Université de Montréal

Building Sample-Efficient Reinforcement Learning

par

Max Allen Schwarzer

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

Thèse présentée en vue de l'obtention du grade de
Philosophiæ Doctor (Ph.D.)
en Informatique

May 28, 2024

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée

Building Sample-Efficient Reinforcement Learning

présentée par

Max Allen Schwarzer

a été évaluée par un jury composé des personnes suivantes :

Glen Berseth

(président-rapporteur)

Aaron Courville

(directeur de recherche)

Marc Bellemare

(codirecteur)

Sarath Chandar

(membre du jury)

Dhruv Batra

(examinateur externe)

(représentant du doyen de la FESP)

Résumé

L'efficacité des données est un défi clé pour l'apprentissage par renforcement profond (DRL), limitant souvent son utilisation aux environnements où des quantités illimitées de données simulées sont disponibles. J'envisage une gamme de solutions pour résoudre ce problème. Nous commençons par proposer une méthode permettant d'exploiter des données non étiquetées pour pré-entraîner des représentations qui sont ensuite affinées sur une petite quantité de données spécifiques à la tâche. Pour apprendre des représentations qui capturent divers aspects de la tâche sous-jacente, j'emplois une combinaison de modélisation des dynamiques latentes et de RL conditionné par objectif non supervisé. Cette approche surpasse nettement les travaux antérieurs combinant le pré-entraînement des représentations hors ligne avec l'affinement spécifique à la tâche, et se compare favorablement à d'autres méthodes de pré-entraînement nécessitant des ordres de grandeur plus de données. Nous identifions ensuite et discutons d'un défaut commun des algorithmes de DRL : une tendance à se fier aux interactions précoces et à ignorer les preuves utiles rencontrées plus tard. Les agents de DRL encourent un risque de surapprentissage par rapport aux expériences antérieures, affectant négativement le reste du processus d'apprentissage. Inspirés par les sciences cognitives, je fais référence à cet effet comme étant *le biais de primauté*. Nous proposons un mécanisme simple mais généralement applicable qui s'attaque au biais de primauté en réinitialisant périodiquement une partie de l'agent. Nous appliquons ce mécanisme aux algorithmes dans les domaines d'action discrets (Atari 100k) et continus (DeepMind Control Suite), améliorant constamment leurs performances. Nous démontrons ensuite que, poussée à l'extrême, cette approche basée sur la réinitialisation permet d'augmenter considérablement les ressources computationnelles même avec des données limitées, un phénomène que j'appelle *franchir le mur du ratio de relecture*. Les algorithmes basés sur cette stratégie sont capables d'exhiber un apprentissage beaucoup plus efficace que les travaux antérieurs, et permettent dans de nombreux cas un échange libre entre

computation et données. Enfin, je conclue en démontrant qu'il est également possible de mettre à l'échelle les réseaux neuronaux utilisés dans le RL efficace en termes de données, simplement en modifiant certains hyperparamètres. En combinaison avec les autres avancées réalisées jusqu'à présent, cela nous permet d'atteindre une efficacité d'apprentissage surhumaine sur Atari 100k même en apprenant purement à partir de zéro et sans utiliser un modèle pour la planification.

Mots-clés. **Apprentissage par renforcement, Apprentissage profond, Efficacité des données, Apprentissage automatique**

Abstract

Data efficiency is a key challenge for deep reinforcement learning (RL), often limiting its use to settings where unlimited quantities of simulated data are available. I consider a range of solutions to address this problem. I begin by proposing a method to leverage unlabeled data to pretrain representations that are then finetuned on a small amount of task-specific data. To learn representations that capture diverse aspects of the underlying task I employ a combination of latent dynamics modelling and unsupervised goal-conditioned RL. This approach significantly surpasses prior work combining offline representation pretraining with task-specific finetuning, and compares favourably with other pretraining methods that require orders of magnitude more data. I then identify and discuss a common flaw of deep RL algorithms: a tendency to rely on early interactions and ignore useful evidence encountered later. Deep RL agents incur a risk of overfitting to earlier experiences, negatively affecting the rest of the learning process. Inspired by cognitive science, I refer to this effect as *the primacy bias*. I propose a simple yet generally-applicable mechanism that tackles the primacy bias by periodically resetting a part of the agent. I apply this mechanism to algorithms in both discrete (Atari 100k) and continuous action (DeepMind Control Suite) domains, consistently improving their performance. I then demonstrate that when taken to the extreme, this reset-based approach allows computational resources to be scaled up enormously even with limited data, a phenomenon which I call *breaking the replay ratio barrier*. Algorithms based on this strategy are able to exhibit far more efficient learning than prior work, and allow computation and data to be freely exchanged in many cases. Finally, I conclude by demonstrating that it is also possible to scale up the neural networks used in sample-efficient RL, simply by changing certain hyperparameters. In combination with the other advances made so far, this allows us to achieve super-human learning efficiency

on Atari 100k even when learning purely from scratch and not using a model for planning.

Keywords. **Reinforcement Learning, Deep Learning, Data Efficiency, Machine Learning**

Contents

List of Tables

List of Figures

Liste des sigles et des abréviations

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
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

Remerciements

I am very grateful for the support I have received at Mila. I particularly wish to thank my advisors Aaron Courville and Marc Bellemare, who have taken immense amounts of time to introduce me to machine learning research and reinforcement learning, and have profoundly shaped my research career thus far. I likewise owe debts to Pablo Castro and Rishabh Agarwal, for their guidance and support during my long period at Google Brain, and for their support in helping me do the last half of my PhD at Google.

I also wish to thank my coauthors, who have made my experience at Mila as productive as it has been, including Evgenii Nikishin, Pierluca D'Oro, Johan Obando-Ceron, Nitarshan Rajkumar, Ankesh Anand, Devon Hjelm, Philip Bachman, Michael Noukhovitch, Rishabh Agarwal, Samuel Lavoie, Christos Tsirigotis, Laurent Charlin, Ankit Vani, Pablo Castro, Pierre-Luc Bacon, Aaron Courville and Marc Bellemare.

I am also extremely thankful for the astonishing computational resources provided to me at Mila, both through the Mila cluster and Compute Canada, and for the resources that were later made available to me at Google.

Finally, the work reported in this thesis would not have been possible without the financial support from: Calcul Quebec, Compute Canada, Borealis, le Fonds de recherche du Québec, the Canada Research Chairs and CIFAR.

Chapter 1

Introduction

Reinforcement learning (RL) is a family of approaches for creating artificial agents for solving sequential decision-making problems by learning to maximize a reward signal. Deep RL combines RL approaches with deep neural networks, and has been immensely successful in scaling RL to approach problems such as playing challenging games (Mnih et al., 2013; Silver et al., 2016a), drug discovery (Popova et al., 2018), flying stratospheric balloons (Bellemare et al., 2020), control for nuclear fusion (Degrave et al., 2022a) and robotic manipulation (Akkaya et al., 2019; Lee et al., 2020). Despite such successes, it remains very challenging to apply deep RL to tackle real-world decision making problems. One key issue limiting the widespread use of deep RL is sample efficiency: many RL algorithms require months (Mnih et al., 2015) to years (Schrittwieser et al., 2020) to millenia (OpenAI et al., 2019) of real-time experience data to achieve acceptable performance, with harder tasks consistently requiring more and more data. Humans, by comparison, are often able to learn the same tasks to an acceptable standard with hours to days of experience; how exactly humans do this is unknown, but it is clear that humans leverage prior knowledge effectively (Dubey et al., 2018), is enormously larger than most RL systems.[1]

In this thesis, I present a collection of works designed to address this issue by improving the sample efficiency of deep reinforcement learning agents, with the ultimate goal of making them more applicable to real-world problems where data is scarce and expensive by matching or surpassing human sample-efficiency. I begin by discussing

[1]The precise number of neurons and synapses in the human brain is not known and varies between individuals, but one may expect something on the order of 240 trillion (Koch, 2004) synapses; by comparison, the largest neural network trained in this thesis has a bit over 10 million parameters.

self-supervised pretraining, in which unlabeled data is leveraged to learn representations that then accelerate reinforcement learning. I continue with a discussion of the primacy bias in deep reinforcement learning, a phenomenon in which reinforcement learning agents overfit to their early experiences, and some techniques that improve sample efficiency by mitigating the primacy bias. I then present an extension of this work, which demonstrates that aggressively countering the primacy bias allows us to benefit from replay ratio scaling, using more compute for fixed amounts of data. Finally, I demonstrate that it is also possible, through a comprehensive set of hyperparameter changes, to scale up the neural networks used in sample-efficient RL, leading to enormous increases in performance and culminating in the creation of an algorithm that is capable of super-human learning efficiency on standard reinforcement learning benchmarks.

1.1. Thesis Outline

I structure the thesis in six parts: a background section, consisting largely of prior work and common terms in reinforcement learning and especially self-supervised learning; four contribution chapters, each consisting of a contextualized reproduction of the main body of a published work; and a conclusion, which summarizes the overall structure and significance of the works presented here. Finally, we present a single, joint bibliography for all chapters, followed by the appendices and supplementary materials of the published works, which are in some cases needed to understand certain discussion points.

- Chapter 2 discusses the pre-requisite concepts and terminology required in order to understand this thesis, particularly aspects relating to representation learning and reinforcement learning, and covers relevant prior work. Readers familiar with these topics should feel empowered to skip this chapter.
- Chapter 3 discusses self-supervised algorithms to pretrain representations for deep reinforcement learning agents from arbitrary offline data, without requiring action or reward supervision, and how best to fine-tune networks pretrained in this fashion to achieve high performance on sample-efficient reinforcement learning tasks. This reproduces the following work:
  - **Max Schwarzer**, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Devon Hjelm, Philip Bachman & Aaron Courville. *Pretraining Representations for Data-Efficient Reinforcement Learning.* NeurIPS (2021).

I led this project and performed the vast majority of experimental work, including all of the results in the main body of the paper.

- Chapter 4 introduces the notion of the primacy bias to deep reinforcement learning, demonstrating that reinforcement learning algorithms may overfit to early experiences, and shows that a family of reset-based solutions can mitigate this and lead to large performance improvements. This reproduces the following work:
  - Evgenii Nikishin*, **Max Schwarzer***, Pierluca D'Oro*, Pierre-Luc Bacon & Aaron Courville. *The Primacy Bias in Deep Reinforcement Learning.* ICML (2022).

  I co-led this project from its inception and performed all Atari experiments, or roughly half of the overall results in the paper.
- Chapter 5 shows that addressing the primacy bias through frequent resets allows the compute used by sample-efficient reinforcement learning algorithms to be scaled up dramatically, leading to large improvements in sample efficiency. This reproduces the following work:
  - Pierluca D'Oro*, **Max Schwarzer***, Evgenii Nikishin, Pierre-Luc Bacon, Marc G Bellemare & Aaron Courville. *Sample-Efficient Reinforcement Learning by Breaking the Replay Ratio Barrier.* ICLR (2023).

  I co-led this project from its inception and performed all Atari experiments, or roughly half of the overall results in the paper.
- Chapter 6 demonstrates that it is simultaneously possible to scale up the neural networks used by sample-efficient reinforcement learning algorithms, simply by applying the techniques from previous chapters and thoroughly re-tuning the parameters of sample-efficient reinforcement learning algorithms to harmonize with larger models. This reproduces the following work:
  - **Max Schwarzer**, Johan Obando-Ceron, Marc G Bellemare, Aaron Courville, Rishabh Agarwal & Pablo Castro. *Bigger, Better, Faster: Human-Level Atari with Human-Level Efficiency.* ICML (2023).

  I led this project and performed the vast majority of experimental work, including all of the results in the main body of the paper.
- Chapter 7 discusses the overall significance of these works and some overarching lessons that may be taken from them, and reflects on the future of sample-efficient reinforcement learning research.

Chapter 2

Background

In this chapter, we will provide an overview of the core concepts required to understand the contributions presented in this proposal. 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).

2.1. 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.

2.1.1. 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 necessary method to improve learning, since methods of the time struggled to directly train deep neural networks (Tesauro, 1992; Bengio, 2009). This would change with the introduction of stabilizing techniques such as the rectified linear unit (e.g., as used in Nair et al., 2010), batch normalization (Ioffe et al., 2015), and skip connections (He et al., 2016; Srivastava et al., 2015), but without them neural networks were often limited to depths of at most a few layers (compared with the hundreds possible at the time of this writing (He et al., 2016)). 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 (generally supervised) task, improving performance (Bengio et al., 2007).

2.1.2. Reconstruction

One of the oldest and most popular techniques for deep representation learning is *reconstruction*, 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 et al., 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).

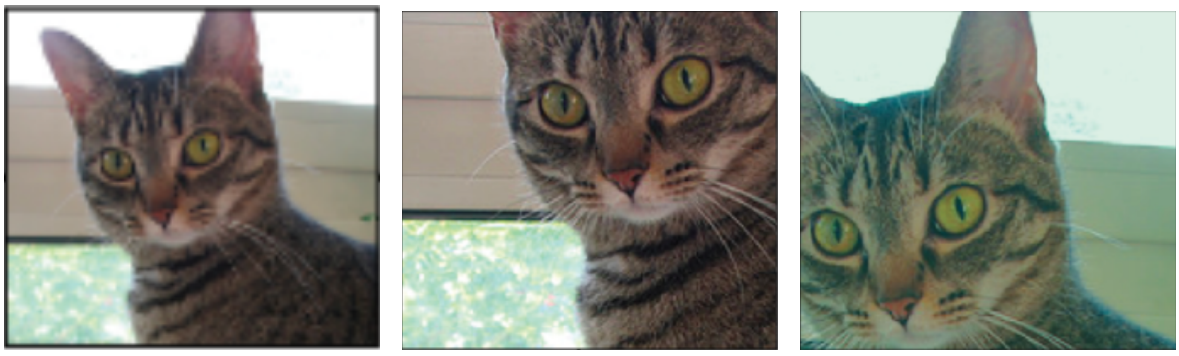


Figure 1. An image from ImageNet (Deng et al., 2009) and two augmented views of the same, as used in AMDIM (Bachman et al., 2019).

2.1.3. 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 for an example due to 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.

2.1.4. 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., n.d.; Brown et al., 2020; OpenAI, 2023). In many cases it has been desirable to only model a portion of this distribution, conditioning on some tokens and predicting others, to enable the use of bidirectional transformer models, as in (Devlin et al., 2019).

Outside of linguistic domains, a wider variety of options have been chosen. While some algorithms have directly used reconstruction (analogous to categorical prediction of tokens), such as PredNet (Lotter et al., 2016) and the Dreamer family (Hafner et al., 2020a; Hafner et al., 2020b; Hafner et al., 2023), others such as CPC and CPC|Action (Oord et al., 2018; Hénaff et al., 2019; Mazoure et al., 2020) combine contrastive objectives (see section 2.1.5) with temporal prediction. It is also possible to combine temporal prediction with the direct (non-constrastive) prediction of latent representations of states at other point in time, as in DeepMDP (Gelada et al., 2019), PBL (Guo et al., 2020) and SPR (Schwarzer et al., 2021a). This often involves either minimizing the mean squared error (DeepMDP and PBL) or maximizing the cosine similarity (SPR) between a prediction of the representation of a future state and its true representation.

2.1.5. 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 et al.,

2010); later, it was adapted by the deep learning community as a tool for representation learning.

2.1.5.1. 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; Chen et al., 2020b; He et al., 2020) 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

$$\mathcal{L}_{\text{InfoNCE}} = \mathbb{E}_X \left[-\log \frac{f_k(x_+, c)}{\sum_{x_j \in X} f_k(x_j, c)} \right] \tag{2.1.1}$$

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

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

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 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

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

using the *cosine similarity* between these projections, 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., 2020) augmented the contrastive framework used in previous works with the use of a polyak-averaged (Polyak et al., 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).

2.1.6. 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., 2020; 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 et al., 2017).

2.1.6.1. 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 et al., 2020b), 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, as in Tarvainen et al., 2017).

Mean Teacher (Tarvainen et al., 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., 2020), showing large boosts in performance 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

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

between the final output layer of the student and that of the teacher, prefiguring both these works 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.

2.1.7. Bootstrap Your Own Latent

Bootstrap Your Own Latent (BYOL, Grill et al., 2020) shows 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

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

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 et al., 2020b) 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.

2.2. 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 et al., 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 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 et al., 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

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

$v_\pi(s) \triangleq \mathbb{E}_\pi[G_t|s_t = s]$. 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: $V(s) = \mathbb{E}_{a\sim\pi(a|s)}[Q(s, a)]$.

2.2.1. 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 et al., 2018), converging at a rate governed by γ.

In these works, 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 et al., 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 $\delta_t \triangleq r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s, a)$.

2.2.2. 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 et al., 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.

2.2.3. 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 et al., 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 here, the classical approach is to minimize the following objective via gradient descent on the parameters of Q:

$$\mathcal{L}_Q(s_t, a_t, r_t, s_{t+1}) = (Q(s_t, a_t) - \mathbb{E}[r_t + \gamma \max_a Q_t(s_{t+1}, a)])^2 \tag{2.2.1}$$

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 et al., 2018).

In practice, this precise objective is not used; we refer the reader to Rainbow (Hessel et al., 2018) for an example of a modern Q-learning algorithm that combines a wide range of algorithmic improvements in search of better performance. Important differences that are relevant to the works presented here are that the loss used by Rainbow is actually a cross-entropy objective on categorical distributions, which captures the agent's uncertainty about its future returns (Bellemare et al., 2017b), that the value target is actually taken from s_{t+n} for some n (generally 3, 5 or 10) with all rewards from t to $t+n$ summed and discounted appropriately, and that numerous other stabilizing measures have been taken to reduce overestimation and improve sample efficiency (Van Hasselt et al., 2016b; Wang et al., 2016b).

Generally, these objectives are 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). Samples are typically drawn from the replay buffer proportionally to a measure of their *priority* (Schaul et al., 2016), to focus training on experiences where the agent has relatively more to learn.

2.2.4. 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:

$$\mathcal{L}_Q(s_t, a_t, r_t, s_{t+1}) = \left(Q(s_t, a_t) - r_t + \gamma Q_t(s_{t+1}, \pi_\theta(s_{t+1})]\right)^2. \tag{2.2.2}$$

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

$$\mathcal{L}_\pi(s) = -Q(s, \pi_\theta(s)) \tag{2.2.3}$$

Note that the parameters being optimized, θ, occur only in the policy network, so the only way this optimization can reduce its loss is by modifying the π_θ to produce actions deemed more favorable by the Q network.

