Université de Montréal

Data-Efficient Reinforcement Learning with Self-Predictive Representations

par Max Schwarzer

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, 2020

Université de Montréal
Faculté des arts et des sciences

Ce mémoire intitulé:

Data-Efficient Reinforcement Learning with Self-Predictive Representations

présenté par:

Max Schwarzer

a été évalué par un jury composé des personnes suivantes:

Phillipe Langlais,	président-rapporteur
Alain Tapp,	directeur de recherche
Aaron Courville,	codirecteur
Laurent Charlin,	membre du jury

Mémoire accepté le: . . . . . .

Résumé

L'efficacité des données reste un défi majeur dans l'apprentissage par renforcement profond. Bien que les techniques modernes soient capables d'atteindre des performances élevées dans des tâches extrêmement complexes, y compris les jeux de stratégie comme le StarCraft, les échecs, le shogi et le go, ainsi que dans des domaines visuels exigeants comme les jeux Atari, cela nécessite généralement d'énormes quantités de données interactives, limitant ainsi l'application pratique de l'apprentissage par renforcement. Dans ce mémoire, nous proposons la SPR, une méthode inspirée des récentes avancées en apprentissage auto-supervisé de représentations, conçue pour améliorer l'efficacité des données des agents d'apprentissage par renforcement profond. Nous évaluons cette méthode sur l'environement d'apprentissage Atari, et nous montrons qu'elle améliore considérablement les performances des agents avec un surcroît de calcul modéré. Lorsqu'on lui accorde à peu près le même temps d'apprentissage qu'aux testeurs humains, un agent d'apprentissage par renforcement augmenté de SPR atteint des performances surhumaines dans 7 des 26 jeux, une augmentation de 350% par rapport à l'état de l'art précédent, tout en améliorant fortement les performances moyennes et médianes. Nous évaluons également cette méthode sur un ensemble de tâches de contrôle continu, montrant des améliorations substantielles par rapport aux méthodes précédentes.

Le chapitre 1 présente les concepts nécessaires à la compréhension du travail présenté, y compris des aperçus de l'apprentissage par renforcement profond et de l'apprentissage auto-supervisé de représentations. Le chapitre 2 contient une description détaillée de nos contributions à l'exploitation de l'apprentissage de représentation auto-supervisé pour améliorer l'efficacité des données dans l'apprentissage par renforcement. Le chapitre 3 présente quelques conclusions tirées de ces travaux, y compris des propositions pour les travaux futurs.

mots-clés: apprentissage profond, apprentissage par renforcement, apprentissage auto-supervisé, apprentissage de représentations

Summary

Data efficiency remains a key challenge in deep reinforcement learning. Although modern techniques have been shown to be capable of attaining high performance in extremely complex tasks, including strategy games such as StarCraft, Chess, Shogi, and Go as well as in challenging visual domains such as Atari games, doing so generally requires enormous amounts of interactional data, limiting how broadly reinforcement learning can be applied. In this thesis, we propose SPR, a method drawing from recent advances in self-supervised representation learning designed to enhance the data efficiency of deep reinforcement learning agents. We evaluate this method on the Atari Learning Environment, and show that it dramatically improves performance with limited computational overhead. When given roughly the same amount of learning time as human testers, a reinforcement learning agent augmented with SPR achieves super-human performance on 7 out of 26 games, an increase of 350% over the previous state of the art, while also strongly improving mean and median performance. We also evaluate this method on a set of continuous control tasks, showing substantial improvements over previous methods.

Chapter 1 introduces concepts necessary to understand the work presented, including overviews of Deep Reinforcement Learning and Self-Supervised Representation learning. Chapter 2 contains a detailed description of our contributions towards leveraging self-supervised representation learning to improve data-efficiency in reinforcement learning. Chapter 3 provides some conclusions drawn from this work, including a number of proposals for future work.

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

Contents

List of Figures

List of Tables

List of Abbreviations

RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
DQN	Deep Q-Network
SAC	Soft Actor-Critic
TD	Temporal Difference
DDPG	Deep Deterministic Policy Gradient
TD3	Twin Delayed DDGP
ALE	Atari Learning Environment
KL	Kullbach-Liebler
MSE	Mean Squared Error
HNS	Human-Normalized Score
DNS	DQN-Normalized Score
SSL	Self-Supervised Learning
EMA	Exponential Moving Average
DM Control	DeepMind Control
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
NCE	Noise-Contrastive Estimation
SSL	Self-Supervised Learning
VAE	Variational Auto-Encoder
MI	Mutual Information
PDF	Probability Density Function
PMF	Probability Mass Function

Acknowledgments

I am very grateful for the support I have received at Mila. I particularly wish to thank Prof. Aaron Courville, who has taken immense amounts of time to introduce me to machine learning research and profoundly shaped my research career thus far. I likewise owe a debt to Philip Bachman, for going above and beyond the call of duty in guiding our joint work on SPR.

I also wish to thank my coauthors, who have made my experience at Mila as productive as it has been: Ankesh Anand, Jose Gallego, Ankit Vani, Rishab Goel, Simon Lacoste-Julien, Devon Hjelm, Philip Bachman, and, of course, Aaron Courville.

I would also like to thank friends at Mila for our interesting and fruitful conversations on machine learning and other topics, including the above as well as but not limited to Yutong Yan, Qizhen Zhang, Eeshan Dhekane, Alex Lamb, Disha Srivastava, Gunshi Gupta, Tristan Deleu, Samarth Sinha, Jingyi He, Phong Nguyen, Nitarshan Rajkumar, Bonnie Li, and Sai Rajeswar.

I am also extremely thankful for the astonishing computational resources provided to me at Mila, both through the Mila cluster and Compute Canada.

Finally, the work reported in this thesis would not have been possible without the financial support from: Hitachi, Calcul Quebec, Compute Canada, the Canada Research Chairs and CIFAR.

1 Introduction

In this chapter, we will provide an overview of the core concepts required to understand the contributions presented in this thesis. Our work lies at the intersection of the fields of representation learning and reinforcement learning, and we structure the overview accordingly. We first introduce developments in representation learning from the 1990s to today, with a particular focus on representation learning methods used in computer vision that this thesis and much related work draw from. We then briefly introduce the field of reinforcement learning, concentrating on reinforcement learning techniques using neural networks, or Deep Reinforcement Learning. We assume knowledge of basic techniques and terminology in deep learning, including feedforward, convolutional, and recurrent neural networks, regularization, and gradient-based optimization, including stochastic gradient descent. We also assume familiarity with traditional supervised learning tasks, such as classification and regression. For readers unfamiliar with these topics, we recommend the book *Deep Learning* (Goodfellow et al., 2016).

1.1 Representation Learning

In this section, we will introduce background material on representation learning in deep learning. We begin with the early developments in the field, including generative pretraining and autoencoding, before moving on to discuss contrastive learning. We conclude with a discussion of recent developments in self-supervised and semi-supervised representation learning, from which this work draws directly. This is not meant to be a complete summary of the history of representations learning; we present only a few key papers, and entirely omit many important historical topics not critical to understanding the techniques used in this thesis, such as Restricted Boltzmann Machines.

Naturally, many of the works considered have used different styles of notation. For convenience, we thus introduce some shared notation; we denote *representations* as z, *targets* used in representation learning objectives as x, and *inputs* to representation learning objectives from which representations are derived as c (when they are separate from x). We denote parameters of neural networks as θ, and encoders as f_θ and decoders as g_θ, where appropriate. We denote a dataset from which x and c are sampled as D. We denote learned distributions, typically parameterized by neural networks, using the subscript θ, as in p_θ and q_θ.

1.1.1 Pretraining

Representation learning as a topic of study in deep learning dates back to early models of multi-layer perceptrons, where *generative pretraining* was proposed as a method to improve learning. Methods of the time struggled to directly train deep neural networks (Tesauro, 1992; Bengio, 2009), as stabilizing techniques such as the rectified linear unit (e.g., as used in Nair and Hinton, 2010), batch normalization (Ioffe and Szegedy, 2015), and skip connections (He et al., 2016; Srivastava et al., 2015) had not yet been adopted. Thus generative pretraining, particularly the layer-wise pretraining proposed by *deep belief nets* (Bengio et al., 2007), served as a means to aid the training of deep networks. By training each layer of neural network individually to reconstruct its own inputs (either the input to the network or the output from the previous layer), proceeding through the network layer-by-layer, the network could be endowed with essentially a better weight initialization, which propagated information through the network in a more efficient fashion. This initialization could then be used to train the network to solve a different (supervised) task, improving performance (Bengio et al., 2007).

1.1.2 Reconstruction

Starting in this period, many works in deep representation learning began to employ *reconstruction* as an objective for representation learning, often (although not always) in the context of autoencoding. In autoencoding, an *encoder* f_θ maps a high-dimensional input x (e.g., an image) to a lower-dimensional representation z (e.g., a 100-dimensional vector). Then, a *decoder* g_θ produces a reconstruction of x, generally by maximizing the likelihood $p_\theta(x|z)$ under a distribution parameterized

by g.[1]

Popular variants, such as the variational autoencoder (VAE, Kingma and Welling, 2013) or Wasserstein autoencoder (WAE, Tolstikhin et al., 2018) introduce a regularization term to ensure that z has a certain desired structure (e.g., that it be normally distributed), and in doing so improve the quality of representations learned (Higgins et al., 2016). For example, the VAE formulates its encoder as parameterizing a distribution $q_\theta(z|x)$ instead of the standard deterministic encoder $f(x)$. The VAE then jointly maximizes $p_\theta(x|z)$ and regularizes q_θ by minimizing the Kullbach-Liebler divergence[2] between $q_\theta(z|x)$ and a *prior* $p(z)$ for the representation: $\mathcal{L}_{VAE} = \mathbb{E}_{x\sim D}\left[\mathbb{E}_{z\sim q_\theta(z|x)}\left[-\log p_\theta(x|z)\right] + \mathrm{KL}(q_\theta(z|x)||p(z))\right]$; the prior is typically chosen to be an isotropic Gaussian distribution, although other choices are not unknown. The WAE, on the other hand, uses a separate critic network to estimate and minimize the Wasserstein distance between $q_\theta(z)$ and $p(z)$.[3]

In more general contexts, input and target output may not be identical. The encoder is given some input c and the decoder is asked to maximize the probability of a different target $p(x|z)$. For example, the *denoising autoencoder* (Vincent et al., 2008) applies noise to the input to f, as a form of regularization. In a task such as video prediction, on the other hand, c might be an individual frame of a video, and x the next frame.

In practical terms, however, reconstruction objectives face a common tendency to disregard smaller objects or visual features, which will typically receive less weight in a calculation of $p(x|z)$. This is particularly true of regularized variants such as variational autoencoders (VAEs), with an entire subfield of research developing around fixing this problem (e.g., Alemi et al., 2018).

1. Many methods in reconstruction calculate the mean squared error between a prediction from a deterministic decoder $\hat{x} = g_\theta(z)$ and x; this is equivalent to maximizing $p(x|z)$ under a fixed-variance Gaussian distribution.

2. The KL divergence is defined as $\mathrm{KL}(q(z)||p(z)) = \mathbb{E}_{z\sim q}[\log \frac{q(z)}{p(z)}]$

3. The Wasserstein distance between two probability measures μ and ν is defined in its minimal form as $W_p(\mu,\nu) = (\inf_{\Gamma\in P(X\sim\mu, Y\sim\nu)} \mathbb{E}_\Gamma[c(X,Y)])$, where Γ is a probability distribution with marginals μ and ν and c is a cost function, chosen here to be a metric. In practice, this must be approximated via duality, using a critic network (Tolstikhin et al., 2018).

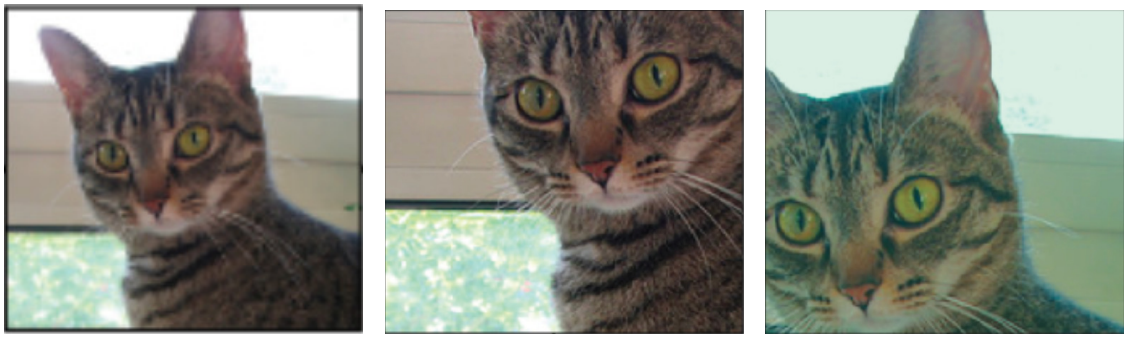

Figure 1.1 – An image from ImageNet (Deng et al., 2009) and two augmented views of the same, as used in AMDIM (Bachman et al., 2019).

1.1.3 Data Augmentation

As representation learning has developed, the use of *data augmentation* has become increasingly foundational. Originally used in supervised learning (see Yaeger et al., 1997; Simard et al., 2003; Krizhevsky et al., 2012), data augmentation stochastically generates alternative "views" of data by adding noise, ideally while preserving quantities of interest (Gontijo-Lopes et al., 2020). When applied to images, as in many of the works to follow, data augmentation typically involves operations such as rotations, cropping, blurring, flipping, and color distortion (e.g., see Grill et al., 2020); when applied to other data, it is common to apply Gaussian noise or other non-spatial distortions. See Fig. 1.1 for an example taken from Bachman et al. (2019).

The use of image augmentation in representation learning can be traced back to early reconstruction-based methods such as the denoising autoencoder (Vincent et al., 2008), in which inputs to an autoencoder are perturbed by noise as a form of regularization. However, image augmentation also enables other classes of representation learning objectives designed to directly exploit the structure created by augmentation.

1.1.4 Temporal Prediction

When temporal structure is present, as in language or video, representation learning techniques that leverage *temporal prediction* may be used. The most well-known such task is without a doubt *language modeling* in natural language processing. In this problem the probability of a string of tokens t is factorized

as $p(t_{1:N}) = \prod_{i=1}^{n} p(t_i|t_{k<i})$. When used for representation learning, a model of the conditional distribution $p_\theta(t_i|t_{k<i})$ is parameterized by a neural network capable of accepting inputs of variable length, such as a recurrent neural network or transformers (Vaswani et al., 2017), allowing parameters to be entirely shared between each $p_\theta(t_i|t_{k<i})$. As t_i is generally a discrete token chosen from a finite vocabulary, the true probability distribution $p(t_i|t_{k<i})$ takes the form of a categorical distribution, which can be conveniently represented by neural networks using the softmax function (Goodfellow et al., 2016). This allows the probability of the data under the neural network model p_θ to be directly maximized via *maximium likelihood estimation*, using $-\sum_{i=1}^{N} \log p_\theta(t_i|t_{k<i})$ as an objective. Networks pretrained in this fashion on large corpora of natural language data have been shown to have representations extremely useful in solving problems such as sentiment analysis, natural language inference, and question answering (Radford et al.; Devlin et al., 2018).

