods have shown to be efficient for deep reinforcement learning [Conti et al., 2017] but they could then take a clear advantage of a lower dimension input to explore faster the parameter space than using raw data [Alvernaz and Togelius, 2017].

3 Learning objectives

In this section, we review what objectives can be used to learn a relevant state representation. A schema detailing the core elements involved in each model's loss function was introduced in Fig. 2 – 5, which highlights the main approaches to be described here. This section touches upon machine learning tools used in SRL such as auto-encoders or siamese networks. A more detailed description of these is later addressed in Section 4.

3.1 Reconstructing the observation

A first idea that can be exploited is the fact that a true state, along with some noise, was used to generate the observation. Under the hypothesis that the noise is not too large, compressing the observation should retain the important information contained in the true state. While this idea is very often exploited with dimensionality reduction algorithms [Fodor, 2002] such as Principal Component Analysis (PCA), we focus here on the recent approaches specifically dealing with state representation.

The PCA algorithm is a linear transformation able to compress and decompress observations with minimal reconstruction error. PCA have been exploited to reduce the dimensionality of the state space during learning [Curran et al., 2016]. By projecting images into a 3- or 4-dimensional space, it is possible to produce a state that is used by a reinforcement learning algorithm and that reduces the convergence time in Super Mario games [Karakovskiy, 2012] and different simulations such as Swimmers or Mountain Car.

Auto-encoders are models that learn to reproduce their input under constraints on their internal representations such as dimensionality constraints (Fig. 2). Their architecture can therefore be used to learn a particular representation in low dimensions s_t by reconstructing o_t.

Simple auto-encoders can be used to learn 2D representation of a real pole from raw images [Mattner et al., 2012] (see Section 4.4 on evaluation scenarios). After training, the encoding vector from the AE is used to learn a controller to balance the pole. An auto-encoder whose internal representation is constrained to represent a position that serves as input to a controller is also presented in [Finn et al., 2015] and [Alvernaz and Togelius, 2017]. The proposed model learns a state representation from raw pixels of respectively a PR2 robot's hand and an agent in the VizDoom environment.

These models, based on auto-encoders that reconstruct the observation at the same time step, can however learn only if the factors of variations are only linked to the actual state, or if very prominent features exists [Lesort et al., 2017]. In order to relax this assumption, it is possible to reconstruct observations from other time steps or to use constraints on the evolution of the state (as will be more detailed in Section 3.2) to focus reconstruction on features relevant to the system dynamics.

An example of auto-encoder tuned to take into account the system dynamics is proposed in [Goroshin et al., 2015] where an AE with a siamese encoder projects sequences of images into a state representation space $\mathcal{S}$ with constraints on the transition between s_t and s_{t+1} to be linear. They use observations at several time steps in order to take time into account in the representation, and predict future observations through a single decoder that reconstructs $\hat{o}_{t+1}$. This makes the model able to learn representations that are related to several time steps and filter out random features of the environment.

The idea of using an auto-encoder to learn a projection into a state space where transitions are assumed to be linear has also been used by [Watter et al., 2015]. The model presented, "Embed To Control" (E2C), consists of a deep generative model that learns to generate image trajectories from a linear latent space.

Extending [Watter et al., 2015], the representation with dynamic constraints can be learned on policy at the same time as a reinforcement learning algorithm learns a task [van Hoof et al., 2016]. They compared different types of auto-encoders to learn visual and tactile state representations and use this representation to learn manipulation task policies for a robot. Sharing the same idea of embedding dynamic constraints into auto-encoders, Deep Variational Bayes Filters (DVBF) are an extension of Kalman filters which learn to reconstruct the observation based on a nonlinear state space using variational inference [Karl et al., 2016]. The reconstruction from a non linear state space based on a model inspired by a Deep Dynamical Model (DDM) [Wahlström et al., 2015] and E2C [Watter et al., 2015] is proposed in [Assael et al., 2015]. It is argued that the model is adapted for better training efficiency and it can learn tasks with complex non-linear dynamics [Assael et al., 2015]. Their result shows improvements over the PILCO model [Deisenroth and Rasmussen, 2011], which learns a state representation by only minimizing the reconstruction error without constraining the latent space.

3.2 Learning a forward model

Last subsection reviewed how reconstructing an observation is useful to learn state representations. Now we will review how temporal dynamics of the system can also help the same purpose. Therefore we present approaches that rely on learning a *forward* model to learn a state space. The general idea is to force states to efficiently encode the information necessary to predict the next state (Fig. 3).

As described in Section 2, in the case of the forward models we study here, the model is used as a proxy for learning s_t. The model firstly makes a projection from the observation space to the state space to obtain s_t and applies a transition to predict $\hat{s}_{t+1}$. The error is computed by comparing the estimated next state $\hat{s}_{t+1}$ with the value of s_{t+1} derived from the next observation o_{t+1} at the next time step.

Note that forward models can benefit from the observation reconstruction objective presented in Section 3.1. As an example, the works previously presented in Section 3.1 [Goroshin et al., 2015, van Hoof et al., 2016, Watter et al., 2015, Assael et al., 2015, Karl et al., 2016] belong to the auto-encoder category of models. However, they all predict future observations to learn representations and therefore, they as well belong to the family of forward models.

The method these works use to combine forward models and auto-encoders consists in mapping o_t to s_t, and then compute the transition, with the help of a_t, to obtain $\hat{s}_{t+1}$. $\hat{s}_{t+1}$ is then remapped onto the pixel space in form of a vector $\hat{o}_{t+1}$. The error is then computed pixel-wise between $\hat{o}_{t+1}$ and o_{t+1}. One common assumption is that the forward model in the learned state space is linear [Goroshin et al., 2015, van Hoof et al., 2016]. The transition is then just a linear combination of s_t and a_t as in Eq. 6. W, U and V are either fixed or learned parameters [van Hoof et al., 2016].

$$\hat{s}_{t+1} = W * \hat{s}_t + U * a_t + V \tag{6}$$

In a similar way, the *Embed to Control* model (E2C) based on variational auto-encoders uses Eq. 6 to compute the mean μ of a distribution and learn supplementary parameters for the variance σ of the distribution [Watter et al., 2015]. Then, $\hat{s}_{t+1}$ is computed with Eq. 7:

$$\hat{s}_{t+1} \sim \mathcal{N}(\mu = W * \hat{s}_t + U * a_t + V, \sigma) \tag{7}$$

Using distributions to compute $\hat{s}_{t+1}$ allows to use the KL-divergence to train the forward model. This method is also used in [Karl et al., 2016] and [Krishnan et al., 2015]. However, the transition model in [Krishnan et al., 2015] considers the KL-divergence

between $P(s_{t+1})$ and $P(\hat{s}_{t+1})$ and does not use the loss of the reconstruction based on o_{t+1} and $\hat{o}_{t+1}$.

The use of a_t is a common feature in most forward models based approaches. In fact, as several future states s_{t+1} from a given state are possible, s_t alone does not contain enough information to predict s_{t+1}. The only approach that gets rid of the need for actions assumes that the transition from s_{t-1} to s_t allows to deduce the transition from s_t to s_{t+1} and uses several past states to predict s_{t+1} [Goroshin et al., 2015]. Actions are therefore implicit in this approach.

Another use of a forward model, connected to an intrinsic curiosity model (ICM) which helps agents explore and discover the environment out of curiosity when extrinsic rewards are sparse or not present at all, is proposed in [Pathak et al., 2017]. In this model, an intrinsic reward signal is computed from the forward model's loss function $\mathcal{L}_{fwd}$ ($\hat{f}$ is the forward function learned by the model, $\hat{\phi}$ is the encoding model):

$$\mathcal{L}_{fwd}(\hat{\phi}(o_{t+1}), \hat{f}(\hat{\phi}(o_t), a_t)) = \frac{1}{2} \parallel \hat{f}(\hat{\phi}(o_t), a_t) - \hat{\phi}(o_{t+1}) \parallel_2^2 \tag{8}$$

It is argued that there is no incentive in this model for s_t to learn to encode any environmental features that cannot influence or are not influenced by the agent's actions. The learned exploration strategy of the agent is therefore robust to uncontrollable aspects of the environment such as the presence of distractor objects, changes in illumination, or other sources of noise in the environment [Pathak et al., 2017].

