Université de Montréal

Unsupervised Representation Learning in Interactive Environments

par Evan Racah

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des arts et des sciences
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique

August, 2019

Résumé

Extraire une représentation de tous les facteurs de haut niveau de l'état d'un agent à partir d'informations sensorielles de bas niveau est une tâche importante, mais difficile, dans l'apprentissage automatique. Dans ce memoire, nous explorerons plusieurs approches non supervisées pour apprendre ces représentations. Nous appliquons et analysons des méthodes d'apprentissage de représentations non supervisées existantes dans des environnements d'apprentissage par renforcement, et nous apportons notre propre suite d'évaluations et notre propre méthode novatrice d'apprentissage de représentations d'état.

Dans le premier chapitre de ce travail, nous passerons en revue et motiverons l'apprentissage non supervisé de représentations pour l'apprentissage automatique en général et pour l'apprentissage par renforcement. Nous introduirons ensuite un sous-domaine relativement nouveau de l'apprentissage de représentations : l'apprentissage auto-supervisé. Nous aborderons ensuite deux approches fondamentales de l'apprentissage de représentations, les méthodes génératives et les méthodes discriminatives. Plus précisément, nous nous concentrerons sur une collection de méthodes discriminantes d'apprentissage de représentations, appelées méthodes contrastives d'apprentissage de représentations non supervisées (CURL). Nous terminerons le premier chapitre en détaillant diverses approches pour évaluer l'utilité des représentations.

Dans le deuxième chapitre, nous présenterons un article de workshop dans lequel nous évaluons un ensemble de méthodes d'auto-supervision standards pour les problèmes d'apprentissage par renforcement. Nous découvrons que la performance de ces représentations dépend fortement de la dynamique et de la structure de l'environnement. À ce titre, nous déterminons qu'une étude plus systématique des environnements et des méthodes est nécessaire.

Notre troisième chapitre couvre notre deuxième article, Unsupervised State Representation Learning in Atari, où nous essayons d'effectuer une étude plus approfondie des méthodes d'apprentissage de représentations en apprentissage par renforcement, comme expliqué dans le deuxième chapitre. Pour faciliter une évaluation plus approfondie des représentations en apprentissage par renforcement, nous introduisons une suite de 22 jeux Atari entièrement labellisés. De plus, nous choisissons de comparer les méthodes d'apprentissage de représentations de façon plus systématique, en nous concentrant sur une comparaison entre méthodes génératives et méthodes contrastives, plutôt que les méthodes générales du deuxième chapitre choisies de façon moins systématique. Enfin, nous introduisons une nouvelle méthode contrastive, ST-DIM, qui excelle sur ces 22 jeux Atari.

Mots-Clés: apprentissage de représentations profondes, apprentissage non supervisé, apprentissage de représentations, apprentissage par renforcement, apprentissage auto-supervisé, contrastives d'apprentissage de représentations non supervisées

Summary

Extracting a representation of all the high-level factors of an agent's state from level-level sensory information is an important, but challenging task in machine learning. In this thesis, we will explore several unsupervised approaches for learning these state representations. We apply and analyze existing unsupervised representation learning methods in reinforcement learning environments, as well as contribute our own evaluation benchmark and our own novel state representation learning method.

In the first chapter, we will overview and motivate unsupervised representation learning for machine learning in general and for reinforcement learning. We will then introduce a relatively new subfield of representation learning: self-supervised learning. We will then cover two core representation learning approaches, generative methods and discriminative methods. Specifically, we will focus on a collection of discriminative representation learning methods called contrastive unsupervised representation learning (CURL) methods. We will close the first chapter by detailing various approaches for evaluating the usefulness of representations.

In the second chapter, we will present a workshop paper, where we evaluate a handful of off-the-shelf self-supervised methods in reinforcement learning problems. We discover that the performance of these representations depends heavily on the dynamics and visual structure of the environment. As such, we determine that a more systematic study of environments and methods is required.

Our third chapter covers our second article, Unsupervised State Representation Learning in Atari, where we try to execute a more thorough study of representation learning methods in RL as motivated by the second chapter. To facilitate a more thorough evaluation of representations in RL we introduce a benchmark of 22 fully labelled Atari games. In addition, we choose the representation learning methods for comparison in a more systematic way by focusing on comparing generative methods with contrastive methods, instead of the less systematically chosen off-the-shelf methods from the second chapter. Finally, we introduce a new contrastive method, ST-DIM, which excels at the 22 Atari games.

Keywords: deep learning, unsupervised learning, representation learning, reinforcement learning, self-supervised learning, constrastive unsupervised representation learning

Table des matières

Table des figures

Liste des tableaux

List of Abbreviations

ALC	Auxiliary Label Classification
CPC	Contrastive Predictive Coding
CURL	Contrastive Unsupervised Representation Learning
DIM	Deep InfoMax
DQN	Deep Q Network
GAN	Generative Adversarial Networks
NCE	Noise Contrastive Estimator
NLL	Negative Log-Likelihood
NLP	Natural Language Processing
PCL	Permutation Contrastive Learning
RL	Reinforcement Learning
ST-DIM	Spatiotemporal Deep InfoMax
VAE	Variational Autoencoder

Acknowledgments

First and foremost, I would like to thank my parents. They have been unconditionally supportive and they always encouraged me to figure out for myself what I wanted to do in life without forcing anything on me, which is something I will forever be grateful for.

I would also like to thank all of my amazing friends from high school and college. I am thankful to them not only for the incredibly fun times we had, but all the invaluable life skills I have learned from them. I cannot overstate the social/soft skills that I took from you.

To Prabhat, Silvia, Yushu, Laleh, and all my former mentors at Lawrence Berkeley Lab, thank you for taking a chance on me and giving me immensely important opportunities that allowed me to get into the field of machine learning and into Mila. Also, thanks to Nelson Max, who was my first nominal undergraduate research supervisor in my last semester at UC Davis. Even though we only met a few times, my research career started with you and Silvia! Opportunities and getting a foot in the door are so invaluable in research and often they are not easy to get, so I am truly grateful to all of you. Also shoutout to my colleagues from LBL: Wahid, Lisa, Mustafa, Thorsten, Shreyas, Debbie, Rollin, Annette, and Jialin. You were all so smart, kind, and helpful.

Furthermore, I am very grateful for the many members of Mila, who I crossed paths with in my time pursuing a master's. Those who I shared a laugh with or argued at length with about disentangling, thank you for the great discussions and the unforgettable memories. I still remember my first few events at Mila that made me feel at home: getting lunch with Rithesh and a beer with James on first day and going to Mila poker on my first Friday. Mila really is an amazing community with a wealth of knowledge, kindness, and humor. Also, thanks to Tristan for proofreading my French abstract.

To my collaborators: Ankesh, Sherjil, Devon, Yoshua, Marc-Alexandre, it was a lot of fun writing the AtariARI paper with y'all. I not only learned a lot about mutual information and contrastive methods, but I really improved as a researcher by learning the importance of frequent communication and learning what is worth spending time on and what is not. Special shoutout to Ankesh for teaching me how to write quick and dirty (but not too dirty) research code and showing me all the best software/workflow tools.

Lastly, I would like to thank my advisor Chris Pal, who has given me an ideal amount of research freedom, while being super chill with all the growing pains I

have had as a master's student.

1 Background and Related Work

1.1 Machine Learning

Artificial intelligence is a subfield in computer science with a long and rich history [Russell and Norvig, 2016]. One of the most recently successful subfields within AI is machine learning. Traditionally in computer science, a programmer specifies an algorithm, which is a set of instructions for the computer to execute. Machine learning, on the other hand, is concerned with shifting this algorithm creation process to the computer, where the goal is to coax the machine into learning these algorithms on its own from input data. Understanding this thesis will require basic knowledge of machine learning, and more specifically deep learning. For a full introduction to machine learning and deep learning, we encourage the reader to check out [Bishop, 2006, Murphy, 2012, Goodfellow et al., 2016].

1.2 Unsupervised Learning

Machine learning is typically divided into two major categories: **supervised learning** and **unsupervised learning**.

Supervised learning usually involves learning from a dataset of pairs, $x, y \sim (X, Y)$ each of which is composed an input example, x paired with a corresponding *label* y, which is usually provided by a human. The goal of supervised learning is then to learn a function to predict y from x, usually by modelling the probability distribution, $p(y|x)$ [Goodfellow et al., 2016].

Unsupervised learning, on the other hand, is concerned with learning without labels using just input examples, X. The goal in this case widely varies, but often deals with estimating or sampling from the distribution $P(X)$ or some specific parts of it [Goodfellow et al., 2016]. Four common goals of unsupervised learning are [Khemakhem et al., 2019]:

1. *Modelling the Data Distribution*
   Modelling the data distribution, $p_{data}(x)$, is often concerned with fitting a model $p_{model}(x)$ to approximate the data distribution. This is often referred to as density estimation.

2. *Sampling*
   Sampling involves learning a model that allows you to approximate sampling from $p_{data}(x)$. This can be accomplished using density estimation if you learn a model you can sample from. Moreover, one directly learn a function made specifically for sampling [Goodfellow et al., 2014, Andrieu et al., 2003].

3. *Reveal Underlying Structure of the Data*
   In this case, we are interested in revealing underlying structure of the data. We can often imagine the data is created by some unknown generative process that takes a few underlying high level conceptual quantities and combines them to form the raw data. In this case we are interested in discovering these latent quantities that are part of this data generating process.

4. *Downstream Task performance*
   In this case, we aim to learn transformations of the data, which we call representations, that are amenable to future prediction tasks. This is commonly called *representation learning*. We cover representation learning in more detail in the next section.

These four goals are interrelated, and not mutually exclusive. For example, many papers model the data distribution and can also sample from the data distribution [Dinh et al., 2017, van den Oord et al., 2016]. Moreover, models known as latent variable models aim to reveal underlying structure, as well as approximately model data distribution and allow you to sample from them [Kingma and Welling, 2013]. The models that accomplish the first and second goals are known as generative models and will be covered in a bit more detail in section 1.5. Models that attempt to learn the underlying structure are often used for downstream tasks which we will show in section 1.5. However, modeling the data distribution may not automatically allow us to learn underlying structure or useful features. For example, a perfect generative model, which could estimate the data distribution perfectly and allow us to sample from it could be a non-parametric model, which stores an infinite number of samples from the data distribution. This model would have no knowledge of the underlying structure even though it would have modeled the data distribution perfectly, as parodied in Gelly et al. [2018].

1.3 Representation Learning

Performance of machine learning algorithms is often dependent on data representations. A representation is often defined as a transformation of the data into another form or space before inputting it to a machine learning algorithm. One example of transforming a raw data to get a representation is what is called feature engineering or computing hand-crafted features [Lowe, 2004, Horn and Schunck, 1981]. Representation learning, on the other hand, is concerned with automatically learning these transformations from the raw data instead of a programmer specifying a function to extract representations. As such, representation learning has been defined as representing data in a way that facilitates pulling out relevant features for downstream prediction tasks [Bengio et al., 2013]. This idea is the fundamental idea behind supervised deep learning, where we train a deep model to learn the appropriate representation and then use it to solve the task at hand in one big end-to-end way. Recently, however, representation learning has become synonymous with unsupervised representation learning, where we aim to learn a representation useful for many tasks without explicitly knowing the task ahead of time. Instead of optimizing the representation for a specific task, we can think of the representation learning process as one that tries to extract the high-level hidden, semantic factors of variation that are involved in the unknown generative process that created the data [Bengio et al., 2013].

1.3.1 What do we want in a representation ?

We want a representation by definition to facilitate success in future supervised tasks. However, it is hard to know a priori what exact traits must be present in a representation to result in high downstream performance. Nevertheless, there are several desirable traits of a representation that are thought to be important for future tasks. We often want a representation that identifies, extracts, and organizes underlying, semantically meaningful factors. Furthermore, we want our representations to be expressive in the sense that they should concisely capture a lot of these salient factors. We also want these representations to be invariant, which means insensitive to noise or other unimportant changes in the raw input. For example, incrementing the pixel value of a few background pixels of a photo of cat should not change the representation's encoding of the identity and traits of the cat.

1.3.2 Disentangling

Another very common desired trait of a representation is that it is disentangled. Bengio et al. [2013] define a disentangled representation as one that untangles the salient factors of variation in the input. These high-level factors are thought to be initially fairly independent and untangled from each other, but they are then highly entangled during the data generative process. As such, learning models that can disentangle these factors can potentially result in models that are more robust to noisy, entangled low-level sensory signals. Disentangling has traditionally been a tough term to define, but currently one of the more agreed-upon definitions is representations that are explicit, modular, and compact [Ridgeway and Mozer, 2018]. Explicit means that the representation encodes high level salient factors in a way that is easily accessible by a downstream classifier. More formally, it means that mapping between factors and elements of the representation can be implemented simply, like with a linear classifier. A representation is modular if different parts of the representation capture only one factor each, and it is compact if each factor is captured by only one part of the representation. Formally, being modular and compact means there is a more or less one-to-one mapping between different parts of the representation and different high-level factors.

1.4 Self-Supervised Learning

Self-supervised learning has emerged as a very popular collection of methods for representation learning [LeCun, 2018]. Due to the novelty of this term and its name, there has been a lot of confusion regarding its official definition and how it is different from unsupervised learning. While there is no agreed upon definition for the term in the machine learning community, practitioners of self-supervised learning are in relative agreement on its rough definition. We will attempt to define the term as such, as well give examples of self-supervised methods throughout this background section. Self-supervised learning methods are a subset of unsupervised learning techniques, which take advantage of structure in the data to create a supervisory signal, allowing self-supervised models to be trained in the same way as supervised learning techniques [Fernando et al., 2017]. This automatically generated supervisory signal comes in the form of “free”, intrinsic labels that are either formed from inherent structure in the data, like temporal order of frames in a video, or come along with data as another modality, like actions paired with each observation in an RL agent. Using these intrinsic labels, we can train a seemingly supervised task. These tasks are often called “pretext” tasks, as they are carefully designed so that solving the task forces the model to learn a good representation. The word pretext is defined as “a reason given in justification of a course of action that is not the real reason” [Dictionary, 1989], so one can think of pretext tasks as tasks that are not important to solve in and of themselves, but are *excuses* to get a model to learn useful representations. Examples of these pretext tasks are determining if video frames out of order [Misra et al., 2016], solving jigsaw puzzles [Noroozi and Favaro, 2016] with the patches of images, and guessing what words neighbor a given word in a sentence [Mikolov et al., 2013]. More examples of self-supervised methods are presented in later sections of the background and throughout this work.

1.5 Generative Models for Representation Learning

Generative models are concerned with learning a joint distribution over all observed variables. One of the most common formulations of generative models in machine learning is a probabilistic model. A probabilistic model is a mathematical specification of the joint distribution of observed data [Kingma and Welling, 2019]. Specifically, if one observes some raw data, like an image, a probabilistic model of this image complete specification of joint distribution of all observed random variables, $P(x_1, x_2, \dots, x_n)$, where x_i is the ith pixel of the image.