Temporal prediction has also been applied to non-linguistic domains, generally using reconstruction-based or contrastive objectives. PredNet (Lotter et al., 2016) uses a reconstruction task, while CPC and CPC|Action combine contrastive objectives with temporal prediction.

1.1.5 Contrastive Learning

In most reconstruction-based approaches, an explicit, normalized model of the target x given representation z is learned, as $p_\theta(x|z)$. In contrastive models, this explicit, normalized reconstruction is replaced with an implicit, unnormalized potential function $f_\theta(x, z)$, trained indirectly to encourage $f_\theta(x, z)$ to match $p(x|z)$ if properly normalized. To do this, $f_\theta(x, z)$ is maximized for x sampled from the true distribution $p_\theta(x|z)$ and minimized for x sampled from a separate *noise* distribution. This method was originally proposed as a means to learn non-normalized statistical models, by Gutmann and Hyvärinen (2010); later, it was adapted by the deep learning community as a tool for representation learning.

Deep Contrastive Learning

In the deep learning formulation of contrastive learning, the distribution $p(x|c)$ to be learned is generally either generated by augmentation (in which case $p(x|c)$

is typically the distribution of possible views of an image of which c is itself an augmented view) (as used in various fashions in Hjelm et al., 2019; Bachman et al., 2019; Hénaff et al., 2019; Chen et al., 2020a,b; He et al., 2019) or by temporal structure (for example, where c is a frame in a video, and $p(x|c)$ is the distribution of future frames) (as in Oord et al., 2018; Anand et al., 2019; Mazoure et al., 2020). Moreover, modeling conditional distributions enables the *noise* distribution to take the form of the marginal $p(x)$, a choice made by all of the methods cited above due to its convenience.

The most common loss function used in contrastive Learning is the InfoNCE loss, introduced by Oord et al. (2018), defined as

$$\mathcal{L}_{\text{InfoNCE}} = \mathbb{E}_X \left[-\log \frac{f_k(x_+, c)}{\sum_{x_j \in X} f_k(x_j, c)} \right] \tag{1.1}$$

Where X is a minibatch of examples drawn uniformly from a dataset D, x_+ is a positive sample drawn from $p(x|c)$, and all other elements of X are negative samples drawn from $p(x)$. In all of the methods considered, f_k typically involves the creation of intermediate representations z_c and z_x, where z_c is the representation used in downstream tasks. The final step of f_k generally consists of the application of a learnable energy function $h_k(z_c, z_x)$, such as the dot product $z_c \cdot z_x$ or a learned bilinear function $z_c W z_x$. If z_c and z_x are generated by different encoders, the encoder used to generate z_c is referred to in this work as the *online encoder* and the encoder used to generate z_x as the *target encoder*. When used with these distributions, minimizing the InfoNCE loss is equivalent to maximizing a lower bound on mutual information between z_x and z_c.[4] However, recent work suggests that mutual information maximization is not critical to the success of contrastive methods (Tschannen et al., 2019).

Deep contrastive learning methods are thus the first deep *self-supervised* methods we will consider, as they use their own representations (z_x) as targets. This stands in sharp contrast to reconstruction and language modeling contexts, where data x is itself used as a target. We will now consider a selection of deep contrastive learning algorithms, focusing on several methods that appeared between June 2018 to June 2020, a period of rapid evolution during which the state-of-the-art top-1

4. This is the origin of the word *info* in the term InfoNCE

accuracy on ImageNet (generally the most common benchmark used, Deng et al., 2009) among self-supervised methods advanced from 48.7% (Oord et al., 2018) to 79.6% (Grill et al., 2020).

One of the first works to introduce deep contrastive learning at a large scale was CPC (Oord et al., 2018). CPC uses a contrastive temporal prediction objective for video and audio data, training its representation of an initial observation c to predict the representations of observations x later in the time series with an autoregressive model, and directly optimizing a sum of InfoNCE objectives, one for each predicted time step in the future. On images, CPC uses essentially an adaptation of the same type of prediction loss, but using image position instead of time. CPC crops images into vertically-overlapping patches, and then uses representations of the upper patch to predict those in the lower patch.

Later approaches to contrastive learning introduce the notion of enforcing *consistency* under augmentation, and of using contrastive losses that exploit the spatial structure of images by forcing global representations to contain local information. Thus, DIM and AMDIM (Hjelm et al., 2019; Bachman et al., 2019) introduce spatial losses. In this DIM-style of spatial loss, multiple targets $p(x_i|c)$ are defined to be the spatial locations in convolutional feature maps at one or more layers of the encoder as it processes the image c; separate InfoNCE losses are computed for each and then averaged. AMDIM augments DIM primarily by using different random augmentations for c and x and in drawing targets x_i from more layers of the encoder, in total dramatically increasing the scale of the method.

More recent innovations to the contrastive learning paradigm, moreover, added *momentum encoders* and emphasized the use of cosine similarity, both of which would be critical for future methods. These methods also achieved state-of-the-art performance without the use of spatially-structured comparisons (e.g., the patch representations used in DIM and AMDIM), leading to substantially simpler algorithms.

SimCLR (Chen et al., 2020a) pioneered the use of auxiliary projection networks; instead of directly calculating $H(z_c, z_x)$ as a dot product or bilinear function of z_c and z_x, SimCLR applies a learnable non-linear function [5] to z_c and z_x, as $g(z_c)$ and $g(z_x)$, where g is trained to minimize the same InfoNCE loss. SimCLR then defines its energy function using the *cosine similarity* between these projections,

5. i.e., a two-layer MLP

as $H(z_c, z_x) = t\frac{g(z_c) \cdot g(z_x)}{\|g(z_c)\|\|g(z_x)\|}$, where t is a fixed temperature hyperparameter. Using the cosine similarity instead of a simple dot product regularizes the objective, preventing the network from easily reducing its loss by modifying the magnitude of its predictions, at the cost of introducing an important hyperparameter t which must be tuned.

MoCo (*Mo*mentum *Co*ntrastive Representation Learning, He et al., 2019) augmented the contrastive framework used in previous works with the use of a polyak-averaged (Polyak and Juditsky, 1992) target encoder to encode targets x, using an exponential moving average (nicknamed *momentum*). Denoting the parameters of the target encoder as θ_m and those of the online encoder as θ_o, the parameters of the target encoder are updated as $\theta_m \leftarrow \tau\theta_m + (1 - \tau)\theta_o$, where τ is a hyperparameter. The primary role of the momentum target network in MoCo is to enable the use of a memory buffer of negative examples which are used to dramatically increase the effective batch size of the algorithm (up to 65,536 examples), although recent work indicates that momentum target encoders can themselves improve training due to their stabilizing influence (Grill et al., 2020). As the difficulty of a contrastive task is directly related to the number of negative samples (see the definition of the InfoNCE loss above; increasing the number of negative examples increases the denominator of the softmax, monotonically increasing the loss), introducing this large memory buffer enables for more finely-tuned representations. Later, innovations from SimCLR were combined with MoCo, leading to a hybrid method with strong performance (Chen et al., 2020c).

1.1.6 Semi-Supervised Learning

Most of the methods considered thus far have been optimized separately from a supervised task such as regression or classification. Instead, they largely focus on pretraining, using a representation learning method separately from the ultimate downstream objective, which is optimized separately either by adapting the entire network (e.g., Bengio et al., 2007) or by training a separate low-capacity algorithm such as a linear classifier with the algorithm's encoder as a preprocessing step applied to inputs (as done during evaluation by Oord et al., 2018; He et al., 2019; Bachman et al., 2019; Chen et al., 2020a).

However, when the goal is to design a representation learning method to solve a

specific downstream task, it is natural to instead jointly optimize a representation learning loss alongside a supervised learning loss. Doing so is generally referred to as *semi-supervised learning*,[6] and may offer improvements to the *data efficiency* for the supervised learning task, reducing the number of (often costly, human-generated) labels necessary to reach a certain level of performance. Both constrastive (e.g., Hénaff et al., 2019; Chen et al., 2020b) and reconstructive (e.g., Rasmus et al., 2015) objectives have been used in semi-supervised learning; in linguistic domains, it is common to employ temporal prediction tasks in the form of language modeling. This has become particularly prominent in problems where labeled data is scarce. When a great deal of unlabeled data is available, the representation learning objective may be optimized over both labeled and unlabeled data, while the supervised learning objective is optimized over only the labeled subset (e.g., Hénaff et al., 2019; Vedantam et al., 2019; Chen et al., 2020b), although some techniques show benefits for representation learning techniques even when no additional unlabeled data is available (see experiments in Hénaff et al., 2019; Chen et al., 2020b; Tarvainen and Valpola, 2017).

Consistency-based Losses

A number of self-supervised learning methods have been proposed in which a "student" network is trained to directly match its outputs to those of a "teacher" network under noise or various other perturbations, predominantly (but not exclusively) in semi-supervised learning. In the most general formulation of this type of objective, the "student" and "teacher" networks are presented with different augmented views of an input, or are otherwise subjected to differing noise, e.g., by dropout (Srivastava et al., 2014). Although invariance to noise or perturbation is a key aspect of many contrastive works (as examined by Wang and Isola, 2020), these methods differ in having no notion of "contrast"; instead, only a divergence is minimized, such as KL divergence (when teacher outputs are interpreted as distributions, as in Sohn et al., 2020; Fortunato et al., 2018) or mean squared error (when they are treated viewed only as vectors Tarvainen and Valpola, 2017).

Mean Teacher (Tarvainen and Valpola, 2017) proposes to instantiate the "teacher" network as an exponential moving average (EMA) of the student (Much as was later done in contrastive learning by He et al., 2019), showing large boosts in performance

6. Not to be confused with *self-supervised learning*, with which it shares acronyms.

on classification tasks with only a small number of labels available. Moreover, they show that using an EMA teacher network leads to a substantial boost in performance over the alternative where the teacher is identical to the student, and demonstrate that subjecting the teacher to additional noise (image augmentation, dropout) greatly improves performance. Mean Teacher is also notable for minimizing the squared error between the final output layer of the student and that of the teacher, prefiguring both this work and later works in pure self-supervised learning (Grill et al., 2020).

Noisy Student (Xie et al., 2020) modifies this general algorithm by adopting an iterated approach, freezing the "teacher" network occasionally and reinitializing a larger "student" network from scratch; after learning from the teacher for a set period of time, the student becomes the new teacher and the cycle begins again. Noisy Student, unlike Mean Teacher, applies no augmentation or noise whatsoever to the inputs to the teacher; instead, Noisy Student uses enormous quantities of additional unlabeled data (300 million images). Also unlike Mean Teacher, Noisy Student uses the teacher to generate inferred labels and trains the student via classification using these labels as targets. As a result of these innovations, Noisy Student shows large improvements of performance even when trained with all ImageNet labels, including large gains on robustness measures.

Fix-Match (Sohn et al., 2020) differs from Noisy Student in returning to the teacher paradigm employed by Mean Teacher. Fix-Match also further refines the distinction between augmentations used for teacher and student; the student is presented with radically distorted images, while the teacher's inputs are subjected to only mild augmentation. Fix-Match also introduces a confidence threshold to avoid training the student when the teacher is uncertain, by only treating the teacher's predictions as equivalent to classification labels if they are sufficiently confident.

1.1.7 Bootstrap Your Own Latent

Bootstrap Your Own Latent (BYOL, Grill et al., 2020) showed that this type of objective is also viable in purely unsupervised learning, outperforming comparable contrastive methods despite having no theoretical incentive to avoid representational collapse. Essentially comparable to contrastive learning methods such as SimCLR but with "contrastive" elements of its loss removed, BYOL directly maximizes the

cosine similarity between representations of different views of an input, achieving state-of-the-art performance with a substantially simpler method.[7] Were representations to collapse to a constant vector the BYOL loss would be minimized, but empirically this never occurs; the precise reason for this remains unknown (private correspondence with authors of Grill et al., 2020).

This results in a dramatically simpler algorithm than that used by comparable contrastive and reconstruction-based alternatives, which respectively need to employ negative samples or train a decoder. Experiments demonstrate that reintroducing negative samples to BYOL in fact *reduces* performance, contrary to arguments by Wang and Isola (2020) that the representational uniformity enforced by negative samples in contrastive methods is critical to their success. Despite the presence of infinitely many spurious minima of the BYOL loss (online and target outputs collapsing to a shared constant vector is sufficient to achieve zero loss), this does not in practice occur, with the exponential moving average target encoder identified as a key reason.

1.2 Reinforcement Learning

Reinforcement learning is a subfield of machine learning that studies how agents learn behavioral patterns, or *policies*, to maximize an objective, or *reward*, while interacting with an environment. In the standard reinforcement learning setting (see Sutton and Barto, 2018), agents interact with a Markov Decision Process (MDP), defined as consisting of a set of states $\mathcal{S}$, a set of possible actions A, a set of possible rewards $\mathcal{R}$, a state transition distribution $p(s'|s,a)$ and a reward distribution $p(r|s',s,a)$.[8]

Agents interact with their environment by choosing actions according to a policy $\pi(a|s)$, which is a probability mass (or density, if $\mathcal{A}$ is infinite) function. After choosing an action, agents observe a reward and the next state; this sequence of state-action-reward-state is commonly referred to as a *transition*. Agents' interactions

7. The authors of BYOL describe their method as minimizing normalized L2 distance, similar to Mean Teacher's choice of MSE (Tarvainen and Valpola, 2017), but also acknowledge that this is equivalent up to a loss scaling factor to maximing cosine similarity.

8. These take the form either of probability mass functions or probability density functions, depending on the finiteness of $\mathcal{S}$ and $\mathcal{R}$.

are commonly divided into *episodes*, sequences of transitions, and are written as $s_0, a_0, r_0, s_1, a_1, r_1, \ldots$. These episodes may be infinite, in which case the quantity of interest for optimization is the average reward received by the agent across timesteps. In the setting considered here, however, the quantity of interest is the discounted sum of rewards (or *return*), defined as $G_t \triangleq r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots = \sum_{k=t}^{\infty} \gamma^{t-k} r_k$, where $\gamma \in [0, 1]$ is a *discount factor* that causes the agent to prioritize nearer rewards (Sutton and Barto, 2018).

It is common for agents to estimate these returns with a learned *value function*, $V : \mathcal{S} \to \mathbb{R}$, which is trained to approximate the true expected return of a state $v_\pi(s) \triangleq \mathbb{E}_\pi[G_t | s_t = s]$. Alternatively, many algorithms instead estimate an action-conditioned variant of the value function, generally denoted as $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, which is trained to approximate $Q^*(s, a) \triangleq \mathbb{E}[G_t | s_t = s, a_t = a]$. When actions are selected according to a policy π, estimating Q is strictly more general than estimating V, as a value function V can be recovered by taking the expectation of Q over actions: $V(s) = \mathbb{E}_{a \sim \pi(a|s)}[Q(s, a)]$.