Forward models are therefore able to learn representations of controllable factors: in order to predict next state, the model must understand the object being controlled. This kind of representation can also be learned through a controllability prior [Jonschkowski et al., 2017].

If a robot acts by applying forces, controllable things should be those whose accelerations correlate with the actions of the robot. Accordingly, a loss function can be defined to minimize the covariance between an action dimension i and the accelerations in the state dimension i. The following formula from [Jonschkowski et al., 2017] makes it explicit:

$$Controllability_i = e^{-cov(a_{t,i}, s_{t+1,i})}, \tag{9}$$

where $cov(a_{t,i}, s_{t+1,i})$ is the covariance between the state $s_{t+1,i}$ at dimension i and time t and $a_{t,i}$ (the action at dimension i that led to such state). Note that here the learned state is assumed to represent an acceleration. Related with this prior also is the notion of *empowerment* [Klyubin et al., 2005], defined as an information-theoretic capacity of an agent's actuation channel to influence its own evolution. The concept of empowerment is related to *accountability* or *agency*, i.e., recognizing when an agent is responsible for originating the change of state in the environment.

3.3 Learning an inverse model

The forward model approach can be turned around and, instead of learning to predict next state (given previous state and action), use current and next states to predict the action between them. The inverse model framework is used in SRL by firstly performing a projection of o_t and o_{t+1} onto learned states s_t and s_{t+1}, and secondly, by predicting the action $\hat{a}_t$ that would explain the transition of s_t into s_{t+1} (Fig. 4). As before, learning this model can impose constraints on the state representation to be able to efficiently predict actions.

An example using inverse models to learn state representations is the Intrinsic Curiosity Module (ICM) [Pathak et al., 2017]. It integrates both an inverse and forward model and the authors argument that using an inverse model is a way to bypass the hard problem of predicting original observations (e.g., pixels in images), since actions have much lower dimension.

A different kind of inverse model is used in [Shelhamer et al., 2017], where the policy gradient algorithm used to learn a controller is augmented with auxiliary gradients from what is called *self-supervised* tasks. In this case, in lack of external supervision, the prediction error resulting from interactions with the environment acts as a self-supervision. They learned a inverse dynamics model to retrieve from o_t and o_{t+1} the action a_t performed between the two successive time steps.

Note that connections among forward and inverse models are important: inverse models can provide supervision to learn representations that the forward model regularizes by learning to predict s_{t+1} [Agrawal et al., 2016]. In practice, this is implemented by decomposing the joint loss function as a sum of the inverse model loss plus the forward model loss [Agrawal et al., 2016]. Conversely, [Zhang et al., 2018] shows in an ablation study that using an inverse model (along with a forward model and an auto-encoder) is the factor that contributes the most to learning a good state representation. Another approach including forward and inverse models, as well as a reconstruction of the observation including multimodal inputs is [Duan, 2017].

3.4 Using feature adversarial learning

Adversarial networks [Goodfellow et al., 2014] can also be used for unsupervised learning of state representations. The use of the Generative Adversarial Network (GAN) framework to learn state

representations is proposed in [Chen et al., 2016]. They present a model named InfoGAN that achieves the disentanglement of latent variables on 3D poses of objects. As described in [Chen et al., 2016], the goal is to learn a generator distribution $P_G(o)$ that matches the real distribution $P_{data}(o)$. Instead of trying to explicitly assign a probability to every o in the data distribution, GANs learn a generator network G that samples from the generator distribution P_G by transforming a noise variable $z \sim P_{noise}(z)$ into a sample $G(z)$. The noise variable has two components. A first one, z_G, randomly sampled from a Gaussian distribution, and a second one with smaller dimension, z_U, sampled from a uniform distribution. The latter is used during training so that the $G(z)$ has a high mutual information with z_U. Then, the sample from z_U has a high correlation with $G(z)$ and can thus be considered as a state representation. This generator is trained by playing against an adversarial discriminator network D that aims at distinguishing between samples from the true distribution P_{data} and the generator distribution P_G. The authors succeed to learn states corresponding to object orientations from sequences of images.

Another example of SRL with Generative Adversarial Networks is presented by [Donahue et al., 2016, Dumoulin et al., 2016]. BiGAN and ALI are extensions of regular GANs to learn the double mapping from image space to latent space, and from latent space to image space. They allow the learned feature representation to be useful for auxiliary supervised discrimination tasks, and competitive with unsupervised and self-supervised feature learning. The BiGAN has also been experimented in [Shelhamer et al., 2017] to learn state representations used for reinforcement learning, but lead to lower performance than their own approach (Section 3.2).

3.5 Exploiting rewards

As opposed to RL, the use of a reward value in SRL is not compulsory. However, it can be used as supplementary information to help differentiating states and to learn task related representations. Rewards are helpful to disentangle meaningful information from a noisy or distracting one, and to tie the representation to a particular task. However, in a multi-task setting, this approach can also be used to learn a generic state representation that is relevant to different tasks.

A *predictable reward prior* which estimates $\hat{r}_{t+1}$ given a state s_t and an action a_t is implemented in [Munk et al., 2016] (along with a forward model) to learn a state for reinforcement learning. Another approach that exploits reward is presented in [Oh et al., 2017]. Besides predicting the reward similarly to [Munk et al., 2016], they also learn to predict the value (discounted sum of future reward) of the next state and exploit this capacity to rely on planning multiple steps for policy learning. The author state that predicting rewards for multiple steps is much easier that predicting observations, while giving the important information for learning a policy.

A dimensionality reduction model called *reward weighted principal component analysis* (rw-PCA), as another way of using rewards for state representation was proposed in [Parisi et al., 2017]. *rwPCA* uses data collected by an RL algorithm and operates a dimensionality reduction strategy which takes reward into account to keep the information into a compressed form. The compressed data is afterwards used to learn a policy.

On the same idea of constructing a task-related representation, [Jonschkowski and Brock, 2015] and [Lesort et al., 2017] use rewards as supplementary information to impose constraints on the state space topology. One of these constraints makes the space more suited to discriminate between states with different rewards. The state space is then particularly adapted to solve a given task. This constraint is called *causality prior* in [Jonschkowski and Brock, 2015] and [Lesort et al., 2017]. It assumes that if we have two different rewards after performing the same action in two different time steps, then the two corresponding states should be differentiated and far away in the representation space (Equation 10).

$$\mathcal{L}_{Caus}(D, \hat{\phi}) = \mathbf{E}[e^{-\|\hat{s}_{t_2} - \hat{s}_{t_1}\|^2} \mid a_{t_1} = a_{t_2}, r_{t_1+1} \neq r_{t_2+1}] \tag{10}$$

3.6 Other objective functions

In this section, we present other approaches assuming various specific constraints for state representation learning. Following [Lake et al., 2016], the learning process can be constrained by prior knowledge (either initially provided by the designer or acquired via learning) to allow the agent to leverage existing common sense, intuitive physics, physical laws, mental states of others, as well as other abstract regularities such as compositionality and causality. This kind of a priori knowledge

is called *prior* [Bengio et al., 2012], [Jonschkowski and Brock, 2015], [Lesort et al., 2017],[Jonschkowski et al., 2017], and is defined through cost functions. These loss functions are applied in the state space in order to impose the required constraints to construct the model projecting the observations in the state space. In the following, $\Delta s_t = s_{t+1} - s_t$ is the difference in between states at times t and $t+1$, and D is a set of observations.