Generative models also can be used in representation learning. The intuition behind their utility in representation learning is that by forcing a representation to be useful for modeling the world, the representation can potentially encode the world at an abstract level and that maybe would be beneficial for certain downstream tasks that require an understanding of the world in order to be solved [Kingma and Welling, 2019]. Most generative approaches for representation learning use what is called a latent variable model. A latent variable model is a probabilistic model that not only tries to model joint distribution over observed variables, but also over latent variables, $z_1, z_2, \dots : P(x, z)$. While observed variables are observed by the model, like pixels, latent variables are unobserved variables that must be inferred. We can think of latent variables as abstract, underlying random variables that describe the data distribution. Often, we can imagine the data as generated by some unknown generative process that combines some hidden, high-level factors along with some noise to create the input data. These high-level factors are modelled using latent variables. Thus, a parametric latent variable model is concerned not only with learning the parameters of the model, but in estimating the latent variables as well. In representation learning, the estimate of these latent variables is used as the representation. One of the most popular neural latent variable models used for representation learning is the variational autoencoder (VAE).

1.5.1 Variational Autoencoders

A VAE learns a latent variable model by maximizing an approximate lower bound on the log marginal likelihood of the data, $\log p(x)$. Learning a latent variable model by directly maximizing the marginal log likelihood: $\log p(x) = \log \int p(x, z)dz$ is intractable to compute because there is no closed form solution and no efficient estimator, as z is continuous and can take on any value. If we knew $p(z|x)$, the log marginal likelihood would be easy to compute, but $p(z|x)$ is also unknown [Kingma and Welling, 2019] However, if we can approximate $p(z|x)$ with a parametric model $q_\psi(z|x)$, then we can learn a lower bound on the log marginal likelihood, $\log p(x)$.

This lower bound is called the evidence lower bound (ELBO):

$$\mathcal{L}_{ELBO} = \log p_\theta(x|z) - D_{\mathrm{KL}}(q(z|x;\psi) \mid\mid p(z)) \leq \log p(x) \tag{1.1}$$

Where $p(z)$ is the prior distribution, often picked to be $\mathcal{N}(0, I)$ an isotropic Gaussian distribution. We parametrize both $q_\psi(z|x)$, which we call the encoder, and $p_\theta(x|z)$, called the decoder, with neural networks. Using the reparameterization trick [Kingma and Welling, 2013], we can train both models end-to-end to maximize this lower bound. First, an image is transformed by the encoder resulting in the mean and variance parameters of the posterior distribution over the latent variables. The latent variables z are sampled using the reparameterization trick [1] [Kingma and Welling, 2013] then this z is transformed by the decoder to obtain $\hat{x}$, a reconstruction of the input x. The ELBO, used as a loss, is then computed and both the encoder and decoder are updated end-to-end to maximize this loss like in any neural network.

Representation learning is concerned with finding a mapping $\phi(x)$ of the data that facilitates better performance on downstream tasks. In the case of VAE's, our $\phi(x)$ is the encoder of the VAE, $q_\psi(z|x)$, where our representation is the mean parameter output by $q_\psi(z|x)$.

1.5.2 Other Generative Methods for Representation Learning

While the original formulation of VAE's remains the most common generative representation learning technique, other variants of the VAE have been developed specifically for the purpose of representation learning. Higgins et al. [2017a], Kim and Mnih [2018], Kumar et al. [2017], Chen et al. [2018] all train VAE's with different modified objectives for $q_\phi(z|x)$ that try to coax it into learning a more *disentangled* representation.

While the VAE and its variants dominate the field of representation learning, other generative methods have been used as well. The most notable of these are various variants of generative adversarial networks (GANs)[Goodfellow et al., 2014]. A GAN consists of a neural network, called the generator, that attempts to transform randomly sampled noise into approximate samples from the dataset, $\hat{x}$

1. where sampling is seen as a deterministic, differentiable function of x and of some independent and identically distributed variables

by attempting to maximize the classification error of another neural network that is trained to discriminate between real samples from the dataset and the neural network generated ones. Chen et al. [2016] learns a representation by forcing the inputs, z, to the generator of a GAN to be distributed in a certain way and have high mutual information with the generated sample, $\hat{x}$. Dumoulin et al. [2017], Donahue et al. [2017] also harness a GAN to learn embeddings of the input data, while also guaranteeing that these embeddings are useful for generating samples.

Generative Self-Supervised Methods

In addition to representation learning methods that are based on an objective that tries to model the distribution of the input data, there are several self-supervised methods that learn representations by conditionally modeling **parts** of the data distribution. Pathak et al. [2016b] learns representations by removing a patch from an image forcing a neural network to generate the missing patch, Zhang et al. [2016] generates the colors of each pixel from an image that has been transformed to gray scale and Zhang et al. [2017] trains an autoencoder to generate a missing RGB channel using the others two channels as input. Lastly, in video, many researchers have used pixel prediction of the future frames in a video to learn representations [Oh et al., 2015, Finn et al., 2016, Srivastava et al., 2015, Lotter et al., 2016, Mathieu et al., 2015].

1.6 Discriminative Models for Representation Learning

One of the main goals of representation learning is to capture high-level features, but ignore more noisy local information. The main goal of generative models, on the other hand, is to try to model every little detail in the raw data. To be sure, generative models for representation learning include other inductive biases like latent variables, but their main objective is modeling every pixel. Intuitively this seems like computational overkill, as a lot of capacity is spent trying to model complex local pixel relationships, while ignoring the high-level semantic context. A few high-level latent variables contain much less information than all the pixels in an image. Hence, modelling p(x|z) (thousands of bits) does not seem like a good idea if you are just interested in learning p(z|x) (10 bits). What if there was a way to learn p(z|x) without using a generative model. Discriminative models for representation learning try to do just that. Discriminative methods for representation learning create a training objective that looks like supervised classification and acts directly in representation space. There are two main approaches to discriminative representation learning: contrastive methods and auxiliary label classification methods.

1.6.1 Contrastive Unsupervised Representation Learning

Contrastive unsupervised representation learning (CURL) methods are algorithms that harness pairs of unlabelled examples that are known to be dependent, like frames in video close together in time or adjacent words in a sentence. Most CURL methods follow a standard recipe where representations from pairs of dependent examples, $\phi(x), \phi(x^+)$ are input to a score function $f(\phi(x), \phi(x^+))$ where the goal is to make the score for dependent pairs higher on average than that for independent pairs, $f(\phi(x), \phi(x^-))$ [Arora et al., 2019]. Losses are then created by constructing a likelihood that is high when the score function behaves as intended and low otherwise. Once one has created this likelihood the loss to be minimized is simply the negative log likelihood:

$$L_{CURL} = -\mathbb{E}_{x,x^+,x^-}\left[\log \frac{\exp f(\phi(x), \phi(x^+))}{\exp f(\phi(x), \phi(x^-)) + \exp f(\phi(x), \phi(x^+))}\right] \tag{1.2}$$

This equation is an example of using just one negative pair and creating a binary classification problem, but this can be extended to use multiple negatives. The score function, $f(x_1, x_2)$ can be a dot product, a concatenation, or a bilinear function. Dependent pairs are often straightforward to acquire using consecutive frames in a video, adjacent patches in an image, or adjacent sentences. Independent pairs are usually selected randomly [Arora et al., 2019].

Contrastive methods trace their origins back to Noise Contrastive Estimation (NCE) [Gutmann and Hyvärinen, 2010]. NCE was originally proposed as a new technique for estimation of unnormalized parametric density estimation models, like energy-based models or markov random fields. Under NCE, the parameters of the density model are estimated by learning a binary classifier that can classify whether a sample is from the dataset or whether it is randomly sampled noise. While this idea was initially proposed for learning parameters of unnormalized explicit density models, it has been extensively used for implicit density generative models, namely generative adversarial networks (GANs)[Goodfellow et al., 2014]. Instead of discriminating between a real datapoint and randomly sampled noise, GANs discriminate between real datapoints and ones generated by a neural network with the goal of approximating samples from the data distribution with a neural network. Most recently, NCE has been adapted to representation learning ; instead of discriminating between real datapoints and generated ones or random noise, we discriminate between matched and unmatched pairs of representations, as shown in equation 1.2. One of the earliest examples of this was permutation contrastive learning (PCL) [Hyvarinen and Morioka, 2017], which is a method for learning transformations of the data for time series data. PCL follows equation 2, where x, x^+ are consecutive examples in time, x^- is a randomly selected example in the sequence, and the score function f is concatenation [Hyvarinen and Morioka, 2017].

Examples of Contrastive Methods in Self-Supervised Learning

While many self-supervised methods are based on intuitive, domain-specific heuristics, a lot of them can simply be reduced to the CURL formulations in equation 1.2. Many self-supervised methods for videos, for instance, can be reduced to contrastive tasks. Misra et al. [2016] and Fernando et al. [2017] train a model to discriminate between sequences of frames in a video that are in order and those that are out of order. These tasks can be derived from equation 1.2 if we consider the score function f to be the concatenation function and positive examples to be triplets of frame features that are in order for [Misra et al., 2016] or sets of frame representations that are out of order [Fernando et al., 2017]. Methods like [Misra et al., 2016, Fernando et al., 2017] that classify between sequences are very straightforward to connect to contrastive methods because they train a classifier, but there are also some self-supervised techniques for video that use what is called a triplet loss. The triplet loss maximizes the L2 distance between representations of dissimilar pairs, while minimizing the distance between similar pairs:

$$\mathcal{L}_{\text{triplet}} = \mathbb{E}_{x,x^+,x^-}\left[\max(||\phi(x) - \phi(x^+)||_2^2 - ||\phi(x) - \phi(x^-)||_2^2, 0)\right] \tag{1.3}$$

Time Contrastive Networks (TCN) [Sermanet et al., 2017] considers videos filmed from multiple angles and uses a triplet loss to discriminate between frames that are close together in time, but filmed at different angles and frames filmed at the same angle but at different time steps.Wang and Gupta [2015] uses the triplet loss to discriminate between spatial patches of a video containing the same object at different time steps and those where only one patch has the object. CURL methods are very prevalent in natural language processing (NLP). Logeswaran and Lee [2018] contrast sentence fragments that are consecutive with random sentence fragments and Mikolov et al. [2013] discriminates between a tuple of a word's embedding with neighboring words' embeddings and a tuple of the word's embedding with random word embeddings.

Connections to Mutual Information

Mutual information between two random variables, u and v is defined as the KL divergence between the joint distribution of the variables and the product of marginals of the variables:

$$I(u;v) = D_{\mathrm{KL}}[p(u,v) \mid\mid p(u)p(v)] \tag{1.4}$$

It can actually be shown that when using a multiple negative version of NCE, InfoNCE [van den Oord et al., 2018]:

$$\mathcal{L}_{InfoNCE} = -\mathbb{E}_{x,x^+,x_i}\left[\log \frac{\exp f(\phi(x),\phi(x^+))}{\exp f(\phi(x),\phi(x^+)) + \sum_{i=0}^{i=k} \exp f(\phi(x),\phi(x_i^-))}\right] \tag{1.5}$$

where there are k negative examples. The optimal score function $f(x_1, x_2)$ actually approximates the density ratio: $\frac{p(x_1,x_2)}{p(x_1)p(x_2)}$, which is the ratio used in mutual information, so then it can be shown that when using InfoNCE, minimizing the loss, $\mathcal{L}_{InfoNCE}$ maximizes a lower bound on mutual information [van den Oord et al., 2018, Poole et al., 2019]: $I(x;x^+) \geq \log(k) - \mathcal{L}_{InfoNCE}$. This bound becomes tighter as, k, the number of negative examples increases. This lower bound is used in Deep InfoMax (DIM) [Hjelm et al., 2019] and CPC [van den Oord et al., 2018].

1.6.2 Auxiliary Label Classification (ALC)

While CURL methods create a classification problem by constructing a likelihood function that captures the intuition of discriminating between positive pairs and negative pairs, there are some methods that work by simply directly solving a classification problem using the features as input and either a label derived from the context of the data or one from some other modality included in the data. Instead of directly contrasting pairs of features, they instead try to classify this "auxiliary" label from the features. Examples of labels derived from the structure of the data are the index of a patch in an image or the difference in time index between two frames in a video. The loss of ALC tasks follow a similar setup to contrastive tasks:

$$\mathcal{L}_{ALC} = -\mathbb{E}_{x_1,x_2}\left[\log \frac{\exp f_j(\phi(x_1),\phi(x_2))}{\sum_{i=1}^{k}\exp f_i(\phi(x_1),\phi(x_2))}\right] \tag{1.6}$$

Where j is the index of the correct class and there are k classes. The difference here is the score function is different for each term in the numerator and denominator, but the input to the score function is the same for all terms. $\phi(x_1)$ and $\phi(x_2)$ are usually embeddings from two or more patches in an image, frames in a video, or even words in a sentence, for example.

Examples of ALC methods in Self-supervised Learning

There are many examples of ALC in classic self-supervised tasks. For example, for static images [Noroozi and Favaro, 2016] create a self-supervised task, whereby an image is cut up into patches, then each patch is encoded into a representation and the patches are shuffled. Then, a neural network is tasked with classifying which of the possible permutations of patch representations is the correct one. Similarly, [Doersch et al., 2015] train a model that takes as inputs embeddings from a pair of patches and must classify the relative position between the two patches. Temporal distance classification [Aytar et al., 2018] processes representations from two frames in a video, x_t and x_{t+k} and classifies the difference in time index between the two (k). [Agrawal et al., 2015, 2016] both harness labels that are not quite derived from the structure of the data, but are essentially "free" and come with every datapoint as a separate modality. Agrawal et al. [2015], for example uses a model that given two frames in a video classifies the direction the camera was moved in, a label, which is included in the data. Agrawal et al. [2016] uses states and actions from a reinforcement learning agent: $x_1, a_1, x_2, a_2, x_3, a_3, \ldots$ and train a model that takes embeddings from two consecutive frames $\phi(x_t), \phi(x_{t+1})$ to classify the corresponding action, a_t that was taken at time t.

1.7 Reinforcement Learning

1.7.1 (Deep) Reinforcement Learning Primer

Reinforcement learning (RL) refers to three things: a subfield of machine learning, a sequential decision-making problem, and a collection of methods to solve the problem. In essence, RL is a problem where the aim is to learn a way to act in a given situation, such that some goal is accomplished or specifically some reward value is maximized [Sutton et al., 1998]. A very straightforward, formal way to model this problem is using Markov decision processes (MDPs). An MDP is composed of an agent, an environment, actions, states, and rewards. The agent is the decision maker, which makes decisions about how to interact with an environment. The agent sequentially receives a state, S_t from the environment, which represents the situation the agent is in. To interact with the environment the agent takes actions, A_t , which it selects based on the state it's in. Based on the action the agent selects, it receives a numerical reward, r_{t+1} along with the next state, S_{t+1} from the environment. S and R are random variables and $A(s)$ is a function of the state. We can form a conditional probability distribution over these random variables, where the probability of reaching state s' and receiving reward r when the agent is at state s and takes action a is given by:

$$p(s', r|s, a) = P(S_t = s, R_t = r|S_{t-1}, A_{t-1} = a) \tag{1.7}$$

This probability distribution fully describes the dynamics of the environment.

The goal in RL is then to maximize the expected return, which is the discounted sum of rewards the agent will receive: $\sum_{t=0}^{\infty} \gamma^t R_t$, where γ $(0 <= \gamma <= 1)$ is the discount rate, which not only favors rewards which happen sooner, but by being between 0 and 1 guarantees the sum converges [Sutton et al., 1998]. The pursuit of this goal is often also referred to as control. When the reward is 0 in all but a few states in an environment the problem is often called a sparse reward problem.