1.2.1 TD Learning

Although a wide variety of methods have been proposed to estimate Q and V, the methods used in this thesis are based on *temporal-difference* (TD) learning. In TD methods, agents learn from individual transitions of the form (s_t, a_t, r_t, s_{t+1}), updating their estimates of $V(s_t)$ or $Q(s_t, a_t)$ using their estimate for the value of s_{t+1}. In the context of value learning, the *TD error* of a transition is defined as $\delta_t \triangleq r_t + \gamma V(s_{t+1}) - V(s)$. With infinite data and certain assumptions about optimization, the correct value function can be learned by iteratively minimizing this error (Sutton and Barto, 2018), converging at a rate governed by γ.

In this work, we largely focus on Q-learning, a variant of TD learning used to jointly estimate the optimal policy π^*, defined as $\pi^* \triangleq \arg\max_\pi \mathbb{E}_{\pi, s_o}[G_0]$, and the optimal value function, defined as $q^*(s, t) = \mathbb{E}_{\pi^*}[G_t | s_t = s, a_t = a]$. When the MDP in question is perfectly known and states and actions can be enumerated, an agent's estimate of Q can iteratively improved by applying the Bellman optimality operator, defined as $Q_{i+1}(S_t, A_t) = \mathbb{E}[r_t + \gamma \max_a Q_i(S_{t+1}, a)]$ (Sutton and Barto, 2018); this operator converges to the optimal policy $\pi = \pi^*$ and Q function $Q = q^*$ at its fixed point. When this is not the case and Q must be learned from data,

this is generally done by minimizing a TD error corresponding to the operator $\delta_t \triangleq r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s, a)$.

1.2.2 Off-Policy Learning

The objectives defined for TD learning contain a number of expectations, over the agent's policy, the reward distribution of the environment, and the environment's transition function. When the MDP in question is fully known, these expectations are not problematic, as they can be directly evaluated. In practice, however, this is often not the case, and these expectations must generally be approximated by samples taken from the agent's interactions with the environment.

Approaches for doing so can be divided into two classes: *on-policy* and *off-policy*, based on whether data used is collected according to the current policy π or some other policy or set of policies. Off-policy methods offer significant theoretical advantages, enabling agents to learn from arbitrarily-collected data, but often suffer from instability, especially when combined with TD learning and function approximation such as neural networks (Sutton and Barto, 2018).

Moreover, some methods are intrinsically off-policy. In Q-learning, for example, the agent's policy is deterministic; although this allows the agent to learn what is strictly the optimal policy, it means that the agent will generally not select a wide-enough variety of actions to explore its environment, potentially causing the best action for a state to go undiscovered. As a result, it is common in Q-learning for agents to collect data in their environment according to stochastic exploration policy based on π but with added noise. As a result, data for Q-learning is *never* sampled according to the agent's actual policy.

1.2.3 Deep Reinforcement Learning

When states and actions can be enumerated, Q can be learned as a $|\mathcal{S}| \times |\mathcal{A}|$ matrix, and V can be learned as a $|\mathcal{A}|$ vector. In more general settings, however, function approximation must be used for Q and V. Among parametric function approximation methods, both linear regression and neural networks are common choices. When linear function approximation is used, some guarantees of stability or performance are available (Sutton and Barto, 2018). However, in practice nonlinear function approximation is required in many cases, particularly those with visual

inputs (for example, see Mnih et al., 2015).

When using parametric function approximation with Q-learning, as we do in this work, it is standard to minimize the following objective via gradient descent on the parameters of Q:

$$\mathcal{L}_Q(s_t, a_t, r_t, s_{t+1}) = (Q(s_t, a_t) - \mathbb{E}[r_t + \gamma \max_a Q_t(s_{t+1}, a)])^2 \tag{1.2}$$

where Q_t is a separate "target" function reflecting an older version of Q. It is key that Q not be modified by gradients taken through the target Q_t, even when Q_t is defined to be the same as Q; failing to do this leads the algorithm to arrive at incorrect solutions (Sutton and Barto, 2018).

In practice, this objective is optimized using samples taken from the environment. Data collected by the agent is placed in a *replay buffer*, a buffer of the agent's most recent experiences; typical buffer sizes may be up to several million transitions. This was originally introduced to stabilize training (Mnih et al., 2015), but recent research suggests that allowing the agent to learn from old experience has a strong positive effect even in newer, more stable methods (Fedus et al., 2020).

Despite this, Deep Q-Learning, or Deep Q Networks (DQN), has thus emerged as a particularly successful method in reinforcement learning. This is largely due to the implementation of a wide range of algorithmic improvements beyond the basic Q-learning algorithm; when combined, in Rainbow (Hessel et al., 2018), the resulting algorithm achieves performance far above a naive version. We refer the reader to Hessel et al. (2018) for a full summary of these improvements; although we take advantage of them, they are largely orthogonal to this work.

1.2.4 Deep Continuous Control

DQN, the method introduced above, is in practice specific to the *discrete control* setting, where $\mathcal{A}$ is finite. This is due to the $\max_a Q(s, a)$ operation that must be performed as part of both optimization and action selection; when $\mathcal{A}$ is discrete and reasonably small, enumerating $Q(s, a)$ for all actions in $\mathcal{A}$ is a reasonably efficient way of finding $\max_a Q(s, a)$.

However, when $\mathcal{A}$ is continuous or otherwise infinite, finding $\max_a Q(s, a)$ is non-trivial. One of the most common alternatives to DQN is to thus learn a separate policy network as $a_t = \pi_\theta(s_t)$ to approximate $\arg\max_a Q(s, a)$. This method, known

as Deep Deterministic Policy Gradient (Lillicrap et al., 2016) leads to the following objective analogous to DQN:

$$\mathcal{L}_Q(s_t, a_t, r_t, s_{t+1}) = \left(Q(s_t, a_t) - r_t + \gamma Q_t(s_{t+1}, \pi_\theta(s_{t+1})]\right)^2. \tag{1.3}$$

π_θ is trained directly to maximize Q, by differentiating through the Q network.

$$\mathcal{L}_\pi(s) = -Q(s, \pi_\theta(s)) \tag{1.4}$$

More refined variants of this algorithm were introduced by Twin Delayed DDGP (TD3, Fujimoto et al., 2018), which introduces techniques to fight value over-estimation and instability due to function approximation error, and Stochastic Actor-Critic (Haarnoja et al., 2018), which learns a stochastic policy $\pi_\theta(a|s)$, using entropy regularization to force π_θ to maintain a certain level of randomness. This results in the following Q-learning loss:

$$\mathcal{L}_Q(s_t, a_t, r_t, s_{t+1}) = \left(Q(s, a) - \mathbb{E}_{a_{t+1}\sim\pi(a|s_{t+1})}[r_t + \gamma Q_t(S_{t+1}, a_{t+1}]\right)^2. \tag{1.5}$$

1.2.5 Representation Learning for Reinforcement Learning

One popular approach to resolving instability in Deep Reinforcement learning is to focus on improving the representations learned by the neural networks used (Lesort et al., 2018). This has in the past taken the form either of directly integrating methods from unsupervised representation learning Oord et al. (2018); Srinivas et al. (2020); Yarats et al. (2019), or in using alternative techniques designed specifically for deep reinforcement learning (see for example Gelada et al., 2019; Dabney et al., 2020; Guo et al., 2020). The most common approach used to combine DRL and representation learning is to jointly optimize a reinforcement learning loss and a representation learning loss (as in this work and Oord et al., 2018; Srinivas et al., 2020; Yarats et al., 2019; Guo et al., 2020; Gelada et al., 2019). In this sense, reinforcement learning with representation learning is treated somewhat analogously to semi-supervised learning, with rewards serving as supervision, the equivalent of labels in a supervised learning context, and the reinforcement learning loss the role of the supervised classification or regression loss. Some methods (e.g., Lee et al., 2019b) employ extra data without supervision by rewards, while others (e.g., this work, Yarats et al., 2019) do without additional data, although it is possible that their performance could be improved if additional data were used.

This approach generally requires choosing a hyperparameter λ governing the weight given to the representation learning loss relative to the reinforcement learning loss. The main alternative formulation would be learning representations with a method from section 1.1 and then doing reinforcement learning using these representations, analogous to a pretraining-based approach in representation learning. Although linear reinforcement learning applied to frozen representations would have some guarantees of stability, it lacks any means of correcting sub-optimal representations, unlike in the standard approach. Moreover, this two-stage process is impractical when only a limited amount of interaction time is available, where it is vital that reinforcement learning progress happen as quickly as possible; by the time a dataset of sufficient size to fully train the representation learning method had been collected, little time would remain for the reinforcement learning algorithm to train.

1.2.6 Data Efficiency

Data efficiency is a major challenge in deep reinforcement learning. Although recent algorithms have been able to effectively solve a number of challenging tasks, such as DotA 2 (OpenAI et al., 2019), Starcraft 2 (Vinyals et al., 2019), and Atari (Badia et al., 2020), they have done so using enormous amounts of experience. As a result, it has become increasingly common to focus on improving the data efficiency of reinforcement learning, generally defined as improving performance with limited environment interaction time. A number of methods have been proposed to this end, from *model-based* methods that aim to accelerate learning by learning an explicit model of environment dynamics and reward distributions (e.g., Kaiser et al., 2019) to tweaked versions of existing algorithms that were previously optimized for performance in large-data regimes (for example, Data-Efficient Rainbow from van Hasselt et al., 2019).

1.2.7 DeepMind Control

Tassa et al. (2018) introduced DeepMind Control (DM Control), a new benchmark for continuous control tasks, adapting the MuJoCo (Todorov et al., 2012) framework. DM Control environments have been widely used in previous work (see Kostrikov et al., 2020; Hafner et al., 2020; Laskin et al., 2020), and represent the

current standard for continuous control. DM Control includes the option to provide agents with either state representations (vectors representing the state of the environment) or pixels (images representing the state of the environment) as inputs, with performance given pixel inputs traditionally lagging far behind performance with state representations (see for example Yarats et al., 2019).

1.2.8 Atari Learning Environment

(Bellemare et al., 2013) introduced the Atari Learning Environment (ALE), a challenging reinforcement learning task in which agents learn to play Atari games using visual inputs. This task is different from many of those traditionally studied in reinforcement learning in that nonlinear function approximation (e.g., neural networks) are key. Atari games are discrete control tasks, in which agents choose between up to 18 actions at each step. The traditional goal, surpassing human performance, has now been achieved on all 57 ALE games (Badia et al., 2020) but only when algorithms are given effectively infinite interaction time. In the limited-time regime, agent performance remains weak (compare results in Badia et al., 2020; Kostrikov et al., 2020).

1.2.9 Evaluation in Atari

Performance on Atari is generally calculated as the human-normalized score, calculated separately on each game as $\frac{\text{agent\_score}-\text{random\_score}}{\text{human\_score}-\text{random\_score}}$. Unlike in many continuous control domains such as DM Control, where algorithm hyperparameters are often selected on a per-task basis, the standard in deep reinforcement learning for Atari has since Mnih et al. (2015) been to use identical hyperparameters on all games. This has important effects on algorithm design; by forcing methods to be successful on a wide range of games, approaches requiring finely-tuned hyperparameters are relatively disadvantaged, and encourages the development of methods that organically tune hyperparameters during training, such as Agent57. As a result, performance by methods that do not adhere to this standard, such as Sunrise (Lee et al., 2020), cannot be compared to that of methods that do.

2 Data-Efficient Reinforcement Learning with Self-Predictive Representations

Authors: **Max Schwarzer***, Ankesh Anand*, Rishab Goel, Devon Hjelm, Aaron Courville and Philip Bachman.

This chapter presents a lengthened version of a joint work with Ankesh Anand, Rishab Goel, Devon Hjelm, Aaron Courville, and Philip Bachman. It will be submitted to the conference track of the 2021 International Conference on Learning Representations, and has been accepted at the 2020 Montreal AI Symposium.

Contributions: The idea was initially conceptualized as a work on temporal contrastive learning with Ankesh Anand, Devon Hjelm, and my advisor Prof. Aaron Courville. I refined the current non-contrastive formulation of the project, wrote code for and performed all of the experiments for Atari listed in the paper, and helped write code for experiments on DeepMind Control. Ankesh Anand adapted code for the rlpyt framework for our project, helped develop our method, and helped write the paper. Rishab Goel wrote code for and ran experiments on DeepMind Control. Devon Hjelm, Prof. Aaron Courville, and Philip Bachman provided advising and helped refine the paper. I am joint first author of the paper with Ankesh Anand.

Affiliation

— Max Schwarzer, Mila, University of Montreal

— Ankesh Anand, Mila, University of Montreal, Microsoft Research

— Rishab Goel, Mila

— Devon Hjelm, Mila, University of Montreal, Microsoft Research

— Aaron Courville, Mila, University of Montreal

— Philip Bachman, Microsoft Research

2.1 Introduction

Deep Reinforcement Learning (deep RL, François-Lavet et al., 2018) has proven to be an indispensable tool for training successful agents on difficult sequential decision-making problems (Bellemare et al., 2013; Tassa et al., 2018). The success of deep RL is particularly noteworthy in highly complex strategic games such as StarCraft (Vinyals et al., 2019) and DoTA2 (OpenAI et al., 2019), where deep RL agents now surpass expert human performance in some scenarios.

Deep RL involves training agents based on large neural networks using large amounts of data (Sutton, 2019), a trend evident across both model-based (Schrittwieser et al., 2019) and model-free (Badia et al., 2020) learning. The sample complexity of such state-of-the-art agents is often incredibly high: MuZero (Schrittwieser et al., 2019) and Agent-57 (Badia et al., 2020) use 10-50 years of experience per game, and OpenAI Five (OpenAI et al., 2019) uses *45,000 years* of experience to accomplish its remarkable performance.

This is clearly impractical: unlike easily-simulated environments such as video games, collecting interaction data for many real-world tasks is costly. Moreover, when given less data, DRL agents' performance is generally far worse; the previous state of the art on data-efficient Atari attained human-level performance on only two games out of 26 when given the same amount of time to learn the game as human players. [1] Thus, making improved *data efficiency* a prerequisite for successful use of deep RL in these settings (Dulac-Arnold et al., 2019).

Meanwhile, new self-supervised representation learning methods have significantly improved data efficiency when learning new vision and language tasks, particularly in low data regimes or semi-supervised learning (Xie et al., 2019; Hénaff et al., 2019; Chen et al., 2020b). Self-supervised methods improve data efficiency by leveraging a nearly limitless supply of training signal from tasks generated on-the-fly, based on "views" drawn from the natural structure of the data (e.g., image patches, data augmentation or temporal proximity, see Doersch et al., 2015; Oord et al., 2018; Hjelm et al., 2019; Tian et al., 2019; Bachman et al., 2019; He et al., 2019; Chen et al., 2020a).