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:

$$\mathcal{L}_Q(s_t, a_t, r_t, s_{t+1}) = \left(Q(s, a) - \mathbb{E}_{a_{t+1} \sim \pi(a|s_{t+1})}[r_t + \gamma Q_t(S_{t+1}, a_{t+1})]\right)^2. \tag{2.2.4}$$

2.2.5. 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., 2021b), or in using alternative techniques designed specifically for deep reinforcement learning (e.g., Schwarzer et al., 2021a; Gelada et al., 2019; Dabney et al., 2021; Guo et al., 2020). The most common approach used to combine DRL and representation learning is to jointly optimize both reinforcement learning and representation learning losses (as in Schwarzer et al., 2021a; Oord et al., 2018; Srinivas et al., 2020; Yarats et al., 2021b; 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., 2019) employ extra data without supervision by rewards, while others (e.g., Schwarzer et al., 2021a; Yarats et al., 2021b) 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 2.1 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.

2.2.6. 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 Hasselt et al., 2019).

2.2.7. DeepMind Control

(Tassa et al., 2020) 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., 2021; Hafner et al., 2020a; 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., 2021b).

2.2.8. Arcade Learning Environment

(Bellemare et al., 2013) introduced the Arcade Learning Environment (ALE), a challenging reinforcement learning task in which agents learn to play Atari 2600 games using visual inputs. This task (often referred to in the community as simply "Atari") is different from many of those traditionally studied in reinforcement learning in that nonlinear function approximation (e.g., neural networks) is critical to success. Atari 2600 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 Atari 2600 games in the ALE (Badia et al., 2020), but only when algorithms are given effectively infinite interaction time. In the limited-time regime, agent performance remains far below human-level performance on many tasks, especially when no pretraining or planning is used (compare results in Badia et al., 2020; Schwarzer et al., 2021a).

2.2.9. Evaluation in the ALE

Performance on the ALE is generally calculated as the human-normalized score, calculated separately on each game as $\frac{\text{agent\_score} - \text{random\_score}}{\text{human\_score} - \text{random\_score}}$, where agent_score refers to the game score achieved by the agent being trained, human_score refers to the score achieved by a human player given two hours of time to learn the game (as reported in Van Hasselt et al., 2016b) and random_score refers to the score achieved by a uniform random policy over all valid actions (generally also taken from Van Hasselt et al., 2016b). 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 the ALE 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., 2021), cannot be compared to that of methods that do.

Chapter 3

Pretraining Representations for Data-Efficient Reinforcement Learning

This chapter reproduces, with some formatting changes, the following work:

- **Max Schwarzer**, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Devon Hjelm, Philip Bachman & Aaron Courville. *Pretraining Representations for Data-Efficient Reinforcement Learning.* NeurIPS (2021).

This work advances the goals of this thesis by introducing self-supervised algorithms to pretrain representations for deep reinforcement learning agents from arbitrary offline data, without requiring action or reward supervision, and demonstrating how best to fine-tune networks pretrained in this fashion to achieve high performance on sample-efficient reinforcement learning tasks.

The accompanying supplemental materials for this work may be found in Appendix A. I led this project and performed the vast majority of experimental work, including all of the results in the main body of the paper.

3.1. Abstract

Data efficiency is a key challenge for deep reinforcement learning. We address this problem by using unlabeled data to pretrain an encoder which is then finetuned on a small amount of task-specific data. To encourage learning representations which capture diverse aspects of the underlying MDP, we employ a combination of latent dynamics modelling and unsupervised goal-conditioned RL. When limited to 100k steps of interaction on Atari games (equivalent to two hours of human experience), our approach

significantly surpasses prior work combining offline representation pretraining with task-specific finetuning, and compares favourably with other pretraining methods that require orders of magnitude more data. Our approach shows particular promise when combined with larger models as well as more diverse, task-aligned observational data – approaching human-level performance and data-efficiency on Atari in our best setting. We provide code associated with this work at https://github.com/mila-iqia/SGI.

3.2. Introduction

Deep reinforcement learning (RL) methods often focus on training networks *tabula rasa* from random initializations without using any prior knowledge about the environment. In contrast, humans rely a great deal on visual and dynamics priors about the world to perform decision making efficiently (Dubey et al., 2018; Lake et al., 2017). Thus, it is not surprising that RL algorithms which learn *tabula rasa* suffer from severe overfitting (Zhang et al., 2018) and poor sample efficiency compared to humans (Tsividis et al., 2017).

Self-supervised learning (SSL) provides a promising approach to learning useful priors from past data or experiences. SSL methods leverage unlabelled data to learn strong representations, which can be used to bootstrap learning on downstream tasks. Pretraining with self-supervised learning has been shown to be quite successful in vision (Hénaff et al., 2019; Grill et al., 2020; Chen et al., 2020a) and language (Devlin et al., 2019; Brown et al., 2020) settings.

Pretraining can also be used in an RL context to learn priors over representations or dynamics. One approach to pretraining for RL is to train agents to explore their environment in an unsupervised fashion, forcing them to learn useful skills and representations (Hansen et al., 2020; Liu et al., 2021b; Campos et al., 2021). Unfortunately, current unsupervised exploration methods require months or years of real-time experience, which may be impractical for real-world systems with limits and costs to interaction — agents cannot be run faster than real-time, may require significant human oversight for safety, and can be expensive to run in parallel. It is thus important to develop pretraining methods that work with practical quantities of data, and ideally that can be applied *offline* to fixed datasets collected from prior experiments or expert demonstrations (as in Stooke et al., 2020).

To this end, we propose to use a combination of self-supervised objectives for representation learning on offline data, requiring orders of magnitude less pretraining data than existing methods, while approaching human-level data-efficiency when finetuned on downstream tasks. We summarize our work below:
RL-aligned representation learning objectives: We propose to pretrain representations using a combination of latent dynamics modeling, unsupervised goal-conditioned reinforcement learning, and inverse dynamics modeling – with the intuition that a collection of such objectives can capture more information about the dynamical and temporal aspects of the environment of interest than any single objective. We refer to our method as **SGI** (**S**PR, **G**oal-conditioned RL and **I**nverse modeling).
Significant advances for data-efficiency on Atari: SGI's combination of objectives performs better than any in isolation and significantly improves performance over previous representation pretraining baselines such as ATC (Stooke et al., 2020). Our results are competitive with exploration-based approaches such as APT or VISR (Liu et al., 2021b; Hansen et al., 2020), which require two to three orders of magnitude more pretraining data and the ability to interact with the environment during training, while SGI can learn with a small offline dataset of exploration data.
Scaling with data quality and model size: SGI's performance also scales with data quality and quantity, increasing as data comes from better-performing or more-exploratory policies. Additionally, we find that SGI's performance scales robustly with model size; while larger models are unstable or bring limited benefits in standard RL, SGI pretraining allows their finetuning performance to significantly exceed that of smaller networks.

We assume familiarity with RL in the following sections.

3.3. Representation Learning Objectives

A wide range of SSL objectives have been proposed for RL which leverage various aspects of the structure available in agent interactions. For example, the temporal dynamics of an environment can be exploited to create a forward prediction task (e.g., Gelada et al., 2019; Guo et al., 2018; Stooke et al., 2020; Schwarzer et al., 2021a) in which an agent is trained to predict its immediate future observations, perhaps conditioned on a sequence of actions to perform.

Structure in RL goes far beyond forward dynamics, however. We propose to combine multiple representation learning objectives, covering different agent-centric and temporal

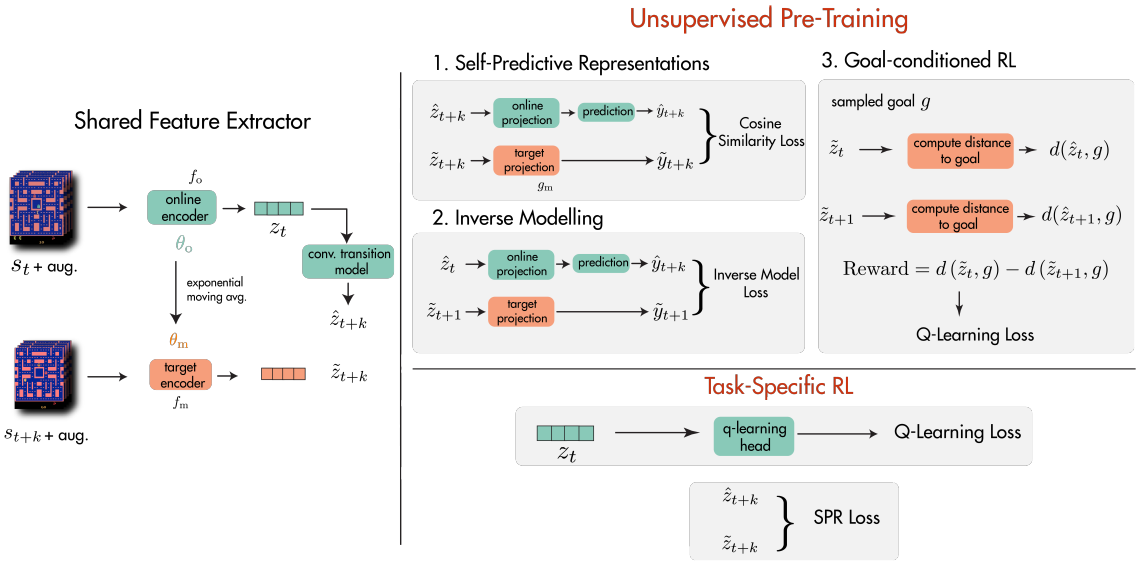


Figure 1. A schematic diagram showing our two stage pretrain-then-finetune method. All unsupervised training losses and task-specific RL use the shared torso on the left.

aspects of the MDP. Based on the intuition that knowledge of an environment is best represented in multiple ways (Hessel et al., 2021; Degris et al., 2012), we expect this to outperform monolithic representation learning methods such as temporal contrastive learning (e.g., Stooke et al., 2020). In deciding which tasks to use, we consider questions an adequate representation should be able to answer about its environment, including:

- If I take action a in state s, what state s' am I likely to see next?
- If I transitioned from state s to state s', what action a did I take?
- What action a would I take in state s so that I reach another state s' as soon as possible

Note that none of these questions are tied to task reward, allowing them to be answered in a fully-unsupervised fashion. Additionally, these are questions about the environment, and not any specific policy, allowing them to be used in offline pretraining with arbitrary behavioral policies.

In general, the first question may be answered by forward dynamics modeling, which as mentioned above is well-established in RL. The second question corresponds to inverse dynamics modeling, which has also been commonly used in the past (Lesort et al., 2018). The third question corresponds to self-supervised goal-conditioned reinforcement learning which has the advantage of being structurally similar to the downstream target task, as both require learning to control the environment.

To facilitate their joint use, we formulate these objectives so that they operate in the latent representation space provided by a shared encoder. We provide an overview of these components in Figure 1 and describe them in greater detail below; we also provide detailed pseudocode in the appendix.

3.3.1. Self-Predictive Representations

SPR (Schwarzer et al., 2021a) is a representation learning algorithm developed for data-efficient reinforcement learning. SPR learns a latent-space transition model, directly predicting representations of future states without reconstruction or negative samples. As in its base algorithm, Rainbow (Hessel et al., 2018), SPR learns a convolutional encoder, denoted as f_o, which produces representations of states as $z_t = f_o(s_t)$. SPR then uses a *dynamics model* h to recursively estimate the representations of future states, as $\hat{z}_{t+k+1} = h(\hat{z}_{t+k}, a_{t+k})$, beginning from $\hat{z}_t \triangleq z_t$. These representations are projected to a lower-dimensional space by a projection function p_o to produce $\hat{y}_{t+k} \triangleq p_o(\hat{z}_{t+k})$.

Simultaneously, SPR uses a *target encoder* f_m to produce target representations $\tilde{z}_{t+k} \triangleq f_m(s_{t+k})$, which are further projected by a target projection function p_m to produce $\tilde{y}_{t+k} \triangleq p_m(\tilde{z}_{t+k})$. SPR then maximizes the cosine similarity between these predictions and targets, using a learned linear prediction function q to translate from $\hat{y}$ to $\tilde{y}$:

$$\mathcal{L}_\theta^{\text{SPR}}(s_{t:t+K}, a_{t:t+K}) = -\sum_{k=1}^{K} \frac{q(\hat{y}_{t+k}) \cdot \tilde{y}_{t+k}}{||q(\hat{y}_{t+k})||_2 \cdot ||\tilde{y}_{t+k}||_2}. \tag{3.3.1}$$

The parameters of these target modules θ_m are defined as an exponential moving average of the parameters θ_o of f_o and p_o: $\theta_m = \tau\theta_m + (1-\tau)\theta_o$.

3.3.2. Goal-Conditioned Reinforcement Learning

Inspired by works such as Dabney et al. (2021) that show that modeling many different value functions is a useful representation learning objective, we propose to augment SPR with an unsupervised goal-conditioned reinforcement learning objective. We define goals g to be normalized vectors of the same size as the output of the agent's convolutional encoder (3,136- or 4,704-dimensional vectors, for the architectures we consider). We use these goals to annotate transitions with synthetic rewards, and train a modified version of Rainbow (Hessel et al., 2018) to estimate $Q(s_t, a, g)$, the expected

return from taking action a in state s_t to reach goal g if optimal actions are taken in subsequent states.

We select goals using a scheme inspired by hindsight experience replay (Andrychowicz et al., 2017), seeking to generate goal vectors that are both semantically meaningful and highly diverse. As in hindsight experience replay, we begin by sampling a state from another trajectory or the future of the current trajectory. However, we take the additional step of applying stochastic noise to encourage goals to lie somewhat off of the current representation manifold. We provide details in the appendix.

3.3.3. Inverse Dynamics Modeling

We propose to use an inverse dynamics modeling task (Lesort et al., 2018), in which the model is trained to predict a_t from s_t and s_{t+1}. Because this is a classification task (in discrete control) or regression task (continuous control), it is naturally not prone to representational collapse, which may complement and stabilize our other objectives. We directly integrate inverse modeling into the rollout structure of SPR, modeling $p(a_{t+k}|\hat{y}_{t+k}, \tilde{y}_{t+k+1})$ for each $k \in (0, \ldots, K-1)$, using a two-layer MLP trained by cross-entropy.

3.4. Related Work

3.4.1. Data-Efficiency

In order to address data efficiency in RL, Kaiser et al. (2019) introduced the Atari 100k benchmark, in which agents are limited to 100,000 steps of environment interaction, and proposed SimPLe, a model-based algorithm that substantially outperformed previous model-free methods. However, Hasselt et al. (2019) and Kielak (2020) found that simply modifying the hyperparameters of existing model-free algorithms allowed them to exceed SimPLe's performance. Later, DrQ (Kostrikov* et al., 2021) found that adding mild image augmentation to model-free methods dramatically enhanced their sample efficiency, while SPR (Schwarzer et al., 2021a) combined data augmentation with an auxiliary self-supervised learning objective. SGI employs SPR as one of its objectives in offline pretraining, leading to significant improvements in data-efficiency.

3.4.2. Exploratory pretraining

A number of recent works have sought to improve reinforcement learning via the addition of an unsupervised *pretraining* stage prior to finetuning on the target task. One common approach has been to allow the agent a period of fully-unsupervised interaction with the environment, during which the agent is trained to maximize a surrogate exploration-based task such as the diversity of the states it encounters, as in APT (Liu et al., 2021b) or ProtoRL (Yarats et al., 2021a), or to learn a set of skills associated with different paths through the environment, as in DIAYN (Eysenbach et al., 2018), VISR (Hansen et al., 2020), and DADS (Sharma et al., 2019). Others have proposed to use self-supervised objectives to generate intrinsic rewards encouraging agents to visit new states; e.g. Pathak et al. (2017) and Burda et al. (2018) use the loss of an inverse dynamics model like that used in SGI, while Sekar et al. (2020) uses the disagreement between an ensemble of latent-space dynamics models. Finally, Campos et al. (2021) report strong results based on massive-scale unsupervised pretraining.

Many of these methods are used to pretrain agents that are later adapted to specific reinforcement learning tasks. However, SGI differs in that it can be used offline and is agnostic to how data is collected. As such, if no pre-existing offline data is available, one of the methods above can be used to generate a dataset for SGI; we use Burda et al. (2018) for this in Section 3.5.2.

3.4.3. Visual Representation Learning

Computer vision has seen a series of dramatic advances in self-supervised representation learning, including contrastive methods (Oord et al., 2018; Hjelm et al., 2019; Bachman et al., 2019; He et al., 2020; Chen et al., 2020a) as well as purely predictive ones (Grill et al., 2020). Variants of these approaches have also been shown to improve performance when coupled with a small quantity of labeled data, in a *semi-supervised* setting (Chen et al., 2020b; Hénaff et al., 2019), and several self-supervised methods have been designed specifically for this case (for example, Sohn et al., 2020; Tarvainen et al., 2017).

These advances have spurred similar growth in methods aimed specifically at improving performance in RL. We refer the reader to Lesort et al. (2018) for a review of earlier methods, including inverse dynamics modeling which is used in SGI. Recent research has focused on leveraging latent-space dynamics modeling as an auxiliary

task. Gelada et al. (2019) propose a simple next-step prediction task, coupled with reward prediction, but found it is prone to latent space collapse and requires an auxiliary reconstruction loss for experiments on Atari. Guo et al. (2020) use a pair of networks for both forward and backward prediction, and show improved performance in extremely large-data multi-task settings. Mazoure et al. (2020) use a temporal contrastive objective for representation learning, and show improvement in continual RL settings. Concurrently, SPR (Schwarzer et al., 2021a) proposed a multi-step latent prediction task with similarities to BYOL (Grill et al., 2020).

Two works similar to ours, Anand et al. (2019) and Stooke et al. (2020), propose reward-free temporal-contrastive methods to pretrain representations. Anand et al. (2019) show that representations from encoders trained with ST-DIM contain a great deal of information about environment states, but they do not examine whether or not representations learned via their method are, in fact, useful for reinforcement learning. However, Stooke et al. (2020) employ a similar algorithm and find only relatively minor improvements in performance compared to standard baselines in the large-data regime; our controlled comparisons show that SGI's representations are far better for data-efficiency. Concurrent to our work, FERM (Zhan et al., 2020) propose contrastive pretraining from human demonstrations in a robotics setting. As FERM is quite similar to ATC, we are optimistic that our improvements over ATC in Atari 100k would translate to FERM's setting. Finally, Yang et al. (2021) propose a family of algorithms for learning representations for DRL from state observations; we believe that these might be promising in DRL from pixels, but they have not yet been adapted to that setting.