In order to do this we often learn two things: a policy and a value function. A policy, for the sake of this thesis, is a parametric function $\pi(a|s;\theta)$, which maps states to probabilities or preferences of taking actions. Throughout the course of training, an agent continues to improve its policy in order to act in a way that maximizes returns. A value function, $v_\pi(s)$ is the expected return the agent will receive if it starts in state s and then follows the policy π. Following in this sense means sampling from π as it interacts with the environment. $v_\pi(s)$ is known as the state-value function, but there is also the action-value function, $q_\pi(s,a)$, which is defined as the expected return if the agent finds itself in state s then takes action a and from then on follows the policy π. In deep RL, q_π and v_π are parameterized by neural networks, which take the state or state and action as input and output a real number estimating the return.

In deep RL, these value function neural networks are often trained using neural fitted Q learning [Riedmiller, 2005] or deep Q-learning, DQN [Mnih et al., 2015], which minimize the following loss:

$$\mathcal{L}_{DQN} = (Q_\theta(s,a) - (r + \gamma max_{a'}\hat{Q}(s',a')))^2 \tag{1.8}$$

[2] where $s', r \sim p(s', r|s, a)$. Another common approach is directly learning a parameterized policy, $\pi(s;\theta)$ instead of a value function. These approaches are called policy gradient methods [Sutton et al., 1998]. Models that learn a policy or value function using just observed states, like Q-learning or policy gradient methods are called model-free methods. Model-based methods, on the other hand, try to explicitly estimate $p(s', r|s, a)$, which can then be sampled from to learn a policy, instead of what model free methods do, which is just directly mapping states to actions [Sutton et al., 1998].

In many cases, the agent cannot observe the true state and observes a low-level sensory information, like an image, which we call the observation denoted by X_t. The states in this case are unobserved latent variables.

While trying to directly maximize a sum of extrinsic, human specified rewards is the most common goal in the field of reinforcement learning, other common goals

2. where $\hat{Q}$ is considered fixed while we update Q_θ

include exploration and imitation learning. Exploration often involves the agent trying to visit as many novel states as possible. This is usually accomplished by maximizing an "intrinsic" reward that tries to measure how infrequently a certain state has been visited [Ostrovski et al., 2017, Pathak et al., 2017a]. Imitation learning is concerned with learning a policy by observing humans completing similar tasks, which is often achieved by maximizing some reward based on how closely the agent has mimicked the human.

1.7.2 Representation Learning for RL

Although DQN has had achieved great success in achieving high scores in video game environments, it is very sample inefficient [François-Lavet et al., 2018]. It is also very brittle in that it only does well on exactly the games it was trained on and any slight alterations to the games would result in poor performance [Leike et al., 2017, Zhang et al., 2018, Kansky et al., 2017]. As such, it seems deep RL would benefit greatly from representation learning. Representation learning is about learning a transformation of the data to make a downstream task easier, so it makes sense to learn a representation that makes a downstream control task more sample efficient and robust. Indeed, some works have begun to think about harnessing representation learning for RL by learning a representation of the state, s in a separate representation learning phase followed by policy or value function learning phase that is learned on top of this learned "state representation" [Lesort et al., 2018]. For example, Cuccu et al. [2019] shows that once one extracts a good state representation, the policy model can be small and simple to train and it excels at Atari games. Jonschkowski and Brock [2015] and Jonschkowski et al. [2017] try to learn state representations by creating several heuristic priors with which they constrain their representation to follow.

Most current research using state representation for control uses generative methods, especially latent variable models. The most common model used in RL for representation learning is the VAE. Several prior works use a VAE and its variations to learn state representations [Ha and Schmidhuber, 2018a, Watter et al., 2015, van Hoof et al., 2016, Duan, 2017, Cuccu et al., 2019]. Thomas et al. [2017] tries to extract a representation from states in a environment that captures how the state changes as actions are taken using a VAE as the base model. Higgins et al. [2017b] uses a β-VAE to discover features that help for transfer policies to new, unseen environments. One of the other common generative methods used in state representation learning is prediction in pixel-space, where one trains a generative model to model the distribution of the observations from the next state given the current observation and action taken: $p(x_{t+1}|x_t, a_t)$. [Oh et al., 2015, Finn et al., 2016], for example, have exploited these pixel prediction models for RL. Since generative models have dominated state representation learning in RL, discriminative methods are more rarely found in the literature. One example of researchers applying discriminative methods is in CPC [van den Oord et al., 2018], where they train a policy gradient method, A3C, with encoded representations from CPC as input instead of raw images from a visual RL environment. While discriminative methods are not common in traditional control approaches, they have seen use in exploration [Pathak et al., 2017a] and imitation learning [Aytar et al., 2018]. In the two articles in this thesis, we will, however, explore discriminative representation learning algorithms for control.

1.8 Evaluating Representations

Representation learning is often defined as the process of learning features useful for downstream tasks, so evaluating representations can sometimes simply be done by measuring the performance on a given downstream task. However, theses tasks are not always known ahead of time, so more and more people have attempted to measure the general, task-agnostic "utility" of a representation. Assessing the utility is often preferred if we do not know a priori what downstream tasks we want to solve or we want a "general purpose" representation. Figuring out what "general purpose" means is difficult and for that reason measuring representations remains a challenge in representation learning. There are usually two main approaches for evaluating representations: qualitative evaluations and quantitative evaluations.

1.8.1 Quantitative Techniques for Evaluating Representations

The most common quantitative approach requires identifying "ground truth" factors of variation. These factors are usually heuristically identified ahead of time by humans as salient, high level context information. Examples of "ground truth" factors of variation include: position, orientation, and color of objects. Harnessing these "labels" at test time, we can evaluate how well a representation encodes and organizes these ground truth factors of variation. Measuring the extent to which a representation encodes factors is often defined as explicitness [Ridgeway and Mozer, 2018].

Explicitness

A representation is defined as explicit if the mapping between factors and elements of the representation can be implemented simply. Generally this means that a simple, shallow classifier like a linear classifier or a random forest can classify the ground truth factors when it receives the representation as input . This "classifier" is often purposely restricted in capacity in order to really test how well a representation has captured these factors. This is because at evaluation time, we want to evaluate a representation not learn a new one. Linear models are shallow and do not learn intermediate representations, whereas deep MLP's, for instance, do. Using a supervised linear classifier on a representation is often called "linear probing" [Alain and Bengio, 2017] and is commonly used in the disentangling literature [Eastwood and Williams, 2018].

In learning disentangled representations, one not only cares about explicitness, how well salient factors are captured, but also how they are organized. In the disentangling literature, this is usually summarized by two many desired traits: modularity and compactness.

Modularity

A representation is modular if each representational unit, r_i (usually one element in a vector) is associated with only one ground truth factor of variation, y_i. $r_i \rightarrow y_i$. More concisely it measures how close to a one-to-one mapping exists between representational units and ground truth vectors. Modularity is often measured in a multitude of ways. For example, the BetaVAE score [Higgins et al., 2017a], the FactorVAE score [Kim and Mnih, 2018], the modularity score [Ridgeway and Mozer, 2018], and the disentanglement score [Eastwood and Williams, 2018] all measure modularity by quantifying how well a single representational unit correlates with a single ground truth variable.

Compactness

A representation is compact if it ensures that a single ground truth factor is represented using only one or a few representational units. $r_i \leftarrow y_i$. In essence, it measures whether there is a one-to-one mapping between ground truth vectors and representational units. SAP score [Kumar et al., 2017], completeness [Eastwood and Williams, 2018], mutual information gap [Chen et al., 2018] all assess compactness

If a representation is both modular and compact, then for the valid representational units, there is a bijection between representational units and ground truth factors. In this manuscript, we mostly focus on explicitness.

1.8.2 Qualitative Techniques for Evaluating Representations

There are many ways to qualitatively inspect how well a representation captures salient factors. These include decoder-based methods, similarity-based measures, and feature map inspection methods.

Decoder-Based Methods

For evaluating explicitness, one common decoder-based qualitative methods is to inspect the reconstruction created by passing the representation through the decoder and checking to see if salient objects and other factors are present in the reconstructed image. For evaluating traditional disentangling traits, like modularity and compactness, one often does what is called latent space interpolation [Dumoulin et al., 2017]. This entails changing just one element of the representation and passing the resulting changed vector back through the decoder and inspecting what has changed in the resulting image. If the representation is nicely disentangled, one might expect to see just one high level feature of the image changing as one representational unit changes, like the x position of a ball or its color. This is a very popular technique that has been used frequently [Chen et al., 2016, Brock et al., 2018, Dumoulin et al., 2017]

Similarity-Based Measures

One technique for similarity-based measures is to visualize low-dimensional embeddings of the representations in your dataset using a technique like t-stochastic neighbor embedding (t-SNE) [Maaten and Hinton, 2008]. Using this visualization, one can see which datapoints' representations are close together in this low-dimensional space and then visually inspect whether these datapoints are semantically similar. Instead of embedding representations to 2 or 3 dimensions, some researchers have visually inspected neighboring datapoints in representation space. This technique is known as finding "nearest neighbors in feature space" and consists simply of computing the L2 distance between a datapoint representation and all the other datapoints in a dataset and then visually inspecting the closest datapoints [Brock et al., 2018, van den Oord et al., 2018].

Feature Map Inspection

There are several visualization techniques that are used when the underlying encoder model is a convolutional neural network. Namely, saliency map methods and feature map inspection methods. Feature map inspection techniques simply involve visually inspecting intermediate feature maps from convolutional neural network to see if important features are captured. Saliency map based methods try to interpret what the CNN is learning by computing the sensitivity of the output of the network with respect to each input pixel [Springenberg et al., 2014, Selvaraju et al., 2017]

In the first chapter, we explore quantitative measures along with qualitative feature map analysis, whereas in the second work, we only focus on quantitative measures.

2 Supervise Thyself: Examining Self-Supervised Representations in Interactive Environments

Supervise Thyself: Examining Self-Supervised Representations in Interactive Environments
Evan Racah, Christopher Pal

This chapter presents a joint work with Christopher Pal. It was accepted to the Self-Supervised Learning Workshop at the International Conference on Machine Learning (ICML 2019)

Contribution: I wrote all the code for all self-supervised methods, data collection, and results processing. I ran all the experiments, processed the results, created the figures, and wrote the entire paper. Christopher Pal proofread the paper and provided high-level guidance.

Affiliation
— Evan Racah, Mila, Université de Montréal
— Christopher Pal, Mila, École Polytechnique de Montréal,Université de Montréal, Element AI

2.1 Abstract

Self-supervised methods, wherein an agent learns representations solely by observing the results of its actions, become crucial in environments which do not provide a dense reward signal or have labels. In most cases, such methods are used for pretraining or auxiliary tasks for ”downstream” tasks, such as control, exploration, or imitation learning. However, it is not clear which method's representations best capture meaningful features of the environment, and which are best suited for which types of environments. We present a small-scale study of self-supervised methods on two visual environments: Flappy Bird and *Sonic The Hedgehog*^{TM}. In particular, we quantitatively evaluate the representations learned from these tasks in two contexts: a) the extent to which the representations capture true state information of the agent and b) how generalizable these representations are to novel situations, like new levels and textures. Lastly, we evaluate these self-supervised features by visualizing which parts of the environment they focus on. Our results show that the utility of the representations is highly dependent on the visuals and dynamics of the environment.

2.2 Introduction

Self-Supervised methods have emerged as powerful methods for pretraining to learn more general representations for complicated downstream tasks in vision [Misra et al., 2016, Fernando et al., 2015, 2017, Wei et al., 2018, Vondrick et al., 2018, Jayaraman and Grauman, 2015, Agrawal et al., 2015, Pathak et al., 2017b, Wang and Gupta, 2015] and NLP [Peters et al., 2018, Subramanian et al., 2018, Mikolov et al., 2013, Conneau and Kiela, 2018]. In interactive environments, they have begun to receive more attention due their ability to learn general features of important parts of the environment without any extrinsic reward or labels [LeCun, 2018]. Specifically, self-supervised methods has been used as auxiliary tasks to help shape the features or add signal to sparse reward problems [Mirowski et al., 2016, Jaderberg et al., 2016, Shelhamer et al., 2016]. They also have been used in unsupervised pretraining for control problems, [Ha and Schmidhuber, 2018b]. Moreover, they have been used in imitation learning to push expert demonstrations and agent observations into a shared feature space [Aytar et al., 2018, Sermanet et al., 2017]. Lastly, they have been used in intrinsic reward exploration to learn a representation well-suited for doing surprisal-based prediction in feature space [Pathak et al., 2017a, Burda et al., 2018a]. In each of these cases, the desired feature space learned with self-supervised methods should capture the agent, objects, and other features of interest, be easy to predict, and generalize to unseen views or appearances. However, existing evaluations of these methods do not really shed light on whether the representations learned by these self-supervised methods really robustly capture these things. Instead, these evaluations only evaluate the utility of these methods on the particular downstream task under study. While these types of tasks have been studied theoretically [Hyvarinen et al., 2018, Arora et al., 2019], they have not really been examined empirically in depth. As such, in this paper we examine a few self-supervised tasks where we specifically try to characterize the extent to which the learned features capture the state of the agent and important objects. Specifically, we measure how well the features: capture the agent and object positions and generalize to unseen environments, and lastly, we qualitatively measure what each self-supervised method is focusing on in the environment. We pick Flappy Bird and *Sonic The Hedgehog*^{TM} because they represent simple and complex games respectively in terms of graphics and dynamics. Also, one can change the level and colors of each to make an "unseen" environment to test generalizability. [1]

2.3 The Self-Supervised Methods We Explore

We explore four different approaches for self-supervision in interactive environments: VAE [Kingma and Welling, 2013], temporal distance classification (TDC)[Aytar et al., 2018] , tuple verification [Misra et al., 2016], and inverse model [Agrawal et al., 2016, Jayaraman and Grauman, 2015, Pathak et al., 2017a]. We also use a randomly initialized CNN as a baseline. All self-supervised models in this study use a *base encoder*, ϕ which is a four layer convolutional neural network, similar in architecture to the encoder used in the VAE in [Ha and Schmidhuber, 2018b]. The encoder takes as input a single frame in pixel space, x and outputs

1. https://github.com/eracah/supervise-thyself

an embedding, $\phi(x)$, where $\phi(x) \in \mathbb{R}^{32}$. Depending on the self-supervised task, certain heads, $f(\phi(x_1), \phi(x_2)$ are placed on top of the encoder, like a deconvolutional decoder or a linear classifier, that take $\phi(x)$ or multiple concatenated $\phi(x)$'s as input.

Random CNN: We use a randomly initialized CNN as a baseline method. In this case, it is an untrained base encoder, ϕ, with randomly initialized weights that are not updated. Random CNN's have been used with varying degrees of success in [Burda et al., 2018a,b]

VAE: VAEs [Kingma and Welling, 2013] are latent variable models that maximize a lower bound of the data likelihood, $p(x)$ by approximating the posterior $p(z|x)$, with a parametric distribution $q(z|x)$ and a prior $p(z)$. VAEs also include a decoder $p(x|z)$, which reconstructs the input x by mapping samples of z back to pixel space. In our setup $q(z|x)$ is parameterized by a Gaussian with mean $\phi(x)$ and the variance is parameterized by a separate fully connected layer on top of the penultimate layer of the base encoder. Also, we use a deconvolutional network, $\hat{x} = g(z)$ to parameterize $p(x|z)$. The VAE is trained by minimizing the KL divergence between the prior, $p(z)$, which we often pick to be an isotropic Gaussian, and the approximate posterior, $q(z|x)$, while also minimizing the negative log-likelihood of $p(x|z)$, like so:

$$\mathcal{L}_{ELBO} = \mathbb{E}_{z \sim q(z|x)}[log\ p(x|z) - D_{KL}[q(z|x)||p(z)]] \tag{2.1}$$

The idea is that if we learn close to factorized latent variables that encode enough information to reliably reconstruct the frame, they will capture important structure of the image, like objects with assumption that objects are independent, which might be too simplifying of an assumption.