- **Slowness Principle**
  The slowness principle assumes that interesting features fluctuate slowly and continuously through time and that a radical change inside the environment has low probability [Wiskott and Sejnowski, 2002, Kompella et al., 2011].

$$\mathcal{L}_{Slowness}(D, \phi) = \mathbb{E}[\| \Delta\ s_t \|^2] \tag{11}$$

  This assumption can have other naming depending on the unit of s_t, e.g., prior of time coherence (time) [Jonschkowski and Brock, 2015, Lesort et al., 2017] or inertia (velocity) [Jonschkowski et al., 2017].

- **Variability**
  The assumption of this prior is that positions of relevant objects vary, and learning state representations should then focus on moving objects [Jonschkowski et al., 2017].

$$\mathcal{L}_{Variability}(D, \phi) = \mathbb{E}[e^{-\|s_{t1} - s_{t2}\|}] \tag{12}$$

  $e^{-distance}$ is used as a similarity measure that is 1 if the distance among states is 0 and goes to 0 with increasing distance between states. Note that this prior is counter-balancing the slowness prior introduced above as the slowness alone would lead to constant values.

- **Proportionality**
  The proportionality prior introduced in [Jonschkowski and Brock, 2015] assumes that for the same action in different states, the reactions to this action will have proportional amplitude or effect. The representation then vary in the same amount for two equal actions in different situations.

$$\mathcal{L}_{Prop}(D, \phi) = \mathbb{E}[(\| \Delta s_{t_2} \| - \| \Delta s_{t_1} \|)^2 | a_{t_1} = a_{t_2}] \tag{13}$$

- **Repeatability**
  This prior states that two identical actions applied at similar states should provide similar state variations, not only in magnitude but also in direction [Jonschkowski and Brock, 2015].

$$\mathcal{L}_{Rep}(D, \phi) = \mathbb{E}[e^{-\|s_{t_2} - s_{t_1}\|^2} \| \Delta s_{t_2} - \Delta s_{t_1} \|^2 |\ a_{t_1} = a_{t_2}] \tag{14}$$

- **Dynamic verification**
  Dynamic verification [Shelhamer et al., 2017] consists in identifying the corrupted observation o_{t_c} in a history of K observations o_t where $t \in [\![0, K]\!]$. Observations are first encoded into states and the sequence is classified by a learned function f to output the corrupted time step. Negative samples are produced by incorporating observations from a wrong time step into the sequence of images. This discriminative approach forces SRL to encode the dynamics in the states.

- **Selectivity**
  States can be learned by using the idea that factors such as objects correspond to 'independently controllable' aspects of the world that can be discovered by interacting with the environment [Thomas et al., 2017]. Knowing the dimension K of the state space, the aim is to train K policies π_k with $k \in [\![1, K]\!]$. The goal is that the policy π_k causes a change in $s_t^{(k)}$ only, and not in any other feature. To quantify the change in $s_t^{(k)}$ when actions are taken according to π_k, the selectivity of a feature k is:

$$\mathcal{L}_{sel}(D, \phi, k) = \mathbb{E}\left[\frac{\| s_{t+1}^{(k)} - s_t^{(k)} \|}{\sum_{k'} \| s_{t+1}^{(k')} - s_t^{(k')} \|} | s_{t+1} \sim P^a_{s_t, s_{t+1}}\right] \tag{15}$$

where $P^a_{s_t,s_{t+1}}$ is the environment transition distribution from s_t to s_{t+1} under action a. The selectivity of $s_t^{(k)}$ is maximal when only that single feature changes as a result of some action. Maximizing the selectivity improves the disentanglement of controllable factors in order to learn a good state representation.

Table 1: Classification of the reviewed SRL models with regards to the learning objectives they implement and the information they use (actions or rewards)

Model	**Actions/Next state constraints**	**Forward model**	**Inverse model**	**Reconstruct observation**	**Predicts next observation**	**Uses rewards**
AE [Mattner et al., 2012]	no	no	no	yes	no	no
Priors [Jonschkowski and Brock, 2015]	yes	no	no	no	no	yes
PVE [Jonschkowski et al., 2017]	yes	no	no	no	no	no
E2C [Watter et al., 2015]	yes	yes	no	yes	yes	no
ML-DDPG [Munk et al., 2016]	yes	yes	no	no	no	yes
VAE/DAE [van Hoof et al., 2016]	yes	yes	no	yes	yes	no
AE [Finn et al., 2015]	yes	no	no	yes	no	no
DVBF [Karl et al., 2016]	yes	yes	no	yes	yes	no
[Goroshin et al., 2015]	yes	yes	no	yes	yes	no
ICM [Pathak et al., 2017]	yes	yes	yes	no	no	no
[Shelhamer et al., 2017]	yes	no	yes	no	no	no
VPN [Oh et al., 2017]	no	yes	no	no	no	yes
DDM [Assael et al., 2015]	yes	yes	no	yes	yes	no
AE [Wahlström et al., 2015]	yes	yes	no	yes	yes	no
[Thomas et al., 2017]	yes	no	no	yes	no	no
PCA [Curran et al., 2015]	no	no	no	yes	no	no

Model	Actions/Next state constraints	Forward model	Inverse model	Reconstruct observation	Predicts next observation	Uses rewards
PCA [Curran et al., 2016]	no	no	no	yes	no	no
rwPCA [Parisi et al., 2017]	no	no	no	yes	no	yes
[Magrans de Abril and Kanai, 2018]	yes	yes	no	no	no	yes
InfoGAN [Chen et al., 2016]	no	no	no	yes	no	no
BiGAN [Donahue et al., 2016]	no	no	no	yes	no	no
[Duan, 2017]	yes	yes	yes	yes	yes	no
DARLA [Higgins et al., 2016]	no	no	no	yes	no	no
[Zhang et al., 2018]	yes	yes	yes	yes	yes	no
AE [Alvernaz and Togelius, 2017]	no	no	no	yes	no	no
world model [Ha and Schmidhuber, 2018]	yes	yes	no	yes	no	no

3.7 Using hybrid objectives

Reconstruction of data in the observation space, forward models, inverse models, exploitation of rewards, and other objective functions presented in the previous sections are different approaches to tackle the state representation learning challenge. However, these approaches are not incompatible, and models often take advantage of several objective functions at the same time.

For instance, interactively *learning to poke by poking* [Agrawal et al., 2016] is an example of empirical learning of intuitive physics using o_t and o_g as current and goal images, respectively, in order to predict the poke action. The latter is composed by the location, angle and length of the action that sets the object in the state of goal image o_g. Simulations shows that using the inverse model or jointly the inverse and forward models improve performance at pushing objects and that when the training data available is reduced, the joint model outperforms the inverse model with a performance comparable to using a considerably larger amount of data.

The authors in [Finn et al., 2015, Goroshin et al., 2015] use the reconstruction of the observation and the slowness principle in their SRL approach. [Goroshin et al., 2015, van Hoof et al., 2016, Watter et al., 2015, Assael et al., 2015, Karl et al., 2016, Ha and Schmidhuber, 2018] combine the reconstruction of observation and forward models. [Jonschkowski and Brock, 2015, Lesort et al., 2017] take advantage of rewards with a causality prior (Eq. 10) and several other objective functions such as the slowness principle, proportionality, and repeatability to learn state representations. We illustrate, as an example, the combination of objective functions from [Watter et al., 2015] in Figure 6.

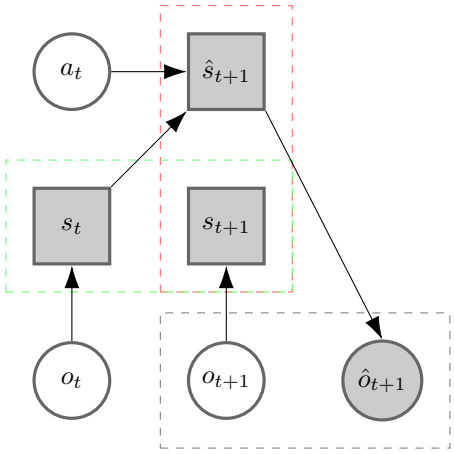

Figure 6: Example of hybrid model (E2C [Watter et al., 2015]) combining a set of previously described *loss term blocks* of previous figures. Several errors are used to learn the state representation using at the same time a forward model error, a reconstruction error and a constraint on the representation that enforces the transition to be linear.