3.5. Experimental Details

In our experiments, We seek to address two main challenges for the deployment of RL agents in the real world (Dulac-Arnold et al., 2020): (1) training the RL agent with a limited budget of interactions in the real environment, and (2) leveraging existing interaction data of **arbitrary quality**.

3.5.1. Environment and Evaluation

To address the first challenge, we focus our experimentation on the Atari 100k benchmark introduced by Kaiser et al. (2019), in which agents are allowed only 100k

steps of interaction with their environment.[1] This is roughly equivalent to the two hours human testers were given to learn these games by Mnih et al. (2015), providing a baseline of human sample-efficiency.

Atari is also an ideal setting due to its complex observational spaces and diverse tasks, with 26 different games included in the Atari 100k benchmark. These factors have led to Atari's extensive use for representation learning and exploratory pretraining (Anand et al., 2019; Stooke et al., 2020; Campos et al., 2021), and specifically Atari 100k for data-efficient RL (e.g., Kaiser et al., 2019; Kostrikov* et al., 2021; Schwarzer et al., 2021a), including finetuning after exploratory pretraining (e.g., Hansen et al., 2020; Liu et al., 2021b), providing strong baselines to compare to.

Our evaluation metric for an agent on a game is *human-normalized score* (HNS), defined as $\frac{agent\_score - random\_score}{human\_score - random\_score}$. We calculate this per game by averaging scores over 100 evaluation trajectories at the end of training, and across 10 random seeds for training. We report both mean (Mn) and median (Mdn) HNS over the 26 Atari-100K games, as well as on how many games a method achieves super-human performance (>H) and greater than random performance (>0). Following the guidelines of Agarwal et al. (2021b) we also report interquartile mean HNS (IQM) and quantify uncertainty via bootstrapping in the appendix.

3.5.2. Pretraining Data

The second challenge pertains to pretraining data. Although some prior work on offline representational pretraining has focused on expert-quality data (Stooke et al., 2020), we expect real-world pretraining data to be of greatly varying quality. We thus construct four different pretraining datasets to approximate different data quality scenarios.

- **(R)andom** To assess performance near the lower limit of data quality, we use a random policy to gather a dataset of 6M transitions for each game. To encourage the agent to venture further from the starting state, we execute each action for a random number of steps sampled from a Geometric($\frac{1}{3}$) distribution.
- **(E)xploratory** To emulate slightly better data that covers a larger range of the state space, we use an exploratory policy. Specifically, we employ the **IDF** (inverse dynamics) variant of the algorithm proposed by (Burda et al., 2018). We log the

[1] Note that sticky actions are disabled under this benchmark.

first 6M steps from an agent trained in each game. This algorithm achieves better-than-random performance on only 70% of tasks, setting it far below the performance of more modern unsupervised exploration methods.

To create higher-quality datasets, we follow Stooke et al. (2020) and use experience gathered during the training of standard DQN agents (Mnih et al., 2015). We opt to use the publicly-available DQN Replay dataset (Agarwal et al., 2020), which contains data from training for 50M steps (across all 57 games, with five different random seeds). Although we might prefer to use data from recent unsupervised exploration methods such as APT (Liu et al., 2021b), VISR (Hansen et al., 2020), or CPT (Campos et al., 2021), none of these works provide code or datasets, making this impractical. We address using data collected from on-task agents with a behavioural cloning baseline in Section 3.6.2, with surprising findings relative to prior work.

- **(W)eak** We first generate a weak dataset by selecting the first 1M steps for each of the 5 available runs in the DQN Replay dataset. This data is generated with an ϵ-greedy policy with high, gradually decaying ϵ, leading to substantial action diversity and many suboptimal exploratory actions. Although the behavioral policies used to generate this agent are not especially competent (see Table 1), they have above-random performance on almost all games, suggesting that that this dataset includes more task-relevant transitions.
- **(M)ixed** Finally, for a realistic best-case scenario, we create a dataset of both medium and low-quality data. To simulate a real-world collection of data from different policies, we concatenate multiple checkpoints evenly spread throughout training of a DQN. We believe this is also a reasonable approximation for data from a modern unsupervised exploration method such as CPT (Campos et al., 2021); as shown in Table 1, the agent for this dataset has performance in between CPT and VISR, with median closer to CPT and mean closer to VISR. This data is also lower quality than the expert data originally used in the method most similar to ours, ATC (Stooke et al., 2020). [2] We create a dataset of 3M steps and a larger dataset of 6M steps; all **M** experiments use the 3M step dataset unless otherwise noted.

We compare the agents used for our datasets to those for unsupervised exploration pretraining baselines in Table 1. We estimate the performance of the Weak and Mixed agents as the average of the corresponding logged evaluations in the Dopamine (Castro

[2] Our data-collection agents are weaker than those used by ATC on seven of the eight games they consider.

Tableau 1. Performance of agents used in pretraining data collection compared to external baselines on 26 Atari games (Kaiser et al., 2019)

Method	Mdn	Mean	>H	>0	Data
Exploratory Pretraining Baselines					
VISR@0	0.056	0.817	5	19	250M
APT@0[1]	0.038	0.476	2	18	250M
CPT@0	**0.809**	**4.945**	**12**	25	4B
Offline Datasets					
Exploratory	0.039	0.042	0	18	6M
Weak[2]	0.028	0.056	0	23	5M
Mixed[2]	0.618	1.266	10	**26**	3M

[1] Calculated from ICLR 2021 OpenReview submission; unreported in arXiv version.

[2] Upper-bound estimate from averaging evaluation performance of corresponding agents in Dopamine.

et al., 2018) baselines. Even our largest dataset is quite small compared to the amounts of data gathered by unsupervised exploration methods (see the "Data" column in Table 1); this is intentional, as we believe that unsupervised interactional data may be expensive in real world applications. We show the performance of the non-random data collection policies in Table 2 (note that a fully-random policy has a score of **0** by definition).

3.5.3. Training Details

We optimize our three representation learning objectives jointly during unsupervised pretraining, summing their losses. During finetuning, we optimize only the reinforcement learning and forward dynamics losses, following Schwarzer et al. (2021a) (see Section 3.6.7), and lower the learning rates for the pretrained encoder and dynamics model by two orders of magnitude (see Section 3.6.6).

We consider the standard three-layer convolutional encoder introduced by Mnih et al. (2015), a ResNet inspired by Espeholt et al. (2018), as well as an enlarged ResNet of the same design. In other respects, our implementation matches that of SPR and is based on its publicly-released code. Full implementation and hyperparameter details are provided in the appendix. We refer to agents by the model architecture and pretraining dataset type used: **SGI-R** is pretrained on Random, **SGI-E** on Exploratory, **SGI-W** on Weak, and **SGI-M** on Mixed. To investigate scaling, we vary the size of the encoder used in **SGI-M**: the standard Mnih et al. (2015) encoder is **SGI-M/S** (for small), our standard ResNet is simply **SGI-M** and using a larger ResNet is **SGI-M/L** (for large)[3]. For **SGI-M/L** we also use the 6M step dataset described earlier. All ablations are conducted in comparison to **SGI-M** unless otherwise noted. Finally, agents without pretraining are denoted **SGI-None**; SGI-None/S would be roughly equivalent to SPR (Schwarzer et al., 2021a).

For baselines, we compare to no-pretraining Atari 100k methods (Kaiser et al., 2019; Hasselt et al., 2019; Kostrikov* et al., 2021; Schwarzer et al., 2021a). For our models trained on Random and Exploratory data we compare against previous pretraining-via-exploration approaches applied to Atari 100k (Liu et al., 2021b; Hansen et al., 2020; Campos et al., 2021). In the higher quality data regime, we compare to recent work on data-agnostic unsupervised pretraining, ATC (Stooke et al., 2020), as well as behavioural cloning (BC).

3.6. Results and Discussion

We find that SGI performs competitively on the Atari-100K benchmark; presenting aggregate results in Table 2, and full per-game data in the appendix. Our best setting, **SGI-M/L**, achieves a median HNS of 0.753, approaching human-level sample-efficiency and outperforming all comparable methods except the recently proposed CPT (Campos et al., 2021). With less data and a smaller model, **SGI-M** achieves a median HNS of 0.679, significantly outperforming the prior method ATC on the same data (**ATC-M**). Meanwhile, **SGI-E** achieves a median HNS of 0.456, matching or exceeding other exploratory methods such as APT (Liu et al., 2021b) and VISR (Hansen et al., 2020), as well as ATC-E.

[3]See the appendix for details on these networks

Tableau 2. HNS on Atari100k for SGI and baselines.

Method	Mdn	Mn	>H	>0	Data
No Pretraining (Finetuning Only)					
SimPLe	0.144	0.443	2	**26**	0
DER	0.161	0.285	2	**26**	0
DrQ	0.268	0.357	2	24	0
SPR	0.415	0.704	7	**26**	0
SGI-None	0.343	0.565	3	26	0
Exploratory Pretraining + Finetuning					
Method	Mdn	Mn	>H	>0	Data
VISR	0.095	1.281	7	21	250M
APT	0.475	0.666[1]	7	**26**	250M
CPT@0[2]	**0.809**	**4.945**	12	25	4000M
ATC-R[3]	0.191	0.472	4	**26**	6M
ATC-E[3]	0.237	0.462	3	**26**	6M
SGI-R	0.326	0.888	5	**26**	6M
SGI-E	0.456	0.838	6	**26**	6M
Offline-data Pretraining + Finetuning					
Method	Mdn	Mn	>H	>0	Data
ATC-W[3]	0.219	0.587	4	**26**	3M
ATC-M[3]	0.204	0.780	5	**26**	3M
BC-M@0	0.139	0.227	0	23	3M
BC-M	0.548	0.858	8	**26**	3M
SGI-W	0.589	1.144	8	**26**	5M
SGI-M/S	0.423	0.914	8	**26**	3M
SGI-M	0.679	1.149	**9**	**26**	3M
SGI-M/L	**0.753**	**1.598**	**9**	**26**	6M

[1] APT claims 0.6955, but we calculate 0.666 from their reported per-game scores.
[2] CPT@0 does not do any finetuning.
[3] Our implementation (see appendix)

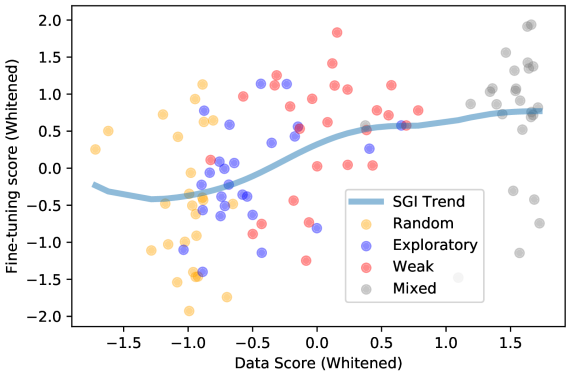


Figure 2. SGI finetuning performance vs. pretraining data score for all combinations of game and dataset. Data score is estimated as clipped return per episode, trend calculated via kernel regression. Values whitened per-game for clarity.

3.6.1. Pretraining data efficiency

SGI achieves strong performance with only limited pretraining data; our largest dataset contains 6M transitions, or roughly 4.5 days of experience. This compares favourably to recent works on unsupervised exploration such as APT or CPT, which require far larger amounts of data and environment interaction (250M steps or 193 days for APT, 4B steps or 8.45 years for CPT). We expect SGI would perform even better if used in these large-data settingss, as we find that it scales robustly with data (see Section 3.6.4).

3.6.2. Behavioural cloning is a strong baseline

Although ATC pretrains with expert data, they did not investigate behavioral cloning as a baseline pretraining objective. We do so on our **Mixed** dataset, the only one to be generated by policies with significantly above-random performance. Behavioral cloning without finetuning (**BC-M@0**) performs poorly, perhaps due to the varying behavioural quality in the dataset. But when finetuned, **BC-M** yields very respectable performance, surpassing **ATC-M** but not **SGI-M**. All fine-tuning settings for BC-M match SGI-M.

3.6.3. Data quality matters

In principle, SGI can be used with any offline dataset but we demonstrate that it scales with the quality of its data. Near the lower bound of data quality where all actions

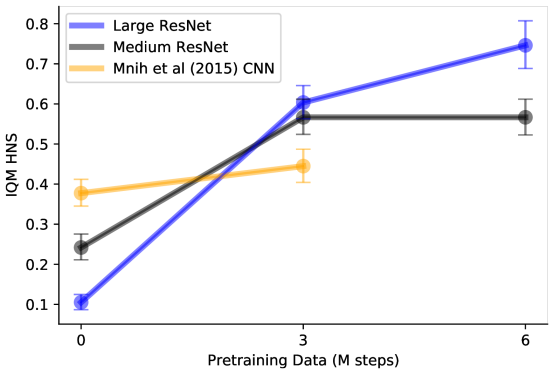


Figure 3. Finetuning performance of SGI for different CNN sizes and amounts of pretrained data from the **Mixed** dataset.

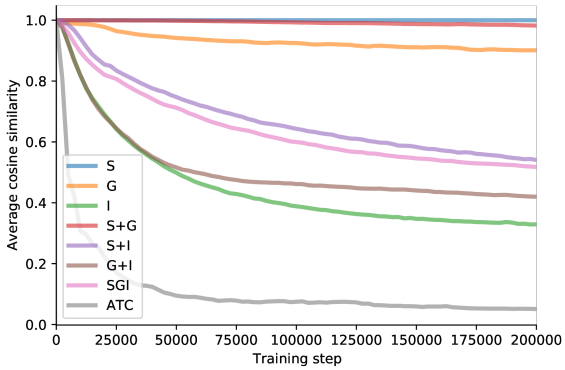


Figure 4. Average cosine similarity between representations over pretraining, averaged across the 26 Atari 100k games. 1 indicates representations are identical, 0 perfect dissimilarity.

are selected randomly, **SGI-R** still provides some benefit over an otherwise-identical randomly-initialized agent (**SGI-None**) on 16 out of 26 games, with a similar median but 57% higher mean HNS. With better data from an exploratory policy, **SGI-E** improves on 16/26 games, gets 33% higher median HNS, and surpasses APT (Liu et al., 2021b) which used 40 times more pretraining data. With similarly weak data but possibly more task-specific transitions, **SGI-W** gets 72% higher median HNS compared to SGI-None and with realistic data from a mixture of policies, **SGI-M** improves to 98%.

Importantly, the pattern we observe is very different from what would be expected for imitation learning. In particular, SGI-W's strong performance shows that expert data is not required. To characterize this, we plot the average clipped reward [4] experienced per episode for each of our pretraining datasets in Figure 2. Normalizing across tasks, we find a strong positive correlation between task reward engagement ($p < 0.0001$) and finetuning performance. Moreover, we find diminishing returns to further task engagement.

3.6.4. Pretraining unlocks the value of larger networks

The three-layer network introduced by Mnih et al. (2015) has become a fixture of deep reinforcement learning, and has been employed by previous works examining pretraining in this field (e.g. Liu et al., 2021b; Stooke et al., 2020). However, we

[4] Unclipped rewards are not available for the offline DQN dataset.

Tableau 3. HNS on Atari 100K for pretraining ablations of SGI.

Pretraining	Mdn	Mean	>H
None	0.343	0.565	3
S	0.009	-0.054	0
G	0.060	0.181	1
I	0.411	0.943	7
S+G	0.029	0.098	0
G+I	0.512	1.004	**9**
S+I	0.629	0.978	8
SGI-M	**0.679**	**1.149**	**9**

find that representational pretraining with this network (**SGI-M/S**) provides only minor benefits compared to training from scratch. In contrast, larger networks struggle without pretraining but shine when pretrained as shown in Figure 3.

This finding is consistent with recent work in unsupervised representation learning for classification, which has observed that unsupervised pretraining benefits disproportionately from larger networks (Chen et al., 2020a). In particular, our results suggest that model size should increase in parallel with amount of pretraining data, matching recent work on scaling in language modeling (Kaplan et al., 2020; Hernandez et al., 2021). SGI thus provides a simple way to use unlabeled data to extend the benefits of large networks, already well-known in the large-data regime (e.g., Schrittwieser et al., 2020; Espeholt et al., 2018), to data-efficient RL.

3.6.5. Combining SGI's objectives improves performance

We test all possible combinations of our three SSL objectives, denoted by combinations of the letters S, G and I to indicate which objectives they employ. Results in Table 3 show that performance monotonically increases as more objectives are used, with inverse dynamics modeling combined with either of the other objectives performing respectably well. This illustrates the importance of using multiple objectives to obtain representational coverage of the MDP.

We note that including inverse modeling appears to be critical, and hypothesize that this is related to representational collapse. To measure this, we plot the average cosine

Tableau 4. HNS on Atari 100K for fine-tuning schemes for SGI.

Method	Mdn	Mean	>H
No pretrain	0.343	0.565	3
Naive	0.429	0.845	8
Frozen	0.499	0.971	8
Reduced LRs	**0.679**	**1.149**	**9**

similarity between representations y_t of different states for several pretraining methods in Figure 4, using our ResNet encoder on the Mixed dataset. We observe that **S**, **G** and **S+G** all show some degree representational collapse, while configurations that include inverse dynamics modeling avoid representational collapse, as does ATC, whose contrastive loss implicitly optimizes for representational diversity (Wang et al., 2020b). Intriguingly, we observe that increased representational diversity does not necessarily improve performance. For example, SGI outperforms **ATC**, **G+I** and **I** in finetuning but has less diverse pretraining representations. We also observe that adding SPR (**S**) consistently pulls representations towards collapse (compare **S+I** and **I**, **S+G** and **G**, and **SGI** and **G+I**); how this relates to performance is a question for future work.

3.6.6. Naively finetuning ruins pretrained representations

We find that properly finetuning pretrained representations is critical, as results in Table 4 show. Although allowing pretrained weights to change freely during finetuning is better than initializing from scratch (**Naive** vs **No Pretrain**), freezing the pretrained encoder (**Frozen**) leads to better performance than either. SGI's approach of reducing finetuning learning rates for pretrained parameters leads to superior performance (**Reduced LRs**, equivalent to **SGI-M**).

We thus hypothesize that representations learned by SGI are being disrupted by gradients early in finetuning, in a phenomenon analogous to catastrophic forgetting (Zenke et al., 2017; Hadsell et al., 2020). As representations may not generalize between different value functions across training (Dabney et al., 2021), allowing the encoder to strongly adapt early in training could make it *worse* at modeling later value functions, compared to the neutral initialization from SGI. We also note that there is a long history in computer vision of employing specialized finetuning hyperparameters (Li et al., 2020; Chu et al., 2016) when transferring tasks.