Temporal Distance Classification (TDC): Temporal Distance Classification (TDC) is a self-supervised method introduced by [Aytar et al., 2018], similar to [Hyvarinen and Morioka, 2016], where the network is asked to predict the relative distance in time between two frames. This is done by learning a linear function, $f(\phi(x_t), \phi(x_{t+\Delta t})$, which predicts Δt, the distance in time between two frames, as a function of the embeddings of the two frames. Δt is sampled from a set of intervals, $\Delta t \in D$

$D = \{[0], [1], [2], [3-4], [5-9]\}$ The idea is that in order to do well at the task, it must learn the features of the input that change over time, which often corresponds to objects [Aytar et al., 2018].

Tuple Verification: Tuple Verification [Misra et al., 2016] is an instance of a temporal order or dynamics verification task, where the network must figure out if a sequence of frames is in order or not. The method works as such: five chronologically ordered, evenly-spaced frames are chosen from a rollout: $\{x_t, x_{t+\Delta t}, x_{t+2\Delta t}, x_{t+3\Delta t}, x_{t+4\Delta t}\}$ In this paper we use an evenly spaced sampling of 5 frames from a sequence of 10 consecutive frames. A binary classification problem is created by shuffling the frames to create sequences that are out of order, which we will call negative examples. The middle three frames in the order: $\{x_{t+\Delta t}, x_{t+2\Delta t}, x_{t+3\Delta t}\}$ is what we call a positive example, whereas sequences where we permute the middle frame with the first frame: $\{x_{t+\Delta t}, x_t, x_{t+3\Delta t}\}$ or permute the middle frame with the last frame: $\{x_{t+\Delta t}, x_{t+4\Delta t}, x_{t+3\Delta t}\}$ are out-of-order sequences that we call negative examples. We ensure that there is a $2:1:1$ ratio between positive samples and the two types of negative examples. The tuple verification model score function, f concatenates the embeddings from the three frames, $f[\{\phi(x_{t+\Delta t}), \phi(x_{t+2\Delta t}), \phi(x_{t+3\Delta t})\}]$ and linearly transforms them. A softmax classifier is then trained to maximize the score of the "ordered" trajectories and minimize the score of the out of order trajectories. Being successful at this task requires knowing how objects transform and move over time, which requires encoding of features corresponding to the appearance and location of objects [Misra et al., 2016].

Inverse Model: The inverse model [Agrawal et al., 2016, Jayaraman and Grauman, 2015, Pathak et al., 2017a] works by taking two consecutive frames from a rollout from an agent, then classifying which action was taken to transition from one frame to other. The model predicts the action by a linear classifier trained on the concatenation of the embeddings of two frames. The idea is that in order to reason about which action was taken, the network must learn to focus on parts of the environment that are controllable and affect the agent [Choi et al., 2018]. This should then result in the network learning features that capture the agent's state and location as well potential obstacles to the agent.

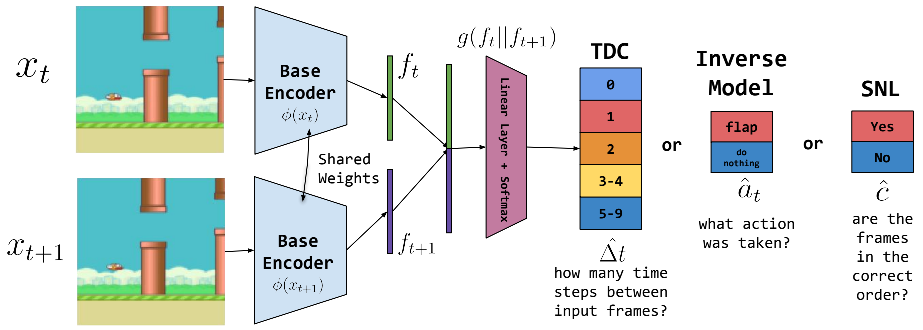


Figure 2.1 – **General architecture for self-supervised embedding**. Shown for Flappy Bird. Two or three frames are each input to the base encoder then the outputs from the encoder, $\phi(x)$ are concatenated and passed to a linear softmax layer that classifies either a) "how many time steps are between a pair of frames ?" for the TDC model [Aytar et al., 2018], b) "what action was taken to go from the first frame to second ?" for the inverse model [Agrawal et al., 2016], or c) "are a triplet of frames in the correct chronological order ?" for the tuple verification model [Misra et al., 2016]

2.4 Experiments/Results

2.4.1 Datasets/Environments

We collect the data for Flappy Bird [Tasfi, 2016] and *Sonic The Hedgehog*^{TM}, the `GreenHillZone` Act 1 level [Nichol et al., 2018] by deploying a random agent to collect 10,000 frames. At train time we randomly select frames from these 10,000 to form each batch. For the generalization experiments in section 2.4.3, we use what we call `FlappyBirdNight`, whereby we change the background, the color of the bird and the color of the pipes. For generalization in Sonic, we use `GreenHillZone` Act 2. We edit the action space of Sonic to be just the two actions: ["Right"] and ["Down", "B"]. This ensures the random agent will get pretty far in the level and actually collect a good diversity of frames. For both games, we resize the frames to 128 x 128. All ground truth position information (y position of bird, x position of pipe, y position of Sonic) is pulled from the gym API of these games and discretized to 16 buckets and represents the relative position of these objects in the frame, not the absolute position in the game. We purposely do not choose the x position of the bird or Sonic because in most frames of the game, the x position is relatively constant, while the background moves.

2.4.2 Extracting Position Information

To show whether the self-supervised methods are capable of encoding the position of important objects, we probe the representations learned from these methods with a linear classifier trained to classify the agent position (bucketed to 16). The results of this experiment are displayed in table 2.1 We find features from the inverse model are the most discriminative for detecting the position of the bird, likely because the inverse model focuses on what parts of the environment it can predictably control. We find features from TDC are the best for localizing the pipe. This is likely the case because TDC focuses on what parts of the environment are most discriminative for separating frames in time and in Flappy Bird, the background and bird stay in one place and the pipes move to the left to simulate the bird moving to the right. Tuple verification features are good for both objects, the pipe and the bird, likely because the position of the pipe relative to the bird is a very important temporal signal, which is discriminative to whether the frames are in order or not. The VAE does not do much better than random features for the small-sized bird, but very respectably for the large pipe, likely due to VAEs preference to capture larger global structure that more contributes to the reconstruction loss.
Sonic, on the other hand, has much more complex dynamics and graphics than Flappy Bird. As a result the classification performance is not as strong. For example, the inverse model does much worse at capturing the position of Sonic. This is likely due to the more inconsistent response of Sonic to action commands. For example, when Sonic is already in the air jumping, the right command has no effect. The ambiguity to which action was called for what pairs of frames, likely causes the inverse model to do bad at its task and thus not learn good features. Moreover, the frame moves up in response to Sonic jumping, so Sonic's exact pixels are not the only thing that change in response to jump, making it tougher for the inverse model to focus in on Sonic. Moreover, sequence verification methods like TDC and tuple verification are also tripped up by Sonic. This is most likely because even though Sonic moves to the right fairly consistently, the background moves in the x position not Sonic. Normally, that would be no problem for TDC and tuple verification, like in Flappy Bird. However, there is no consistent landmark in the background for these methods to use like the pipes in Flappy Bird. VAEs also do worse than they do at Flappy Bird. However, they do relatively better than any other self-supervised methods. Likely, this is because they are not affected by weird dynamics of the

environment.

2.4.3 Generalizing to New Levels and Colors

We can also show how well these features generalize to new situations. Theoretically, if the features are truly, robustly capturing objects of interest, changing the level or the colors of the environment, should not affect a linear classifier's ability to localize objects of interest. We test this out by looking at zero-shot linear probe accuracy with the background, pipe, and bird colors changed for Flappy Bird and on a new level for Sonic.We see these results in table 2.2. Unsurprisingly, we find the performance decreases for all self-supervised methods in Flappy Bird. Surprisingly, the features from TDC generalize better than the inverse model for classifying bird's location. Potentially, this is because the color of the bird changes and the features from the inverse model are more specific to the exact appearance of the bird from the setup it was trained on. TDC features, on the hand, may encode the bird based on where it is relative to the pipes, and less so on exactly how it looks and for the same reason, TDC features are able to capture the pipes, despite their different color. The VAE features' performance unsurprisingly drops for both objects, as the global structure that they learn to encode completely changes with the new colors in the `FlappyBirdNight` setup.

2.4.4 Qualititative Inspection of Feature Maps

We show qualitative inspection by superimposing a frame's feature map on top of the frame itself. We pick the most compelling feature map for each frame, which we show in figures 2.2 and 2.3. Confirming our hypothesis from 2.4.2, we see for Flappy Bird, the inverse model feature map focuses on the bird and the TDC feature map focuses on the pipe. Interestingly enough, tuple verification keys in on the top half of the pipe and the VAE activates on everything in the frame, but the pipes. For Sonic, things are not as clean and interpretable. As we see in figure 2.3, the inverse model feature map and the TDC one focus in on a cloud, perhaps mistaking it for Sonic, and the tuple verification map keys in on nothing at all. The VAE feature map, unsurprisingly, activates on important, ubiquitous objects for reconstructing the frame, like the tree and the bush. None of these representative feature maps key in on Sonic himself, which agrees with the poor quantitative classification accuracy

Table 2.1 – Extracting Position Information We train a linear classifier on top of feature spaces from each method and measure the classification accuracy for the various position of various objects in the game, y position of the bird and x position of the first pipe for Flappy Bird and y position of Sonic in Sonic

Method	Flappy Bird		Sonic
	Bird Y Pos Acc (%)	Pipe X Pos Acc(%)	Y Pos (%)
RandCNN	35.57	52.13	23.88
VAE	36.20	76.52	**53.82**
Inv Model	**91.67**	81.64	25.83
tuple verification	75.41	87.08	11.2
TDC	56.06	**92.36**	10.42

Table 2.2 – Generalizing Extracting Position to New Levels and Colors We see how well the trained linear classifiers do in a zero shot transfer to new colors for Flappy Bird and a new level for Sonic

Method	FlappyBirdNight		Sonic GreenHillZone Act 2
	Bird Y Pos Acc (%)	Pipe X Pos Acc(%)	Y Pos (%)
RandCNN	34.03	7.41	28.31
VAE	2.53	2.0	**34.57**
Inv Model	36.5	8.90	27.28
tuple verification	1.88	1.74	14.4
TDC	**46.9**	**16.22**	13.59

results in table 2.1.

2.5 Related Work

This paper is not the first paper to quantitatively and qualitatively compare features from self-supervised methods in interactive environments. [Burda et al., 2018a] compare the feature spaces learned from VAEs, Inverse Models, and Random CNN's and raw pixels (but not sequence verification methods) across a large variety of games. They even measure the generalization of these feature spaces to new, unseen environments. However, all their evaluations are in the context of how well an agent explores its environment with this feature space using extrinsic rewards and other measures of exploration. Additionally, [Shelhamer et al., 2016] studies VAEs, Inverse Models, and a sequence verification task in RL environments, but they evaluate these self-supervised methods as auxiliary tasks paired with a traditional extrinsic reward policy gradient algorithm, A3C [Mnih et al., 2016], using empirical returns from extrinsic rewards as a measure of utility of each feature space. Lastly, trying to infer the position of salient objects has been explored a lot in robotics in a field called state inference [Jonschkowski and Brock, 2015, Jonschkowski et al., 2017]. Moreover, [Raffin et al., 2018, Lesort et al., 2018] look into using some self-supervised tasks for state inference, but they mostly measure performance on control tasks; they do not measure direct correlation or classification accuracy of the features to the true position of the object.

2.6 Conclusion

We have shown comparing methods on interactive environments reveals intriguing things about the self-supervised methods as well as the environments themselves. Particularly, we expose various traits of environments that some self-supervised tasks can take advantage of and others cannot. For example, inverse models are very good at localizing what they can control even when it is small, but only when the dynamics are simple and predictable and the appearance of the agent itself is consistent. Temporal distance classifiers are very good at capturing things that move very predictably in time. Tuple verification encoders are good at capturing small and large objects in environments with pretty consistent graphics and dynamics. VAEs learn good features when the objects are big and repeatably show up in the scene with consistent appearance.

2.6.1 Future Work

The very different behaviors of each method depending on the traits of the environment warrants further study covering more environments with more diverse appearances and dynamics, as well as a wider range of self-supervised methods. In addition, the fact that some methods excel at capturing or generalizing better than others depending on the environment motivates exploring potentially combining these methods by having a shared encoder body with multiple self-supervised heads. We hope that this study can open the door to more extensive, rigorous approaches for studying the capability of self-supervised methods and that its results can inspire new methods that learn even better features.