Table 1 summarizes all the reviewed models by showing, for each one, which proxies or surrogate functions have been used for learning: reconstruction of observation, prediction of the future (forward model) and/or retrieving actions (inverse model), and what kind of information is used: action and/or rewards.

4 Building blocks of State Representation Learning

In this section, we cover various implementation aspects relevant to state representation learning and its evaluation. We refer to specific surrogate models, loss function specification tools or strategies that help constraining the information bottleneck and generalizing when learning low-dimensional state representations,

4.1 Learning tools

We first detail a set of models that through an auxiliary objective function, help learning a state representation. One or several of these learning tools can be integrated in broader SRL approaches as was previously described.

4.1.1 Auto-encoders (AE)

Auto-encoders (AE) are a common tool used to learn state representations that are widely used for dimensionality reduction [Hinton et al., 2006, Wang et al., 2012, Wang et al., 2016]. Their objective is to output a reproduction of the input. Its architecture is composed by an encoder and a decoder. The encoder projects the input to a latent space representation (often in lower dimension than the input), which is re-projected to the output afterwards by the decoder. In our problem setting, o_t is the input, s_t the latent representation, and $\hat{o}_t$ is the output. The dimensionality of the latent representation can be chosen depending on the dimension of the state representation we want to learn and enforcing it in such case. The AE will then automatically learn a compact representation by minimizing the reconstruction error between input and output. The usual loss function $\mathcal{L}$ to measure the reconstruction error is the mean squared error (MSE) between input and output, computed pixel-wise. However, it can be any norm.

$$Loss = \mathcal{L}(x, \hat{x}) \tag{16}$$

Auto-encoders are used in different SRL settings [Finn et al., 2015, Mattner et al., 2012]; PCA can also be considered as a particular case of auto-encoder [Curran et al., 2016].

4.1.2 Denoising auto-encoders (DAE)

The main issue of auto-encoders is the risk of finding an easy but not satisfying solution to minimize the pixel reconstruction error. This occurs when the decoder reconstructs a kind of *average* looking dataset. To make the training more robust to this kind of mean optimization solution, denoising auto-encoders (DAE) [Vincent et al., 2008, Vincent et al., 2010] can be used. This architecture adds noise to the input and makes the "average" image a more corrupted solution than the original AE. The DAE architecture is used in [van Hoof et al., 2016] to learn visual and tactile state representations. The authors compared state representations learned by a DAE and a variational auto-encoder (VAE) by using the learned states in a reinforcement learning setting. They found that, in most cases, DAE state representation models gather less rewards than those with VAE state representations.

4.1.3 Variational auto-encoders (VAE)

The SRL literature has also benefited from the variational inference used in variational auto-encoders (VAE) [Kingma and Welling, 2013, Jimenez Rezende et al., 2014] to learn a mapping from observations to state representations. A VAE is an auto-encoder with probabilistic hidden cells: it interprets $\mathcal{S}$ as a set sampled from distribution $P(s_t|o_t)$. It then approximates $P(s_t|o_t)$ with a model $q_\theta(s_t|o_t)$ called the approximate posterior or recognition model. θ represents the parameters of the model, which, for instance, can be a neural network. The VAE also provides a generator which approximates $P(o_t|s_t)$ with a model p_ϕ. ϕ represents the parameters of the generator. Both models p and q are then trained by minimizing the error between o_t and $\hat{o}_t$ and the KL divergence between $q_\theta(s_t|o_t)$ and the normal distribution $\mathcal{N}(\mu = 0, \sigma = \mathbb{I})$ (where μ is the mean of the distribution, σ its covariance matrix and $\mathbb{I}$ the identity matrix). VAE-related models that do not use exactly the original VAE, but variations of it, are [Watter et al., 2015, Assael et al., 2015, Krishnan et al., 2015, van Hoof et al., 2016, Karl et al., 2016, Higgins et al., 2016].

4.1.4 Siamese networks

Siamese networks [Chopra et al., 2005] consist of two or more identical networks that share their parameters, i.e., have the exact same weights. The objective of the siamese architecture is not to classify input data, but to differentiate between the inputs (*same* versus *different* class or condition, for example). This kind or architecture is useful to impose constraints in the latent space of a neural network. For example it can be used to learn similarity metrics or time dependencies, as it is done in time-contrastive networks [Sermanet et al., 2017].

In the context of SRL, siamese networks can be employed to implement some priors previously presented in Section 3.6. For example, two siamese networks can be used to compute a similarity loss and optimize the slowness principle (or temporality prior) between s_t and s_{t+1} as in [Lesort et al., 2017]. In [Goroshin et al., 2015] they use three siamese networks to compute three consecutive states at the same time that are fed into another model that predicts the next state.

4.2 Observation/action spaces

This section presents a summary of the dimensionality of the observation, state and action spaces, as well as the applications in which the reviewed papers are evaluated (Table 2). The continuity or discreetness of the action space is also shown. These are good proxies to assess the complexity of the problem tackled: the higher the dimensionality of observation and action, as well as the smaller we want the dimension of the state to be, the harder is the task of learning a state representation because much more information will need to be processed and filtered in order to keep only the information that is substantial.

We note also that the reviewed literature often presents results with presumably higher dimensionality of learned states than theoretically needed (e.g. using state of dimension 6 for a 2 joints robotic arm). The dimensionality of the state may seem obvious when we are learning a state that should (according to the task) correlate with a clear dimension (position, distance, angle) in the environment. However, deciding the dimensionality of the state space is not always trivial when we are learning more abstract states with no clear dimension associated to it. For instance, visually representing the state associated to an Atari game scene in a complex situation is not as easy to interpret nor assess in comparison to the dimensionality of states associated to a position of an arm, its angle or velocity. Indeed, since the learning objective we reviewed are just proxies to

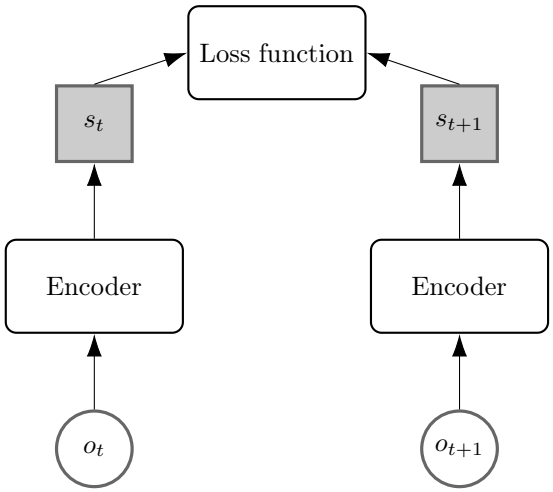

Figure 7: Representation of siamese encoders: networks with tied (shared) parameters. In SRL, siamese networks allow to set constraints via loss functions among several states, e.g. in robotic priors.

guide state representation learning, they can lead to something different that the ideal and minimal state representation. In particular, increasing the capacity of the model by augmenting the dimensionality of the state above the dimension of the true state can lead to a better optimization of the learning objectives.