Tableau 5. HNS on Atari 100K for finetuning ablations of SGI.

Finetune SSL	Mdn	Mean	>H
Without SGI pretraining			
None	0.161	0.315	2
S Only	0.343	0.565	3
With SGI pretraining			
None	0.452	1.114	8
SGI	0.397	1.011	8
S Only	**0.679**	**1.149**	**9**

3.6.7. Not all SSL objectives are beneficial during finetuning

Although SGI uses **S** during finetuning, we experiment with a variant that optimizes only the standard DQN objective, roughly equivalent to using DrQ (Kostrikov* et al., 2021) with DQN hyperparameters set to match SGI. We find that finetuning with **S** has a large impact with or without pretraining (compare **None** and **S Only** entries in Table 5.). Although, SGI without **S** manages to achieve roughly the same mean human-normalized score as SGI with **S**, removing **S** harms performance on performance on 19 out of 26 games and reduces median normalized score by roughly 33%. We also find no benefit to using all of SGI's constituent objectives during finetuning (**All Losses** in Table 5) compared to using **S** alone, although the gap between them is not statistically significant for metrics other than median.

3.7. Conclusion

We present SGI, a fully self-supervised (reward-free) approach to representation learning for reinforcement learning, which uses a combination of pretraining objectives to encourage the agent to learn multiple aspects of environment dynamics. We demonstrate that SGI enables significant improvements on the Atari 100k data-efficiency benchmark, especially in comparison to unsupervised exploration approaches which require orders of magnitude more pretraining data. Investigating the various components of SGI, we find that it scales robustly with higher-quality pretraining data and larger models, that

all three of the self-supervised objectives contribute to the success of this approach, and that careful reduction of fine-tuning learning rates is critical to optimal performance.

3.8. Acknowledgements

We would like to thank Mila and Compute Canada for computational resources. We would like to thank Rishabh Agarwal for useful discussions. Aaron Courville would like to acknowledge financial support from Microsoft Research and Hitachi. We would also like to thank Wancong (Kevin) Zhang for his helpful comments to the codebase on GitHub.

Chapter 4

The Primacy Bias in Deep Reinforcement Learning

This chapter reproduces, with some formatting changes, the following work:

- Evgenii Nikishin*, **Max Schwarzer***, Pierluca D'Oro*, Pierre-Luc Bacon & Aaron Courville. *The Primacy Bias in Deep Reinforcement Learning.* ICML (2022).

This work advances the goals of this thesis by finding and resolving a key issue hindering the sample-efficiency of deep reinforcement learning: the primacy bias. It demonstrates that reinforcement learning algorithms may overfit to early experiences, and shows that a family of reset-based solutions can mitigate this and lead to large performance improvements.

The accompanying supplemental materials for this work may be found in Appendix B. I co-led this project and performed all Atari experiments, or roughly half of the overall results in the paper.

4.1. Abstract

This work identifies a common flaw of deep reinforcement learning (RL) algorithms: a tendency to rely on early interactions and ignore useful evidence encountered later. Because of training on progressively growing datasets, deep RL agents incur a risk of overfitting to earlier experiences, negatively affecting the rest of the learning process. Inspired by cognitive science, we refer to this effect as *the primacy bias*. Through a series of experiments, we dissect the algorithmic aspects of deep RL that exacerbate

this bias. We then propose a simple yet generally-applicable mechanism that tackles the primacy bias by periodically resetting a part of the agent. We apply this mechanism to algorithms in both discrete (Atari 100k) and continuous action (DeepMind Control Suite) domains, consistently improving their performance.

4.2. Introduction

Bob is learning a difficult passage on a guitar to rehearse it with his band. After long nights of practice, he is able to play it but with straining finger positions and unclean sound. Alice, another fellow guitarist, shows him a less fatiguing and nicer-sounding way to play the passage which Bob is theoretically able to understand and execute well. Nonetheless, in subsequent rehearsal sessions, Bob's unconscious mind automatically resorts to the first bad-sounding solution, discouraging him and his bandmates from trying more technically challenging pieces of music. Bob is experiencing an instance of the *primacy bias*, a cognitive bias demonstrated by studies of human learning (Marshall et al., 1972; Shteingart et al., 2013).

The outcomes of first experiences can have long-lasting effects on subsequent learning and behavior. Thanks to Alice, Bob has already collected new data sufficient for improving his performance on that passage and guitar playing in general; however, because of the priming provided by his long training nights with a bad technique, he is unable to leverage his new experience. The primacy bias generates a vicious circle: since Bob cannot improve his guitar skills in the face of new data, he will not be able to collect more interesting data by playing more challenging guitar passages, crippling his overall learning.

The central finding of this paper is that deep reinforcement learning (RL) algorithms are susceptible to a similar bias. We refer to the primacy bias in deep RL as a tendency to overfit early interactions with the environment preventing the agent from improving its behavior on subsequent experiences. The consequences of this phenomenon compound: an agent with poor performance will collect data of poor quality and the poor data will amplify the difficulty of recovering from an overfitted solution.

Standard components of deep RL algorithms magnify the effect of the primacy bias. For instance, experience replay (Mnih et al., 2015) allows efficient data reuse but exposes the agent to its initial samples more than recent ones. In the interest of sample efficiency, deep RL algorithms often additionally use a high replay ratio (Hasselt et al.,

2019; Fedus et al., 2020) updating the agent more times on the same data. Such design choices can improve the agent's performance but come with a risk of exacerbating the effects of early interactions.

How can a deep RL algorithm avoid the primacy bias? Coming back to Bob, he could re-establish his learning progression by simply forgetting his bad practices and directly learning from newer experience. Similarly, deep RL agents affected by the primacy bias can forget parts of a solution which was derived by overfitting to early experiences before continuing the learning process.

As a remedy for the primacy bias, we propose a *resetting* mechanism allowing the agent to forget a part of its knowledge. Our strategy is simple and compatible with any deep RL algorithm equipped with a replay buffer: we periodically re-initialize the last layers of an agent's neural networks, while maintaining the experience within the buffer.

Despite its simplicity, this resetting mechanism consistently improves performance of agents on benchmarks including the discrete-action ALE (Bellemare et al., 2013) and the continuous-action DeepMind Control Suite (Tassa et al., 2020). Our strategy imposes no additional computational costs and requires only two implementation choices: which parts of the neural networks to reset and how often to reset them. We also show that resetting enables training regimes with higher replay ratio and longer n-step targets (Sutton et al., 2018), where an agent without resets would be overfitting otherwise.

To summarize the contributions of this paper, we:

(1) Provide demonstrations of the existence of the primacy bias in deep RL, a tendency of an agent to harm its future decision making by overfitting to early data and ignoring subsequent interactions;
(2) Expose plausible causes of this phenomenon and show how algorithmic aspects in modern deep RL amplify its consequences;
(3) Propose a mechanism for alleviating the primacy bias by periodically resetting a part of the agent;
(4) Empirically demonstrate both qualitative and quantitative improvements in performance when applying resets to strong baseline algorithms.

4.3. Preliminaries

We adopt the standard formulation of reinforcement learning (Sutton et al., 2018) under the Markov decision process (MDP) formalism where the agent observes a state s from a space $\mathcal{S}$, chooses an action a from a space $\mathcal{A}$, and receives a reward r according to a mapping $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The environment then transitions into a state s' according to a distribution $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ and the interaction continues. The initial state s_0 is sampled from a distribution $\rho \in \Delta(\mathcal{S})$. The goal of the agent is to learn a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the expected discounted sum of rewards $\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$ with $\gamma \in [0, 1)$.

RL methods typically learn an action-value function

$$Q_\pi(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a \right],$$

through *temporal-difference* (TD) learning (Sutton, 1988) by minimizing the difference between $Q_\pi(s,a)$ and $\mathbb{E}_{p(s'|s,a),\pi(a'|s')} \left[r(s,a) + \gamma Q_\pi(s',a')\right]$. Many RL algorithms store past experiences in a *replay buffer* (Mnih et al., 2015) that increases sample efficiency by leveraging a single data point more than once. The number of resampling times of given data is controlled by the *replay ratio* (Hasselt et al., 2019; D'Oro et al., 2020) which plays a critical role in the algorithm's performance. Set too low, the agent would underuse the data it has and become sample-inefficient; set too high, the agent would overfit the existing data.

The core idea behind temporal-difference methods is *bootstrapping*, learning from agent's own value estimates without the need to wait until the end of the interaction. TD learning can be generalized by using *n-step* targets $\mathbb{E}_\pi \left[r(s_t,a_t) + \gamma r(s_{t+1}, a_{t+1}) + \cdots + \gamma^n Q_\pi(s_{t+n},a_{t+n})\right]$. Here, n controls a trade-off between the (statistical) bias of Q_π estimates and the variance of the sum of future rewards.

In the rest of the paper, we consider deep RL algorithms where Q_π and π (when needed) are modelled by neural network function approximators.

4.4. The Primacy Bias

The main goal of this paper is to understand how the learning process of deep reinforcement learning agents can be disproportionately impacted by initial phases of training due to an effect called the primacy bias.

The Primacy Bias in Deep RL: *a tendency to overfit early experiences that damages the rest of the learning process.*

This definition is wide-ranging: the primacy bias has multiple roots and leads to multiple negative effects for the training of an RL agent, but they are all connected to improper learning from early data.

The rest of this section presents two experiments, with the goal of demonstrating the existence and the dynamics of the phenomenon in isolation. First, we show that excessive training of an agent on the very first interactions can fatally damage the rest of the learning process. The second experiment demonstrates that data collected by an agent impacted by the primacy bias is adequate for learning, although the agent cannot leverage it due to its accumulated overfitting.

4.4.1. Heavy Priming Causes Unrecoverable Overfitting

The degree of reliance of an agent on early data is a crucial factor in determining how much any primacy effect is going to affect the learning process. At the same time, in the interest of sample efficiency, it is vital to leverage initial experiences at their full potential to expedite the training. This trade-off is particularly evident for algorithms with a replay buffer, which can be used to update the agent several times before interacting further with the environment.

To uncover in an explicit way the effect of the primacy bias in RL, we probe excessive reliance to early data to its extreme: could *overfitting on a single batch of early data be enough to entirely disrupt an agent's learning process*?

To investigate this question, we train Soft Actor-Critic (Haarnoja et al., 2018) on the `quadruped-run` environment from DeepMind Control suite (DMC) (Tassa et al., 2020). We use default hyperparameters, which imply a single update for both policy and value function per step in the environment. Then, we train an identical version of the algorithm in an experimental condition that we refer to as *heavy priming*: after

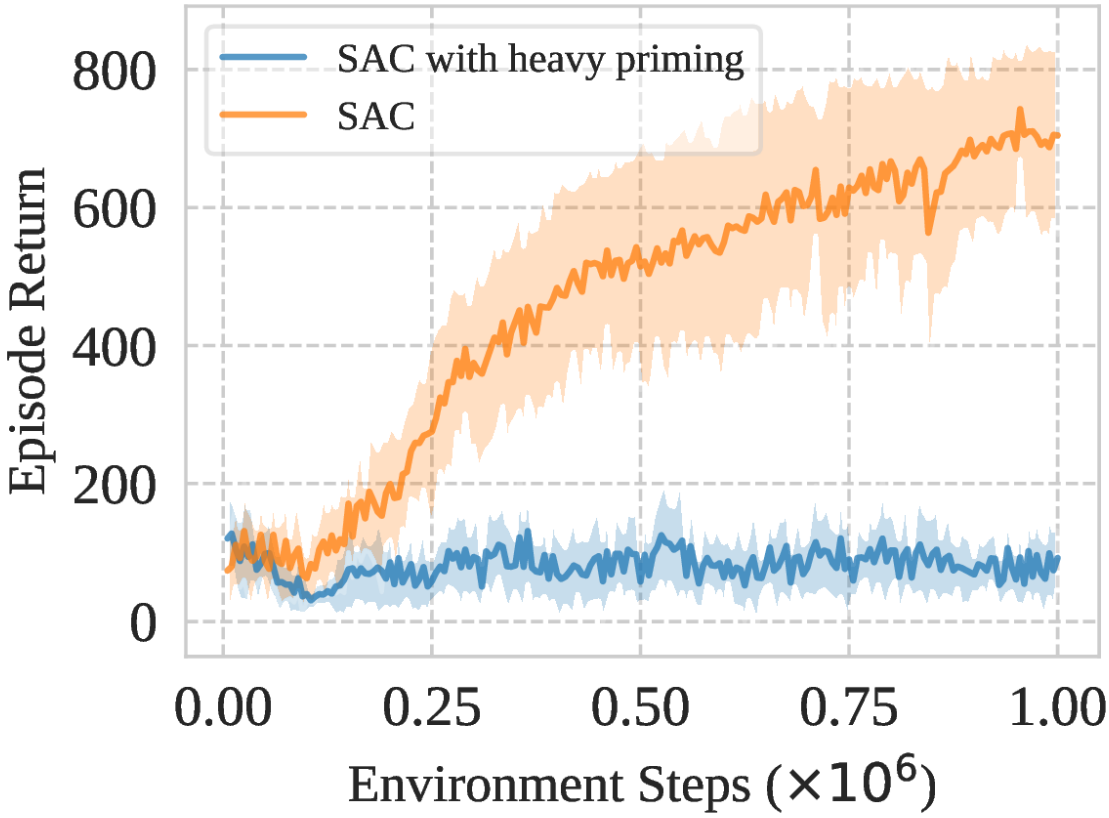


Figure 1. Undiscounted returns on `quadruped-run` for SAC with and without *heavy priming* on the first 100 transitions. An agent extremely affected by the primacy bias is unable to learn even after collecting hundreds of thousands of new transitions. Mean and std are estimated over 10 runs.

collecting 100 data points, we update the agent 10^5 times using the resulting replay buffer, before resuming standard training. Figure 1 shows that even after collecting and training on almost one million new transitions, the agent with heavy priming is unable to recover from the initial overfitting.

This experiment conveys a simple message: overfitting to early experiences might inexorably damage the rest of the learning process. Even if no practical implementation would use such a large number of training steps on such a limited dataset, we will see in Section 4.6 that even a relatively small number of updates per step can cause similar issues. The finding suggests that the primacy bias has compounding effects: an overfitted agent gathers worse data that itself leads to less efficient learning that further damages the ability to learn and so on.

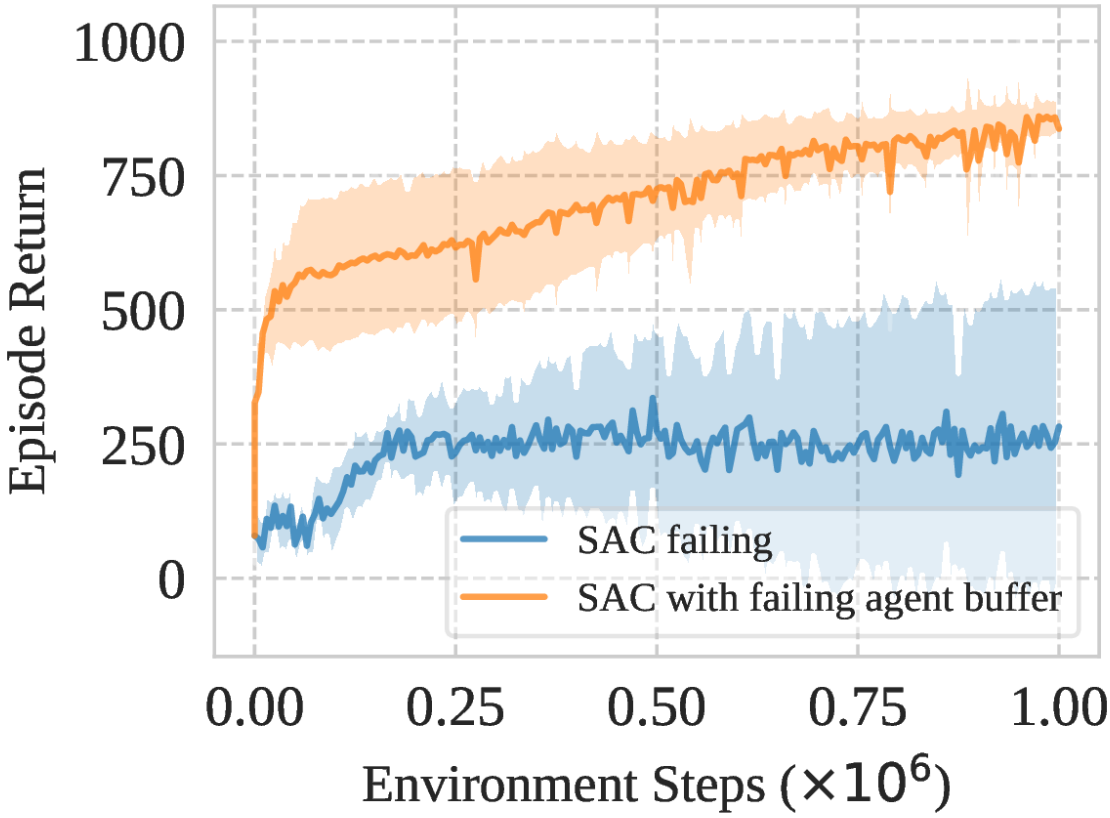


Figure 2. Undiscounted returns on `quadruped-run` for SAC trained with 9 updates per step. `SAC failing` is a standard agent; `SAC with failing agent buffer` is an agent initialized with the replay buffer of the first agent, which allows it to learn quickly. Mean and std are estimated over 10 runs.

4.4.2. Experiences of Primed Agents are Sufficient

Once the agent is heavily impacted by the primacy bias, it might struggle to reach satisfying performance. But is the data collected by an overfitted agent unusable for learning? To answer this question, we train a SAC agent with 9 updates per step in the MDP: due to the primacy bias, this agent performs poorly. Then, we initialize the same agent from scratch but use the data collected by the previous SAC agent as its initial replay buffer. Figure 2 demonstrates that returns collected by this agent improve rapidly approaching the optimal task performance.

This experiment articulates that *the primacy bias is not a failure to collect proper data per se, but rather a failure to learn from it.* The data stored in the replay buffer is in principle enough to have better performance but the overfitted agent lacks the ability to distill it into a better policy. In contrast, the randomly initialized neural networks

are not affected by the primacy bias and thus capable of fully leveraging the collected experience.