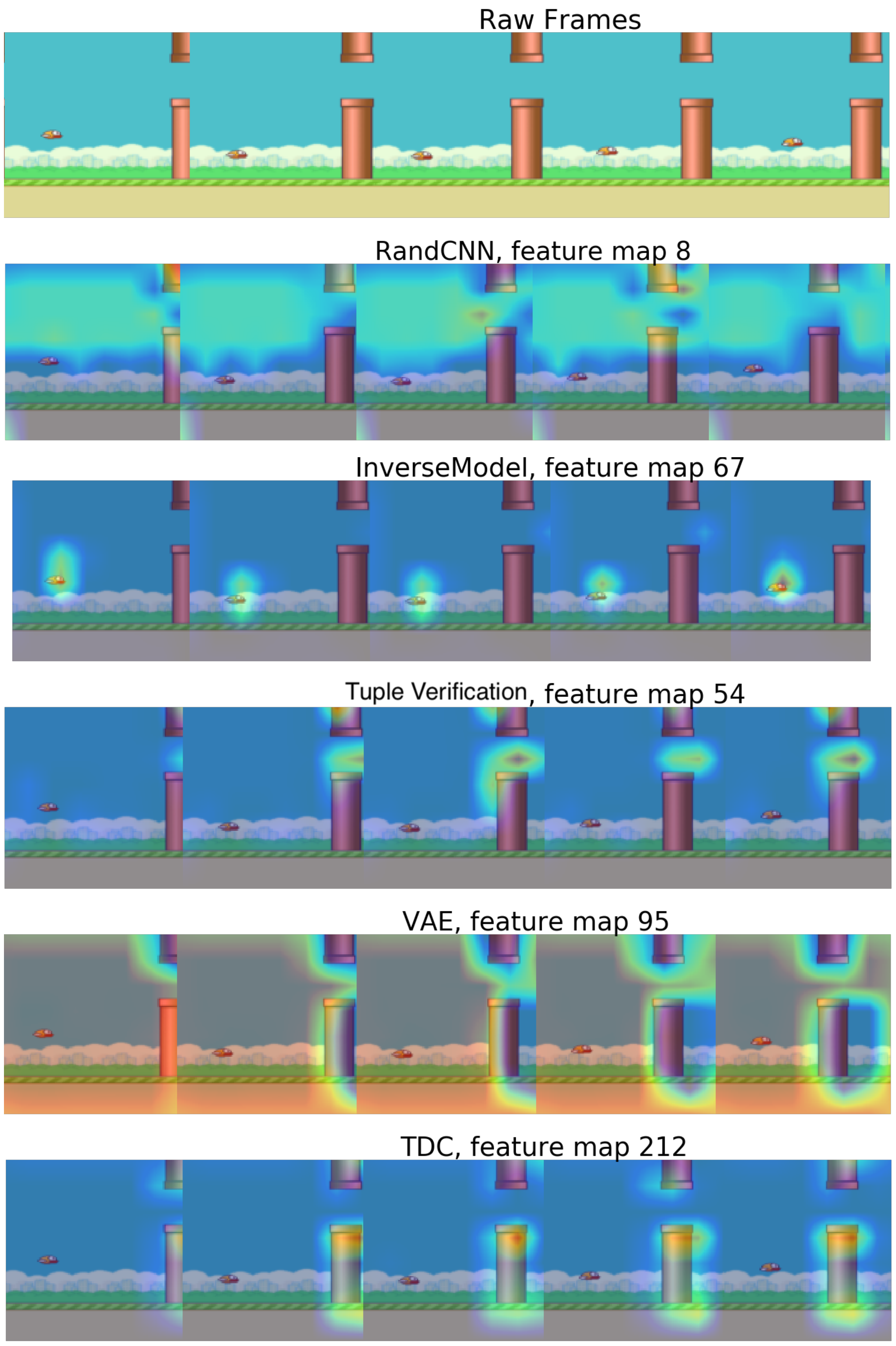


Figure 2.2 – Qualititative Inspection of Feature Maps Flappy Bird feature maps from the last convolutional layer of the encoder superimposed on top of a sequence of frames they are a function of. Red pixels are high values, blue are low values

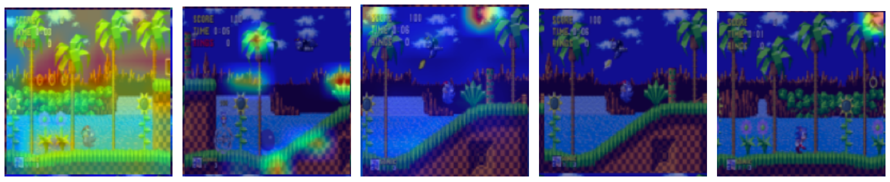


Figure 2.3 – Sonic feature maps from the last convolutional layer of the encoder superimposed on top of the frames they are a function of for from left: random CNN, VAE, inverse Model, tuple verification, and temporal distance classification. Red is high values, blue are low values

3 Unsupervised State Representation Learning in Atari

Unsupervised State Representation Learning in Atari

Ankesh Anand[1], **Evan Racah**[1], Sherjil Ozair [1], Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm

This chapter presents joint work with Ankesh Anand, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. It is an accepted paper to Neural Information Processing Systems (NeurIPS 2019) - Conference Track.

Contribution:
The initial idea of adding labels to all Atari games was conceived by Yoshua and Sherjil. I identified and tested all the 308 state variable labels for all 22 games for the AtariARI benchmark. I also coded up the interface for the gym wrappers to expose these labels during evaluation and I coded up the linear probes for evaluation. Moreover, I coded up the baselines random cnn, supervised, pixel-prediction, and VAE. I also did all post-processing, data analysis, and created all the tables in the paper. Lastly, I wrote all of section 3.4 and I contributed to the writing of section 3.5,3.6, and 3.7. Ankesh coded up and ran experiments for all contrastive tasks except CPC. He was the first to identify the ”incompleteness problem” in PCL-like tasks, which led to the conception of ST-DIM. Ankesh and I together conceived the idea of ST-DIM. Ankesh coded up ST-DIM and all its ablations and ran all the ST-DIM experiments. Ankesh coded up and trained the PPO agent and the interface to collect episodes and representations from the PPO agent. Ankesh created all figures and diagrams for the paper including the ablations diagram. He also acquired the commented source code for many of the Atari games. Ankesh also wrote some other sections of the paper. Sherjil coded up the CPC baseline. He also changed the code of the other contrastive tasks to incorporate InfoNCE instead of NCE. Sherjil also wrote the introduction, related work, and methods section,

1. equal contribution

as well as being a major contributor the discussion in 3.7. Sherjil also contributed some labels to Pong and Video Pinball. Yoshua proofread the paper and provided detailed feedback. Marc proofread the paper, provided feedback, and helped launch experiments on the MSR cluster. Devon provided high-level advice in weekly and sometimes twice-weekly meetings and helped with the text of the paper.

Affiliation

— Ankesh Anand, Mila, Université de Montréal

— Evan Racah, Mila, Université de Montréal

— Sherjil Ozair, Mila, Université de Montréal

— Yoshua Bengio, Mila, Université de Montréal

— Marc-Alexandre Côté, Microsoft Research

— R Devon Hjelm, Microsoft Research, Mila, Université de Montréal

3.1 Abstract

State representation learning, or the ability to capture latent generative factors of an environment, is crucial for building intelligent agents that can perform a wide variety of tasks. Learning such representations without supervision from rewards is a challenging open problem. We introduce a method that learns state representations by maximizing mutual information across spatially and temporally distinct features of a neural encoder of the observations. We also introduce a new benchmark based on Atari 2600 games where we evaluate representations based on how well they capture the ground truth state variables. We believe this new framework for evaluating representation learning models will be crucial for future representation learning research. Finally, we compare our technique with other state-of-the-art generative and contrastive representation learning methods.

3.2 Introduction

The ability to perceive and represent visual sensory data into useful and concise descriptions is considered a fundamental cognitive capability in humans [Marr, 1982, Gordon and Irwin, 1996], and thus crucial for building intelligent agents [Lake et al., 2017]. Representations that succinctly reflect the true state of the environment should allow agents to learn to act in those environments with fewer interactions, and effectively transfer knowledge across different tasks in the environment.

Recently, deep representation learning has led to tremendous progress in a variety of machine learning problems across numerous domains [Krizhevsky et al., 2012, Amodei et al., 2016, Wu et al., 2016, Mnih et al., 2015, Silver et al., 2016]. Typically, such representations are often learned via end-to-end learning using the signal from labels or rewards, which makes such techniques often very sample-inefficient. In contrast, human learning in the natural world appears to require little to no explicit supervision for perception [Gross, 1968].

Unsupervised [Dumoulin et al., 2017, Kingma and Welling, 2013, Dinh et al., 2017] and self-supervised representation learning [Pathak et al., 2016a, Doersch and Zisserman, 2017, Kolesnikov et al., 2019] have emerged as an alternative to supervised versions which can yield useful representations with reduced sample complexity. In the context of learning state representations [Lesort et al., 2018], current unsupervised methods rely on generative decoding of the data using either VAEs [Watter et al., 2015, Higgins et al., 2017b, Ha and Schmidhuber, 2018a, Duan, 2017] or prediction in pixel-space [Oh et al., 2015, Finn et al., 2016]. Since these objectives are based on reconstruction error in the pixel space, they are not incentivized to capture abstract latent factors and often default to capturing pixel level details.

In this work, we leverage recent advances in self-supervision that rely on scalable estimation of mutual information [Belghazi et al., 2018, van den Oord et al., 2018, Hjelm et al., 2019, Veličković et al., 2018], and propose a new contrastive state representation learning method named Spatiotemporal Deep Infomax (ST-DIM), which maximizes the mutual information across both the spatial and temporal axes.

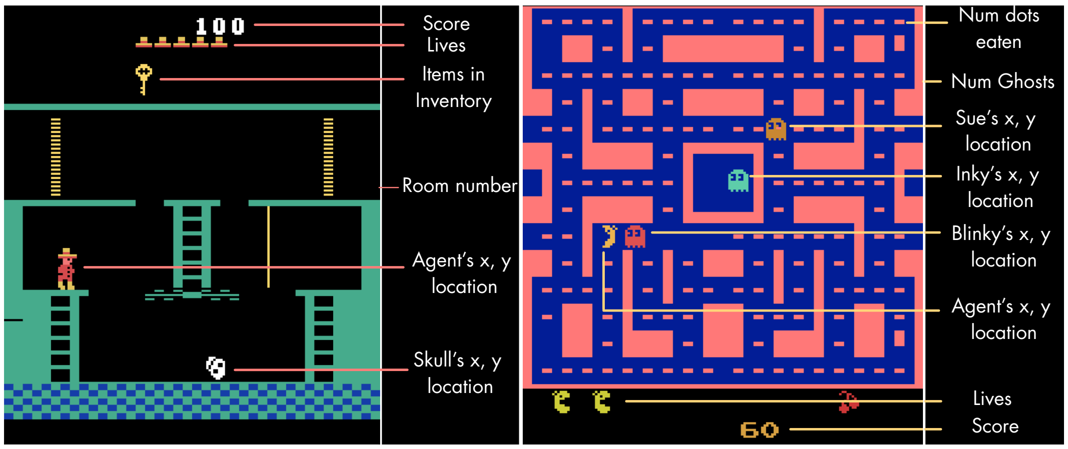


Figure 3.1 – We use a collection of 22 Atari 2600 games to evaluate state representations. We leveraged the source code of the games to annotate the RAM states with important state variables such as the location of various objects in the game. We compare various unsupervised representation learning techniques based on how well the representations linearly-separate the state variables. Shown above are examples of state variables annotated for Montezuma's Revenge and MsPacman.

To systematically evaluate the ability of different representation learning methods at capturing the true underlying factors of variation, we propose a benchmark based on Atari 2600 games using the Arcade Learning Environment [ALE, Bellemare et al., 2013]. A simulated environment provides access to the underlying generative factors of the data, which we extract using the source code of the games. These factors include variables such as the location of the player character, location of various items of interest (keys, doors, etc.), and various non-player characters, such as enemies (see figure 3.1). Performance of a representation learning technique in the Atari representation learning benchmark is then evaluated using *linear probing* [Alain and Bengio, 2017], i.e. the accuracy of linear classifiers trained to predict the latent generative factors from the learned representations.

Our contributions are the following

1. We propose a new self-supervised state representation learning technique which exploits the spatial-temporal nature of visual observations in a reinforcement learning setting.
2. We propose a new state representation learning benchmark using 22 Atari 2600 games based on the Arcade Learning Environment (ALE).
3. We conduct extensive evaluations of existing representation learning techniques on the proposed benchmark and compare with our proposed method.

3.3 Related Work

Unsupervised representation learning via mutual information estimation: Recent works in unsupervised representation learning have focused on extracting latent representations by maximizing a lower bound on the mutual information between the representation and the input. Belghazi et al. [2018] estimate the mutual information with neural networks using the Donsker-Varadhan representation of the KL divergence [Donsker and Varadhan, 1983], while Chen et al. [2016] use the variational bound from Barber and Agakov [2003] to learn discrete latent representations. Hjelm et al. [2019] learn representations by maximizing the Jensen-Shannon divergence between joint and product of marginals of an image and its patches. van den Oord et al. [2018] maximize mutual information using a multi-sample version of noise contrastive estimation [Gutmann and Hyvärinen, 2010, Ma and Collins, 2018]. See [Poole et al., 2019] for a review of different variational bounds for mutual information.

State representation learning: Learning better state representations is an active area of research within robotics and reinforcement learning. Jonschkowski and Brock [2015] and Jonschkowski et al. [2017] propose to learn representations using a set of handcrafted robotic priors. Several prior works use a VAE and its variations to learn a mapping from observations to state representations [Higgins et al., 2018, Watter et al., 2015, van Hoof et al., 2016]. Thomas et al. [2017] aims to learn the representations that maximize the causal relationship between the distributed policies and the representation of changes in the state. Recently, Cuccu et al. [2019] shows that visual processing and policy learning can be effectively decoupled in Atari games. Nachum et al. [2019] connects mutual information estimators to representation learning in hierarchical RL. Our work is also closely related to recent work in learning object-oriented representations [Burgess et al., 2019].

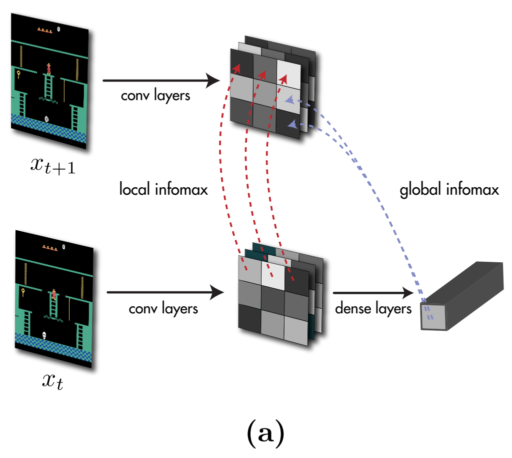
Evaluation frameworks of representations: Evaluating representations is an open problem, and doing so is usually domain specific. In vision tasks, it is common to evaluate based on the presence of linearly separable label-relevant information, either in the domain the representation was learned on [Coates et al., 2011] or in transfer learning tasks [Xian et al., 2018, Triantafillou et al., 2017]. In NLP, the SentEval [Conneau and Kiela, 2018] and GLUE [Wang et al., 2019] benchmarks provide a means of providing a more linguistic-specific understanding of what the model has learned, and these have become a standard tool in NLP research. Our evaluation framework can be thought of as a GLUE-like benchmarking tool for RL, providing a fine-grained understanding of how well the RL agent perceives the objects in the scene. Analogous to GLUE in NLP, we anticipate that our benchmarking tool will be useful in RL research for better designing components of agent learning.

3.4 Spatiotemporal Deep Infomax

We assume a setting where an agent interacts with an environment and observes a set of high-dimensional observations $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ across several episodes. Our goal is to learn an abstract representation of the observation that captures the underlying latent generative factors of the environment.

This representations should focus on high-level semantics (e.g., the concept of agents, enemies, objects, score, etc.) and ignore the low-level details such as the precise texture of the background, which warrants a departure from the class of methods that rely on a generative decoding of the full observation. Prior work in neuroscience [Friston, 2005, Rao and Ballard, 1999] has suggested that the brain maximizes *predictive information* [Bialek and Tishby, 1999] at an abstract level to avoid sensory overload. Predictive information, or the mutual information between consecutive states, has also been shown to be the organizing principle of retinal ganglion cells in salamander brains [Palmer et al., 2015]. Thus our representation learning approach relies on maximizing an estimate based on a lower bound on the mutual information over consecutive observations x_t and x_{t+1}.