Motivated by successes in semi-supervised and self-supervised learning (Tarvainen and Valpola, 2017; Xie et al., 2019; Grill et al., 2020), we train better state

1. Roughly two hours per game.

representations for RL by forcing representations to be temporally predictive and consistent when subject to data augmentation. Specifically, we extend a strong model-free agent by adding a dynamics model which predicts future latent representations provided by a parameter-wise exponential moving average of the agent itself. We also add data augmentation to the future prediction task, which enforces consistency across different views of each observation. Contrary to some methods (Kaiser et al., 2019; Hafner et al., 2019), our dynamics model operates entirely in the latent space and doesn't rely on reconstructing raw states.

We evaluate our method, which we call Self-Predictive Representations (SPR), on 26 games in the Atari 100k benchmark (Kaiser et al., 2019), where agents are allowed only 100k steps of environment interaction (producing 400k frames of input) per game, which roughly corresponds to two hours of real-time experience. Notably, the human experts in Mnih et al. (2015) and Van Hasselt et al. (2016) were given the same amount of time to learn these games, so a budget of 100k steps permits a reasonable comparison in terms of data efficiency.

In our experiments, we augment a modified version of Data-Efficient Rainbow (DER) (van Hasselt et al., 2019) with the SPR loss, and evaluate versions of SPR with and without data augmentation. We find that each version is superior to controlled baselines. When coupled with data augmentation, SPR achieves a median score of 0.415, which is a state-of-the-art result on this benchmark, outperforming prior methods by a significant margin. Notably, SPR also outperforms human expert scores on 7 out of 26 games while using roughly the same amount of in-game experience. We now describe our overall approach in detail.

2.1.1 Deep Q-Learning

We focus on the Atari Learning Environment (Bellemare et al., 2013), a challenging setting where the agent takes discrete actions while receiving purely visual, pixel-based observations. A prominent method for solving Atari, Deep Q Networks (DQN, Mnih et al., 2015), trains a neural network Q_θ to approximate the agent's current *Q-function* (policy evaluation) while updating the agent's policy greedily with respect to this Q-function (policy improvement). This involves minimizing the error between predictions from Q_θ and a target value estimated by Q_ξ,

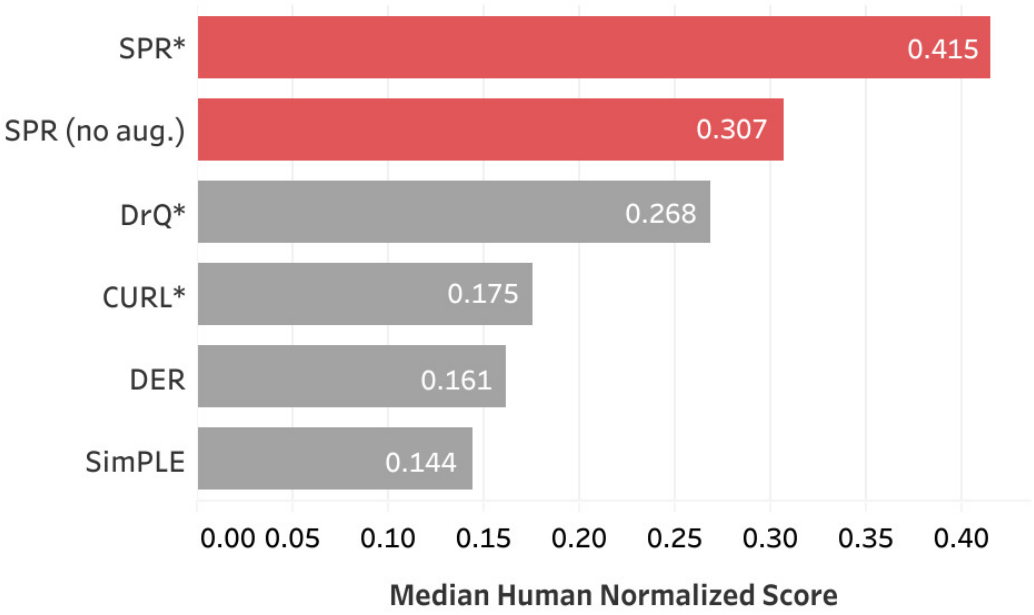

Figure 2.1 – Median Human-Normalized scores of different methods across 26 games in the Atari 100k benchmark (Kaiser et al., 2019), averaged over 10 random seeds. Each method is allowed access to only 100k environment steps or 400k frames per game. (*) indicates that the method uses data augmentation. SPR achieves a state-of-art-result on median human-normalized score, improving over the previous best, DrQ (Kostrikov et al., 2020), by 55%. Note that without data augmentation SPR still outperforms prior methods that use data augmentation.

an earlier version of the network:

$$\mathcal{L}_{DQN} = \Big(Q_\theta(o_t, a_t) - (r_t + \gamma \max_a Q_\xi(o_{t+1}, a))\Big)^2. \tag{2.1}$$

Various improvements have been made over the original DQN: Distributional RL (Bellemare et al., 2017) models the full distribution of future reward rather than just the mean, Dueling DQN (Wang et al., 2016) decouples the *value* of a state from the *advantage* of taking a given action in that state, Double DQN (Van Hasselt et al., 2016) modifies the Q-learning update to avoid overestimation due to the max operation, among many others. Rainbow (Hessel et al., 2018) consolidates these improvements into a single combined algorithm and has been adapted to work well in data-limited regimes (van Hasselt et al., 2019).

We also evaluate SPR on DeepMind Control (DM Control, Tassa et al., 2018), a

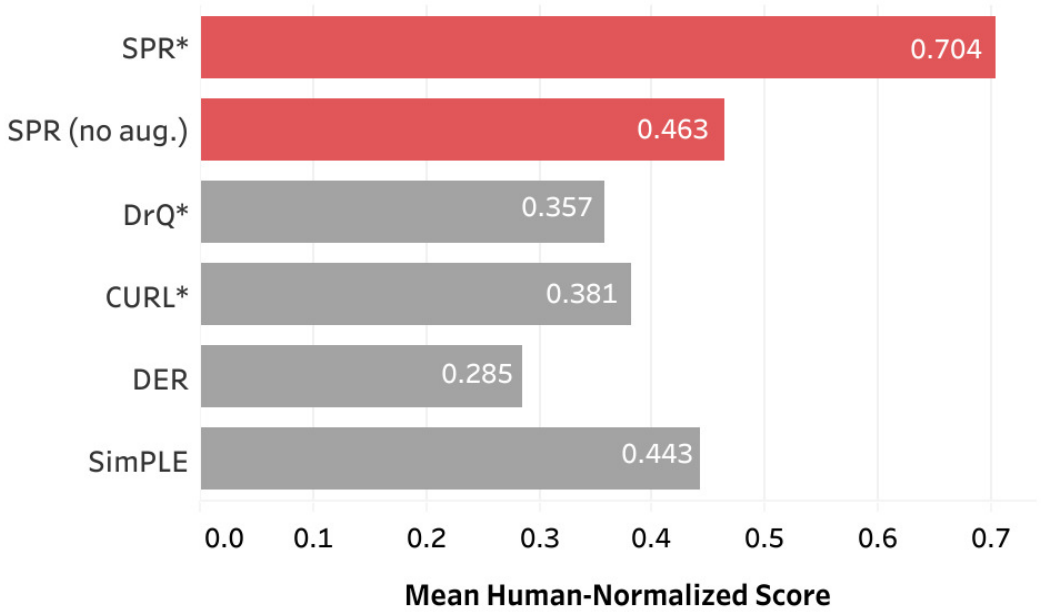

Figure 2.2 – Mean Human-Normalized scores of different methods across 26 games in the Atari 100k benchmark (Kaiser et al., 2019), averaged over 10 random seeds. Each method is allowed access to only 100k environment steps or 400k frames per game. (*) indicates that the method uses data augmentation. SPR achieves a state-of-art-result on mean human-normalized score, improving over the previous best, SimPLe (Kaiser et al., 2019), by 59%. Note that without data augmentation SPR still outperforms prior methods that use data augmentation.

continuous control task. As DQN is designed for discrete control settings, it cannot be used in DM Control. However, a number of methods comparable to DQN have been proposed for continuous control; we use Soft Actor-Critic (SAC, Haarnoja et al., 2018), as employed by DrQ (Kostrikov et al., 2020).

2.1.2 Self-Predictive Representations

For our auxiliary loss, we start with the intuition that encouraging state representations to be predictive of future observations given future actions should improve the data efficiency of RL algorithms. Let $(s_{t:t+K}, a_{t:t+K})$ denote a sequence of $K+1$ previously experienced states and actions sampled from a replay buffer, where K is the maximum number of steps into the future which we want to predict. Our method has four main components which we describe below:

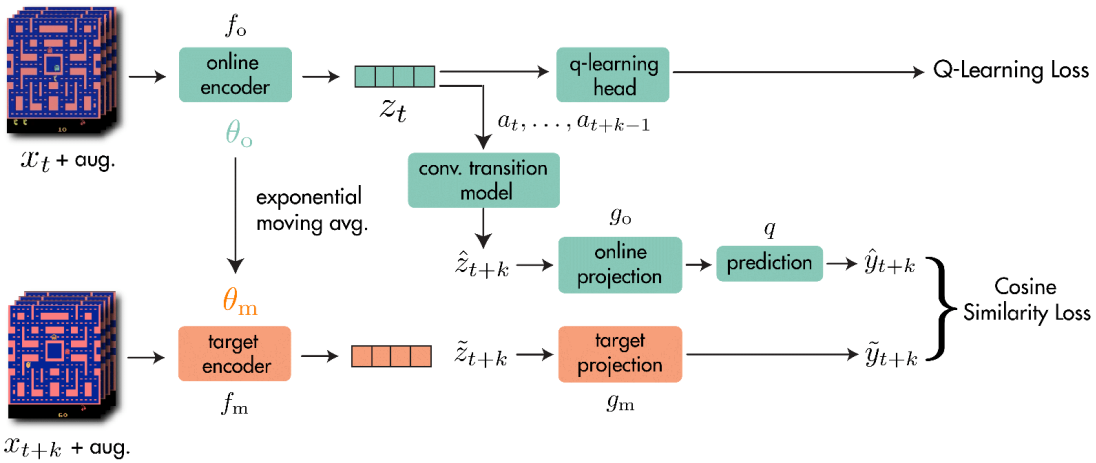

Figure 2.3 – An illustration of the full SPR method. Representations from the online encoder are used in the reinforcement learning task and for prediction of future representations from the target encoder via the transition model. The target encoder and projection head are defined as an exponential moving average of their online counterparts and are not updated via gradient descent. For brevity, we illustrate only the k^{th} step of future prediction, but in practice we compute the loss over all steps from 1 to K. Note: our implementation in Atari includes g_o in the Q-learning head.

— **Online and Target networks**: We use an *online encoder* f_o to transform observed states s_t into representations $z_t \triangleq f_o(s_t)$. We use these representations in an objective that encourages them to be *predictive* of future observations up to some fixed temporal offset K, given a sequence of K actions to perform. We augment each observation s_t independently when using data augmentation. In most cases, we find it beneficial to follow prior work (Tarvainen and Valpola, 2017; Grill et al., 2020) by computing target representations for future states using a separate *target encoder* f_{m}, whose parameters are an exponential moving average (EMA) of the online encoder parameters. Denoting the parameters of f_{o} as θ_{o}, those of f_{m} as θ_{m}, and the EMA coefficient as $\tau \in [0, 1)$, the update rule for θ_{m} is:

$$\theta_{\text{m}} \leftarrow \tau\theta_{\text{m}} + (1 - \tau)\theta_{\text{o}}. \tag{2.2}$$

Note that this means the target encoder is not updated via gradient descent. A special case of interest is $\tau = 0$, in which case the target encoder is identical to the online encoder with a stopgradient applied to its outputs. We find this case to perform best in some settings where stability is not a concern.

— **Transition Model:** For the prediction objective, we generate a sequence of K predictions $\hat{z}_{t+1:t+K}$ of future state representations $\tilde{z}_{t+1:t+K}$ using an action-conditioned *transition model* h. We compute $\hat{z}_{t+1:t+K}$ iteratively: $\hat{z}_{t+k+1} \triangleq h(\hat{z}_{t+k}, a_{t+k})$, starting from $\hat{z}_t \triangleq z_t \triangleq f_o(s_t)$. We compute $\tilde{z}_{t+1:t+K}$ by applying the target encoder f_m to the observed future states $s_{t+1:t+K}$: $\tilde{z}_{t+k} \triangleq f_m(s_{t+k})$. The transition model and prediction loss operate in the latent space, thus avoiding pixel-based reconstruction objectives. We describe the architecture of h in section 2.1.3.

— **Projection Heads:** We use online and target projection heads g_o and g_m (Kaiser et al., 2019) to project online and target representations to a smaller latent space, and apply an additional *prediction head* q (Grill et al., 2020) to the online projections to predict the target projections:

$$\hat{y}_{t+k} \triangleq q(g_o(\hat{z}_{t+k})),\ \forall \hat{z}_{t+k} \in \hat{z}_{t+1:t+K}; \quad \tilde{y}_{t+k} \triangleq g_m(\tilde{z}_{t+k}),\ \forall \tilde{z}_{t+k} \in \tilde{z}_{t+1:t+K}. \tag{2.3}$$

The target projection head parameters are given by the same update rule as the online and target encoders.

— **Prediction Loss**: We compute the future prediction loss for SPR by summing over cosine similarities [2] between the predicted and observed representations at timesteps $t+k$ for $1 \leq k \leq K$:

$$\mathcal{L}^{\text{SPR}}(s_{t:t+K}, a_{t:t+K}) = -\sum_{k=1}^{K} \left(\frac{\tilde{y}_{t+k}}{||\tilde{y}_{t+k}||_2} \right)^\top \left(\frac{\hat{y}_{t+k}}{||\hat{y}_{t+k}||_2} \right), \tag{2.4}$$

where $\tilde{y}_{t+k}$ and $\hat{y}_{t+k}$ are computed from $(s_{t:t+K}, a_{t:t+K})$ as we just described.

We call our method Self-Predictive Representations (SPR), following the self-predictive nature of the objective. During training, we combine the SPR loss with the Q-learning loss for Rainbow. The SPR loss affects f_o, g_o, q and h. The Q-learning loss affects f_o and the Q-learning head, which contains additional layers specific to Rainbow. Denoting the Q-learning loss from Rainbow as $\mathcal{L}^{\text{RL}}$, our full optimization objective is: $\mathcal{L}_\theta^{\text{total}} = \mathcal{L}_\theta^{\text{RL}} + \lambda \mathcal{L}_\theta^{\text{SPR}}$.

Compared to prior work (Kostrikov et al., 2020; Laskin et al., 2020), our method can leverage data augmentations more effectively by encouraging consistency between

2. Cosine similarity is proportional to the "normalized L2" loss used in BYOL (Grill et al., 2020)

representations of different augmented views. We empirically verify this via a controlled comparison to DrQ (see section 2.3.4). It should be noted that SPR can still be used in contexts where data augmentation is unavailable or counterproductive. Compared to related work on contrastive representation learning, SPR does not use negative samples, which may require careful design of contrastive tasks, large batch sizes (Chen et al., 2020a), or the use of a buffer to emulate large batch sizes (He et al., 2019).