4.5. Have You Tried Resetting It?

The previous section provided controlled experiments demonstrating the primacy bias phenomenon and outlined its consequences. We now present a simple technique that mitigates the effect of this bias on an agent's training. The solution, which we call *resetting* in the rest of the paper, is summarized as follows:

> Given an agent's neural network, periodically re-initialize the parameters of its last few layers while preserving the replay buffer.

The next section analyzes both quantitatively and qualitatively the performance improvements provided by resetting as a mean to address the overfitting to early data.

4.6. Experiments

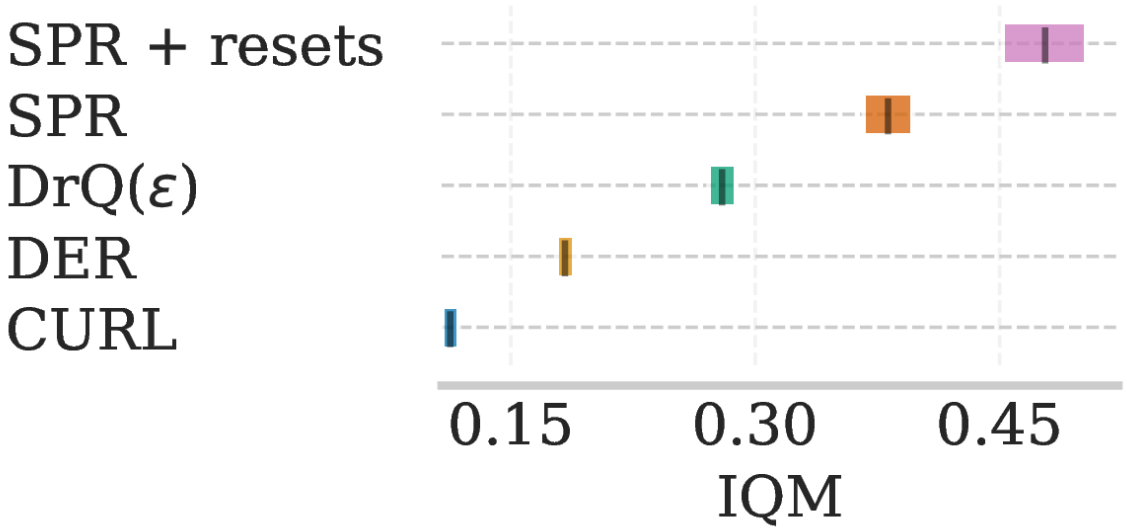


Figure 3. Point estimates and 95% bootstrap confidence intervals for the performance of SPR with resets and prior methods on Atari 100k. Results for SPR and SPR + resets are over 20 seeds per game; others are taken from Agarwal et al. (2021b) and use 100 seeds.

The goals of our experiments are mostly twofold. First, we demonstrate across different algorithms and domains the performance gains of using resets as a remedy

Method	IQM	Median	Mean
SPR + resets	**0.478** (0.46, 0.51)	**0.512** (0.42, 0.57)	**0.911** (0.84, 1.00)
SPR	0.380 (0.36, 0.39)	0.433 (0.38, 0.48)	0.578 (0.56, 0.60)
DrQ(ϵ)	0.280 (0.27, 0.29)	0.304 (0.28, 0.33)	0.465 (0.46, 0.48)
DER	0.183 (0.18, 0.19)	0.191 (0.18, 0.21)	0.351 (0.34, 0.36)
CURL	0.113 (0.11, 0.12)	0.102 (0.09, 0.12)	0.261 (0.25, 0.27)

Tableau 1. Point estimates and 95% bootstrap confidence intervals for the performance of SPR with resets and prior methods on Atari 100k. Results for SPR and SPR + resets are over 20 seeds per game; others are taken from Agarwal et al. (2021b) and use 100 seeds.

for the primacy bias; then, we analyze the learning dynamics induced by resetting, including its effects on TD learning and interaction with critical design choices of RL algorithms such as the replay ratio and n-step TD targets. Finally, we provide an ablation study of the proposed resetting strategy.

4.6.1. Setup

We focus on two domains: discrete control, represented by the 26-task Atari 100k benchmark (Kaiser et al., 2019), and continuous control, represented by the DeepMind Control Suite (Tassa et al., 2020). We apply resets to three baseline algorithms: SPR (Schwarzer et al., 2021a) for Atari, and SAC (Haarnoja et al., 2018) and DrQ (Kostrikov* et al., 2021) for continuous control from dense states and raw pixels respectively. The appendix provides all environment names and the number of training steps in each domain.

Since both the architectures and the number of training iterations vary across methods, the reset strategy needs slight customization. For SPR, we reset only the final linear layer of the 5-layer Q-network over the course of training spaced 2×10^4 steps apart; for SAC, *we reset agent's networks entirely* every 2×10^5 steps since the networks have only 3 layers; for DrQ, we reset the last 3 out of 7 layers of the policy and value networks 10 times over the course of training[1]. SAC and DrQ re-initialize target

[1]In fact, the reset periodicity in a few environments is higher due to action repeats (default is 2), a practice following (Hafner et al., 2019) used in the codebase we build upon, but using the action repeat of 2 for all environments delivers the same results.

networks and both Q-networks (due to the use of double Q-learning (Van Hasselt et al., 2016a)); SPR does not have these extra networks. We also reset the corresponding optimizer statistics (Kingma et al., 2015). Experiments in the appendix, however, show the relative robustness to these design choices.

The replay buffer is preserved between resets; SPR and SAC store in the buffer all prior interactions, while DrQ includes only the most recent 100k transitions due to memory limitations of storing image observations. SAC and DrQ sample transitions uniformly from the buffer, while SPR uses prioritized experience replay (Schaul et al., 2016). The difference between buffer configurations suggests that effects of resets hold for varying buffer sizes and sampling schemes.

After resetting, we do not perform any form of pre-training for the newly initialized parameters (Igl et al., 2021) and return directly to standard training, including keeping intact the cycle between environment interactions and agent updates. To provide rigorous evaluation of all algorithms, we follow the guidelines of Agarwal et al. (2021b) with the focus on the interquartile mean (IQM) of the performance across tasks.

4.6.2. Resets Consistently Improve Performance

Tables 1 and 2 report the aggregated results for the three algorithms. In both tables, we report the best results over different values of replay ratio and n for methods with and without resets. The empirical evidence suggests that resets mitigate the primacy bias and provide significant benefits across a wide range of tasks (discrete or continuous action spaces), input types (raw images or dense features), replay buffer configurations (matching or shorter than total number of steps, prioritized replay or random sampling), and depth of the employed neural networks (deep convolutional networks or 3-layer fully-connected networks). Remarkably, the magnitude of improvement provided by resets for SPR is comparable to advancements of prior algorithms, while not requiring additional computation costs.

4.6.3. Learning Dynamics of Agents with Resets

At first glance, resetting may appear as a drastic (if not wasteful) measure as the agent must learn the parameters of the randomly initialized layers from scratch each time. We show in this section how, against all odds, this strategy still leads to improved performance and fast learning in a wide range of situations.

Method	IQM	Median	Mean
SAC + resets	**656** (549, 753)	**617** (538, 681)	**607** (547, 667)
SAC	501 (389, 609)	475 (407, 563)	484 (420, 548)
DrQ + resets	**762** (704, 815)	**680** (625, 731)	**677** (632, 720)
DrQ	569 (475, 662)	521 (470, 600)	535 (481, 589)

Tableau 2. Point estimates and 95% bootstrap confidence intervals for the performance of SAC and DrQ with and without resets on DMC tasks. Results are computed over 10 seeds per task.

Figure 4 gives four representative examples of the learning trajectories induced by resets. By default, SAC uses policy and value function architectures which typically contain three layers; in this context, we found that resetting *the whole network* is an effective strategy. For environments in which the primacy bias does not appear to be an issue, such as `cheetah-run`, resetting causes some spikes of reduced performance in the learning curves, but the agent is able to return to the previous state in just a few thousands of steps. Instead, when dealing with environments where the algorithm is susceptible to the primacy bias, such as `hopper-hop` and `humanoid-run`, resetting not only brings performance back to the previous level but also allows the agent to surpass it.

But why is an RL agent able to recover so fast after each reset? A decisive factor for the effectiveness of resetting resides in preserving the replay buffer across iterations. Indeed, periodically emptying the replay buffer is highly detrimental for performance, as we show in the appendix. We conjecture that a model-based perspective can offer an explanation: since the replay buffer can be seen as a non-parametric model of the world (Vanseijen et al., 2015; Hasselt et al., 2019), after a reset, the agent forgets the behaviors learned in the past while preserving its model in the buffer as the core of its knowledge. On the neural network training side, Zhang et al. (2019) observe that learning mostly amounts to recovering the right representations – that is, with the preserved buffer and representations, learning an actuator might be relatively straightforward.

Resets affect learning in a generally positive way, by triggering a virtuous circle. After resetting, the agent is free from the negative priming provided by its past training iterations: it can better leverage the data collected so far, thus improving its performance and unlocking the possibility to generate higher quality data for its future updates.

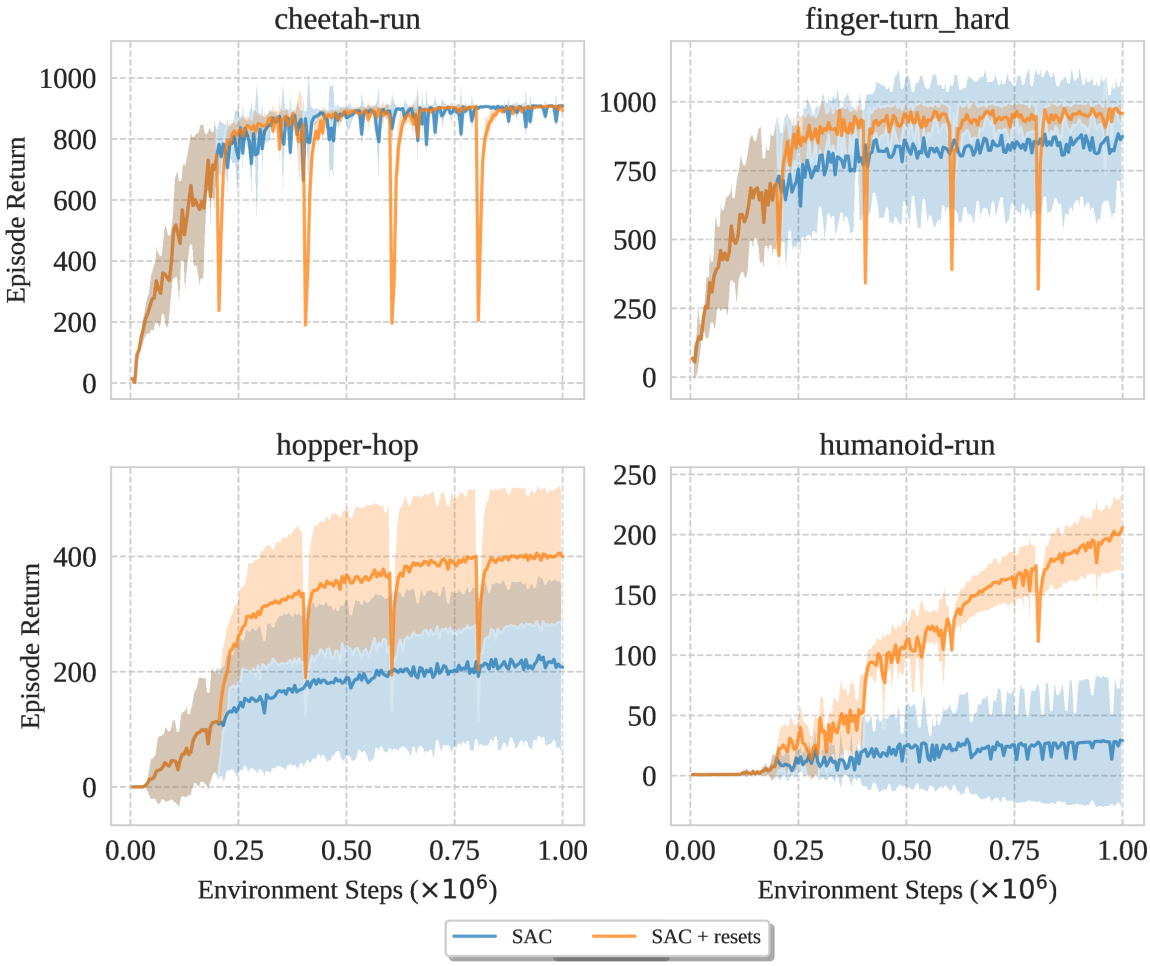


Figure 4. Four examples showing diverse effects of resets for SAC (32 updates per step, resetting every 2×10^5 steps) on DMC tasks. After each reset, performance recovers quickly due to keeping the replay buffer. In `cheetah-run`, the baseline agent consistently succeeds at the task and resets provide no major benefit. In all other tasks, resets increase performance and often reduce variance. Mean and std are estimated over 10 runs.

If the primacy bias is a special form of overfitting, resets can be seen as a tailor-made form of regularization. The appendix shows that the particular nature of resets allows the agent to overcome the primacy bias even when other forms of regularization such as L2 and dropout would not.

On a practical side, resets are an easy-to-use strategy for addressing the primacy bias. Their use requires making only two choices: the periodicity of the resets and the number of layers of the neural networks to be reinitialized. Moreover, the infrequent re-initialization of a neural network comes with no additional computational cost.

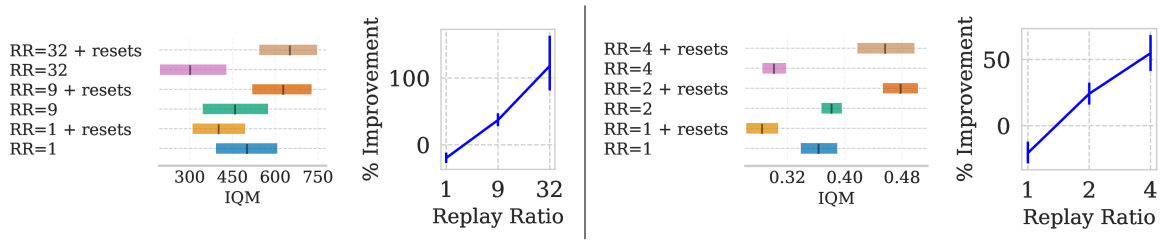


Figure 5. Performance of SAC (left) and SPR (right) and with and without resets for different replay ratios and a fixed default n. The right-hand plots visualize the percent improvement gained by adding resets. Agents with higher replay ratio are more prone to the primacy bias and hence benefit more from mitigating it.

4.6.4. Elements Behind the Success of Resets

The large improvement in performance provided by resets across algorithms and environments naturally raises a question about the conditions under which they are maximally useful. To find an answer, we focus the discussion on the interaction of resets with crucial algorithmic aspects and hyperparameters impacting the risk of overfitting.

4.6.4.1. Replay ratio. The initial experiments in section 4.4 suggest that the degree of reliance on early data is a critical determinant of the strength of the primacy bias. As a consequence, we observe that the impact of resets depends heavily on the *replay ratio*, the number of gradient updates per each environment step. fig. 5 shows that the higher the replay ratio is, the larger the effects from resets: they improve SPR's performance by over 40% at four updates/step and allow SAC to achieve its highest performance at the high replay ratio of 32, where adding resets increases performance by over 100%. Even when pushing SAC to the extreme replay ratios of 128 and 256, where learning is barely possible in most environments, resets allow the agent to obtain

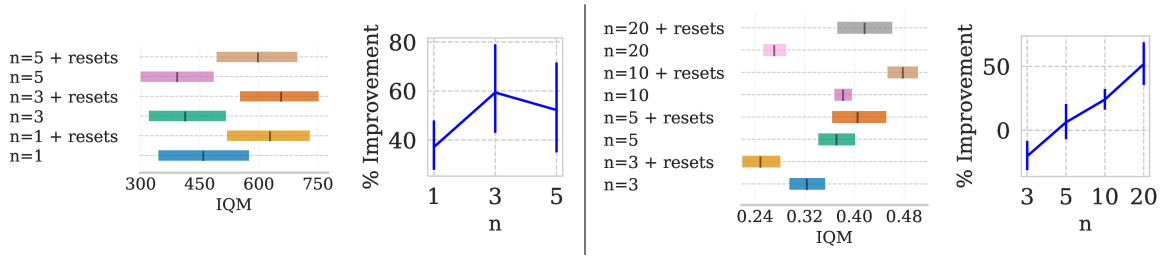


Figure 6. Performance of SAC (left) and SPR (right) with and without resets for different n-step target lengths and a fixed replay ratio (9 for SAC, default 2 for SPR). The right-hand plots visualize the percent improvement gained by adding resets. As the target variance increases with n, the agent becomes more susceptible to the primacy bias and benefits more from mitigating it.

reasonable performance (see appendix). Resets thus allow less careful tuning of this parameter and improve sample efficiency by performing more updates per each data point without being severely affected by the primacy bias.

4.6.4.2. n-step targets. The parameter n in TD learning controls a bias-variance trade-off, with larger values of n decreasing the bias in value estimates but increasing the variance (and vice versa). We hypothesize that an agent learning from higher variance targets would be more prone to overfitting to the initial data and the effects of resets would increase with increasing n. Figure 6 confirms the intuition and demonstrates up to 40% improvement for SPR for $n = 20$ and opposed to no improvement for $n = 3$. Likewise, SAC attains 50–60% improvement for increased values of n compared to 40% for the default $n = 1$.

The results with varying replay ratios and n-step targets suggest that resets reshape the hyperparameter landscape creating a new optimum with higher performance.

4.6.4.3. TD failure modes. Temporal-difference learning, when employed jointly with function approximation and off-policy training, is known to be potentially unstable (Sutton et al., 2018). In sparse-reward environments, the critic network might converge to a degenerate solution because of bootstrapping mostly on it's own outputs (Kumar et al., 2020a); having the non-zero reward data might not sufficient to escape a collapse. For example, Figure 7 (left) demonstrates the behavior of DrQ in `cartpole-swingup_sparse`, where a collapsed agent makes no learning progress. However, after a manual examination of a replay buffer, we found that the agent was reaching goal states in roughly 2% of trajectories. This observation provides evidence for an explanation that *mitigating the primacy bias addresses more the issues of optimization than exploration*. Likewise, if divergence in temporal difference learning occurs, it is essentially unrecoverable by standard optimization, with predicted values failing to decay to normal magnitudes after hundreds of thousands of steps. Figure 7 shows that adding resets solves this problem by giving the agent a second chance to find a stable solution. Even though there exist works studying in detail the pathological behaviors of TD learning (Bengio et al., 2020) and this is not the main scope of our work, resets naturally address the outlined failures.