3.4.1 Maximizing mutual information across space and time



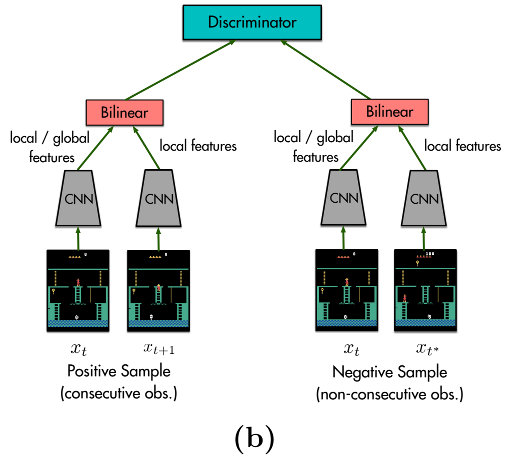


Figure 3.2 – A schematic overview of SpatioTemporal DeepInfoMax (ST-DIM). (a) shows the two different mutual information objectives: local infomax and global infomax. (b) shows a simplified version of the contrastive task we use to estimate mutual information. In practice, we use multiple negative samples.

Given a mutual information estimator, we follow DIM [Hjelm et al., 2019] and maximize a sum of patch-level mutual information objectives. The global objectives maximize the mutual information between the full observation at time t with small patches of the observation at time $t+1$. The representations of the small image patches are taken to be the hidden activations of the convolutional encoder applied to the full observation. The layer is picked appropriately to ensure that the hidden activations only have a limited receptive field corresponding to $1/16^{th}$ the size of the full observations. The local objective maximizes the mutual information between the local feature at time t with the corresponding local feature at time $t+1$. Figure 3.2 is a visual depiction of our model which we call Spatiotemporal Deep Infomax (ST-DIM).

It has been shown that mutual information bounds can be loose for large values of the mutual information [McAllester and Statos, 2018] and in practice fail to capture all the relevant features in the data [Ozair et al., 2019] when used to learn representations. To alleviate this issue, our approach constructs multiple small mutual information objectives (rather than a single large one) which are easier to estimate via lower bounds, which has been concurrently found to work well in the context of semi-supervised learning [Bachman et al., 2019].

For the mutual information estimator, we use infoNCE [van den Oord et al., 2018], a multi-sample variant of noise-contrastive estimation [Gutmann and Hyvärinen, 2010] that was also shown to work well with DIM. Let $\{(x_i, y_i)\}_{i=1}^N$ be a paired dataset of N samples from some joint distribution $p(x, y)$. For any index i, (x_i, y_i) is a sample from the joint $p(x, y)$ which we refer to as *positive examples*, and for any $i \neq j$, (x_i, y_j) is a sample from the product of marginals $p(x)p(y)$, which we refer to as *negative examples*. The InfoNCE objective learns a score function $f(x, y)$ which assigns large values to positive examples and small values to negative examples by maximizing the following bound [see van den Oord et al., 2018, Poole et al., 2019, for more details on this bound],

$$\mathcal{I}_{NCE}(\{(x_i, y_i)\}_{i=1}^N) = \sum_{i=1}^{N} \log \frac{\exp f(x_i, y_i)}{\sum_{j=1}^{N} \exp f(x_i, y_j)} \tag{3.1}$$

The above objective has also been referred to as *multi-class n-pair loss* [Sohn, 2016, Sermanet et al., 2018] and *ranking-based NCE* [Ma and Collins, 2018], and is similar to MINE [Belghazi et al., 2018] and the JSD-variant of DIM [Hjelm et al., 2019].

Following van den Oord et al. [2018] we use a bilinear model for the score function $f(x, y) = \phi(x)^T W \phi(y)$, where ϕ is the representation encoder. The bilinear model combined with the InfoNCE objective forces the encoder to learn *linearly predictable* representations, which we believe helps in learning representations at the semantic level. In our context, the positive examples correspond to pairs of consecutive observations (x_t, x_{t+1}) and negative samples correspond to pair to pair of non-consecutive observations (x_t, x_{t^*}), where t^* is a randomly sampled time index from the episode. For ST-DIM, the final score function for the global objective is $f_g(x_t, x_{t+1}) = \phi(x_t)^T W_g \phi_{l,m,n}(x_{t+1})$ and the score function of the local objective is $f_l(x_t, x_{t+1}) = \phi_{l,m,n}(x_t)^T W_l \phi_{l,m,n}(x_{t+1})$, where $\phi_{l,m,n}$ is the feature map at the l^{th} layer at the (m, n) spatial location.

3.5 The Atari Annotated RAM Interface (AtariARI)

Measuring the usefulness of a representation is still an open problem, as a core utility of representations is their use as feature extractors in tasks that are different from those used for training (e.g., *transfer learning*). Measuring classification performance, for example, may only reveal the amount of class-relevant information in a representation, but may not reveal other information useful for segmentation. It would be useful, then, to have a more *general* set of measures on the usefulness of a representation, such as ones that may indicate more general utility across numerous real-world tasks. In this vein, we assert that in the context of dynamic, visual, interactive environments, the capability of a representation to capture the underlying high-level factors of the state of an environment will be generally useful for a variety of downstream tasks such as prediction, control, and tracking.

We find video games to be a useful candidate for evaluating visual representation learning algorithms primarily because they are spatiotemporal in nature, which is (1) more realistic compared to static i.i.d. datasets and (2) prior work Hyvärinen et al. [2004], Locatello et al. [2019] have argued that without temporal structure, recovering the true underlying latent factors is undecidable. Apart from this, video games also provide ready access to the underlying ground truth states, unlike real-world datasets, which we need to evaluate performance of different techniques.

Annotating Atari RAM: ALE does not explicitly expose any ground truth state information. However, ALE does expose the RAM state (128 bytes per timestep) which are used by the game programmer to store important state information such as the location of sprites, the state of the clock, or the current room the agent is in. To extract these variables, we consulted commented disassemblies [Whalen and Taylor, 2008] (or source code) of Atari 2600 games which were made available by Engelhardt [2019] and Jentzsch and CPUWIZ [2019]. We were able to find and verify important state variables for a total of 22 games. Once this information is acquired, combining it with the ALE interface produces a wrapper that can automatically output a state label for every example frame generated from the game. We make this available with an easy-to-use *gym* wrapper, which returns this information with no change to existing code using *gym* interfaces. Table 3.1 lists the 22 games along with the categories of variables for each game. We describe the meaning of each category in the next section.

State variable categories: We categorize the state variables of all the games among six major categories: agent localization, small object localization, other localization, score/clock/lives/display, and miscellaneous. **Agent Loc.** (agent localization) refers to state variables that represent the x or y coordinates on the screen of any sprite controllable by actions. **Small Loc.** (small object localization) variables refer to the x or y screen position of small objects, like balls or missiles. Prominent examples include the ball in Breakout and Pong, and the torpedo in Seaquest. **Other Loc.** (other localization) denotes the x or y location of any other sprites, including enemies or large objects to pick up. For example, the location of ghosts in Ms Pacman or the ice floes in Frostbite. **Score/Clock/Lives/Display** refers to variables that track the score of the game, the clock, or the number of remaining lives the agent has, or some other display variable, like the oxygen meter in Seaquest. **Misc.** (Miscellaneous) consists of state variables that are largely specific to a game, and don't fall within one of the above mentioned categories. Examples include the existence of each block or pin in Breakout and Bowling, the room number in Montezuma's Revenge, or Ms. Pacman's facing direction.

Probing: Evaluating representation learning methods is a challenging open problem. The notion of *disentanglement* [Bengio, 2009, Bengio et al., 2013] has emerged as a way to measure the usefulness of a representation [Eastwood and Williams, 2018, Higgins et al., 2018]. In this work, we focus only on *explicitness*, i.e the degree to which underlying generative factors can be recovered using a *linear* transformation from the learned representation. This is standard methodology in the self-supervised representation learning literature [Doersch and Zisserman, 2017, van den Oord et al., 2018, Caron et al., 2018, Kolesnikov et al., 2019, Hjelm et al., 2019]. Specifically, to evaluate a representation we train linear classifiers predicting each state variable, and we report the mean F1 score.

3.6 Experimental Setup

We evaluate the performance of different representation learning methods on our benchmark. Our experimental pipeline consists of first training an encoder, then freezing its weights and evaluating its performance on linear probing tasks. For each identified generative factor in each game, we construct a linear probing task where the representation is trained to predict the ground truth value of that factor. Note that the gradients are not backpropagated through the encoder network, and only used to train the linear classifier on top of the representation.

3.6.1 Data preprocessing and acquisition

We consider two different modes for collecting the data: (1) using a random agent (steps through the environment by selecting actions randomly), and (2) using a PPO [Schulman et al., 2017] agent trained for 50M timesteps. For both these modes, we ensure there is enough data diversity by collecting data using 8 differently initialized workers. We also add additional stochasticity to the pretrained PPO agent by using an ϵ-greedy like mechanism wherein at each timestep we take a random action with probability ϵ[2].

3.6.2 Methods

In our evaluations, we compare the following methods:

1. Randomly-initialized CNN encoder (RANDOM-CNN).
2. Variational autoencoder (VAE) [Kingma and Welling, 2013] on raw observations.

2. For all our experiments, we used $\epsilon = 0.2$.

3. Next-step pixel prediction model (PIXEL-PRED) inspired by the ”No-action Feedforward” model from Oh et al. [2015].
4. Contrastive Predictive Coding (CPC) [van den Oord et al., 2018], which maximizes the mutual information between current latents and latents at a future timestep.
5. SUPERVISED model which learns the encoder and the linear probe using the labels. The gradients are backpropagated through the encoder in this case, so this provides a best-case performance bound.

All methods use the same base encoder architecture, which is the CNN from [Mnih et al., 2013], but adapted for the full 160x210 Atari frame size. To ensure a fair comparison, we use a representation size of 256 for each method. As a sanity check, we include a blind majority classifier (MAJ-CLF), which predicts label values based on the mode of the train set. More details in Appendix, section 3.9.

3.6.3 Probing

We train a different 256-way[3] linear classifier with the representation under consideration as input. We ensure the distribution of realizations of each state variable has high entropy by pruning any variable with entropy less than 0.6. We also ensure there are no duplicates between the train and test set. We train each linear probe with 35,000 frames and use 5,000 and 10,000 frames each for validation and test respectively. We use early stopping and a learning rate scheduler based on plateaus in the validation loss.

3.7 Results

We report the F1 averaged across all categories for each method and for each game in Table 3.2 for data collected by random agent. In addition, we provide a breakdown of probe results in each category, such as small object localization or score/lives classification in Table 3.3 for the random agent. We include the corresponding tables for these results with data collected by a pretrained PPO agent

3. Each RAM variable is a single byte thus has 256 possible values ranging from 0 to 255.

in tables 3.6 and 3.7. The results in table 3.2 show that ST-DIM largely outperforms other methods in terms of mean F1 score. In general, contrastive methods (ST-DIM and CPC) methods seem to perform better than generative methods (VAE and PIXEL-PRED) at these probing tasks. We find that RandomCNN is a strong prior in Atari games as has been observed before [Burda et al., 2018a], possibly due to the inductive bias captured by the CNN architecture empirically observed in [Ulyanov et al., 2018]. We find similar trends to hold on results with data collected by a PPO agent. Despite contrastive methods performing well, there is still a sizable gap between ST-DIM and the fully supervised approach, leaving room for improvement from new unsupervised representation learning techniques for the benchmark.

3.8 Discussion

Ablations: We investigate two ablations of our ST-DIM model: Global-T-DIM, which only maximizes the mutual information between the global representations and JSD-ST-DIM, which uses the NCE loss Hyvärinen and Pajunen [1999] instead of the InfoNCE loss, which is equivalent to maximizing the Jensen Shannon Divergence between representations. We report results from these ablations in Figure 3.3. We see from the results in that 1) the InfoNCE loss performs better than the JSD loss and 2) contrasting spatiotemporally (and not just temporally) is important across the board for capturing all categories of latent factors.

We found ST-DIM has two main advantages which explain its superior performance over other methods and over its own ablations. It captures small objects much better than other methods, and is more robust to the presence of easy-to-exploit features which hurts other contrastive methods. Both these advantages are due to ST-DIM maximizing mutual information of patch representations.

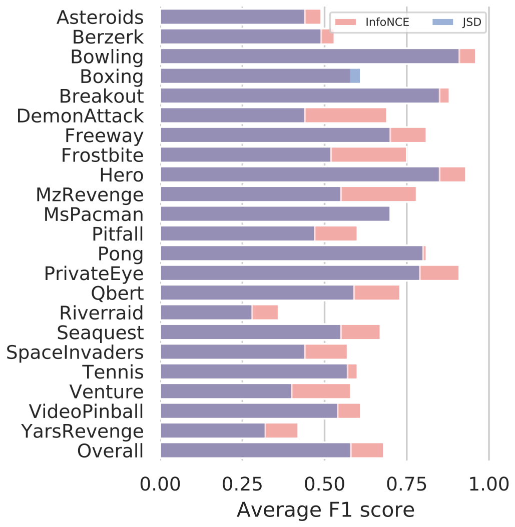
Capturing small objects: As we can see in Table 3.3, ST-DIM performs better at capturing small objects than other methods, especially generative models like VAE and pixel prediction methods. This is likely because generative models try to model every pixel, so they are not penalized much if they fail to model the few pixels that make up a small object. Similarly, ST-DIM holds this same advantage over Global-T-DIM (see Table 3.9), which is likely due to the fact that Global-T-DIM is not penalized if its global representation fails to capture features from some patches of the frame.

Robust to presence of easy-to-exploit features: Representation learning with mutual information or contrastive losses often fail to capture all salient features if a few easy-to-learn features are sufficient to saturate the objective. This phenomenon has been linked to the looseness of mutual information lower bounds [McAllester and Statos, 2018, Ozair et al., 2019] and *gradient starvation* [Combes et al., 2018]. We see the most prominent example of this phenomenon in Boxing. The observations in Boxing have a clock showing the time remaining in the round. A representation which encodes the shown time can perform near-perfect predictions without learning any other salient features in the observation. Table 3.4 shows that CPC, Global T-DIM, and ST-DIM perform well at predicting the clock variable. However only ST-DIM does well on encoding the other variables such as the score and the position of the boxers.

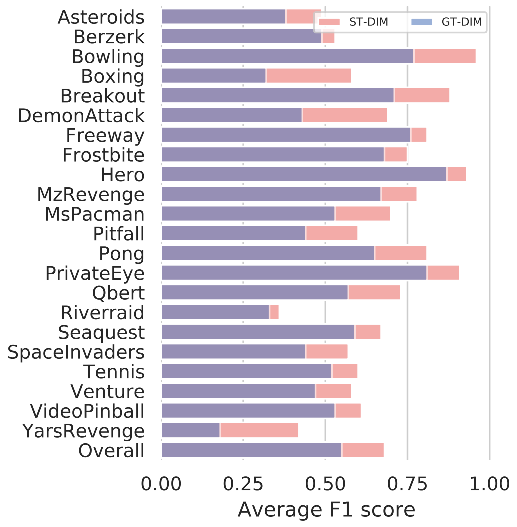
We also observe that the best generative model (PIXEL-PRED) does not suffer from this problem. It performs its worst on high-entropy features such as the clock and player score (where ST-DIM excels), and does slightly better than ST-DIM on low-entropy features which have a large contribution in the pixel space such as player and enemy locations. This sheds light on the qualitative difference between contrastive and generative methods: contrastive methods prefer capturing high-entropy features (irrespective of contribution to pixel space) while generative methods do not, and generative methods prefer capturing large objects which have low entropy. This complementary nature suggests hybrid models as an exciting direction of future work.

3.9 Architecture Details

All architectures below use the same encoder architecture as a base, which is the one used in Mnih et al. [2013] adapted to work for the full 160x210 frame size as shown in figure 3.4.