Algorithm 1: Self-Predictive Representations

Denote parameters of online encoder f_o and projection g_o as θ_o
Denote parameters of target encoder f_m and projection g_m as θ_{m}
Denote parameters of transition model h, predictor q and Q-learning head as ϕ
Denote the maximum prediction depth as K
initialize replay buffer B
while *Training* **do**
  collect experience (s, a, r, s') with (θ_o, ϕ) and add to buffer B
  sample minibatch $(s, a, r, s') \sim B$
  if *augmentation* **then**
    $s \leftarrow \text{augment}(s)$
  end
  $z_0 \leftarrow f_\theta(s_0)$ `// online representations`
  $l \leftarrow 0$
  for *k in (1, . . . , K)* **do**
    $\hat{z}_k \leftarrow h(\hat{z}_{k-1}, a_{k-1})$ `// latent states via transition model`
    $\tilde{z}_k \leftarrow f_m(s_k)$ `// target representations`
    $\hat{y}_k \leftarrow q(g_o(\hat{z}_k))$, $\tilde{y}_k \leftarrow g_m(\tilde{z}_k)$ `// projections`
    $l \leftarrow l - \left(\frac{\tilde{y}_k}{||\tilde{y}_k||_2}\right)^\top \left(\frac{\hat{y}_k}{||\hat{y}_k||_2}\right)$ `// SPR loss at step` k
  end
  $l \leftarrow \lambda l + \text{RL loss}(s, a, r, s'; \theta_o)$ `// Add RL loss for batch with` θ_o
  $\theta_o, \phi \leftarrow \text{optimize}((\theta_o, \phi), l)$ `// update online parameters`
  $\theta_m \leftarrow \tau\theta_o + (1 - \tau)\theta_m$ `// update target parameters`
end

2.1.3 Transition Model Architecture

For the transition model h, we apply a convolutional network directly to the $64 \times 7 \times 7$ spatial output of the convolutional encoder f_o. The network comprises

two 64-channel convolutional layers with 3×3 filters, with batch normalization (Ioffe and Szegedy, 2015) after the first convolution and ReLU nonlinearities after each convolution. We append a one-hot vector representing the action taken to each location in the input to the first convolutional layer, similar to Schrittwieser et al. (2019). We use a maximum prediction depth of $K = 5$, and we truncate calculation of the SPR loss at episode boundaries to avoid encoding environment reset dynamics into the model. Encoders for DeepMind Control generally output a vector representation rather than a spatial feature map (e.g., as in Lillicrap et al., 2016), so we instead use a two-layer multilayer perceptron applied to the 50-dimensional vector representation. We use ReLU nonlinearities and batch normalization after the first layer, as in Atari. As actions are continuous rather than discrete, we concatenate actions themselves to the input to the first layer of the MLP.

2.1.4 Data Augmentation

When using augmentation, we use the same set of image augmentations as in DrQ from Kostrikov et al. (2020), consisting of small random shifts and color jitter; see Figure 2.4 for an example of the augmentation used. We found it important to normalize activations to lie in $[0, 1]$ at the output of the convolutional encoder and transition model when using augmentation, as in Schrittwieser et al. (2019). We use Kornia (Riba et al., 2020) for efficient GPU-based data augmentations.

When not using augmentation, we find that SPR performs better when dropout with probability 0.5 is applied at each layer in the online and target encoders. This is consistent with Laine and Aila (2016); Tarvainen and Valpola (2017), who find that adding noise inside the network is important when not using image-specific augmentation, as proposed by Bachman et al. (2014). We found that applying dropout in this way was not helpful when using image-specific augmentation.

2.1.5 Implementation Details

For our Atari experiments, we largely follow van Hasselt et al. (2019) for DQN hyperparameters, with several exceptions. We follow DrQ (Kostrikov et al., 2020) by: using the 3-layer convolutional encoder from Mnih et al. (2015), using 10-step returns instead of 20-step returns for Q-learning, and not using a separate DQN target network when using augmentation. We also perform two gradient steps

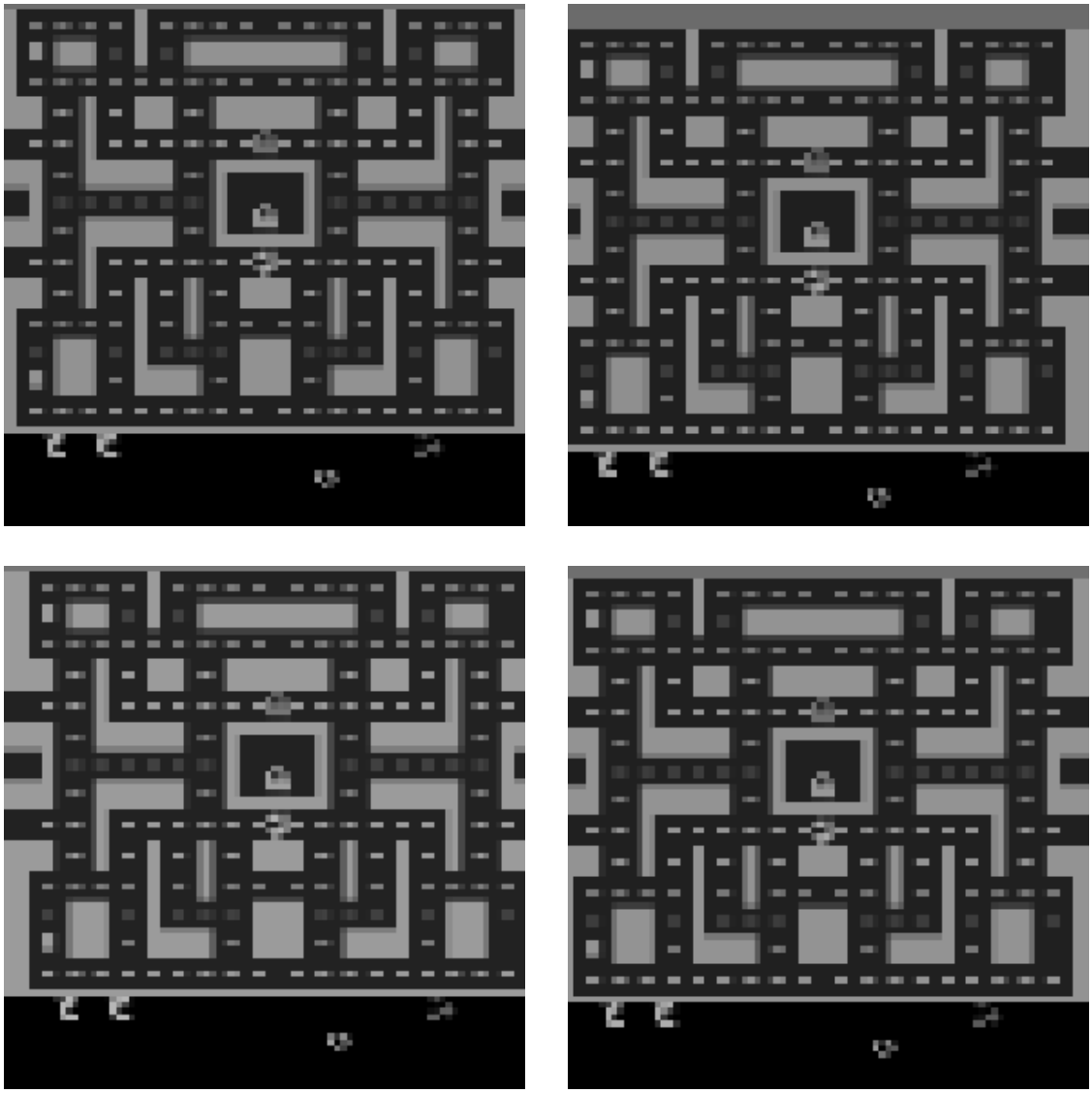

Figure 2.4 – Upper-left: a preprocessed image taken from the Atari game Ms Pacman, as it would be presented to the DQN used in SPR. Others: Augmented views of this image under the augmentation scheme used for SPR (Kostrikov et al., 2020).

per environment step instead of one. We show results for this configuration with and without augmentation in Table 2.4, and confirm that these changes are not themselves responsible for our performance. We reuse the first layer of the DQN MLP head as the SPR projection head g_o. When using dueling DQN (Wang et al., 2016), g_o concatenates the outputs of the first layers of the value and advantage heads. When these layers are noisy (Fortunato et al., 2018), g_o does not use the noise parameters. Finally, we parameterize the predictor q as a linear layer. For $\mathcal{L}_\theta^{\text{total}} = \mathcal{L}_\theta^{\text{RL}} + \lambda \mathcal{L}_\theta^{\text{SPR}}$, we use $\lambda = 2$ based on early experiments.

For our DM Control experiments, we adapt the publicly available DrQ codebase (see link in Kostrikov et al., 2020) based on SAC. Following results from DrQ showing that type of encoder used is not important when using augmentation, we replace the convolutional portion of the encoder used by DrQ with the three-layer convolutional network used in Atari, which is more computationally efficient. Unlike some other approaches that introduce a separate optimization step for a representation learning objective, we follow our approach in Atari by jointly optimizing a sum of the critic (value learning) and SPR losses. Due to the dramatically-varying scales of the critic loss, we find that the SPR loss weight λ must be tuned separately per environment, although $\lambda = 200$ is the most commonly successful value. As the first layer of the Q-head in SAC is action-conditioned, we instead define our projection g_o as a separate MLP. Other hyperparameters follow Atari.

Our implementation is based on `rlpyt` (Stooke and Abbeel, 2019) and `PyTorch` (Paszke et al., 2019).

2.2 Results

2.2.1 Sample-Efficient Atari

We test SPR on the sample-efficient Atari setting introduced by Kaiser et al. (2019) and van Hasselt et al. (2019). In this task, only 100,000 environment steps of training data are available – equivalent to 400,000 frames, or just under two hours – compared to the typical standard of 50,000,000 environment steps, or roughly 39 days of experience. When used without image augmentation, SPR demonstrates scores comparable to the previous best result from Kostrikov et al. (2020). When

Table 2.1 – Hyperparameters for SPR on Atari, with and without augmentation.

Parameter	**Setting (for both variations)**
Gray-scaling	True
Observation down-sampling	84x84
Frames stacked	4
Action repetitions	4
Reward clipping	[-1, 1]
Terminal on loss of life	True
Max frames per episode	108K
Update	Distributional Q
Dueling	True
Support of Q-distribution	51
Discount factor	0.99
Minibatch size	32
Optimizer	Adam
Optimizer: learning rate	0.0001
Optimizer: β_1	0.9
Optimizer: β_2	0.999
Optimizer: ϵ	0.00015
Max gradient norm	10
Priority exponent	0.5
Priority correction	$0.4 \rightarrow 1$
Exploration	Noisy nets
Noisy nets parameter	0.5
Training steps	100K
Evaluation trajectories	100
Min replay size for sampling	2000
Replay period every	1 step
Updates per step	2
Multi-step return length	10
Q network: channels	32, 64, 64
Q network: filter size	8×8, 4×4, 3×3
Q network: stride	4, 2, 1
Q network: hidden units	256
Non-linearity	ReLU
Target network: update period	1
λ (SPR loss coefficient	2
K (Prediction Depth)	5

Parameter	**With Augmentation**	**Without Augmentation**
Data Augmentation	Random shifts ($\pm$4 pixels) & Intensity(scale=0.05)	None
Dropout	0	0.5
τ (EMA coefficient)	0	0.99

combined with image augmentation, SPR achieves a median human-normalized score of 0.415, which is a new state-of-the-art result on this task. SPR achieves super-human performance on seven games in this data-limited setting: Boxing, Krull, Kangaroo, Road Runner, James Bond, Up N Down, and Crazy Climber, compared to a maximum of two for any previous methods, and achieves scores higher than DrQ (the previous state-of-the-art method) on 23 out of 26 games. See Table 2.2 for a full list of scores. For consistency with previous works, we report human and random scores from Wang et al. (2016), and compare against SimPLe (Kaiser et al., 2019), Data-Efficient Rainbow (DER, van Hasselt et al., 2019), Overtrained Rainbow (OTRainbow, Kielak, 2020), CURL (Srinivas et al., 2020) and DrQ (Kostrikov et al., 2020).

Atari Evaluation

We evaluate the performance of different methods by computing the average episodic return at the end of training. It is common to normalize scores with respect to expert human scores to account for different scales of scores in each game. The human-normalized performance of an agent on a game is calculated as $\frac{\text{agent score} - \text{random score}}{\text{human score} - \text{random score}}$ and then aggregated across the 26 games by taking their mean or median. It is common to report the median human-normalized performance, as the median is less susceptible to outliers. However, we find that in some games human scores are so high that differences between methods are washed out when normalizing scores. This makes it difficult for scores in these games, such as Alien, Asterix, and Seaquest, to influence aggregate metrics.

Moreover, we find that the use of the median, as opposed to mean or some other aggregate metric, has some traits that make it undesirable in the sample-efficient setting (and perhaps in the regular setting as well; see (Badia et al., 2020)). In particular, performance on the median human-normalized game score is affected only by a small number of games, such as Pong and Battlezone. On many other games, such as Alien and Seaquest, agents rarely achieve strong enough performance to influence the median, while on games such as Krull, Kung Fu Master, and James Bond agents almost always achieve performance far above the median. We hypothesize that this is due to some games simply being relatively less tractable for DQNs, perhaps due to human visual and game-based priors being relatively more important.

Table 2.2 – Mean episodic returns on the 26 Atari games considered by Kaiser et al. (2019) after 100k environment steps. Results for SPR are recorded at the end of training and averaged over 10 random seeds, including Human-Normalized Score (HNS) and DQN-Normalized Score (DNS). Best results for each game and metric are **bolded**. SPR outperforms prior methods on all aggregate metrics, and exceeds expert human performance on 7 out of 26 games while using a similar amount of experience.