Table 2: Settings of each approach: characteristics of each environment and state representation dimensions

Reference	**Observation Dimension**	**State Dimension**	**Action Dimension**	**Environment**	**Data**
Priors [Jonschkowski and Brock, 2015]	16*16*3	2	25 discrete	Slot cars, mobile robot localization	Raw images
PVE[Jonschkowski et al., 2017]	Unavailable	5	Discrete	Inverted pendulum, ball in cup, cart-pole	Raw images
E2C [Watter et al., 2015]	40*40*3	8	Discrete	Agent with obstacle	Raw images
E2C [Watter et al., 2015]	48*48*3	8	Discrete	Inverted pendulum	Raw images
E2C [Watter et al., 2015]	80*80*3	8	Discrete	Cart-pole	Raw images
E2C [Watter et al., 2015]	128*128*3	8	Discrete	3 link arm	Raw images
[van Hoof et al., 2016]	20*20*3	3	Continuous	Pendulum swing-up	Raw images
[van Hoof et al., 2016]	228	3	Continuous	Real-robot manipulation task	Tactile data
ML-DDPG [Munk et al., 2016]	18 or 24	6	2 discrete	2 link arm	Joint position
ML-DDPG [Munk et al., 2016]	192 or 308	96	36 discrete	Octopus	Joint position

Reference	Observation Dimension	State Dimension	Action Dimension	Environment	Data
[Finn et al., 2015]	240*240*3	32	Continuous	Robotics manipulation tasks	Raw images
DVBF [Karl et al., 2016]	16*16*3	3	Unavailable	Pendulum	Raw images
DVBF [Karl et al., 2016]	16*16*3	2	Unavailable	Bouncing ball	Raw images
DVBF [Karl et al., 2016]	16*16*3	12	Unavailable	2 bouncing balls	Raw images
[Goroshin et al., 2015]	3 frames of 32*32	2	2 discrete	NORB dataset	Raw images
ICM [Pathak et al., 2017]	42*42*3	3	4 discrete	3D VizDoom navigation game	Raw images
ICM [Pathak et al., 2017]	42*42*3	2	14 discrete	Mario Bros	Raw images
[Shelhamer et al., 2017]	Unavailable	Un.	Unavailable	Atari	Raw images
VPN [Oh et al., 2017]	3*10*10	Un.	4 discrete	2D navigation	Raw images
VPN [Oh et al., 2017]	4*84*84	Un.	4 discrete	Atari	Raw images
[Curran et al., 2016]	Unavailable	4	5 discrete	Mountain car 3D	Unavailable
[Curran et al., 2016]	Unavailable	12	243 discrete	6 link swimmer	Unavailable
[Higgins et al., 2017]	64*64*3	64	8 discrete	Mujoco, Jaco arm	Raw images
[Higgins et al., 2017]	84*84*3	64	99 discrete	DeepMind Lab	Raw images
rwPCA [Parisi et al., 2017]	21*21*3	Auto.	2 continuous	Picking a coin and putting it on a goal	Raw images
[Parisi et al., 2017]	20	Auto.	2 continuous	Hit a ball with a ping-pong paddle	Position of objects
[Magrans de Abril and Kanai, 2018]	40	2	Continuous	Explore a 3 room simulated 2D map	Position of agent
[Duan, 2017]	240*240*4	512	4 continuous	Poking cube	Raw images + depth
[Zhang et al., 2018]	10*10*9	256	4 discrete	Maze	feature vector
[Zhang et al., 2018]	Unavailable	200	continuous	Mujoco	joints

4.3 Evaluating learned state representations

This section provides a review of validation metrics and embedding quality evaluation techniques used across the literature. These are summarized in Table 3.

The most common way of evaluating the quality of the learned state space is by letting an agent use the states to learn a control task, and thus assessing whether the representation is general enough to be transferable. This method is for example applied to evaluate the performance of an SRL algorithm using reinforcement learning [Jonschkowski and Brock, 2015, Jonschkowski et al., 2017, Munk et al., 2016, van Hoof et al., 2016, Finn et al., 2015, Pathak et al., 2017, Shelhamer et al., 2017, Oh et al., 2017, Parisi et al., 2017, Assael et al., 2015].

Table 3: Evaluation methods for state representation learning and their respective objective.

Metric	**Evaluation objective**
Task performance [Jonschkowski and Brock, 2015, Jonschkowski et al., 2017, Munk et al., 2016, van Hoof et al., 2016, Finn et al., 2015, Pathak et al., 2017, Shelhamer et al., 2017, Oh et al., 2017, Parisi et al., 2017, Assael et al., 2015, Higgins et al., 2016, Zhang et al., 2018, Alvernaz and Togelius, 2017]	Assesses that the information needed to solve a task is contained in s_t. The evaluation is done in reinforcement learning algorithms.
Disentanglement metric score [Higgins et al., 2016]	Measures the disentanglement of the latent factors. It assumes that generative factors are known and interpretable. Used in transfer learning, object recognition.
Distortion [Indyk, 2001]	Measures the preservation of local and global geometry coherence in unsupervised representation learning.
NIEQA (Normalization Independent Embedding Quality Assessment) [Zhang et al., 2012]	Measures the local & global neighborhood embedding quality assessment; not limited to isometric embeddings. Used in manifold learning.
KNN-MSE [Lesort et al., 2017]	Measures the degree of preservation of the same neighbors in between the latent space and the ground truth.
Supervised regression [Jonschkowski et al., 2017]	Evaluates the performance of a supervised regression between the learned states and the ground truth.

However, this approach is often very costly and inefficient in terms of time, computation and data. Also, various state-of-the-art RL algorithms may be applied to learn a policy and may result in very different performances for a given state representation. The uncertainty inherent to RL therefore makes RL algorithms sufficient but not practical nor appropriate to be a necessary condition to validate a particular state representation. In consequence, it would be desirable to have an intermediate manner to assess the representation that is independent of the algorithm applied to complete the task and there are, indeed, several more direct ways to assess the learned state space. For example, visual assessment of the representation's quality can be done using a Nearest-Neighbors approach as in [Sermanet et al., 2017, Pinto et al., 2016]. The idea is to look at the nearest neighbors in the learned state space, and for each neighbor, retrieve their corresponding observation. Visual inspection can then reveal if these two observations indeed correspond to nearest neighbors in the ground truth state space $\tilde{s}$ we intend to learn.

While the nearest neighbor coherence can be assessed visually, KNN-MSE is a quantitative metric derived from this qualitative information [Lesort et al., 2017]. Using the ground truth state value for every observation, KNN-MSE measures the distance between the value of an observation and the value of the nearest neighbor observations retrieved in the learned state space. A low distance means that a neighbor in the ground truth is still a neighbor in the learned representation, and thus, local coherence is conserved.

For an observation o, KNN-MSE is computed using its associated learned state $s = \phi(o)$ as follows:

$$\text{KNN-MSE}(s) = \frac{1}{k} \sum_{s' \in KNN(s,k)} ||\tilde{s} - \tilde{s}'||^2 \tag{17}$$

where $\text{KNN}(s, k)$ returns the k nearest neighbors of s (chosen with the Euclidean distance) in the learned state space $\mathcal{S}$, $\tilde{s}$ is the ground truth associated to s, and $\tilde{s}'$ is the ground truth associated to s'.

One of the characteristics that a good representation should possess is to produce a disentangled representation of variation factors. The evaluation of these characteristics can be done using the selectivity prior (see Section 3.6 and Eq. 15) from [Thomas et al., 2017]. This prior cares about the independence among variations of the representation under each action. However, it is applicable mainly if actions are known to be independent.

Another way to quantitatively compare the degree of disentanglement reached by a model is using the disentanglement metric score [Higgins et al., 2016]. It assumes that the data is generated by a process in which the generative factors are known, interpretable, and that some are conditionally independent. In order to measure the disentanglement, it uses a simple low-capacity and low VC-dimension linear classifier's accuracy (reported as *disentanglement metric score*). The classifier's goal is to predict the generative factor that was kept fixed for a given difference between pairs of representations from the same latent factor.

Other metrics from the area of manifold learning can be used, such as distortion [Indyk, 2001] and NIEQA [Zhang et al., 2012]; both share the same principle as two quantitative measures of the global quality of a representation: the representation space should, as much as possible, be an undistorted version of the original space.

Distortion [Indyk, 2001] gives insight of the quality of a representation by measuring how the local and global geometry coherence of the representation changes with respect to the ground truth. It was designed in the *embeddings* context as a natural and versatile paradigm for solving problems over metric spaces.