(a) InfoNCE vs JSD



(b) Effect of Spatial Loss

Figure 3.3 – Different ablations for the ST-DIM model

1. **Linear Probe**:
   The linear probe is a linear layer of width 256 with a softmax activation and trained with a cross-entropy loss.
2. **Majority Classifier** (maj-clsf):
   The majority classifier is parameterless and just uses the mode of the distribution of classes from the training set for each state variable and guesses that mode for every example on the test set at test time.
3. **Random-CNN**:
   The Random-CNN is the base encoder with randomly initialized weights and no training
4. **VAE and Pixel-Pred**:
   The VAE and Pixel Prediction model use the base encoder plus each have an extra 256 wide fully connected layer to parameterize the log variance for the VAE and to more closely resemble the *No Action Feed Forward* model from Oh et al. [2015]. In addition bith models have a deconvolutional network as a decoder, which is the exact transpose of the base encoder in figure 3.4.
5. **CPC**:
   CPC uses the same architecture as described in van den Oord et al. [2018] with our base encoder from figure 3.4 being used as the image encoder g_{enc}.

6. **ST-DIM (and its ablations)**:
   ST-DIM and the two ablations, JSD-ST-DIM and Global-T-DIM, all use the same architecture which is the base encoder plus a 1x256x256 bilinear layer.
7. **Supervised**:
   The supervised model is our base encoder plus our linear probe trained end-to-end with the ground truth labels.
8. **PPO Features** (section 3.13):
   The PPO model is our base encoder plus two linear layers for the policy and the value function, respectively.

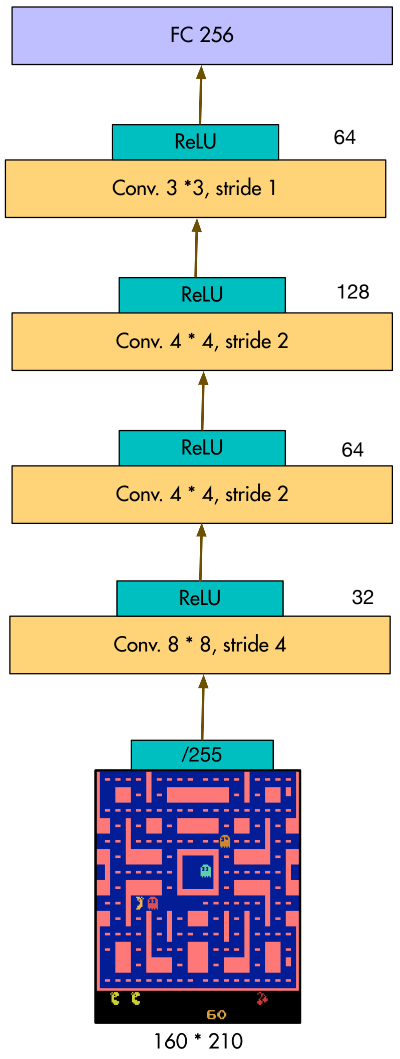


Figure 3.4 – The base encoder architecture used for all models in this work

3.10 Preprocessing and Hyperparameters

We preprocess frames primarily in the same way as described in Mnih et al. [2013], with the key difference being we use the full 160x210 images for all our experiments instead of downsampling to 84x84. Table 3.5 lists the hyper-parameters we use across all games. For all our experiments, we use a learning rate scheduler based on plateaus in the validation loss (for both contrastive training and probing).

Compute infrastructure: We run our experiments on a autoscaling-cluster with multiple P100 and V100 GPUs. We use 8 cores per machines to distribute data collection across different workers.

3.11 Results with Probes Trained on Data Collected By a Pretrained RL agent

In addition to evaluating on data collected by a random agent, we also evaluate different representation learning methods on data collected by a pretrained PPO [Schulman et al., 2017] agent. Specifically, we use a PPO agent trained for 50M steps on each game. We choose actions stochastically by sampling from the PPO agent's action distribution at every time step, and inject additional stochasticity by using an ϵ-greedy mechanism with $\epsilon = 0.2$. Table 3.6 shows the game-by-game breakdown of mean F1 probe scores obtained by each method in this evaluation setting. Table 3.7 additionally shows the category-wise breakdown of results for each method. We observe a similar trend in performance as observed earlier with a random agent.

3.12 More Detailed Ablation Results

We expand on the results reported on different ablations (JSD-ST-DIM and Global-T-DIM) of STDIM in the main text, and provide a game by game breakdown of results in Table 3.8, and a category-wise breakdown in Table 3.9.

3.13 Probing Pretrained RL Agents

We make a first attempt at examining the features that RL agents learn. Specifically, we train linear probes on the representations from PPO agents that were trained for 50 million frames. The architecture of the PPO agent is described in section 3.9. As we see from table 3.10, the features perform poorly in the probing tasks compared to the baselines. Kansky et al. [2017], Zhang et al. [2018] have also argued that model-free agents have trouble encoding high level state information. However, we note that these are preliminary results and require thorough investigation over different policies and models.

3.14 Conclusion

We present a new representation learning technique which maximizes the mutual information of representations across spatial and temporal axes. We also propose a new benchmark for state representation learning based on the Atari 2600 suite of games to emphasize learning multiple generative factors. We demonstrate that the proposed method excels at capturing the underlying latent factors of a state even for small objects or when a large number of objects are present, which prove difficult for generative and other contrastive techniques, respectively. We have shown that our proposed benchmark can be used to study qualitative and quantitative differences between representation learning techniques, and hope that it will encourage more research in the problem of state representation learning.

Acknowledgements

We are grateful for the collaborative research environment provided by Mila and Microsoft Research. We thank Aaron Courville, Chris Pal, Remi Tachet, Eric Yuan, Chinwei-Huang, Khimya Khetrapal and Tristan Deleu for helpful discussions and feedback during the course of this work. We would also like to thank the developers of PyTorch Paszke et al. [2017] and Weights&Biases.

Table 3.1 – Number of ground truth labels available in the benchmark for each game across each category. Localization is shortened for local. See section 3.5 for descriptions and examples for each category.

GAME	AGENT LOCAL.	SMALL OBJECT LOCAL.	OTHER LOCAL.	SCORE/CLOCK LIVES DISPLAY	MISC	OVERALL
ASTEROIDS	2	4	30	3	3	41
BERZERK	2	4	19	4	5	34
BOWLING	2	2	0	2	10	16
BOXING	2	0	2	3	0	7
BREAKOUT	1	2	0	1	31	35
DEMONATTACK	1	1	6	1	1	10
FREEWAY	1	0	10	1	0	12
FROSTBITE	2	0	9	4	2	17
HERO	2	0	0	3	3	8
MONTEZUMAREVENGE	2	0	4	4	5	15
MSPACMAN	2	0	10	2	3	17
PITFALL	2	0	3	0	0	5
PONG	1	2	1	2	0	6
PRIVATEEYE	2	0	2	4	2	10
QBERT	3	0	2	0	0	5
RIVERRAID	1	2	0	2	0	5
SEAQUEST	2	1	8	4	3	18
SPACEINVADERS	1	1	2	2	1	7
TENNIS	2	2	2	2	0	8
VENTURE	2	0	12	3	1	18
VIDEOPINBALL	2	2	0	2	0	6
YARSREVENGE	2	4	2	0	0	8
TOTAL	39	27	124	49	70	308

Table 3.2 – Probe F1 scores averaged across categories for each game (data collected by random agents)

Game	MAJ-CLF	RANDOM-CNN	VAE	PIXEL-PRED	CPC	ST-DIM	SUPERVISED
ASTEROIDS	0.28	0.34	0.36	0.34	0.42	**0.49**	0.52
BERZERK	0.18	0.43	0.45	**0.55**	**0.56**	0.53	0.68
BOWLING	0.33	0.48	0.50	0.81	0.90	**0.96**	0.95
BOXING	0.01	0.19	0.20	0.44	0.29	**0.58**	0.83
BREAKOUT	0.17	0.51	0.57	0.70	0.74	**0.88**	0.94
DEMONATTACK	0.16	0.26	0.26	0.32	0.57	**0.69**	0.83
FREEWAY	0.01	0.50	0.01	**0.81**	0.47	**0.81**	0.98
FROSTBITE	0.08	0.57	0.51	0.72	**0.76**	**0.75**	0.85
HERO	0.22	0.75	0.69	0.74	0.90	**0.93**	0.98
MONTEZUMAREVENGE	0.08	0.68	0.38	0.74	0.75	**0.78**	0.87
MSPACMAN	0.10	0.49	0.56	**0.74**	0.65	0.72	0.87
PITFALL	0.07	0.34	0.35	0.44	0.46	**0.60**	0.83
PONG	0.10	0.17	0.09	0.70	0.71	**0.81**	0.87
PRIVATEEYE	0.23	0.70	0.71	0.83	0.81	**0.91**	0.97
QBERT	0.29	0.49	0.49	0.52	0.65	**0.73**	0.76
RIVERRAID	0.04	0.34	0.26	**0.41**	**0.40**	0.36	0.57
SEAQUEST	0.29	0.57	0.56	0.62	**0.66**	**0.67**	0.85
SPACEINVADERS	0.14	0.41	0.52	**0.57**	0.54	**0.57**	0.75
TENNIS	0.09	0.41	0.29	0.57	**0.60**	**0.60**	0.81
VENTURE	0.09	0.36	0.38	0.46	0.51	**0.58**	0.68
VIDEOPINBALL	0.09	0.37	0.45	0.57	0.58	**0.61**	0.82
YARSREVENGE	0.01	0.22	0.08	0.19	0.39	**0.42**	0.74
MEAN	0.14	0.44	0.40	0.58	0.61	**0.68**	0.82

Table 3.3 – Probe F1 scores for different methods averaged across all games for each category (data collected by random agents)

		RANDOM					
Category	MAJ-CLF	CNN	VAE	PIXEL-PRED	CPC	ST-DIM	SUPERVISED
Small Loc.	0.14	0.19	0.18	0.31	0.42	**0.51**	0.66
Agent Loc.	0.12	0.31	0.32	0.48	0.43	**0.58**	0.81
Other Loc.	0.14	0.50	0.39	0.61	0.66	**0.69**	0.80
Score/Clock/Lives/Display	0.13	0.58	0.54	0.76	0.83	**0.87**	0.91
Misc.	0.26	0.59	0.63	0.70	0.71	**0.75**	0.83

Table 3.4 – Breakdown of F1 Scores for every state variable in Boxing for ST-DIM, CPC, and Global-T-DIM, an ablation of ST-DIM that removes the spatial contrastive constraint, for the game Boxing

METHOD	VAE	PIXEL-PRED	CPC	GLOBAL-T-DIM	ST-DIM
CLOCK	0.03	0.27	0.79	0.81	**0.92**
ENEMY_SCORE	0.19	0.58	0.59	**0.74**	0.70
ENEMY_X	0.32	0.49	0.15	0.17	**0.51**
ENEMY_Y	0.22	**0.42**	0.04	0.16	0.38
PLAYER_SCORE	0.08	0.32	0.56	0.45	**0.88**
PLAYER_X	0.33	0.54	0.19	0.13	**0.56**
PLAYER_Y	0.16	**0.43**	0.04	0.14	0.37

Table 3.5 – Preprocessing steps and hyperparameters

Parameter	**Value**
Image Width	160
Image Height	210
Grayscaling	Yes
Action Repetitions	4
Max-pool over last N action repeat frames	2
Frame Stacking	None
End of episode when life lost	Yes
No-Op action reset	Yes
Batch size	64
Sequence Length (CPC)	100
Learning Rate (Training)	3e-4
Learning Rate (Probing, non supervised)	5e-2
Learning Rate (Probing, supervised)	3e-4
Entropy Threshold	0.6
Encoder training steps	70000
Probe training steps	35000
Probe test steps	10000

Table 3.6 – Probe F1 scores for all games for data collected by a pretrained PPO (50M steps) agent

GAME	MEAN AGENT REWARDS	MAJ-CLF	RANDOM-CNN	VAE	PIXEL-PRED	CPC	ST-DIM	SUPERVISED
ASTEROIDS	489862.00	0.23	0.31	0.35	0.31	0.38	**0.40**	0.56
BERZERK	1913.00	0.13	0.33	0.35	0.39	0.38	**0.43**	0.61
BOWLING	29.80	0.23	0.61	0.51	0.81	0.90	**0.98**	0.98
BOXING	93.30	0.05	0.30	0.32	0.57	0.32	**0.66**	0.87
BREAKOUT	580.40	0.09	0.34	0.59	0.47	0.55	**0.66**	0.87
DEMONATTACK	428165.00	0.03	0.19	0.18	0.26	0.43	**0.58**	0.76
FREEWAY	33.50	0.01	0.36	0.02	**0.60**	0.38	**0.60**	0.76
FROSTBITE	3561.00	0.13	0.57	0.46	0.70	**0.74**	0.69	0.85
HERO	44999.00	0.12	0.54	0.60	0.68	**0.86**	0.77	0.96
MZREVENGE	0.00	0.08	0.68	0.58	0.72	**0.77**	**0.76**	0.88
MSPACMAN	4588.00	0.07	0.34	0.36	**0.52**	0.45	0.49	0.71
PITFALL	0.00	0.16	0.39	0.37	0.53	0.69	**0.74**	0.92
PONG	21.00	0.02	0.10	0.24	0.67	0.63	**0.79**	0.87
PRIVATEEYE	-10.00	0.24	0.71	0.69	0.87	0.83	**0.91**	0.99
QBERT	30590.00	0.06	0.36	0.38	0.39	**0.51**	0.48	0.65
RIVERRAID	20632.00	0.04	0.25	0.21	**0.34**	0.31	0.22	0.59
SEAQUEST	1620.00	0.29	0.64	0.58	**0.75**	0.69	**0.75**	0.90
SPACEINVADERS	2892.50	0.02	0.28	0.30	**0.41**	0.32	**0.41**	0.65
TENNIS	-4.30	0.15	0.25	0.13	**0.65**	0.63	**0.65**	0.61
VENTURE	0.00	0.05	0.32	0.36	0.37	0.50	**0.59**	0.69
VIDEOPINBALL	356362.00	0.13	0.36	0.42	**0.56**	**0.57**	0.54	0.79
YARSREVENGE	5520.00	0.03	0.14	0.26	0.23	0.38	**0.43**	0.74
MEAN	-	0.11	0.38	0.38	0.54	0.56	**0.62**	0.78

Table 3.7 – Probe F1 scores for different methods averaged across all games for each category (data collected by a pretrained PPO (50M steps) agent

CATEGORY	MAJ-CLF	RANDOM-CNN	VAE	PIXEL-PRED	CPC	ST-DIM	SUPERVISED
Small Loc.	0.10	0.13	0.14	0.27	0.31	**0.41**	0.65
Agent Loc.	0.11	0.34	0.34	0.48	0.45	**0.54**	0.83
Other Loc.	0.14	0.47	0.38	0.56	0.58	**0.61**	0.74
Score/Clock/Lives/Display	0.05	0.44	0.50	0.71	0.74	**0.80**	0.90
Misc.	0.19	0.53	0.57	0.62	0.65	**0.67**	0.83

Table 3.8 – Probe F1 scores for different ablations of ST-DIM for all games averaged across each category (data collected by random agents)