4.6.4.4. What and how to reset. Our particular choice of the resetting strategy calls for a number of ablations. This paragraph provides only the conclusions while

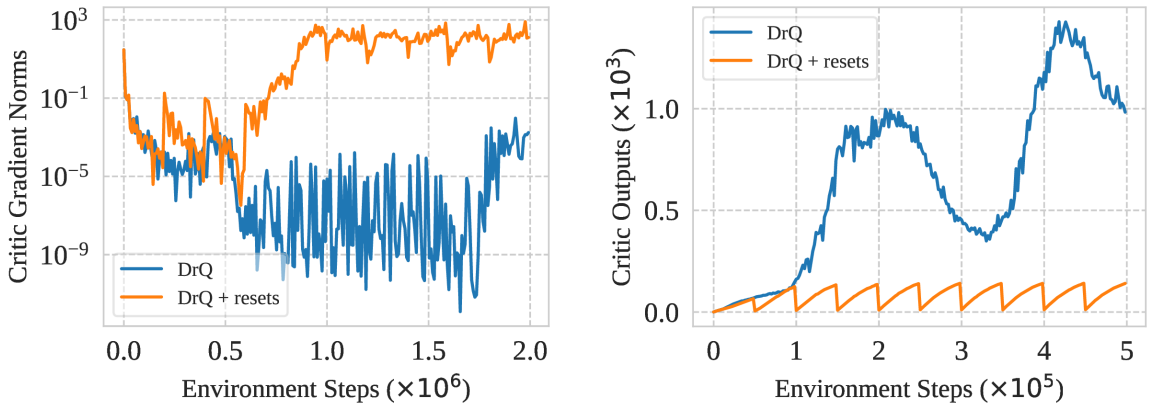


Figure 7. Examples of TD failure modes and how resets address them. **Left:** A run with TD collapse in a sparse-reward task `cartpole-swingup_sparse`. Even in the presence of non-zero rewards in the buffer, the agent without resets cannot learn a non-trivial critic. **Right:** A run with TD divergence in `walker-stand`. Even with double Q-learning, the critic might severely overestimate the action values. On both plots, DrQ without resets achieves near-zero returns, while DrQ + resets learns a near-optimal policy. The examples are not cherry-picked, such patterns of behavior occur frequently.

the appendix presents supporting figures. The number of layers to reset is a domain-dependent choice. For the SAC algorithm learning from dense state features, it is possible to reset the agent entirely. Resetting in SPR attains the best performance for the last layer only (out of 5 total layers), while for DrQ resetting the last layer is slightly worse than resetting last 3 out of 7 layers. We conjecture that the difference lies in the degree of representation learning required for each domain: a significant chunk of knowledge in Atari is contained in the agent's representations; it might be notably easier to learn features in DeepMind Control, especially when dealing with dense states. In DrQ, when resetting both actor and critic, resetting critic proved to be slightly more important than the actor; likely because the DrQ encoder learns from critic loss only, a practice proposed in (Yarats et al., 2021b).

Another seemingly important choice is whether to reset the state of the optimizer. We find that resetting the optimizer has almost no impact on training because the moment estimates are updated quickly. Regarding the resets frequency, the optimal choice should depend on how fast an algorithm can recover and how much it is affected by the primacy bias; we found that sometimes even a single reset improves the performance of baseline agents. We briefly experimented with resetting a random subnetwork and observed that the performance was either comparable or worse than with resetting the last layers. Lastly, when sampling new weights after a reset, it is natural to use a new

random seed; we observed that even with the same seed resets alleviate the primacy bias supporting the conclusions of Bjorck et al. (2022) that pathologies in deep RL algorithms are not due to problems with the initialization. Overall, while we see certain differences when varying the discussed design choices, resetting showed itself to be robust the choice of hyperparameters.

4.6.5. Summary

In short, the experimental results suggest that resets improve expected returns across a diverse set of environments and algorithms. They act as a form of regularization thus preventing overfitting to early data, unlock new hyperparameter configurations with possibly higher performance and sample efficiency, and address the optimization challenges arising in deep RL. While with a more thorough hyperparameter search and additional modifications it is possible to even further improve the performance upon the baselines, it is exciting to see that the proposed simple resetting scheme gives benefits comparable to previous algorithmic advancements.

4.7. Related Work

The primacy bias in deep RL is intimately related to memorization, optimization in RL, and cognitive science. Various aspects of our work existed in the literature.

4.7.1. Overfitting in RL

Generalization and overfitting have many faces in deep reinforcement learning. Generalization of values to similar states is a setting where the most classical form of overfitting can arise when using function approximation (Sutton et al., 2018). Kumar et al. (2020a) and Lyle et al. (2022a) show that an approximator for value function gradually loses its expressivity due to bootstrapping in TD learning; we conjecture that this amplifies the effect of first data points. Dabney et al. (2021) propose to treat holistically the sequence of value prediction tasks, arguing that if an agent focuses too much on a single prediction problem might overspecialize learned representations. Laroche et al. (2022) show that policy gradient updates can be slow in unlearning previous behaviors. Song et al. (2019) study observational overfitting by examining saliency maps in visual domains and argue that agents might pick up spurious correlations for decision making. The prioneering work (Farahmand et al., 2008) and a more recent one (Liu et al., 2021c)

argue in favor of using regularization in RL. Finally, overfitting can happen in settings with learning from offline datasets (Fujimoto et al., 2019) and with multiple tasks (Teh et al., 2017). Surveys by Kirk et al. (2021) and Zhang et al. (2018) give a taxonomy of generalization aspects in RL.

Techniques similar to our resets also existed before. Anderson et al. (1993) propose to reset individual neurons of a Q-network based on a variance criterion and observes faster convergence. Forms of non-uniform sampling including re-weighting recent samples (Wang et al., 2020a) and prioritized experience replay (PER) (Schaul et al., 2016) can be seen as a way to mitigate the primacy bias. We observe that SPR, built on top of Rainbow (Hessel et al., 2018) and already using PER, still benefits from resets. Igl et al. (2021) provide demonstrations in the supervised case that learning from a pretrained network can be worse than learning from scratch for non-stationary datasets and propose a method ITER that fully resets the agent's network in an on-policy buffer-free setting with distillation from the previous generation as a knowledge transfer mechanism. The difference contrasts the approach with our resetting scheme that uses a replay buffer as a basis for knowledge transfer after a reset.

Our work sheds light on another special form of overfitting in deep RL and proposes a simple solution for mitigating it.

4.7.2. Forgetting mechanisms

In contrast to the well-known phenomenon of catastrophic forgetting (French, 1999), several works have noted the opposite effect of catastrophic memorization (Robins, 1993; Sharkey et al., 1995) similar to the primacy bias. Achille et al. (2018) observe the existence of critical phases in learning that have a determining effect on the resulting network. Erhan et al. (2010) notice higher sensitivity of the trained network with respect to first datapoints. While the effect of the early examples in supervised problems might be present, the consequences of overfitting to initial experiences in deep RL would be much more drastic because the agent itself collects the data to learn from. Dohare et al. (2021) adjust stochastic gradient descent for the continual learning setting; we highlight that in continual learning the agent does not have influence over the stream of data, while the primacy bias in deep RL arises because data collection is in the training loop.

The idea of resetting subnetworks recently received more attention in supervised learning. Taha et al. (2021) studies this process from an evolutionary perspective and shows increased performance in computer vision tasks. Zhou et al. (2022) show

that some degree of forgetting might improve generalization and draw a connection to the emergence of compositional representations (Ren et al., 2019). Zhang et al. (2019) demonstrate that resetting different layers affect differently the performance of a network. Alabdulmohsin et al. (2021) demonstrate that resets increase margins of training examples and induce convergence to flatter minima.

These works complement the evidence about the existence of the primacy bias in deep RL and add to our analysis of the regularization effect of resets.

4.7.3. Cognitive science

The primacy bias (also known as the *primacy effect*) is a well-studied cognitive bias in human learning (Marshall et al., 1972). Given a sequence of facts, humans often form generalizations based on the first ones and pay less attention to the later ones. Asch (1961) shows that this tendency can foster a creation of harmful prejudices by examining the difference in responses after presenting the same data but in different order. Shteingart et al. (2013) argue that outcomes of first experiences affect future decision making and have a substantial and lasting effect on subsequent behavior. Yalnizyan-Carson et al. (2021) use RL as a framework to test a hypothesis that some degree of forgetting in natural brains is beneficial for decision making. Resets can be linked to cultural transmission between generations (Kirby, 2001), where an agent before a reset transmits its knowledge to an agent after the reset through a buffer. Lastly, studies of a critical period (Johnson et al., 1989) show that if a child fails to develop a skill during a particular stage of its development, it might much more difficult to acquire the skill later, drawing the connection to proper learning from early experiences.

Even though humans and RL systems learn under different conditions, our paper provides evidence that artificial agents exhibit the primacy bias noted in humans.

4.8. Future Work and Limitations

This paper focuses on empirical investigation of the primacy bias phenomenon. An exciting avenue for future work is developing theoretical understanding of risks associated to overfitting to first experiences. Likewise, deriving guarantees for learning with resets similarly to the results of Li et al. (2021) in games and Besson et al. (2018) for bandits would make the technique more theoretically sound.

Our version of resets is appealing because of its simplicity. However, the reset periodicity is a hyperparameter that an RL practitioner needs to choose. A version of the technique based on the feedback from the RL system or even meta-learning the resetting strategy can potentially improve the performance even further.

Finally, we note that brief collapses in performance induced by resetting may be undesirable from a regret minimization perspective. Potential remedies include having a period of offline post-training after each reset or sampling actions from an interpolation between pre- and post-reset agents for some period of time after each reset.

4.9. Conclusion

This paper identifies the primacy bias in deep RL, a damaging tendency of artificial agents to overfit early experiences. We demonstrate dangers associated with this form of overfitting and propose a simple solution based on resetting a part of the agent. The experimental evidence across domains and algorithms suggests that resetting is an effective and generally applicable technique in RL.

The general trend in RL research for many years was to first establish a sound algorithm for the tabular case and then use a neural network for representing parts of the agent. The last step was not rarely seen as just a technical detail. The primacy bias, however, is a phenomenon specific to RL with function approximation. The findings of our paper point at the importance of studying the profound interaction of reinforcement and deep learning. Similarly to techniques like Batch Normalization (Ioffe et al., 2015) that revolutionized training of supervised models, future progress might come not only from advancements in core RL but rather by approaching the problem in conjunction. It is striking that something as simple as periodic resetting improves performance so drastically, suggesting that there is still much to understand about the behavior of deep RL.

Overall, this work sheds light on the learning processes of deep RL agents, unlocks training regimes that were unavailable without resets, and opens possibilities for further studies improving both understanding and performance of deep reinforcement learning algorithms.

Acknowledgements

The authors thank Rishabh Agarwal, David Yu-Tung Hui, Ilya Kostrikov, Ankit Vani, Zhixuan Lin, Tristan Deleu, Wes Chung, Mandana Samiei, Hattie Zhou, Marc G. Bellemare for insightful discussions and useful suggestions on the early draft, Compute Canada for computational resources, and Sara Hooker for Isaac Asimov's quote. This work was partially supported by Facebook CIFAR AI Chair, Samsung, Hitachi, IVADO, and Gruppo Ermenegildo Zegna.

We acknowledge the Python community (Van Rossum et al., 1995; Oliphant, 2007) for developing the core set of tools that enabled this work, including JAX (Bradbury et al., 2018; Babuschkin et al., 2020), Jupyter (Kluyver et al., 2016), Matplotlib (Hunter, 2007), numpy (Oliphant, 2006; Van Der Walt et al., 2011), pandas (McKinney, 2012), and SciPy (Jones et al., 2014).

Chapter 5

Sample-Efficient Reinforcement Learning by Breaking the Replay Ratio Barrier

This chapter reproduces, with some formatting changes, the following work:

- Pierluca D'Oro*, **Max Schwarzer***, Evgenii Nikishin, Pierre-Luc Bacon, Marc Bellemare & Aaron Courville. *Sample-Efficient Reinforcement Learning by Breaking the Replay Ratio Barrier.* ICLR (2023).

This work advances the goals of this thesis by demonstrating that correctly addressing the primacy bias allows dramatically more computational resources to be used for a fixed amount of data, leading to large improvements in sample efficiency through *scaling by resetting*.

The accompanying supplemental materials for this work may be found in Appendix C. I co-led this project and performed all Atari experiments, or roughly half of the overall results in the paper.

5.1. Abstract

Increasing the replay ratio, the number of updates of an agent's parameters per environment interaction, is an appealing strategy for improving the sample efficiency of deep reinforcement learning algorithms. In this work, we show that fully or partially resetting the parameters of deep reinforcement learning agents causes better replay ratio scaling capabilities to emerge. We push the limits of the sample efficiency of carefully-modified algorithms by training them using an order of magnitude more updates than usual, significantly improving their performance in the Atari 100k and

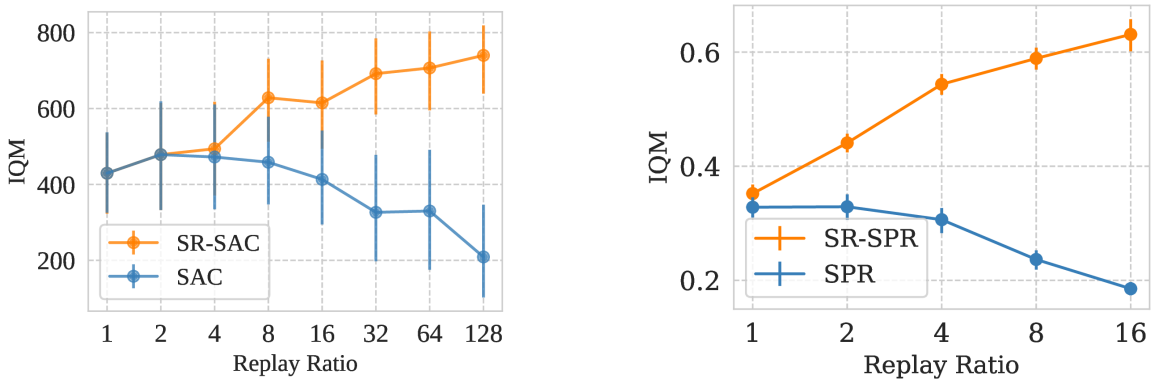


Figure 1. Scaling behavior of SAC and SR-SAC in the DeepMind Control Suite (DMC15-500k) benchmark, and of SPR and SR-SPR in the Atari 100k benchmark (5 seeds for point for SAC and SR-SAC, at least 20 seeds for point for SPR and SR-SPR, 95% bootstrapped C.I.).

DeepMind Control Suite benchmarks. We then provide an analysis of the design choices required for favorable replay ratio scaling to be possible and discuss inherent limits and tradeoffs.

5.2. Introduction

In many real world scenarios, each interaction with the environment comes at a cost, and it is desirable for deep reinforcement learning (RL) algorithms to learn with a minimal amount of samples (François-Lavet et al., 2018). This can be naturally achieved if an algorithm is able to leverage more computational resources during training to improve its performance. Given the online nature of deep RL, there is a peculiar way to aim at having such behavior: to train the agent for longer, given a dataset of experiences, before interacting with the environment again.

A method based on this idea can be said to be *scaling the replay ratio*, the number of updates of an agent's parameters for each environment interaction. Despite generally providing limited benefit when applied to standard baselines (Fedus et al., 2020; Kumar et al., 2020a), replay ratio scaling has been shown to bring performance improvements to well-tuned algorithms. Recent approaches were able to achieve better sample efficiency by increasing it to higher values, up to 8 for discrete control (Kielak, 2020) or 20 for continuous control (Chen et al., 2021b; Smith et al., 2022).

In this paper, we show that it is possible, with minimal but careful modifications to model-free algorithms mostly based on parameter resets (Ash et al., 2020; Nikishin et al., 2022), to reach new levels of replay ratio scaling and push the sample efficiency limits of

deep RL. Both in continuous control, with SAC in DeepMind Control Suite (Haarnoja et al., 2018; Tassa et al., 2020), and discrete control, with SPR in Atari 100k (Schwarzer et al., 2021a; Kaiser et al., 2020), we *break the replay ratio barrier*, unlocking a training regime in which orders of magnitude of additional agent updates can be used to increase the performance of an algorithm for a given budget of interactions with the environment. By doing so, we obtain better aggregated scores than strong baselines, with a general blueprint to improve sample efficiency of potentially any off-policy deep RL algorithm.

To understand how this can be feasible, it is useful to reflect on one of the most common patterns observed in the development of deep RL algorithms (Mnih et al., 2015). With a few exceptions, researchers typically ground their methods on the well-established dynamic programming mathematical machinery, combining it with optimization strategies common in deep learning. However, the RL setting is inherently different from the one in which most deep learning architectures and optimization methods were developed. In deep RL, neural networks have to deal with dynamic datasets, whose composition changes over the course of training; their training actively determines the value of future inputs, but also the value of future targets. We argue that the recently identified tendency of neural networks to lose their ability to learn and generalize from new information during training (Chaudhry et al., 2018; Ash et al., 2020; Berariu et al., 2021; Igl et al., 2021; Dohare et al., 2022; Lyle et al., 2022a; Lyle et al., 2022b; Nikishin et al., 2022), against which most RL methods deploy no countermeasures, has been the main roadblock in achieving better sample efficiency through replay ratio scaling.

After presenting and evaluating our algorithmic solution leading to better replay ratio scaling, we discuss some of the aspects of thinking about deep RL algorithms under the lens of this paradigm. We show some examples of algorithm design decisions important, or not important, for effective replay ratio scaling to be possible, with particular attention to the role of online interaction. Then, we visualize in an explicit way the data-computations tradeoff implied by this approach and, after having shown the potential of replay ratio scaling, we discuss its inherent limits.