NIEQA (Normalization Independent Embedding Quality Assessment) [Zhang et al., 2012] is a more complex evaluation than distortion that measures the local geometry quality and the global topology quality of a representation. NIEQA local part checks if the representation is locally equivalent to an Euclidean subspace that preserves the structure of local neighborhoods. NIEQA objectives are aligned with KNN-MSE [Lesort et al., 2017], as a measure to assess the quality of the representation, especially locally. The global NIEQA measure is also based on the idea of preserving original structure in the representation space, but instead of looking at the neighbors, it samples "representative" points in the whole state space. Then, it considers the preservation of the geodesic distance between those points in the state space.

One last mechanism to assess SRL methods is to use supervised learning to learn a regression from the learned representation to its ground-truth [Jonschkowski et al., 2017]. The training and test sets are separated into two datasets to evaluate if the regression can generalize to unseen states. The assumption is that this regression measures how well meaningful features are encoded in the state. A good generalization would show a good encoding.

4.4 Evaluation scenarios

Datasets used to validate state representation learning include varied, but mainly simulated, environments because they are easier to reproduce and generate. Unlike in image recognition challenges where MNIST digits or ImageNet datasets prevail, in state representation learning, a varied set of regular video games or visuomotor tasks in robotics can be found as a test suite for robotics control. Examples of simulated environments include, among others:

- Pendulum (Inverted or classical): The goal is to represent the state of the pendulum [Watter et al., 2015, Jonschkowski et al., 2017, van Hoof et al., 2016, Mattner et al., 2012]. The pendulum starts in a random position, and the objective is to swing it up so it stays upright (there is no specified reward threshold at which the task is considered solved).

- Cart-Pole: consists of an inverted pendulum attached to a cart which moves along a frictionless track; the system is controlled applying +1 or -1 force to the cart, and a reward of +1 is provided for every time step that the pole remains upright. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center[1] ([Watter et al., 2015], [Jonschkowski et al., 2017]).

- Atari games [Bellemare et al., 2013]: mostly low (2D) dimensional simulated environments with different agents and goals. In these games, states can be represented through different variables (time in achieving a task, amount of bonus, keeping alive, etc.) [Shelhamer et al., 2017, Oh et al., 2017].

- More advanced test games include *VizDoom*, where the levels passed, reward accumulated and exploration levels are used as evaluation metrics [Pathak et al., 2017, Alvernaz and Togelius, 2017]. Likewise, Mario Benchmark [Karakovskiy, 2012] is a platform designed for reinforcement learning based on the "Super Mario Bros" video game. This test suite is for example experimented in [Curran et al., 2016, Pathak et al., 2017].

[1] https://github.com/openai/gym/wiki/Leaderboard#pendulum-v0

- Other evaluation benchmarks tested in the reviewed works in this survey include simulated octopus arms [Engel et al., 2006, Munk et al., 2016], labyrinths [Thomas et al., 2017], navigation grids [Magrans de Abril and Kanai, 2018, Oh et al., 2017], driving cars [Jonschkowski and Brock, 2015], or *mountain car* scenarios [Curran et al., 2016]. Another example is the *bouncing ball*, where the goal is to learn a representation of one bouncing ball position in 2D (x,y) [Karl et al., 2016].

- In the robotics domain we can find benchmarks on robot manipulation skills [Finn et al., 2015, van Hoof et al., 2016] such as Baxter pushing a button [Lesort et al., 2017], grasping [Finn et al., 2015], stabilizing [van Hoof et al., 2016], poking objects [Agrawal et al., 2016, Duan, 2017] or balancing a real pendulum [Mattner et al., 2012]. Nevertheless, some approaches achieve to learn in real environment scenarios, for instance, with mobile robots that explore an arena [Jonschkowski and Brock, 2015].

Many of the latter simulated scenarios are part of Universe and OpenAI Gym [Brockman et al., 2016] or DeepMind Labs [Beattie et al., 2016]. These benchmarking tasks used in the most prominent state representation learning literature are summarized in Table 2.

5 Discussion and future trends

In this section, we first discuss the implications of SRL for autonomous agents and the assessment, comparison and reproducibility of the representation learned. Finally, we explore the consequences of SRL on the interpretability of machine learning algorithms.

5.1 SRL models for autonomous agents

SRL methods provide unsupervised tools for autonomous agents to learn representations about the environment without extra annotations. They need, however, that the agent gathers data to learn. Therefore, the role of environment exploration is an important dimension to investigate in SRL. If the space is not sufficiently explored by the agent, acquisition of varied observations and exposure to actions that lead to optimal performance can be hindered [Pathak et al., 2017].

One way to incorporate exploration in SRL is to integrate curiosity or intrinsic motivations [Oudeyer et al., 2007] in the algorithm that collects data. The overall idea of this approach is to complement the extrinsic reward by an intrinsic reward that favors states where SRL makes the most progress. This is done for example in the Intrinsic Curiosity Module (ICM) [Pathak et al., 2017] by defining an intrinsic reward linked to the forward model error which encourages exploration. This approach is improved in [Magrans de Abril and Kanai, 2018] by balancing this exploratory behavior with an homeostatic drive to also favor actions that lead to familiar state-action pairs. The reverse question of how the learned state space can influence the performance of intrinsic motivation approaches [Péré et al., 2018] is also relevant. The automatic exploration, designed to maximize the quality of a learned state representation, is a field to be further explored in order to build high quality representations.

Another approach to gather enough relevant data could be to perform data augmentation by adding data from simulation; however, the problem is to make the model benefit from simulation data for real life applications, a problem that is known as the *reality gap* [Mouret et al., 2013]. Nevertheless, using both kinds of data was shown to improve results in particular applications [Bousmalis et al., 2017]. An interesting research direction is therefore to study how to exploit simulation data to improve SRL for real world applications.

Another problem to ultimately perform SRL autonomously (i.e., without manual parameter tuning) is the choice of the state representation dimension, which is made empirically in most reviewed approaches. The challenge of deciding the dimensionality automatically can be related to a bias-variance trade-off, as the dimensionality of the representation constrains the capacity of the model. Indeed, increasing the states dimension augments the capacity of the model, which, as a result, will be better at reducing the training error, but also leads to overfitting. As discussed in Section 4.2, learning criteria can be better optimized by models with large capacity, and thus, an automatic process is prone to over estimate the dimension needed.

To avoid choosing manually the state dimension, it is possible to choose automatically a number of features from a larger set such that they have a certain variance and are orthogonal [Parisi et al., 2017]. It can be done by using PCA to produce a set of features in which the most significant ones

are selected with respect to the chosen variance. PCA can also be modified to select reward related components [Parisi et al., 2017]. Although the variance has to be fixed a priori, the authors claim that this is usually easier than choosing the state dimension. Extending this technique to other state representation approach could be an interesting research direction.

5.2 Assessment, comparison and reproducibility in SRL

The assessment challenge of SRL is two-sided. First, there is no easy nor certified way for validating a learned representation. Secondly, the lack of common evaluation frameworks makes a fair comparison between approaches difficult.

As mentioned in Section 4.3, the most objective method to evaluate the quality of representations is to check if the state representation learned can be used by an RL algorithm to solve a task more efficiently. However, this assessment is uncertain and unstable given the stochasticity of reinforcement learning algorithms [Henderson et al., 2017]. Moreover, it is not obvious which RL algorithm is the best choice, and thus, several should be used in the comparison. In practice, a large amount of policy evaluation runs is therefore required in order to provide a robust assessment, which is possible in simulation but is seldom applicable on real robots, given the robots fragility and experimentation time involved. In this case, it is therefore interesting to use several of the other measures presented in section 4.3, that only give partial information on the state representation quality, but are possible to apply for a comparison with a ground truth state.