Game	Random	Human	SimPLe	DER	OTRainbow	CURL	DrQ	SPR (no Aug)	SPR
Alien	227.8	7127.7	616.9	739.9	824.7	558.2	771.2	**847.2**	801.5
Amidar	5.8	1719.5	88.0	**188.6**	82.8	142.1	102.8	142.7	176.3
Assault	222.4	742.0	527.2	431.2	351.9	600.6	452.4	**665.0**	571.0
Asterix	210.0	8503.3	**1128.3**	470.8	628.5	734.5	603.5	820.2	977.8
Bank Heist	14.2	753.1	34.2	51.0	182.1	131.6	168.9	**425.6**	380.9
BattleZone	2360.0	37187.5	5184.4	10124.6	4060.6	14870.0	12954.0	10738.0	**16651.0**
Boxing	0.1	12.1	9.1	0.2	2.5	1.2	6.0	12.7	**35.8**
Breakout	1.7	30.5	16.4	1.9	9.8	4.9	16.1	12.9	**17.1**
ChopperCommand	811.0	7387.8	**1246.9**	861.8	1033.3	1058.5	780.3	667.3	974.8
Crazy Climber	10780.5	35829.4	62583.6	16185.3	21327.8	12146.5	20516.5	**43391.0**	42923.6
Demon Attack	152.1	1971.0	208.1	508.0	711.8	817.6	**1113.4**	370.1	545.2
Freeway	0.0	29.6	20.3	**27.9**	25.0	26.7	9.8	16.1	24.4
Frostbite	65.2	4334.7	254.7	866.8	231.6	1181.3	331.1	1657.4	**1821.5**
Gopher	257.6	2412.5	771.0	349.5	**778.0**	669.3	636.3	774.5	715.2
Hero	1027.0	30826.4	2656.6	6857.0	6458.8	6279.3	3736.3	5707.4	**7019.2**
Jamesbond	29.0	302.8	125.3	301.6	112.3	**471.0**	236.0	367.2	365.4
Kangaroo	52.0	3035.0	323.1	779.3	605.4	872.5	940.6	1359.5	**3276.4**
Krull	1598.0	2665.5	**4539.9**	2851.5	3277.9	4229.6	4018.1	3123.1	3688.9
Kung Fu Master	258.5	22736.3	**17257.2**	14346.1	5722.2	14307.8	9111.0	15469.7	13192.7
Ms Pacman	307.3	6951.6	**1480.0**	1204.1	941.9	1465.5	960.5	1247.7	1313.2
Pong	-20.7	14.6	**12.8**	-19.3	1.3	-16.5	-8.5	-16.0	-5.9
Private Eye	24.9	69571.3	58.3	97.8	100.0	**218.4**	-13.6	52.6	124.0
Qbert	163.9	13455.0	**1288.8**	1152.9	509.3	1042.4	854.4	606.6	669.1
Road Runner	11.5	7845.0	5640.6	9600.0	2696.7	5661.0	8895.1	10511.0	**14220.5**
Seaquest	68.4	42054.7	**683.3**	354.1	286.9	384.5	301.2	580.8	583.1
Up N Down	533.4	11693.2	3350.3	2877.4	2847.6	2955.2	3180.8	6604.6	**28138.5**
Median HNS	0.000	1.000	0.144	0.161	0.204	0.175	0.268	0.307	**0.415**
Mean HNS	0.000	1.000	0.443	0.285	0.264	0.381	0.357	0.463	**0.704**
Median DNS	0.000	0.994	0.118	0.142	0.103	0.142	0.131	0.225	**0.361**
Mean DNS	0.000	23.382	0.232	0.239	0.197	0.325	0.171	0.336	**0.510**
# Superhuman	0	N/A	2	2	1	2	2	5	**7**

This behavior is well-known to make the mean human-normalized score a poor predictor of overall performance, as it is dominated by outliers with very high scores (Badia et al., 2020). However, it perhaps counter-intuitively also makes the median quite sensitive to individual games; in our reported results without augmentation, for example, weakened performance on Frostbite alone could result in a median score of 0.24, while improved performance on only Battle Zone could yield a median score of 0.38. This applies to other algorithms as well; for our baseline DQN with augmentation, similar changes could lead to median scores ranging between 0.435 and 0.245. We thus also provide scores normalized not by human performance but by scores from the original DQN as reported by Wang et al. (2016) [3] We also report the number of games with super-human performance, an important measure of data efficiency given that 400k frames corresponds to roughly the same amount of environment interaction given to human testers.

Additionally, we note that the standard evaluation protocol of evaluating over only 500,000 frames per game is problematic, as the quantity we are trying to measure is expected return over episodes. Due to the very long lengths of some episodes (up to 108,000 frames), this method may collect as few as four complete episodes. This problem is compounded by the fact that better policies tend to have longer episodes on many games, leading stronger algorithms to experience even greater variance in this estimate of expected episodic returns. As variance is already a concern in deep RL (see Henderson et al., 2018), we propose evaluating over 100 episodes irrespective of their length, but we report results using the standard metric.

Finally, in the interest of replicability, we provide full bootstrap distributions to estimate the uncertainty in our headline result. We construct 1000 synthetic sets of results by independently resampling results for each game, and then recalculate aggregate metrics for each set of results. Distributions can be seen in Figures 2.5-2.8. We find that the variance of the resulting distributions of aggregate metrics is quite large, with 95% confidence intervals given in Table 2.3.

3. Although a natural choice, normalizing by scores from Data Efficient Rainbow is undesirable; its very poor results on several games result in extremely high normalized scores and make the mean unusable as a metric.

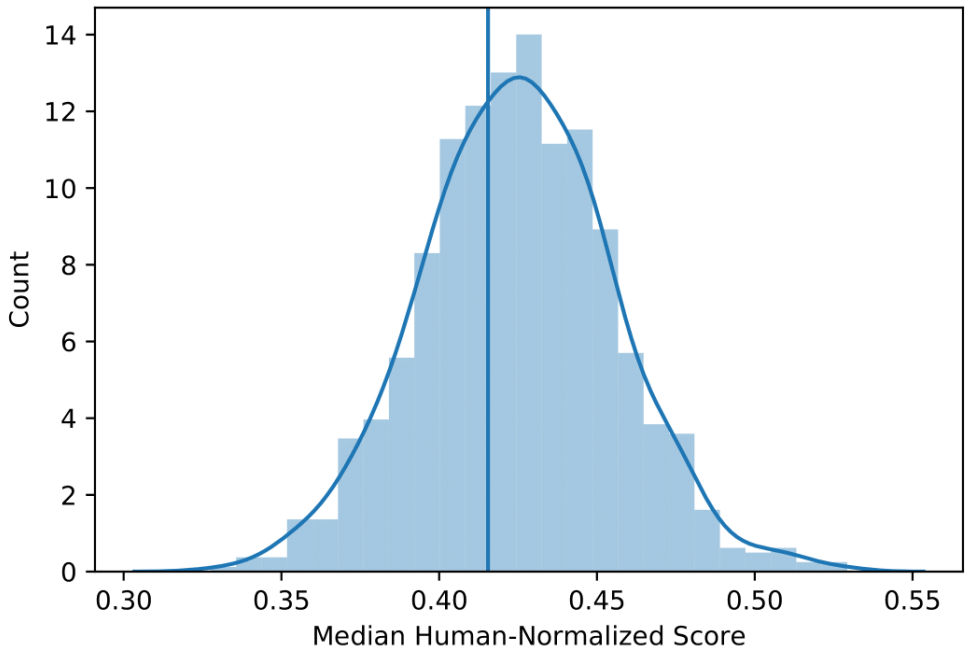

Figure 2.5 – A bootstrapped estimate of the distribution of SPR's median Human-Normalized score on Atari when using augmentation, averaged over 10 random seeds. Vertical line denotes the original value.

Table 2.3 – 95% percentile bootstrap confidence intervals for aggregate metrics for SPR with augmentation on Atari.

Metric	Point Estimate	95% Confidence Interval
Median Human-Normalized Score	0.415	(0.365, 0.484)
Mean Human-Normalized Score	0.704	(0.617, 0.809)
Median DQN-Normalized Score	0.361	(0.316, 0.386)
Mean DQN-Normalized Score	0.510	(0.410, 0.635)

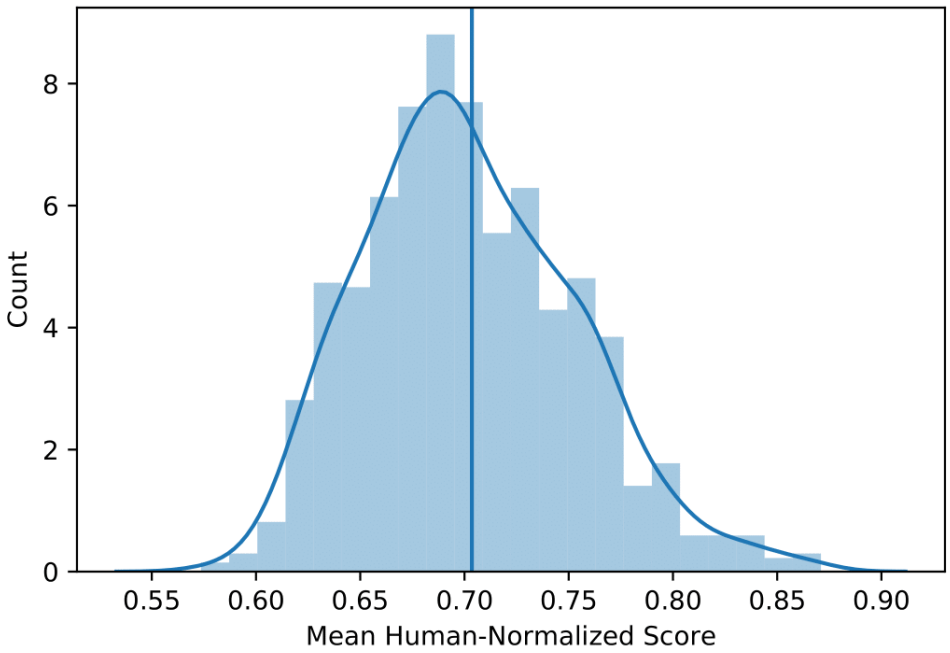

Figure 2.6 – A bootstrapped estimate of the distribution of SPR's mean Human-Normalized score on Atari when using augmentation, averaged over 10 random seeds. Vertical line denotes the original value.

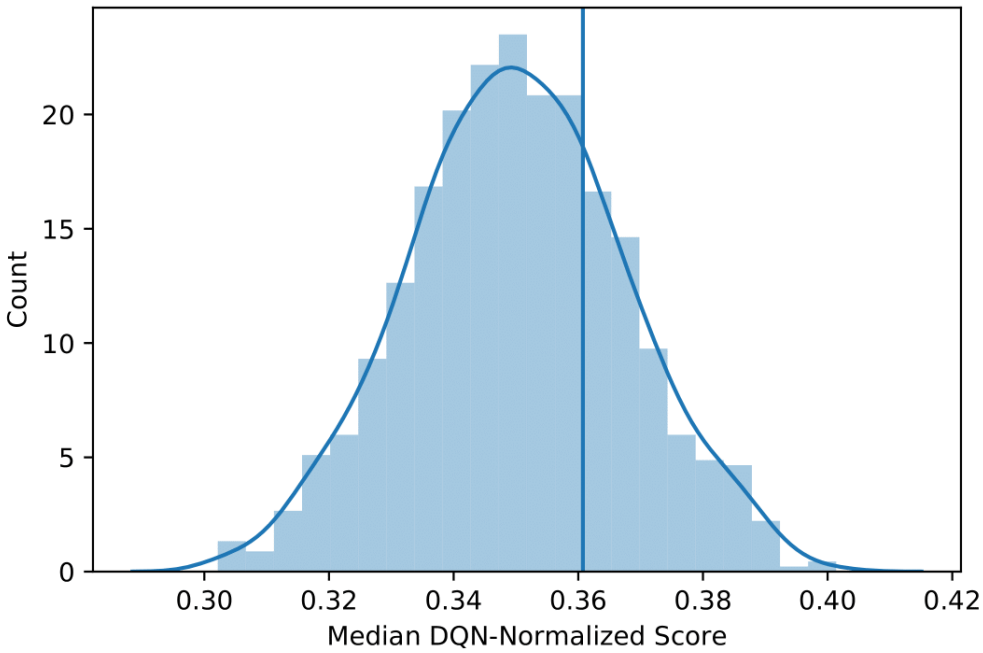

Figure 2.7 – A bootstrapped estimate of the distribution of SPR's median DQN-Normalized score on Atari when using augmentation, averaged over 10 random seeds. Vertical line denotes the original value.

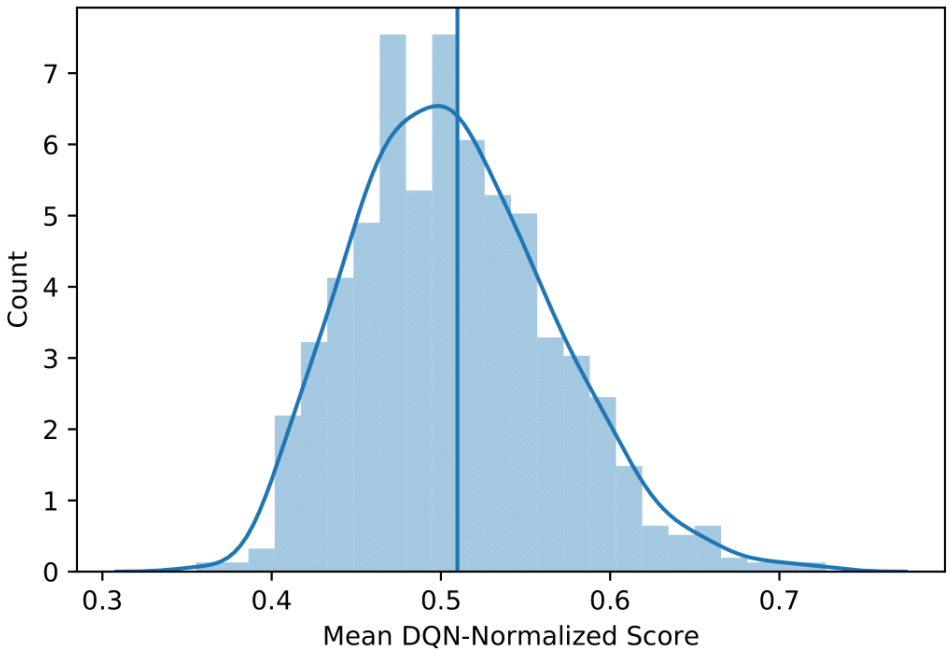

Figure 2.8 – A bootstrapped estimate of the distribution of SPR's mean DQN-Normalized score performance on Atari when using augmentation, averaged over 10 random seeds. Vertical line denotes the original value.

Table 2.4 – Scores on the 26 Atari games under consideration for our base DQN without SPR with and without augmentation, compared to previous methods. The high mean DQN-normalized score of our DQN without augmentation is due largely to a very high score on Private Eye, without which it would be comparable to DER.

Variant	Human-Normalized Score		DQN-Normalized Score	
	median	mean	median	mean
Our DQN (no aug)	0.204	0.240	0.149	0.374
OTRainbow	0.204	0.264	0.103	0.197
DER	0.161	0.285	0.142	0.239
Our DQN (w/ aug)	0.346	0.480	0.278	0.284
DrQ	0.268	0.357	0.131	0.171

Controlled baselines

To ensure that the hyper-parameter changes we make to the DER baseline are not solely responsible for our improved performance, we perform controlled experiments using the same hyper-parameters and same random seeds but with SPR disabled. We find that our DQN without augmentation is slightly stronger than Data-Efficient Rainbow and comparable to Overtrained Rainbow, while with augmentation enabled our results are somewhat stronger than those of DrQ (unsurprisingly so, as DrQ did not employ a number of the innovations included in Rainbow). None of these methods, however are close to the performance of SPR. Insofar as that our hyperparameters yield better performance than those used by previous methods, they may be considered an auxiliary contribution of this work. We show performance for these baselines in Table 2.4, and visualize the robust improvements granted by SPR in Figure 2.9.