5.3. Related Work

Loss of Ability to Learn and Generalize in Neural Networks A growing body of evidence suggests that artificial neural networks lose their ability to learn and generalize during training. The phenomenon is not clearly visible when learning with

a static dataset on a fixed task, but it starts appearing when the data distribution changes. In the continual learning setting, an alleviation of the problem by partially resetting the network parameters already provides a consistent improvement (Ash et al., 2020). Berariu et al. (2021) provides an in-depth study of how this phenomenon happens, including how many training updates are required for the performance of a network on future tasks to be unrecoverably damaged. The phenomenon becomes even more prominent in deep RL, where it has been identified in multiple settings. In the context of on-policy algorithms, it has been investigated as a consequence of *transient non-stationarity* and mitigated via self-distillation (Igl et al., 2021); in off-policy RL, it has been studied under the name of *capacity loss* (Lyle et al., 2022a), counteracted by the use of auxiliary tasks; in the sparse reward setting, it has been mitigated by post-training policy distillation (Lyle et al., 2022b). To address what they call *loss of plasticity*, Dohare et al. (2022) proposes a variation of backpropagation compatible with continual learning, also applying it to the continual RL context. In this paper, we primarily leverage a periodic hard resetting method (Zhou et al., 2022), investigated in Nikishin et al. (2022) as a solution to *the primacy bias* phenomenon. Our work demonstrates that addressing this phenomenon allows for increased sample efficiency by scaling the replay ratio to much higher values than other model-free methods. We report in Appendix C.1 a more precise summary and glossary of the different related definitions from previous work.

Scaling in Deep and Reinforcement Learning The topic of understanding and exploiting the scaling behavior of a deep learning algorithm's performance with respect to the amount of resources used for training has recently gained attention. Hestness et al. (2017) pioneered the idea of empirically studying and predicting performance when increasing a model's size, and subsequent work investigated scaling with respect to both model and dataset size, as well as training time (Kaplan et al., 2020; Bahri et al., 2021; Djolonga et al., 2021). Recent work in language modeling has also highlighted the importance of having high-quality data and the right training setup for efficient scaling to be possible (Hoffmann et al., 2022). In RL, scaling with respect to model size has been investigated in the offline setting for decision transformers (Lee et al., 2022) and with respect to planning-time in model-based RL (Hamrick et al., 2021). For what concerns replay ratio scaling, moderately increasing the replay ratio for standard baselines has been shown to be a competitive data-efficient baseline for both discrete and continuous control when compared to model-based RL methods (Holland et al., 2018; Hasselt et al.,

2019; Kielak, 2020; D'Oro et al., 2020), despite clear limitations (Kumar et al., 2020a). Recent approaches in continuous control leveraged high replay ratios as a strategy to improve sample efficiency through the use of ensembles of value functions (Chen et al., 2021b; Hiraoka et al., 2022; Wu et al., 2022) or normalization strategies (Smith et al., 2022); we argue that explicitly alleviating the progressive loss of ability to learn and generalize pushes the replay ratio scaling capabilities much further than those techniques can achieve.

5.4. Effective Replay Ratio Scaling with Resets

Most off-policy deep RL algorithms make use of a replay buffer (Lin, 1992) for storing transitions encountered over (a window of) an agent's lifespan. At a fixed frequency, such methods sample a batch of transitions from the buffer, update the parameters of the agent by following the gradient of some loss function, and let the agent interact again with the environment before adding new experience to the buffer. The number of agent updates per environment step is usually called *replay ratio*[1] (Wang et al., 2016a; Fedus et al., 2020), and most standard algorithms are trained with a value around 1 (Mnih et al., 2015; Haarnoja et al., 2018). It is natural to view increasing the replay ratio beyond these values as a way to improve sample efficiency. For ease of discussion, we now explicitly state and give a name to this idea, which has been an object of interest in previous studies (Hasselt et al., 2019; Kumar et al., 2020a).

Replay Ratio Scaling

Change in an agent's performance caused by doing more updates for a fixed number of environment interactions.

This definition does not have any positive connotation per se; any deep RL algorithm will have a certain replay ratio scaling behavior, and a desirable property for an algorithm is to have particularly *favorable* replay ratio scaling, so that its performance can improve by increasing the replay ratio.

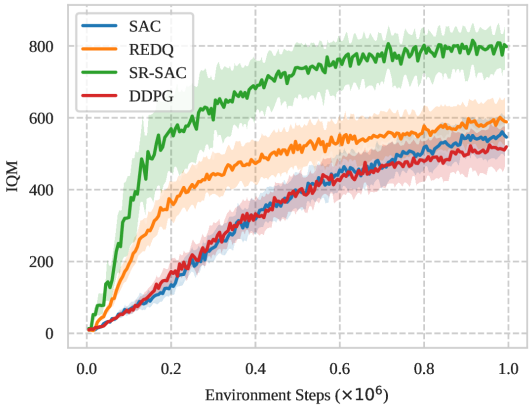
In contrast to other performance scaling properties analyzed for deep learning algorithms (Kaplan et al., 2020), replay ratio scaling is intertwined with the online RL paradigm: if the agent has a significantly better data-collection policy due to more

[1]Related quantities are also known as update-to-data (UTD) ratio (Chen et al., 2021b; Smith et al., 2022).

training, the next collected sample will be potentially different with respect to the one collected if doing less training before the interaction; by this virtue, also future learning will be directly impacted by the presence of different data in the replay buffer. In other words, this type of scaling can only be understood by considering the interaction of an agent with an environment: training more on a small dataset of interactions, without any further collection of data, will eventually lead to challenges associated to off-policy learning (Ostrovski et al., 2021); but training more *while the data is collected* can drastically change the stream of incoming data and the overall learning dynamics.

Given its appeal, what are the limiting factors to increasing the replay ratio? We argue that the main factor inhibiting effective replay ratio scaling in existing deep RL algorithms has been *the progressive loss of the ability to learn and generalize in neural networks* (Dohare et al., 2022; Lyle et al., 2022b; Nikishin et al., 2022). It has been shown that this property hinders a neural network's performance under task switches (Ash et al., 2020; Berariu et al., 2021) and, from the perspective of a neural network employed by the agent, what is deep RL if not a long sequence of related but distinct tasks (Dabney et al., 2021)?

Recent studies showed that, even under smooth task changes, the more training has been done on a previous task, the worse the performance will eventually be in a new task (Ash et al., 2020; Berariu et al., 2021). Since higher replay ratio correspond to an increased amount of training, this gives a natural explanation to the limit in increasing it. The ability to learn and generalize can, however, be restored. For instance, Nikishin et al. (2022) periodically reset the network's parameters, with a frequency that is fixed with respect to the number of *environment steps*. In this work, we argue that the key to surprisingly effective replay ratio scaling is a periodic restoration of the ability to learn and generalize of the network, via partial (Ash et al., 2020) or total (Nikishin et al., 2022) resets of its parameters, with a reset frequency that only depends on the number of *updates* and thus implicitly also on the replay ratio. This means the more an algorithm updates its neural networks, the more frequent the restoration of its ability to learn and generalize will be, leading to better performance, as we now show in practice.



DMC15-500k

Method	IQM	Median	Mean
SR-SAC	**740** (642, 818)	**667** (573, 742)	**658** (589, 722)
REDQ	511 (440, 577)	493 (442, 544)	494 (452, 534)
SAC	391 (334, 448)	424 (376, 468)	424 (386, 461)
DDPG	392 (334, 445)	410 (364, 454)	408 (371, 442)

DMC15-1M

Method	IQM	Median	Mean
SR-SAC	**805** (726, 867)	**729** (628, 790)	**710** (643, 775)
REDQ	586 (514, 649)	546 (490, 596)	539 (498, 576)
SAC	535 (467, 597)	525 (471, 567)	519 (480, 557)
DDPG	514 (450, 572)	492 (440, 540)	489 (450, 526)

Figure 2 & Tableau 1. Performance of SR-SAC and of standard baselines on the DMC15 benchmark. (5 seeds for SR-SAC, 20 for all other algorithms, 95% bootstrapped C.I.).

5.5. Replay Ratio Scaling Drastically Improves Sample Efficiency

We apply two different reset strategies to two standard continuous control and discrete control algorithms and study their replay ratio scaling behavior. We consider Soft Actor-Critic (SAC) (Haarnoja et al., 2018), which optimizes an actor and a critic by maximizing policy entropy alongside the environment's reward, and SPR (Schwarzer et al., 2021a), a model-free DQN-based reinforcement learning algorithm that augments a sample-efficient variant of Rainbow (Hasselt et al., 2019) with a model-based latent dynamics prediction objective designed to improve representation learning in the low-data regime. The two curves in Figure 1 show that it is possible, with the same algorithm, to almost double the performance for the same number of environment steps, by just varying the replay ratio. We call the modified versions of these two algorithms *Scaled-by-Resetting SAC* (SR-SAC) and *Scaled-by-Resetting SPR* (SR-SPR). In the rest of this section, we are going to describe the precise the details of the reset strategies that we employ for the two algorithms, as well as the benchmarks to which they are applied, by describing our decisions first in continuous control and then in discrete control. For evaluation and comparisons, we follow the protocol suggested by Agarwal et al. (2021b).

5.5.1. Continuous Control

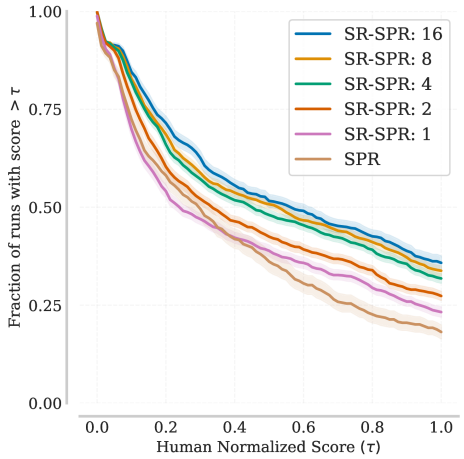
The DMC15 Benchmark To appropriately compare the performance of different algorithms, we consider a benchmark based on 15 environments from DeepMind Control Suite (Tassa et al., 2020). Our selection of tasks, reported in Table 5, is a set for which discussing sample efficiency is sensible (i.e., neither immediately solvable nor unsolvable by common deep RL algorithms). For ease of comparison, we specialize the benchmark to DMC15-500k, in which 5×10^5 interactions with the environment are allowed, and DMC15-1M, in which 10^6 interactions are allowed.

Reset Strategy We adapt the approach of Nikishin et al. (2022), and completely reset all the agent parameters every 2.56×10^6 of its updates. This lets us avoid individually specifying the moments at which resets should happen for different replay ratios. In terms of environment steps, resets will just occur more often at higher replay ratios. For instance, for replay ratio 128 (128x higher than what typically used by SAC), a reset occurs once every 20000 steps of interaction with the environment.

Results In Figure 2, we compare a version of SR-SAC that uses a replay ratio of 128 to standard deep RL baselines. This also includes the recently proposed REDQ (Chen et al., 2021b), which obtained state-of-the-art sample efficiency by using a replay ratio of 20. At any budget of interactions with the environment, SR-SAC compares favorably with REDQ, despite being a simpler algorithm. SR-SAC establishes a new state-of-the-art result for model-free continuous control. Following (Agarwal et al., 2021b) we focus on interquartile mean (IQM) performance, defined as the 25% trimmed mean performance over all runs on all considered tasks, and report 95% bootstrap confidence intervals.

5.5.2. Atari 100k

Reset Strategy We follow Nikishin et al. (2022) in performing one reset every 40,000 updates; at replay ratio 16, the highest considered, this corresponds to a reset every 2,500 environment steps, or roughly once every three minutes of interaction. However, Nikishin et al. (2022) only reset a subset of the agent's parameters when training on the ALE, leaving the agent's convolutional encoder untouched by resets. While this leaves the encoder vulnerable to plasticity loss, fully resetting the encoder is impractical, as Nikishin et al. (2022) observe. As an intermediate solution, we apply soft resets, using a variant of *Shrink and Perturb* (Ash et al., 2020) in which



Atari 100k			
Method	IQM	Median	Mean
SR-SPR	**0.632** (0.60, 0.66)	**0.685** (0.60, 0.77)	**1.272** (1.18, 1.37)
IRIS	0.501 (0.44, 0.56)	0.289 (0.25, 0.41)	1.046 (0.96, 1.13)
SPR	0.380 (0.36, 0.39)	0.433 (0.38, 0.48)	0.578 (0.56, 0.60)
DrQ(ϵ)	0.280 (0.27, 0.29)	0.304 (0.28, 0.33)	0.465 (0.46, 0.48)
DER	0.183 (0.18, 0.19)	0.191 (0.18, 0.21)	0.351 (0.34, 0.36)

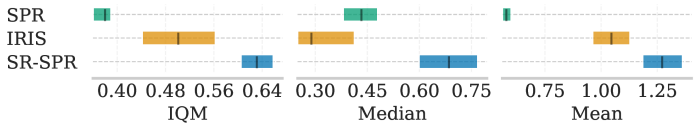


Figure 3 & Tableau 2. Performance profiles (left, higher is better) of SR-SPR at various replay ratios, and 95% C.I.s of SR-SPR: 16 and of standard baselines on Atari 100k (right, 20 seeds for SR-SPR and SPR, 5 seeds for IRIS, 100 seeds for all other algorithms as taken from Agarwal et al. (2021b))

encoder parameters are interpolated between their previous value and a random re-initialized parameter vector on each reset: $\theta^t = \alpha\theta^{t-1} + (1-\alpha)\phi$, $\phi \sim$ `initializer`. This formulation is different from that used by (Ash et al., 2020) but allows easy interpolation between completely resetting a layer and leaving it unchanged; we use $\alpha = 0.8$ by default. We examine the impact of this decision in Section 5.6.2.

Target Networks By default, SPR does not employ a separate target network, unlike traditional DQNs (Mnih et al., 2015). However, we find that this leads replay ratio scaling to stop improving performance at relatively low replay ratios, which we hypothesize is due to fundamental variance in optimization limiting the accuracy to which the value function may be estimated. To alleviate it, we directly adopt the target strategy employed by SR-SAC, with an exponential moving average (EMA) target network with coefficient $\tau = 0.005$, which we find allows beneficial replay ratio scaling out to at least replay ratio 16. Moreover, following (Ghavamzadeh et al., 2011), SR-SPR also uses its target network for action selection. We elaborate on this design decision in Section 5.6.2.

Results Figure 3 shows performance profiles of SR-SPR at various replay ratios, demonstrating that replay ratio scaling consistently improves performance up to at least replay ratio 16. We also compare a version of SR-SPR that uses replay ratio 16 to standard baselines (DrQ, DER, Kostrikov et al., 2022; Hasselt et al., 2019) and recent work (IRIS, Micheli et al., 2023) in table 2. SR-SPR establishes a new state-of-the-art for

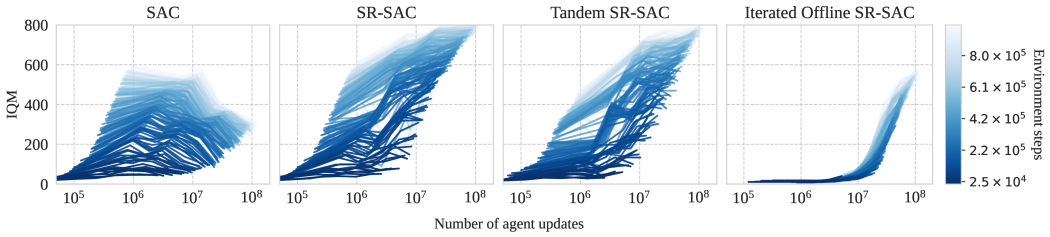


Figure 4. Scaling behavior of SAC, SR-SAC and its tandem and iterated offline variations in the DMC15 benchmark. Each individual line shows performance at a given number of environment steps, denoted by color, across different numbers of agent updates. Each point in a line is obtained by measuring performance with a different replay ratio for that number of environment steps. Each line is computed over 5 seeds.

model-free control on Atari 100k, and rivals prior work that has aggressively pretrained on additional data (Liu et al., 2021a; Schwarzer et al., 2021b). We present full results and per-game scores for SR-SPR in table 2, and show training curves in fig. 7. We report IQM performance, as well as plotting a performance profile (Agarwal et al., 2021b), which visualizes the full distribution of performance across all runs[2] and demonstrates that increasing SR-SPR 's replay ratio comprehensively improves performance.

5.6. Algorithm Design in Light of Replay Ratio Scaling

5.6.1. Analyzing the Importance of Online Interaction

When training with high replay ratios and short reset intervals, the training regime an agent is subjected to begins to resemble offline RL; the agent is primarily learning from data collected by policies unrelated to its own, with only a small amount of online data available to correct its policy. Given many classical analyses from offline RL (Levine et al., 2020), it is perhaps surprising that an agent trained in a pseudo-offline setting with no explicit regularization towards conservatism (e.g., Kumar et al., 2020b) can learn successfully. What is then the role of the incoming stream of interactions? To gain some understanding, in this section we attack the problem from different angles and study the scaling behavior of variants of SR-SAC. We consider different data collection patterns and how interleaving them with agent optimization changes the

[2]Broadly speaking, a transposed and clipped plot of the cumulative distribution function of performance.

training dynamics. The appendix also presents a comparisons with NFQI (Riedmiller, 2005) and with the online use of an offline RL algorithm.

5.6.1.1. Iterated Offline Setting. Changing the replay ratio in a deep RL algorithm can be seen as a specific way of increasing the proportion of offline training an agent is subject to. Specifically, the agent's parameters are updated a number of times exactly equal to the replay ratio before a new sample is collected. This implies a uniform distribution of the number of offline updates across time steps. Is this an important variable for determining the replay ratio scaling behavior of an algorithm?

To answer this question, we resort to what we call *iterated offline* RL (Matsushima et al., 2021; Riedmiller et al., 2021), which alternates between purely offline updates and data collection. In this setting, a certain value of replay ratio is not distributed uniformly during the course of the interactions with the environment. Instead, the agent is not updated during data collection, and an amount of updates equal to the one that would be due in that data collection time frame in virtue of the replay ratio is applied completely offline, right after each reset.

As visible in Figure 4, the iterated offline paradigm has a different replay ratio scaling behavior. Applying a very large number of updates with a fixed dataset, with an algorithm such as SAC, incurs serious risk of generating a degenerate policy, not able to outperform the previous one. As exemplified in Figure 6, in the absence of any mechanism for stopping this natural degeneration to happen, this circle is broken only when enough data is collected. Collecting enough data in this sense is possible for easy tasks such as `hopper-stand` and `walker-run`, with a cost in sample efficiency, or impossible on hard tasks such as `humanoid-stand` and `quadruped-walk`. This is unfortunate: the iterated offline RL paradigm can be quite useful in practical settings, in which the agent is allowed to only collected batches of data without any update (perhaps for safety reasons); however, current backbone algorithms (such as SAC) are not currently compatible with such a setting, that thus leads to favorable replay scaling only when closer to the online setting. This explains the sudden increase in the curve of Figure 4, when the number of agent updates, and consequently the reset frequency, becomes large enough. An interesting question, left for future work, is whether this behavior could change if combined with conservative algorithms created for the offline RL setting.