	JSD-ST-DIM	GLOBAL-T-DIM	ST-DIM
ASTEROIDS	0.44	0.38	**0.49**
BERZERK	0.49	0.49	**0.53**
BOWLING	0.91	0.77	**0.96**
BOXING	**0.61**	0.32	0.58
BREAKOUT	0.85	0.71	**0.88**
DEMONATTACK	0.44	0.43	**0.69**
FREEWAY	0.70	0.76	**0.81**
FROSTBITE	0.52	0.68	**0.75**
HERO	0.85	0.87	**0.93**
MONTEZUMAREVENGE	0.55	0.67	**0.78**
MSPACMAN	**0.70**	0.53	**0.70**
PITFALL	0.47	0.44	**0.60**
PONG	**0.80**	0.65	**0.81**
PRIVATEEYE	0.79	0.81	**0.91**
QBERT	0.59	0.57	**0.73**
RIVERRAID	0.28	0.33	**0.36**
SEAQUEST	0.55	0.59	**0.67**
SPACEINVADERS	0.44	0.44	**0.57**
TENNIS	0.57	0.52	**0.60**
VENTURE	0.40	0.47	**0.58**
VIDEOPINBALL	0.54	0.53	**0.61**
YARSREVENGE	0.32	0.18	**0.42**
MEAN	0.58	0.55	**0.68**

Table 3.9 – Different ablations of ST-DIM. F1 scores for for each category averaged across all games (data collected by random agents)

	JSD-ST-DIM	GLOBAL-T-DIM	ST-DIM
Small Loc.	0.44	0.37	**0.51**
Agent Loc.	0.47	0.43	**0.58**
Other Loc.	0.64	0.53	**0.69**
Score/Clock/Lives/Display	0.69	0.76	**0.86**
Misc.	0.64	0.66	**0.74**

Table 3.10 – Probe results on features from a PPO agent trained on 50 million timesteps compared with a majority classifier and random-cnn baseline. The probes for all three methods are trained with data from the PPO agent that was trained for 50M frames

	MAJ-CLF	RANDOM-CNN	PRETRAINED-RL-AGENT
ASTEROIDS	0.23	**0.31**	**0.31**
BERZERK	0.13	**0.33**	0.30
BOWLING	0.23	**0.61**	0.48
BOXING	0.05	**0.30**	0.12
BREAKOUT	0.09	**0.34**	0.23
DEMONATTACK	0.03	**0.19**	0.16
FREEWAY	0.01	**0.36**	0.26
FROSTBITE	0.13	**0.57**	0.43
HERO	0.12	**0.54**	0.42
MONTEZUMAREVENGE	0.08	**0.68**	0.07
MSPACMAN	0.06	**0.34**	0.26
PITFALL	0.16	**0.39**	0.23
PONG	0.02	**0.10**	0.09
PRIVATEEYE	0.24	**0.71**	0.31
QBERT	0.06	**0.36**	0.34
RIVERRAID	0.04	**0.25**	0.10
SEAQUEST	0.29	**0.64**	0.50
SPACEINVADERS	0.02	**0.28**	0.19
TENNIS	0.15	0.25	**0.66**
VENTURE	0.05	**0.32**	0.08
VIDEOPINBALL	0.13	**0.36**	0.21
YARSREVENGE	0.03	**0.14**	0.09
MEAN	0.11	**0.38**	0.27

4 Conclusion

In this work we first introduced the field of unsupervised representation learning and we talked about why representation learning is important for machine learning in general. Moreover, we discussed its potential for use in reinforcement learning. We then discussed several common approaches for representation learning, namely generative methods and discriminative methods. We also touched on an emerging subfield in representation learning: self-supervised learning. Furthermore, we focused on a special subset of self-supervised techniques called contrastive unsupervised representation learning (CURL) methods. Lastly, we closed the background section by discussing approaches for evaluating the utility of a representation.

In our first article, Supervise Thyself, we took several off-the-shelf self-supervised methods, applied them to reinforcement learning problems, learned representations and then evaluated them. We discovered that in some cases self-supervised techniques can learn very useful representations. However, we realized that not only was each self-supervised technique only good at representing one concept, but that their success and what they represented depended heavily on the dynamics and graphics of the environment. We concluded that we needed to look at a larger, more general suite of environments as well as a general class of methods.

Our second article, Unsupervised State Representation Learning in Atari, extended the work of the first paper in several ways. First, we proposed a large benchmark of 22 Atari games with labels for all the high-level factors in order to allow for a more thorough evaluation of representations learned by these representation learning methods. Furthermore, we decided to focus on a more general class of self-supervised methods: contrastive methods, instead of a random collection of off-the-shelf self-supervised methods. This allowed us to appropriately ablate our methods and make more general conclusions. Lastly, we identified a common failure mode in contrastive unsupervised representation learning (CURL) and contributed a new CURL method that empirically did not suffer this limitation.

Representation learning has the potential to be a very powerful tool for machine learning. Exactly how to learn and measure these representations is still a challenge. In this work we have shown some promising methods of extracting and evaluating these representations. However, representation learning is still in its infancy. Particularly, there is still no agreement in the community on the best way to evaluate representations nor is there a unified agreed upon definition for key terms like self-supervision or disentangling. Representation learning specifically for RL is even less mature as it has yet to be definitively shown that pretrained self-supervised representations are competitive with model-free reinforcement learning approaches, which optimize the sum of rewards end-to-end. We hope that future work can build upon our promising results by refining not only contrastive methods, but evaluating them by more thoroughly testing them to ensure they facilitate high performance on downstream tasks that we care about.

Bibliographie

Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 37–45, 2015.

Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, pages 5074–5082, 2016.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *International Conference on Learning Representations (Workshop Track)*, 2017.

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182, 2016.

Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1-2):5–43, 2003.

Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning. *arXiv preprint arXiv:1902.09229*, 2019.

Yusuf Aytar, Tobias Pfaff, David Budden, Tom Le Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by watching youtube. *arXiv preprint arXiv:1805.11592*, 2018.

Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910*, 2019.

David Barber and Felix Agakov. The im algorithm: A variational approach to information maximization. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, pages 201–208, Cambridge, MA, USA, 2003. MIT Press. URL http://dl.acm.org/citation.cfm?id=2981345.2981371.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *Proceedings of the 35th International Conference on Machine Learning*, pages 531–540, 2018. URL http://proceedings.mlr.press/v80/belghazi18a.html.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Yoshua Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 35(8):1798–1828, 2013.

William Bialek and Naftali Tishby. Predictive information. *arXiv preprint cond-mat/9902341*, 1999.

Christopher M Bishop. *Pattern recognition and machine learning.* springer, 2006.

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

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018a.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018b.

Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018.

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

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

Jongwook Choi, Yijie Guo, Marcin Moczulski, Junhyuk Oh, Neal Wu, Mohammad Norouzi, and Honglak Lee. Contingency-aware exploration in reinforcement learning. *arXiv preprint arXiv:1811.01483*, 2018.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

Remi Tachet des Combes, Mohammad Pezeshki, Samira Shabanian, Aaron Courville, and Yoshua Bengio. On the learning dynamics of deep neural networks. *arXiv preprint arXiv:1809.06848*, 2018.

Alexis Conneau and Douwe Kiela. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*, 2018.

Giuseppe Cuccu, Julian Togelius, and Philippe Cudré-Mauroux. Playing atari with six neurons. *International Conference on Autonomous Agents and Multiagent Systems*, 2019.

Oxford English Dictionary. Oxford english dictionary. *Simpson, JA & Weiner, ESC*, 1989.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *International Conference on Learning Representations (ICLR)*, 2017.

Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2051–2060, 2017.

Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *International Conference on Learning Representations (ICLR)*, 2017.

Monroe D Donsker and SR Srinivasa Varadhan. Asymptotic evaluation of certain markov process expectations for large time. iv. *Communications on Pure and Applied Mathematics*, 36(2):183–212, 1983.

Wuyang Duan. Learning state representations for robotic control: Information disentangling and multi-modal learning. Master's thesis, Delft University of Technology, 2017.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *International Conference on Learning Representations (ICLR)*, 2017.

Cian Eastwood and Christopher KI Williams. A framework for the quantitative evaluation of disentangled representations. 2018.

Steve Engelhardt. BJARS.com Atari Archives. `http://bjars.com`, 2019. [Online; accessed 1-March-2019].

Basura Fernando, Efstratios Gavves, Jose M Oramas, Amir Ghodrati, and Tinne Tuytelaars. Modeling video evolution for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5378–5387, 2015.

Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-supervised video representation learning with odd-one-out networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5729–5738. IEEE, 2017.

Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in neural information processing systems*, pages 64–72, 2016.

Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, Joelle Pineau, et al. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, 2018.

Karl Friston. A theory of cortical responses. *Philosophical transactions of the Royal Society B: Biological sciences*, 360(1456):815–836, 2005.

Sylvain Gelly, Karol Kurach, Marcin Michalski, and Xiaohua Zhai. Memgen: Memory is all you need. *arXiv preprint arXiv:1803.11203*, 2018.

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

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

Robert D Gordon and David E Irwin. What's in an object file ? evidence from priming studies. *Perception & Psychophysics*, 58(8):1260–1277, 1996.

Charles G Gross. Learning, perception, and the brain, 1968.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, pages 2450–2462, 2018a.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018b.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5):6, 2017a.

Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1480–1490. JMLR. org, 2017b.

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

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *International Conference on Learning Representations (ICLR)*, 2019.

Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

Aapo Hyvarinen and Hiroshi Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ica. In *Advances in Neural Information Processing Systems*, pages 3765–3773, 2016.

Aapo Hyvärinen and Petteri Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12(3):429–439, 1999.

Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent component analysis*, volume 46. John Wiley & Sons, 2004.

Aapo Hyvarinen, Hiroaki Sasaki, and Richard E Turner. Nonlinear ica using auxiliary variables and generalized contrastive learning. *arXiv preprint arXiv:1805.08651*, 2018.

AJ Hyvarinen and Hiroshi Morioka. Nonlinear ica of temporally dependent stationary sources. Proceedings of Machine Learning Research, 2017.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Dinesh Jayaraman and Kristen Grauman. Learning image representations tied to ego-motion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1413–1421, 2015.

Thomas Jentzsch and CPUWIZ. Atariage atari 2600 forums, 2019. URL http://atariage.com/forums/forum/16-atari-2600/.

Rico Jonschkowski and Oliver Brock. Learning state representations with robotic priors. *Autonomous Robots*, 39(3):407–428, 2015.

Rico Jonschkowski, Roland Hafner, Jonathan Scholz, and Martin Riedmiller. Pves: Position-velocity encoders for unsupervised learning of structured state representations. *arXiv preprint arXiv:1705.09805*, 2017.

Ken Kansky, Tom Silver, David A Mély, Mohamed Eldawy, Miguel Lázaro-Gredilla, Xinghua Lou, Nimrod Dorfman, Szymon Sidor, Scott Phoenix, and Dileep George. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1809–1818. JMLR. org, 2017.

Ilyes Khemakhem, Diederik P Kingma, and Aapo Hyvärinen. Variational autoencoders and nonlinear ica: A unifying framework. *arXiv preprint arXiv:1907.04809*, 2019.

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

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *ArXiv*, abs/1906.02691, 2019.

Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. *arXiv preprint arXiv:1901.09005*, 2019.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

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

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

Yann LeCun. Learning world models with self-supervised learning, 2018. Presented at ICML worlshop on Generative Modeling in RL.

Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. Ai safety gridworlds. *arXiv preprint arXiv:1711.09883*, 2017.

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

Francesco Locatello, Stefan Bauer, Mario Lucic, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. *International Conference on Machine Learning*, 2019.

Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. *arXiv preprint arXiv:1803.02893*, 2018.

William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016.

David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

Zhuang Ma and Michael Collins. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. *arXiv preprint arXiv:1809.01812*, 2018.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.* Henry Holt and Co., Inc., 1982. ISBN 0716715678.

Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

David McAllester and Karl Statos. Formal limitations on the measurement of mutual information. *arXiv preprint arXiv:1811.04251*, 2018.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*. MIT Press, 2013.

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

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2019.

Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta learn fast: A new benchmark for generalization in rl. *arXiv preprint arXiv:1804.03720*, 2018.

Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016.

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in neural information processing systems*, pages 2863–2871, 2015.

Georg Ostrovski, Marc G Bellemare, Aäron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2721–2730. JMLR. org, 2017.

Sherjil Ozair, Corey Lynch, Yoshua Bengio, Aaron Van den Oord, Sergey Levine, and Pierre Sermanet. Wasserstein dependency measure for representation learning. *arXiv preprint arXiv:1903.11780*, 2019.

Stephanie E Palmer, Olivier Marre, Michael J Berry, and William Bialek. Predictive information in a sensory population. *Proceedings of the National Academy of Sciences*, 112(22):6908–6913, 2015.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016a.

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016b.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. 2017a.

Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6024–6033. IEEE, 2017b.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Ben Poole, Sherjil Ozair, Aäron Van den Oord, Alexander A Alemi, and George Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, 2019.

Antonin Raffin, Ashley Hill, René Traoré, Timothée Lesort, Natalia Díaz-Rodríguez, and David Filliat. S-rl toolbox: Environments, datasets and evaluation metrics for state representation learning. *arXiv preprint arXiv:1809.09369*, 2018.

Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79, 1999.

Karl Ridgeway and Michael C Mozer. Learning deep disentangled embeddings with the f-statistic loss. In *Advances in Neural Information Processing Systems*, pages 185–194, 2018.

Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia ; Pearson Education Limited,, 2016.

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

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626, 2017.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. *arXiv preprint arXiv:1704.06888*, 2017.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141. IEEE, 2018.

Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. Loss is its own reward: Self-supervision for reinforcement learning. *arXiv preprint arXiv:1612.07307*, 2016.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1857–1865, 2016.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852, 2015.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*, 2018.

Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 2. MIT press Cambridge, 1998.

Norman Tasfi. Pygame learning environment. `https://github.com/ntasfi/PyGame-Learning-Environment`, 2016.

Valentin Thomas, Jules Pondard, Emmanuel Bengio, Marc Sarfati, Philippe Beaudoin, Marie-Jean Meurs, Joelle Pineau, Doina Precup, and Yoshua Bengio. Independently controllable factors. *arXiv preprint arXiv:1708.01289*, 2017.

Eleni Triantafillou, Richard Zemel, and Raquel Urtasun. Few-shot learning through an information retrieval lens, 2017.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9446–9454, 2018.

Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. In *NIPS*, 2016.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Herke van Hoof, Nutan Chen, Maximilian Karl, Patrick van der Smagt, and Jan Peters. 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. IEEE, 2016.

Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *European Conference on Computer Vision*, pages 402–419. Springer, 2018.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7.

Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2015.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pages 2746–2754, 2015.

Donglai Wei, Joseph J Lim, Andrew Zisserman, and William T Freeman. Learning and using the arrow of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8052–8060, 2018.

Zach Whalen and Laurie N Taylor. Playing the past. *History and Nostalgia in Video Games. Nashville, TN: Vanderbilt University Press*, 2008.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning - a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–1, 2018. ISSN 1939-3539. doi: 10.1109/tpami.2018.2857768. URL http://dx.doi.org/10.1109/TPAMI.2018.2857768.

Amy Zhang, Yuxin Wu, and Joelle Pineau. Natural environment benchmarks for reinforcement learning. *arXiv preprint arXiv:1811.06032*, 2018.

Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer, 2016.

Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, volume 1, page 5, 2017.