2.2.2 DeepMind Control

We also evaluate SPR on the DeepMind Control (Tassa et al., 2018), testing over 19 different environments. We focus on the data-efficient setting where only 100k environment steps are given to the agent, as asymptotic (e.g., 500k step) performance is already essentially perfect for many environments. As our code is based on that of DrQ but contains some modifications, we run controlled experiments against

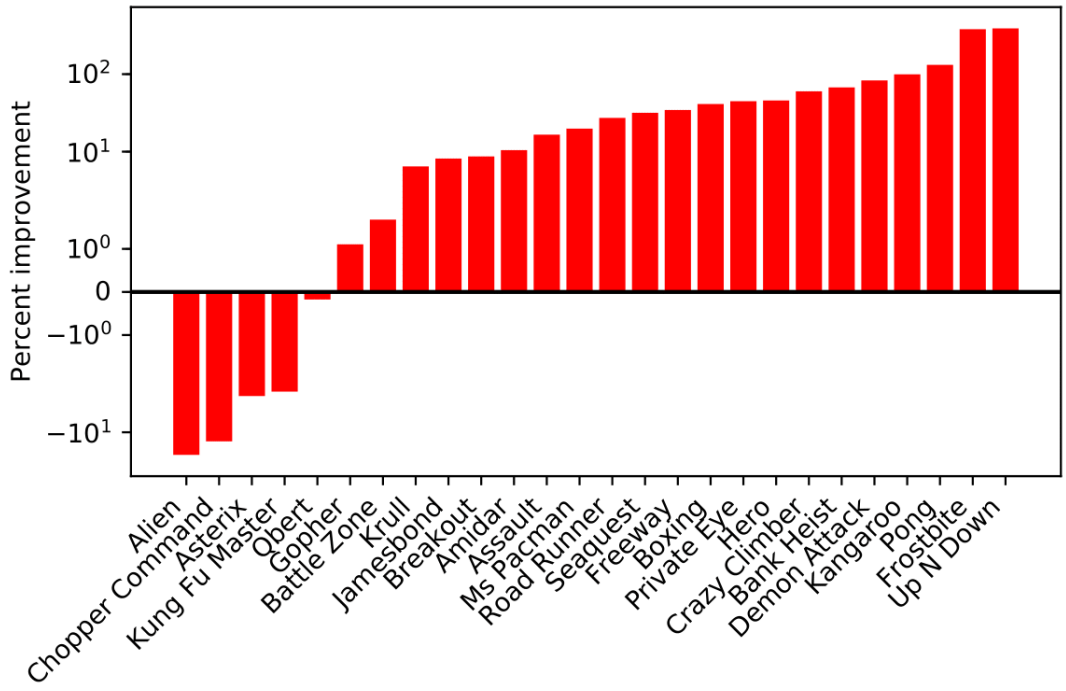

Figure 2.9 – SPR performance on Atari compared to our baseline DQN, averaged over 10 random seeds. Both SPR and the baseline are given augmented data. We plot $100 \cdot \frac{\text{SPR\_score} - \text{baseline\_score}}{\text{baseline\_score}}$, using a symmetric log scale.

Table 2.5 – Mean episodic returns on 19 DeepMind Control environments after 100k environment steps. The results are recorded at the end of training and averaged over five random seeds.

Environment	DrQ	SPR
acrobot swingup	**7.39**	5.36
cartpole balance	963.44	**990.14**
cartpole balance sparse	**1000.0**	**1000.0**
cartpole swingup	808.72	**809.44**
cartpole swingup sparse	**154.96**	140.26
cheetah run	347.25	**473.30**
cup in catch	**972.62**	968.06
finter turn easy	369.58	**394.76**
hopper hop	0.93	**152.16**
hopper stand	**520.47**	449.83
pendulum swingup	63.8	**235.6**
quadruped walk	63.89	**107.5**
quadruped run	72.97	**82.19**
walker stand	646.53	**961.33**
walker walk	**837.90**	818.94
finger spin	871.4	**982.54**
reacher easy	**530.5**	506.52
reacher hard	365.82	**434.9**
walker run	300.37	**338.73**

DrQ using its default settings. We find that SPR improves performance on 12 out of 19 environments, and degrades performance on only 7. We show full results in Table 2.5 and Figure 2.10.

2.3 Discussion

2.3.1 The role of the exponential moving average encoder

We find that using an EMA target encoder to be beneficial in most but not all circumstances, with the stability of training appearing to play a defining role. In

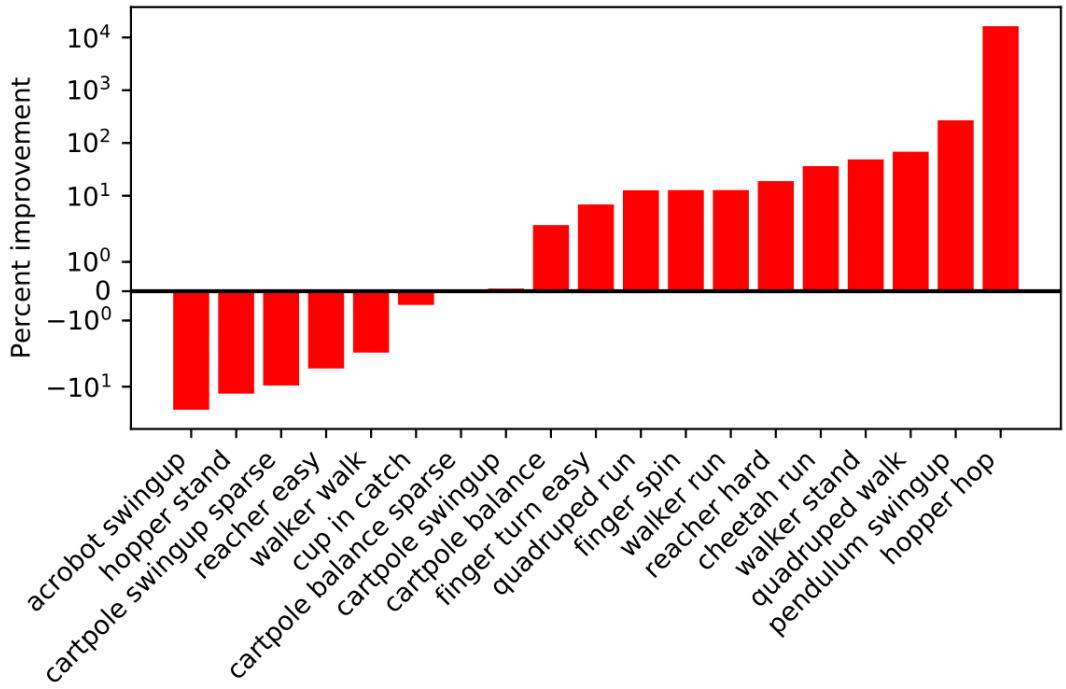

Figure 2.10 – SPR performance compared to the DrQ baseline, averaged over five random seeds. We plot $100 \cdot \frac{\text{SPR\_score} - \text{DrQ\_score}}{\text{DrQ\_score}}$, using a symmetric log scale.

this sense, using an EMA target encoder can be understood as being comparable to the role of the target network in deep reinforcement learning, where it serves to stabilize learning. Assuming a pefectly correct transition model h (i.e., that $h(f_o(s_t), a_t) = f_o(s_{t+1})$) [4] our choice of a linear layer as predictor q softly constrains the output of the projection layer $g_o(\hat{z})$ to be a linear transformation of the target projection $g_m(\tilde{z})$. This thus serves to limit how quickly the network's representations can change, a form of stabilization different from and weaker than that of traditional target networks in DRL, which instead limit how quickly the network's *outputs* can change. However, we should thus expect using an EMA target encoder to slow down learning when instability is not a concern, much like target networks limit how quickly new information can be propagated through an agent's estimates of value.

In continuous control, which is notoriously unstable (see for example Fujimoto et al., 2018), we found that using an EMA target encoder [5] greatly improved performance, and quickly discontinued experiments without it. In discrete control, on the other hand, we find that stability is less of a concern. Kostrikov et al. (2020) showed that data augmentation provides sufficient stabilization as to obviate the need for a separate DQN target network. We observe a similar result; when augmentation is in use, not using an EMA target encoder is slightly superior, even when the DQN is not using a separate target network. When augmentation is disabled, however, we find that using an EMA target encoder provides a boost in performance (see Tables 2.6 and 2.7).

2.3.2 Propagating gradients through targets is harmful

When not using an EMA target encoder, we find that allowing gradients to propagate through target representations leads to catastrophic reductions in performance both with and without augmentation, as can be seen in Tables 2.6 and 2.7. This can be informatively contrasted to other works, such as DeepMDP (Gelada et al., 2019), which employs a somewhat similar future prediction objective but propagates gradients through target representations as well. As a result, DeepMDP had a strong tendency to exhibit representational collapse; collapse to a constant

4. Obviously this does not occur in practice and is only possible in deterministic environments, but is a useful example

5. $\tau = 0.01$

Table 2.6 – Scores on the 26 Atari games under consideration for variants of SPR with different target encoder schemes, without augmentation.

Variant	Human-Normalized Score		DQN-Normalized Score	
	median	mean	median	mean
SPR ($\tau = 0.99$)	0.307	0.463	0.225	0.336
No Stopgradient SPR	0.208	0.375	0.233	0.301
SPR ($\tau = 0$)	0.228	0.512	0.246	0.312

Table 2.7 – Scores on the 26 Atari games under consideration for variants of SPR with different target encoder schemes, with augmentation.

Variant	Human-Normalized Score		DQN-Normalized Score	
	median	mean	median	mean
SPR ($\tau = 0$)	0.415	0.704	0.361	0.510
No Stopgradient SPR	0.278	0.515	0.231	0.344
SPR ($\tau = 0.99$)	0.396	0.622	0.287	0.356

vector minimizes both the DeepMDP and SPR prediction losses, and propagating gradients makes this solution far simpler to find. By contrast, contrastive methods such as CPC (Oord et al., 2018; Guo et al., 2018) are less likely to suffer from this problem, as they are inherently less prone to representational collapse (indeed, collapse to a constant vector leads to very high losses with InfoNCE). However, we consistently find that contrastive methods are outperformed by SPR (see next section).

2.3.3 Representational Collapse

One of the key questions surrounding non-contrastive self-supervised learning methods such as BYOL and SPR is the reason for their apparent stability. Although representational collapse to a fixed vector minimizes the SPR and BYOL losses, this solution appears never to be arrived at. Empirically, we find that adding SPR has little-to-no effect on the *homogeneity* of representations learned by a DQN, which we define as the average cosine similarity between representations of different states in each minibatch. We log this measure at the first layer of the Q head in Atari

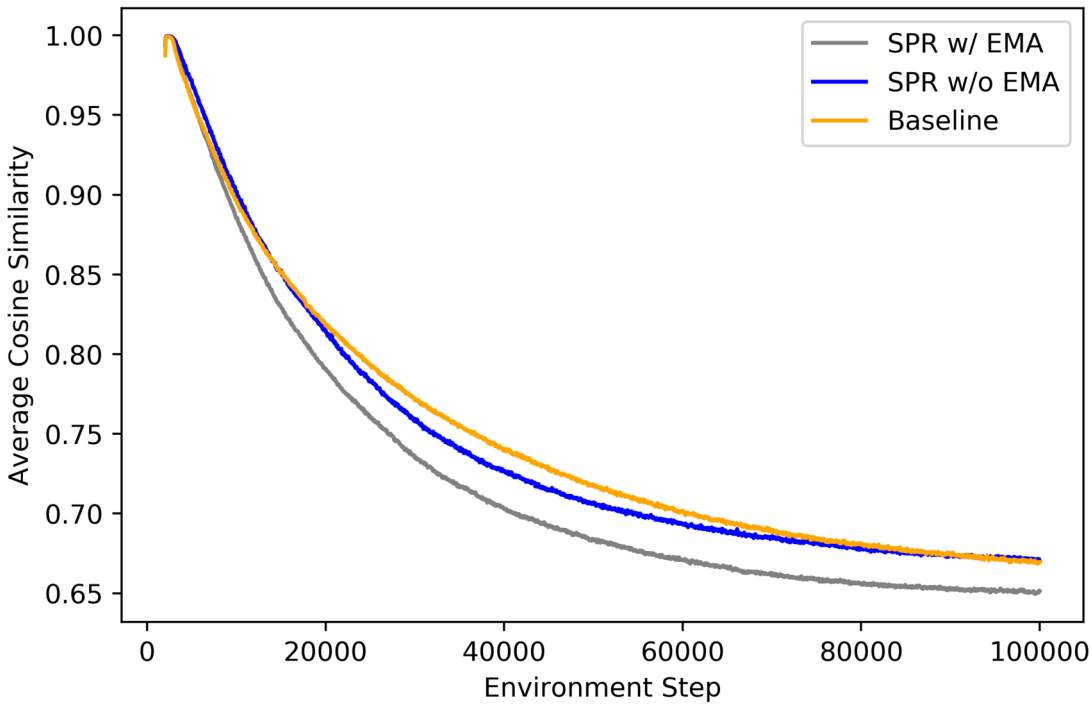

Figure 2.11 – Average cosine similarity between representations of different states for two variants of SPR with augmentation, $\tau = 0$ and $\tau = 0.99$, and our base DQN, averaged over a subset of 10 games. Results averaged over 10 random seeds per game.

(i.e., the projection layer used by SPR) on a subset of 10 games over the course of training (see Figure 2.11) and find that SPR if anything slightly increases the diversity of representations when used with an EMA target encoder and has no effect when not. Insofar as that there is a tendency for representations to be highly homogeneous, this is already present in the base DQN and is not caused by SPR.

2.3.4 Dynamics modeling is key

A key distinction between SPR and other recent approaches leveraging representation learning for reinforcement learning, such as CURL (Srinivas et al., 2020) and DRIML (Mazoure et al., 2020), is our use of an explicit multi-step dynamics model. We test two ablated versions of SPR, one with no dynamics modeling and one that models only a single step. Each of these variants has degraded performance compared to five-step SPR, with extended dynamics modeling improving performance (see Table 2.8). However, there is evidence that a BYOL-style objective (i.e.,

Table 2.8 – Scores on the 26 Atari games under consideration for variants of SPR with ablated temporal prediction.

Variant	Human-Normalized Score		DQN-Normalized Score	
	median	mean	median	mean
SPR	0.415	0.704	0.361	0.510
1-step SPR	0.301	0.570	0.346	0.337
Non-temporal SPR	0.271	0.507	0.295	0.326
No SPR	0.346	0.480	0.278	0.284

SPR without temporal prediction) provides at least some benefit on most aggregate metrics, perhaps due to providing additional supervision to learn representations invariant to augmentation.

2.3.5 Comparison with contrastive losses

Although many recent works in representation learning have employed contrastive learning, we find that SPR consistently outperforms both temporal and non-temporal variants of contrastive losses (see Table 2.9), including CURL (Srinivas et al., 2020).