Comparing approaches from published results is also particularly hard because of the high variability of the environments and data used in the different approaches (as illustrated in Table 2). This points to the need of an evaluation framework incorporating several tasks and several evaluation metrics similar to the ones proposed for reinforcement learning such as the *DeepMind Control Suite* [Tassa et al., 2018]. Reproducibility guidelines with proper experimental techniques and reporting procedures, as pointed in [Henderson et al., 2017] for RL, should also be defined for SRL. In the mean time, as there is not yet an ideal method for state representation assessment, researchers should at least make public their simulation environment (with possible ground truth), and use simulation settings from other approaches to provide fairer comparisons and facilitate the method reproducibility. Furthermore, we strongly encourage authors to entirely describe their experiments, in particular, report their data and models' characteristics.

5.3 Providing interpretable systems

In 2018, European Union regulations on algorithmic decision-making include a "right to explanation", "right to opt-out" and "non discrimination" of models [Goodman and Flaxman, 2016]. Artificial intelligence research is thus granted with an opportunity to further provide meaningful explanations to why algorithms work the way they do. The interpretability of results in machine learning is however a challenging problem that needs proper definition [Lipton, 2016]. In any case, monitoring the degree to which AI systems show the same thinking flows as humans is invaluable and crucial; not only to explain how human cognition works, but also to help AI make better and more fair decisions [Lake et al., 2016].

We define interpretability in the SRL context as the capacity for a human to be able to link a variation in the representation to a variation in the environment, and be able to know why the representation was sensitive to this variation. As SRL is designed to be able to give this level of interpretability, it could help improving the understanding we have about learning algorithms' output. Indeed, the higher the dimension, the less interpretable the result is for humans. Therefore, the dimensionality reduction induced by SRL, coupled with the link to the control and possible disentanglement of variation factors, could be highly beneficial to improve our understanding capacity of the decisions made by algorithms using this state representation.

6 Conclusion

State Representation Learning algorithms are designed to find a way to compress high-dimensional observation data into a low and meaningful dimensional space for controlled systems. These models only require the agent's observations, its performed actions and, optionally, the reward of the associated task.

This work aims at presenting an accessible guide to learn about SRL approaches, the existing tools for evaluation and the common simulation settings used as benchmark. We presented the learning objectives of the state of the art approaches on SRL and their resemblances and differences. We discussed afterwards the use of SRL for autonomous agents, the difficulties for comparing existing approaches and the interpretability of results.

A general advice when building SRL models would be to integrate as many learning objectives as possible, depending on the available data. As an example, one could use a reconstruction objective for linking the state space to the observations, combined with a predictive objective (forward model) to capture dynamics, and a reward-based objective to apprehend the effects of actions performed. More general priors could also be added to force the state space to be coherent and understandable for humans. While many models integrate several of these objectives, no proposed model currently includes all of them together.

As SRL is designed to automatically learn representations from a set of unlabeled observations, it could be used in future work to learn from evolving environments and could be a step towards continual or lifelong learning. Another area to explore in the future is the integration of exploration strategies for data collection specifically designed to be able to improve the state representation learned.

7 Acknowledgements

This research is funded by the DREAM project under the European Union's Horizon 2020 research and innovation program under grant agreement No 640891. We acknowledge Olivier Sigaud, Antonin Raffin, Cynthia Liem and other colleagues for insightful and detailed feedback.

References

[Achille and Soatto, 2017] Achille, A. and Soatto, S. (2017). A Separation Principle for Control in the Age of Deep Learning. *ArXiv e-prints*.

[Agrawal et al., 2016] Agrawal, P., Nair, A., Abbeel, P., Malik, J., and Levine, S. (2016). Learning to poke by poking: Experiential learning of intuitive physics. *CoRR*, abs/1606.07419.

[Alvernaz and Togelius, 2017] Alvernaz, S. and Togelius, J. (2017). Autoencoder-augmented Neuroevolution for Visual Doom Playing. *ArXiv e-prints*.

[Assael et al., 2015] Assael, J.-A. M., Wahlström, N., Schön, T. B., and Deisenroth, M. P. (2015). Data-efficient learning of feedback policies from image pixels using deep dynamical models. *NIPS Deep Reinforcement Learning Workshop*.

[Beattie et al., 2016] Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Legg, S., and Petersen, S. (2016). Deepmind lab. *CoRR*, abs/1612.03801.

[Bellemare et al., 2013] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents.

[Bengio et al., 2012] Bengio, Y., Courville, A. C., and Vincent, P. (2012). Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538.

[Bohg et al., 2017] Bohg, J., Hausman, K., Sankaran, B., Brock, O., Kragic, D., Schaal, S., and Sukhatme, G. S. (2017). Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291.

[Bousmalis et al., 2017] Bousmalis, K., Irpan, A., Wohlhart, P., Bai, Y., Kelcey, M., Kalakrishnan, M., Downs, L., Ibarz, J., Pastor, P., Konolige, K., et al. (2017). Using simulation and domain adaptation to improve efficiency of deep robotic grasping. *arXiv preprint arXiv:1709.07857*.

[Brockman et al., 2016] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *CoRR*, abs/1606.01540.

[Böhmer et al., 2015] Böhmer, W., Springenberg, J. T., Boedecker, J., Riedmiller, M., and Obermayer, K. (2015). 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*, pages 1–10.

[Chen et al., 2016] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180.

[Chopra et al., 2005] Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1.

[Conti et al., 2017] Conti, E., Madhavan, V., Petroski Such, F., Lehman, J., Stanley, K. O., and Clune, J. (2017). Improving Exploration in Evolution Strategies for Deep Reinforcement Learning via a Population of Novelty-Seeking Agents. *ArXiv e-prints*.

[Curran et al., 2016] Curran, W., Brys, T., Aha, D., Taylor, M., and Smart, W. D. (2016). Dimensionality reduced reinforcement learning for assistive robots.

[Curran et al., 2015] Curran, W., Brys, T., Taylor, M., and Smart, W. (2015). Using PCA to Efficiently Represent State Spaces. *ArXiv e-prints*.

[Deisenroth and Rasmussen, 2011] Deisenroth, M. P. and Rasmussen, C. E. (2011). Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 465–472, USA. Omnipress.

[Donahue et al., 2016] Donahue, J., Krähenbühl, P., and Darrell, T. (2016). Adversarial feature learning. *CoRR*, abs/1605.09782.

[Duan, 2017] Duan, W. (2017). Learning state representations for robotic control. *M. Thesis*.

[Dumoulin et al., 2016] Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., and Courville, A. (2016). Adversarially learned inference. *arXiv preprint arXiv:1606.00704*.

[Engel et al., 2006] Engel, Y., Szabo, P., and Volkinshtein, D. (2006). Learning to control an octopus arm with gaussian process temporal difference methods. In *Advances in neural information processing systems*, pages 347–354.

[Finn et al., 2015] Finn, C., Tan, X. Y., Duan, Y., Darrell, T., Levine, S., and Abbeel, P. (2015). Deep spatial autoencoders for visuomotor learning. *CoRR*, abs/1509.06113.

[Fodor, 2002] Fodor, I. K. (2002). A survey of dimension reduction techniques. Technical report, Lawrence Livermore National Lab., CA (US).

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. pages 2672–2680.

[Goodman and Flaxman, 2016] Goodman, B. and Flaxman, S. (2016). European union regulations on algorithmic decision-making and a" right to explanation". *arXiv preprint arXiv:1606.08813*.

[Goroshin et al., 2015] Goroshin, R., Mathieu, M., and LeCun, Y. (2015). Learning to linearize under uncertainty. *CoRR*, abs/1506.03011.

[Ha and Schmidhuber, 2018] Ha, D. and Schmidhuber, J. (2018). World Models. *ArXiv e-prints*.

[Henderson et al., 2017] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2017). Deep reinforcement learning that matters. *CoRR*, abs/1709.06560.

[Higgins et al., 2016] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2016). beta-vae: Learning basic visual concepts with a constrained variational framework.

[Higgins et al., 2017] Higgins, I., Pal, A., Rusu, A. A., Matthey, L., Burgess, C. P., Pritzel, A., Botvinick, M., Blundell, C., and Lerchner, A. (2017). DARLA: Improving Zero-Shot Transfer in Reinforcement Learning. *ArXiv e-prints*.