Recent work has suggested that contrastive learning objectives such as InfoNCE implicitly optimize two quantities; the invariance of the representation to shifts or distortions, and the *uniformity* of the representation (Wang and Isola, 2020), and has suggested optimizing for these metrics individually. We note that the SPR loss is conceptually similar to the invariance loss suggested by Wang and Isola (2020), but lacks a comparable term to encourage representations to be distinct, which the theory suggests would be catastrophic. However, SPR outperforms the contrastive method CURL (Srinivas et al., 2020) by a large margin. To further examine this discrepancy, we implement four contrastive controls in our codebase:

— A temporal contrastive objective that learns an action-conditioned transition model and optimizes the InfoNCE (Oord et al., 2018) using negative examples from other time steps in the same trajectory as well as from other trajectories (this is roughly similar to the approach taken by CPC|Action, but using the InfoNCE loss)

— A temporal contrastive objective that directly draws target representations from one state after the encoder, without learning an explicit transition model (roughly similar to the approach taken by STDIM (Anand et al., 2019) or DRIML (Mazoure et al., 2020).

— A non-temporal contrastive objective similar to CURL (Srinivas et al., 2020). Targets are taken from different augmented views of the same image.

— A soft contrastive approach inspired by the repulsion objective proposed by Wang and Isola (2020). We optimize the repulsion objective jointly with the SPR loss, which is quite close to the "invariance" objective they propose. We use $t = 2$ and a weight equal to that given to the SPR loss, based on hyperparameters used by Wang and Isola (2020).

We use the same hyperparameters as SPR, including augmentation. We follow SimCLR (Chen et al., 2020a) in using normalized representations in calculating the contrastive loss, with a temperature of $\tau = 0.1$, and in using projection layers, defined identically to SPR. We also use a predictor, following the controls in Grill et al. (2020).

We show results for these experiments in Table 2.9. We find that SPR consistently outperforms all controls by large margins, consistent with experiments conducted by BYOL that indicate that repulsion provided by negative samples actively harms performance in their case (Grill et al., 2020). Perhaps more surprisingly, however, the contrastive approaches tested fail even to achieve the same level of performance as their base DQN; we hypothesize that the repulsive aspect of the contrastive losses somehow interferes with the DQN objective.

2.4 Related Work

2.4.1 Data-Efficient RL:

A number of works have sought to improve sample efficiency in deep RL. SiMPLe (Kaiser et al., 2019) learns an explicit pixel-level transition model for Atari to generate simulated training data, achieving strong results on several games in the 100k frame setting. However, both van Hasselt et al. (2019) and Kielak (2020)

Table 2.9 – Scores on the 26 Atari games under consideration for various contrastive alternatives to SPR implemented in our codebase.

Variant	Human-Normalized Score		DQN-Normalized Score	
	median	mean	median	mean
SPR	0.415	0.704	0.361	0.510
Base DQN	0.346	0.480	0.278	0.284
5-step contrastive	0.172	0.506	0.142	0.239
1-step contrastive	0.231	0.473	0.213	0.280
Non-temporal contrastive	0.200	0.379	0.179	0.268
SPR with repulsion	0.176	0.422	0.144	0.271

demonstrate that variants of Rainbow (Hessel et al., 2018) tuned for sample efficiency can achieve comparable or superior performance.

In the context of continuous control, several works propose to leverage a latent-space model trained on reconstruction loss to improve sample efficiency (Hafner et al., 2019; Lee et al., 2019a; Hafner et al., 2020). Most recently, DrQ (Kostrikov et al., 2020) and RAD (Laskin et al., 2020) have found that applying modest image augmentation can substantially improve sample efficiency in reinforcement learning, yielding better results than prior model-based methods. Data augmentation has also been found to improve generalization of reinforcement learning methods (Combes et al., 2018; Laskin et al., 2020) in multi-task and transfer settings. We show that data augmentation can be more effectively leveraged in reinforcement learning by forcing representations to be consistent between different augmented views of an observation while also predicting future latent states.

2.4.2 Representation Learning in RL:

Representation learning has a long history of use in reinforcement learning (For a detailed overview of previous methods, see Lesort et al., 2018). For example, CURL (Srinivas et al., 2020) recently proposed a combination of image augmentation and a contrastive loss to perform representation learning for RL. However, follow-up results from RAD (Laskin et al., 2020) suggest that most of the benefits of CURL come from its use of image augmentation rather than its contrastive loss.

CPC (Oord et al., 2018), CPC|Action (Guo et al., 2018), ST-DIM (Anand et al., 2019) and DRIML (Mazoure et al., 2020) propose to optimize various temporal contrastive losses in reinforcement learning environments. We perform an ablation comparing such temporal contrastive losses to our method in section 2.3.5. Kipf et al. (2019) propose to learn object-oriented contrastive representations by training a structured transition model based on a graph neural network.

SPR bears some resemblance to Deep MDP (Gelada et al., 2019), which trains a transition model with an unnormalized L2 loss to predict representations of future states, along with a reward prediction objective. However, DeepMDP uses its online encoder for prediction targets as well rather than employing a target encoder, and is thus prone to representational collapse (sec. C.5 in Gelada et al. (2019)). To mitigate this issue, DeepMDP relies on an additional observation reconstruction objective. In contrast, our model is self-supervised, trained entirely in the latent space, and uses a normalized loss. Our ablations (see section 2.3.1) demonstrate that using an EMA target encoder has a large impact on our method, making it another key difference between SPR and DeepMDP.

SPR is also similar to PBL (Guo et al., 2020), which also directly predicts representations of future states. However, PBL uses a separate target network trained via gradient descent, whereas SPR generates its own targets. Moreover, PBL studies multi-task generalization in the asymptotic limits of data, whereas SPR is concerned with single-task performance in low data regimes, using 0.01% as much data as PBL. Unlike PBL, SPR is also designed to work with data augmentation, similarly to Mean Teachers or BYOL (Tarvainen and Valpola, 2017; Grill et al., 2020), which empirically provides a large boost in performance.

3 Conclusion

In this work, we introduced Self-Predictive Representations (SPR), a self-supervised representation learning algorithm designed to improve the data efficiency of deep reinforcement learning agents. SPR learns representations that are both temporally predictive and consistent across different views of environment observations, directly predicting representations of future states produced by a target encoder based on the agent itself. SPR achieves state-of-the-art performance on the 100k steps Atari benchmark, demonstrating significant improvements over prior work, and also provides improvements in challenging continuous control tasks in the DeepMind Control suite. Our experiments show that SPR is highly robust, and is able to outperform the previous state of the art even without data augmentation.

We therefore believe that SPR opens up a variety of avenues for future work. Recent work in both visual (Chen et al., 2020b) and language representation learning (Brown et al., 2020) has suggested that self-supervised models trained on large datasets perform exceedingly well on downstream problems with limited data, often outperforming methods trained using only task-specific data. Future works could similarly exploit large corpora of unlabelled data, perhaps from multiple MDPs or raw videos, to further improve the performance of RL methods in low-data regimes. As the SPR objective does not require supervision by reward data, it could be directly applied in such settings.

Another interesting direction is to use the transition model learned by SPR for planning. MuZero (Schrittwieser et al., 2019) has demonstrated that planning with a model supervised via reward and value prediction can work extremely well given sufficient (massive) amounts of data. It remains unclear whether such models can work well in low-data regimes, and whether augmenting such models with self-supervised objectives such as SPR can improve their data efficiency. Moreover, SPR also offers a more light-weight approach to model learning than MuZero; as the predictions generated by SPR are themselves the inputs to the agent's Q-function, they represent a form of indirect value prediction. Thus, it may be possible to

avoid the costly and potentially-disruptive step of training the transition model to explicitly predict value, as undertaken by methods such as MuZero (Schrittwieser et al., 2019).

Bibliography

Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. Fixing a broken elbo. In *International Conference on Machine Learning*, pages 159–168, 2018. Cited on page 3.

Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. Unsupervised state representation learning in atari. In *NeurIPS*, 2019. Cited on pages 6, 45, and 47.

Philip Bachman, Ouais al Sharif, and Doina Precup. Learning with pseudo-ensembles. *Advances in Neural Information Processing Systems (NIPS)*, 2014. Cited on page 26.

Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *NeurIPS*, 2019. Cited on pages vii, 4, 6, 7, 8, and 19.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. *arXiv preprint arXiv:2003.13350*, 2020. Cited on pages 16, 17, 19, 30, and 32.

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, 2013. Cited on pages 17, 19, and 20.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. *ICML*, 2017. Cited on page 21.

Yoshua Bengio. Learning deep architectures for ai. *Machine Learning*, 2(1):1–127, 2009. Cited on page 2.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007. Cited on pages 2 and 8.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. Cited on page 48.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *ICML*, 2020a. Cited on pages 6, 7, 8, 19, 25, and 45.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020b. Cited on pages 6, 9, 19, and 48.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020c. Cited on page 8.

Remi Tachet des Combes, Philip Bachman, and Harm van Seijen. Learning invariances for policy generalization. *arXiv preprint arXiv:1809.02591*, 2018. Cited on page 46.

Will Dabney, André Barreto, Mark Rowland, Robert Dadashi, John Quan, Marc G Bellemare, and David Silver. The value-improvement path: Towards better representations for reinforcement learning. *arXiv preprint arXiv:2006.02243*, 2020. Cited on page 15.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. Cited on pages vii, 4, and 7.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. Cited on page 5.

Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. Cited on page 19.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019. Cited on page 19.

William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. *arXiv preprint arXiv:2007.06700*, 2020. Cited on page 14.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. In *ICLR*, 2018. URL `https://openreview.net/forum?id=rywHCPkAW`. Cited on pages 9 and 28.

Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *arXiv preprint arXiv:1811.12560*, 2018. Cited on page 19.

S Fujimoto, H van Hoof, and D Meger. Addressing function approximation error in actor-critic methods. *Proceedings of Machine Learning Research*, 80:1587–1596, 2018. Cited on pages 15 and 41.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. *ICML*, 2019. Cited on pages 15, 41, and 47.

Raphael Gontijo-Lopes, Sylvia J Smullin, Ekin D Cubuk, and Ethan Dyer. Affinity and diversity: Quantifying mechanisms of data augmentation. *arXiv preprint arXiv:2002.08973*, 2020. Cited on page 4.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. Cited on pages 1 and 5.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent:

A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020. Cited on pages 4, 7, 8, 10, 11, 19, 23, 24, 45, and 47.

Daniel Guo, Bernardo Avila Pires, Bilal Piot, Jean-bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap latent-predictive representations for multitask reinforcement learning. *arXiv preprint arXiv:2004.14646*, 2020. Cited on pages 15 and 47.

Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Bernardo A Pires, and Rémi Munos. Neural predictive belief representations. *ICML*, 2018. Cited on pages 42 and 47.

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. Cited on page 5.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018. Cited on pages 15 and 22.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *ICML*, 2019. Cited on pages 20 and 46.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *ICLR*, 2020. Cited on pages 16 and 46.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. Cited on page 2.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019. Cited on pages 6, 8, 9, 19, and 25.

Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with

contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019. Cited on pages 6, 9, and 19.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. Cited on page 32.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, 2018. Cited on pages 14, 21, and 46.

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. 2016. Cited on page 3.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *ICLR*, 2019. Cited on pages 6, 7, and 19.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. Cited on pages 2 and 26.

Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model based reinforcement learning for atari. In *ICLR*, 2019. Cited on pages vii, ix, 16, 20, 21, 22, 24, 28, 30, 31, and 45.

Kacper Piotr Kielak. Do recent advancements in model-based deep reinforcement learning really improve data efficiency?, 2020. URL `https://openreview.net/forum?id=Bke9u1HFwB`. Cited on pages 30 and 45.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. 2013. Cited on page 3.

Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. *arXiv preprint arXiv:1911.12247*, 2019. Cited on page 47.

Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020. Cited on pages vii, 16, 17, 21, 22, 24, 26, 27, 28, 30, 41, and 46.

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. Cited on page 4.

Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016. Cited on page 26.

Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020. Cited on pages 16, 24, and 46.

Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019a. Cited on page 46.

Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019b. Cited on page 15.

Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning, 2020. Cited on page 17.

Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Franois Goudou, and David Filliat. State representation learning for control: An overview. *Neural Networks*, 108, 2018. Cited on pages 15 and 46.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR (Poster)*, 2016. Cited on pages 15 and 26.

William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. 2016. Cited on page 5.

Bogdan Mazoure, Remi Tachet des Combes, Thang Doan, Philip Bachman, and R Devon Hjelm. Deep reinforcement and infomax learning. *arXiv preprint arXiv:2006.07217*, 2020. Cited on pages 6, 43, 45, and 47.

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), 2015. Cited on pages 14, 17, 20, and 26.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. Cited on page 2.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. Cited on pages 6, 7, 8, 15, 19, 42, 44, and 47.

OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019. URL `https://arxiv.org/abs/1912.06680`. Cited on pages 16 and 19.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. Cited on page 28.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992. Cited on page 8.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. Cited on page 5.

Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in neural information processing systems*, pages 3546–3554, 2015. Cited on page 9.

Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *The IEEE Winter Conference on Applications of Computer Vision*, 2020. Cited on page 26.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019. Cited on pages 19, 26, 48, and 49.

Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *Icdar*, volume 3, 2003. Cited on page 4.

Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020. Cited on pages 9 and 10.

Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020. Cited on pages 15, 30, 43, 44, 45, and 46.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1), 2014. Cited on page 9.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015. Cited on page 2.

Adam Stooke and Pieter Abbeel. rlpyt: A research code base for deep reinforcement learning in pytorch. *arXiv preprint arXiv:1909.01500*, 2019. Cited on page 28.

Richard Sutton. The bitter lesson. *Incomplete Ideas (blog)*, 2019. `http://www.incompleteideas.net/IncIdeas/BitterLesson.html`. Cited on page 19.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL `http://incompleteideas.net/book/the-book-2nd.html`. Cited on pages 11, 12, 13, and 14.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017. Cited on pages 9, 11, 19, 23, 26, and 47.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018. Cited on pages 16, 19, 21, and 37.

Gerald Tesauro. Practical issues in temporal difference learning. In *Advances in neural information processing systems*, pages 259–266, 1992. Cited on page 2.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019. Cited on page 19.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. Cited on page 16.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=HkL7n1-0b`. Cited on page 3.

Michael Tschannen, Josip Djolonga, Paul K Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. *arXiv preprint arXiv:1907.13625*, 2019. Cited on page 6.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016. Cited on pages 20 and 21.

Hado P van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in reinforcement learning? In *NeurIPS*, 2019. Cited on pages 16, 20, 21, 26, 28, 30, and 45.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. Cited on page 5.

Ramakrishna Vedantam, Karan Desai, Stefan Lee, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. Probabilistic neural symbolic models for interpretable visual question answering. In *International Conference on Machine Learning*, pages 6428–6437, 2019. Cited on page 9.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008. Cited on pages 3 and 4.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 2019. Cited on pages 16 and 19.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere, 2020. Cited on pages 9, 11, 44, and 45.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *ICML*, 2016. Cited on pages 21, 28, 30, and 32.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*, 2019. Cited on page 19.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020. Cited on page 10.

Larry S Yaeger, Richard F Lyon, and Brandyn J Webb. Effective training of a

neural network character classifier for word recognition. In *Advances in neural information processing systems*, pages 807–816, 1997. Cited on page 4.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. *arXiv preprint arXiv:1910.01741*, 2019. Cited on pages 15 and 17.