[Hinton et al., 2006] Hinton, G. E., Osindero, S., and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554.

[Indyk, 2001] Indyk, P. (2001). Algorithmic applications of low-distortion geometric embeddings. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 10–33. IEEE.

[Jimenez Rezende et al., 2014] Jimenez Rezende, D., Mohamed, S., and Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *ArXiv e-prints*.

[Jonschkowski and Brock, 2015] Jonschkowski, R. and Brock, O. (2015). Learning state representations with robotic priors. *Auton. Robots*, 39(3):407–428.

[Jonschkowski et al., 2017] Jonschkowski, R., Hafner, R., Scholz, J., and Riedmiller, M. A. (2017). PVEs: Position-Velocity Encoders for Unsupervised Learning of Structured State Representations. *CoRR*, abs/1705.09805.

[Karakovskiy, 2012] Karakovskiy, Sergey, T. J. (2012). The Mario AI benchmark and competitions. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):55–67.

[Karl et al., 2016] Karl, M., Soelch, M., Bayer, J., and van der Smagt, P. (2016). Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data. *ArXiv e-prints*.

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-Encoding Variational Bayes. *ArXiv e-prints*.

[Klyubin et al., 2005] Klyubin, A. S., Polani, D., and Nehaniv, C. L. (2005). Empowerment: A universal agent-centric measure of control. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 128–135. IEEE.

[Kompella et al., 2011] Kompella, V. R., Luciw, M. D., and Schmidhuber, J. (2011). Incremental slow feature analysis: Adaptive and episodic learning from high-dimensional input streams. *CoRR*, abs/1112.2113.

[Krishnan et al., 2015] Krishnan, R. G., Shalit, U., and Sontag, D. (2015). Deep Kalman Filters. *ArXiv e-prints*.

[Lake et al., 2016] Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2016). Building Machines That Learn and Think Like People. *ArXiv e-prints*.

[Lesort et al., 2017] Lesort, T., Seurin, M., Li, X., Díaz-Rodríguez, N., and Filliat, D. (2017). Unsupervised state representation learning with robotic priors: a robustness benchmark. *CoRR*, abs/1709.05185.

[Lillicrap et al., 2015] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971.

[Lipton, 2016] Lipton, Z. C. (2016). The Mythos of Model Interpretability. *ArXiv e-prints*.

[Magrans de Abril and Kanai, 2018] Magrans de Abril, I. and Kanai, R. (2018). Curiosity-driven reinforcement learning with homeostatic regulation. *ArXiv e-prints*.

[Mattner et al., 2012] Mattner, J., Lange, S., and Riedmiller, M. A. (2012). Learn to swing up and balance a real pole based on raw visual input data. In *Neural Information Processing - 19th International Conference, ICONIP 2012, Doha, Qatar, November 12-15, 2012, Proceedings, Part V*, pages 126–133.

[Mnih et al., 2016] Mnih, V., Agapiou, J., Osindero, S., Graves, A., Vinyals, O., Kavukcuoglu, K., et al. (2016). Strategic attentive writer for learning macro-actions. *arXiv preprint arXiv:1606.04695*.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

[Mouret et al., 2013] Mouret, J., Koos, S., and Doncieux, S. (2013). Crossing the reality gap: a short introduction to the transferability approach. *CoRR*, abs/1307.1870.

[Munk et al., 2016] Munk, J., Kober, J., and Babuska, R. (2016). Learning state representation for deep actor-critic control. In *Proceedings of the 55th Conference on Decision and Control (CDC)*, pages 4667–4673. IEEE.

[Oh et al., 2017] Oh, J., Singh, S., and Lee, H. (2017). Value Prediction Network. *ArXiv e-prints*.

[Oudeyer et al., 2007] Oudeyer, P.-Y., Kaplan, F., and Hafner, V. (2007). Intrinsic motivation systems for autonomous mental development. *Evolutionary Computation, IEEE Transactions on*, 11(2):265–286.

[Parisi et al., 2017] Parisi, S., Ramstedt, S., and J., P. (2017). Goal-driven dimensionality reduction for reinforcement learning. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*.

[Pathak et al., 2017] Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *ICML*.

[Pinto et al., 2016] Pinto, L., Gandhi, D., Han, Y., Park, Y., and Gupta, A. (2016). The curious robot: Learning visual representations via physical interactions. *CoRR*, abs/1604.01360.

[Péré et al., 2018] Péré, A., Forestier, S., Oudeyer, P.-Y., and Sigaud, O. (2018). Unsupervised learning of goal spaces for intrinsically motivated goal exploration.

[Rusu et al., 2016] Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. *CoRR*, abs/1606.04671.

[Sermanet et al., 2017] Sermanet, P., Lynch, C., Hsu, J., and Levine, S. (2017). Time-contrastive networks: Self-supervised learning from multi-view observation. *CoRR*, abs/1704.06888.

[Shelhamer et al., 2017] Shelhamer, E., Mahmoudieh, P., Argus, M., and Darrell, T. (2017). Loss is its own reward: Self-supervision for reinforcement learning. *arXiv preprint arXiv:1612.07307*.

[Stulp and Sigaud, 2013] Stulp, F. and Sigaud, O. (2013). Robot Skill Learning: From Reinforcement Learning to Evolution Strategies. *Paladyn*, 4(1):49–61.

[Sutton, 1998] Sutton, R. S. (1998). *Introduction to reinforcement learning*, volume 135.

[Tassa et al., 2018] Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., and Riedmiller, M. (2018). DeepMind Control Suite. *ArXiv e-prints*.

[Thomas et al., 2017] Thomas, V., Pondard, J., Bengio, E., Sarfati, M., Beaudoin, P., Meurs, M., Pineau, J., Precup, D., and Bengio, Y. (2017). Independently controllable factors. *CoRR*, abs/1708.01289.

[van Hoof et al., 2016] van Hoof, H., Chen, N., Karl, M., van der Smagt, P., and Peters, J. (2016). Stable reinforcement learning with autoencoders for tactile and visual data. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3928–3934.

[Vincent et al., 2008] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1096–1103, New York, NY, USA. ACM.

[Vincent et al., 2010] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408.

[Wahlström et al., 2015] Wahlström, N., Schön, T. B., and Deisenroth, M. P. (2015). From Pixels to Torques: Policy Learning with Deep Dynamical Models. *ArXiv e-prints*.

[Wang et al., 2012] Wang, J., He, H., and Prokhorov, D. V. (2012). A folded neural network autoencoder for dimensionality reduction. In Chan, J. H. and Tan, A.-H., editors, *INNS-WC*, volume 13 of *Procedia Computer Science*, pages 120–127. Elsevier.

[Wang et al., 2016] Wang, Y., Yao, H., and Zhao, S. (2016). Auto-encoder based dimensionality reduction. *Neurocomput.*, 184(C):232–242.

[Watter et al., 2015] Watter, M., Springenberg, J., Boedecker, J., and Riedmiller, M. (2015). Embed to control: A locally linear latent dynamics model for control from raw images. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2746–2754. Curran Associates, Inc.

[Wiskott and Sejnowski, 2002] Wiskott, L. and Sejnowski, T. J. (2002). Slow feature analysis: Unsupervised learning of invariances. *Neural Comput.*, 14(4):715–770.

[Yang et al., 2017] Yang, X., Ramesh, P., Chitta, R., Madhvanath, S., Bernal, E. A., and Luo, J. (2017). Deep multimodal representation learning from temporal data. *CoRR*, abs/1704.03152.

[Zhang et al., 2018] Zhang, A., Satija, H., and Pineau, J. (2018). Decoupling dynamics and reward for transfer learning. In *Proceedings of the 6th International Conference on Learning Representations (ICLR) workshops*.

[Zhang et al., 2012] Zhang, P., Ren, Y., and Zhang, B. (2012). A new embedding quality assessment method for manifold learning. *Neurocomputing*, 97:251–266.