5.6.1.2. Tandem Setting. With high replay ratios, an agent's training begins to resemble offline RL: although the agent still has the possibility to interact with the environment, it is very infrequent relative to the amount of training. Thus, an agent after a reset has a small stream of interactions collected by the agent itself, while the vast majority of its data was collected by potentially unrelated agents. How important, then, is this small stream of online interaction? To answer this question, we leverage the tandem setting, as presented in Ostrovski et al. (2021). Two copies of the same agent, identical apart from the initialization, are created. With the same algorithm (SR-SAC in this case), they are trained on the replay buffer collected by the *active agent*. The *passive agent* thus never directly interacts with environment, and cannot collect data to correct its own misconceptions about the environment.

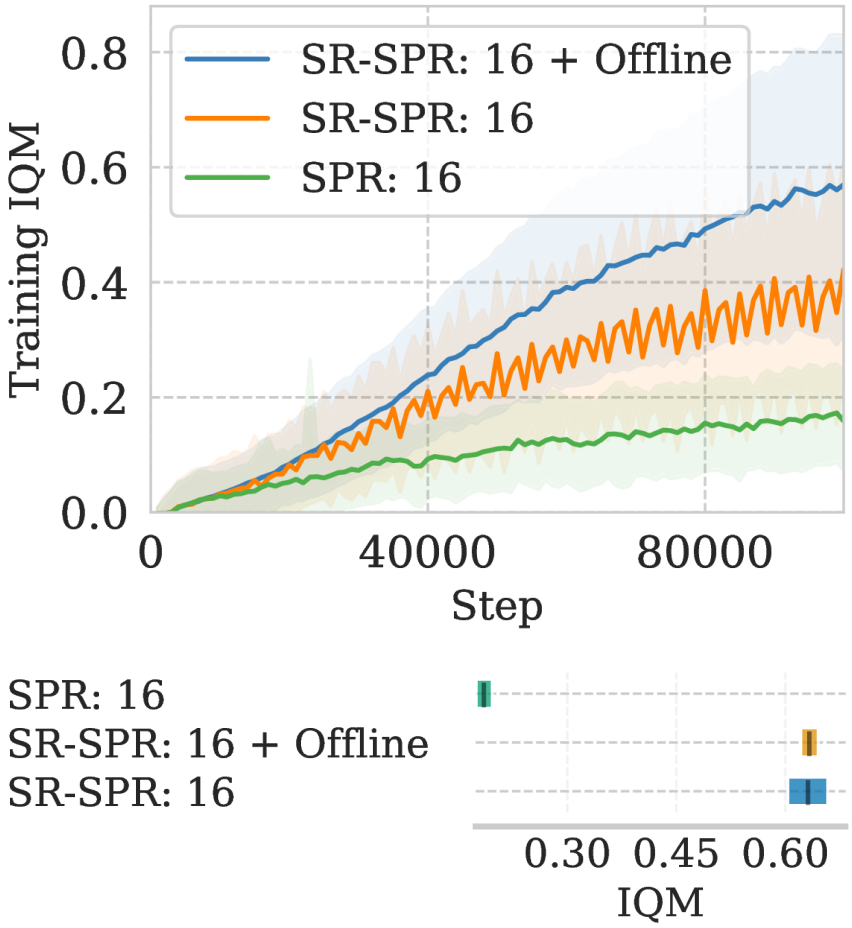


Figure 5. Learning curves (top) and evaluation performance (bottom) at replay ratio 16 for SPR and SR-SPR with and without offline updates after each reset.

As shown in Figure 4, the behavior of Tandem SR-SAC offers an alternative perspective on the importance of online interactions: despite the performance of the algorithm being hurt, the overall replay ratio scaling capabilities remain similar. We can look at the performance of the passive agent to understand what the exact effect of online interactions is on training. As evident in the environments from Figure 6, especially in `hopper-stand` and `quadruped-walk`, there is a qualitative difference between the behavior of an active agent (blue curve) and a passive agent (green curve): right after a reset, with the initial high replay ratio training, the performance of both agents is greatly improved; after a few thousands steps, training remains stable for the active agent but causes performance collapse in the passive agent. This experiment thus demonstrates the power of having online interactions as an *implicit regularization mechanism*.

For the design of future replay ratio-scalable algorithms, one should keep in mind that it is indeed possible to scale an algorithm potentially affected by extreme off-policyness; however, online data collection slows down performance collapse when training aggressively, as shown in both the iterated offline and the tandem experiments.

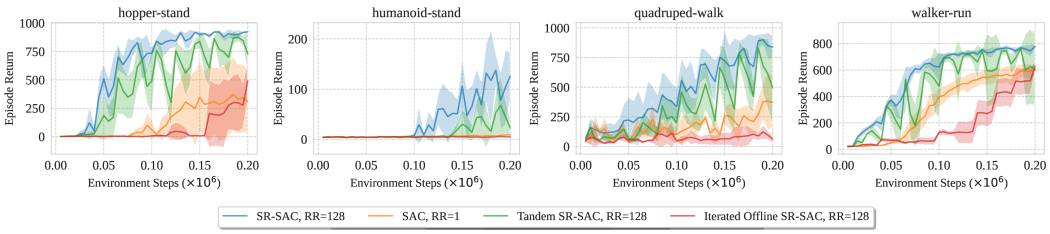


Figure 6. Examples of behaviors of SR-SAC and its tandem and iterated offline variations on four environments from DMC15. (5 runs, ± std).

5.6.1.3. Alternative Combinations of Offline and Online Updates. The iterated offline setting can be seen as the extreme in which all of the updates are done offline, compared to the even distribution used in the online setting. What if we use an intermediate strategy?

For SR-SPR, we find that directly mixing offline and online RL by performing half the updates allotted to each interval immediately after each reset can actually improve performance by some metrics, such as training return (see Figure 5 upper), by mitigating the performance drop otherwise experienced after each reset. Although we find that this has essentially no impact on final evaluation performance (Figure 5 lower), it may allow SR-SPR to be used when cumulative regret is important.

5.6.2. What is Required for Replay Ratio Scaling in Discrete Control?

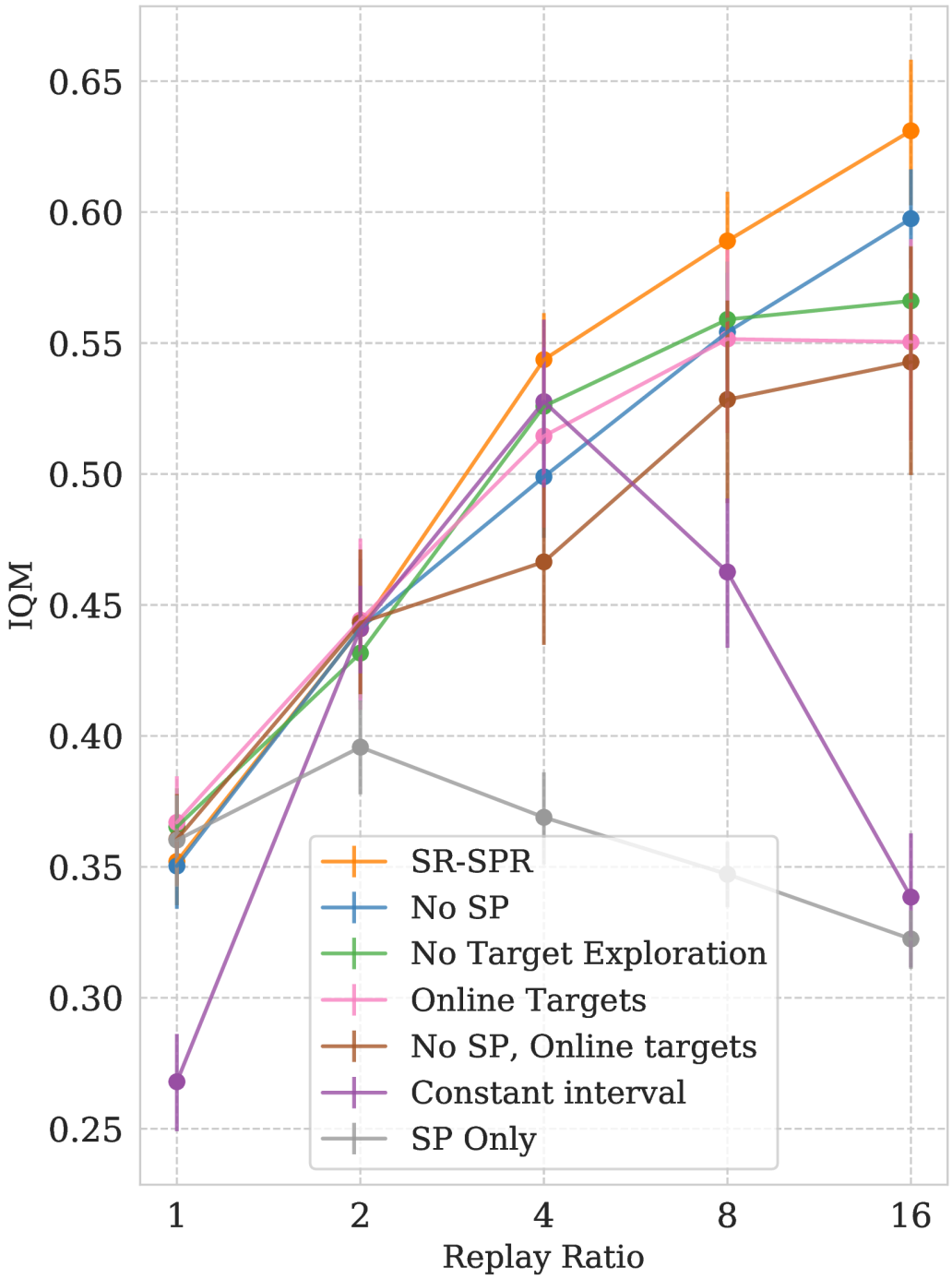


Figure 7. The replay ratio scaling behavior of SR-SPR with various components ablated.

Although replay ratio scaling is relatively straightforward for SR-SAC, achieving robust replay ratio scaling for SR-SPR requires more complex design decisions due to its shorter training period and more complex function approximation. As a result, unlike SR-SAC, SR-SPR contains additional modifications from the variant of SPR used by Nikishin et al. (2022). We study the impact of these design decisions on scaling behavior and report results in Figure 7.

Inspired by the findings of Berariu et al. (2021) that plasticity loss is concentrated in the final layers of the network but affects all layers, we apply Shrink and Perturb (SP) to the encoder; this is responsible for roughly a constant increase of IQM 0.04 past replay ratio 4. We note however that applying Shrink and Perturb alone to all the parameters of the network is not sufficient to enable beneficial scaling; it is important that at least the network's final layers be completely reset. We explain this using the observations from (Berariu et al., 2021) that the last layers are more responsible for the loss of plasticity.

That said, the most important factor in allowing SR-SPR to continue scaling well is its use of a target network. This effect is primarily due to better action selection through the target network; we found that the stabilizing effect on optimization was a less important factor. This is reminiscent of speedy Q-learning (Ghavamzadeh et al., 2011), where the use of an exponential moving average policy was shown to improve convergence speed, and can also be understood in relationship to the policy churn phenomenon (Schaul et al., 2022) (see Figure 2 in the appendix).

Meanwhile, removing both Shrink and Perturb and the target network is roughly equivalent to taking the method of Nikishin et al. (2022) but setting reset intervals as in SR-SPR. As Figure 7 suggests, this alone suffices to yield some replay ratio scaling but not as efficient compared to SR-SPR. However, maintaining a fixed reset interval (in terms of environment steps) when varying replay ratio, as done by Nikishin et al. (2022), leads to poor performance at replay ratios above 4.

Intriguingly, we note that these modifications are beneficial specifically for replay ratio scaling ; at replay ratios 1 or 2 they do not improve performance (although for the most part they do not significantly harm performance either). We thus hypothesize that there may be other modifications to complex algorithms such as SPR that could be made to further improve replay ratio scaling properties, but that are today not in widespread use because they do not improve performance in standard low replay ratio settings.

5.6.3. Visualizing the Data/Compute Tradeoff

If an order of magnitude more of updates can be used for improving the performance of an algorithm, additional tradeoffs start to emerge. The type of computations that replay ratio scaling implies are fundamentally different than other concepts of scaling, (e.g., about larger models): scaling here is inherently sequential. Thus, obtaining more hardware does not help faster execution of the algorithm. When collecting new transitions is not very expensive, the choice between collecting new samples in the environment and spending more time updating an agent could become nontrivial.

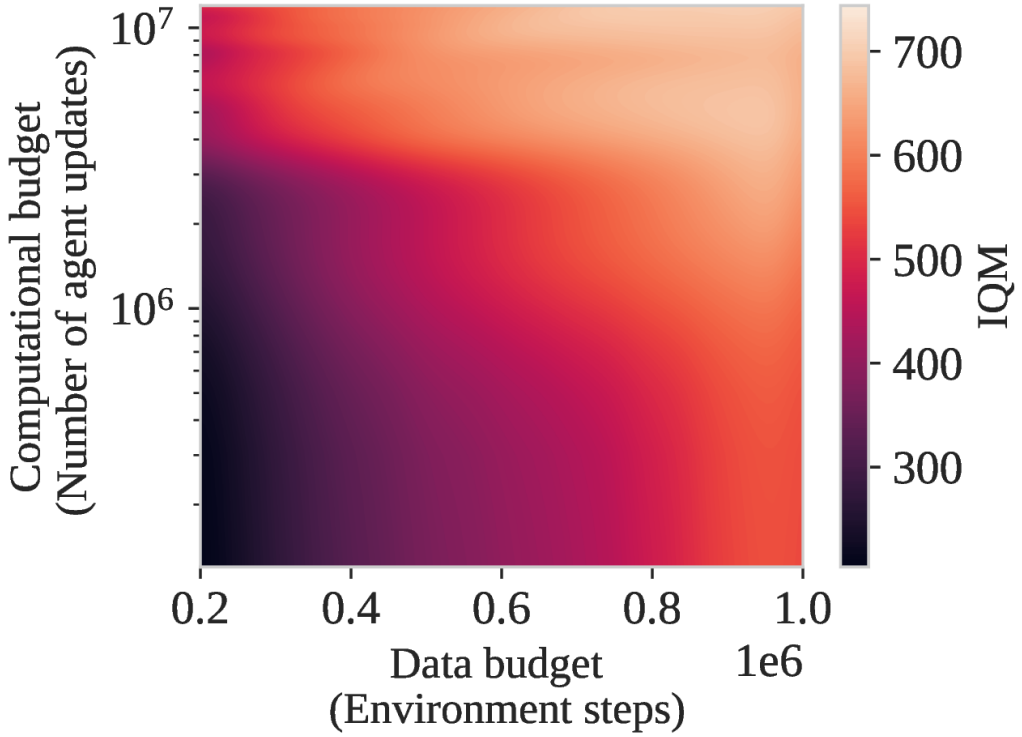


Figure 8. Performance of SR-SAC in DMC15 as a function of the number of interactions and of the number of agent updates, determined by the replay ratio.

We visualize this tradeoff in Figure 8. The plot is obtained by combining runs of SR-SAC with doubling replay ratio from 0.25 to 128, and considering, for a fixed data budget (in terms of environment steps), the total computational budget (in terms of total number of agent updates at that point), as well as the achieved performance. There exists multiple ways to achieve the same level of performance, as denoted by the color. This plot shows that resets provide a knob on replay ratio scaling and allows to tradeoff data for computation. If, for a given problem, sample efficiency is

more important than computational considerations, one can spend about two orders of magnitude of additional agent updates to obtain the same performance that can be obtained by waiting for 800000 additional samples to be collected from the environment. The peculiar feature of the approach we advocate for in this paper is that it allows to act on this tradeoff with an algorithm basically as simple as the employed backbone.

5.7. The Limits of Replay Ratio Scaling

We have seen what becomes possible when higher level of replay ratio scaling are unlocked by resets. What are the limits of this paradigm? First of all, replay ratio scaling is always possible up to a finite value, at which there is simply not enough information left to be extracted from the existing dataset of experience. Current methods, including the one proposed in this paper, are not able to automatically identify when this limit is reached, and they are therefore still subject to performance collapse when increasing the replay ratio too much. Second, replay ratio scaling cannot go beyond the intrinsic limitations of the given deep RL algorithm: for example, if the task is simply impossible to solve because of hard credit assignment or exploration, then replay ratio scaling is only of limited help. Third, the strategy we proposed for replay ratio scaling is based on keeping the entire history of interactions with the environment in the replay buffer. While this is feasible for the kind of sample-efficiency benchmarks that we have used in this paper, it might also require special consideration to be applied to larger problems; for instance, it is possible to keep a large replay buffer on permanent storage, albeit at the cost of slower batch retrieval. Lastly, replay ratio scaling can inherently become time-consuming for a training agent, which can limit the applicability of methodologies like ours to settings requiring high-frequency interactions with an environment.

5.8. Conclusions

In this paper, we have shown that, by leveraging partial or full resets of an agent's parameters, it is possible to unlock new levels of favorable replay ratio scaling and, consequently, of sample-efficiency for model-free deep RL algorithms. We demonstrated this by a careful evaluation on the DeepMind Control Suite and Atari 100k benchmarks, where our approach (SR-SAC and SR-SPR) demonstrated far superior performance compared to strong baselines, with minimal amounts of additional algorithmic complexity. Then, we discussed which algorithmic design choices are important for achieving

such levels of replay ratio scaling with a deep RL algorithm, as well as the tradeoffs implied by this paradigm. Through our empirical analysis, we showed the value of online data collection, offering a perspective on its relationship with offline RL (Levine et al., 2020).

More generally, this paper is about how to leverage a discovery for the design of future deep RL algorithms. We believe this work to be an example of how the development of effective deep RL methods should be achieved not only through extending existing algorithms or creating new ones, but also through the discovery of new phenomena related to deep RL systems, and of techniques for exploiting them to increase performance. It is natural to wonder whether deeper understanding or exploitation of surprising empirical properties (Ostrovski et al., 2021; Schaul et al., 2022) beyond the one behind this work could lead to the emergence of new capabilities in deep RL algorithms.

Acknowledgments. The authors thank Zhixuan Lin for adapting the REDQ baseline, Nathan U. Rahn, Rishabh Agarwal, David Yu-Tung Hui, Jesse Farebrother for insightful discussions and useful suggestions on the early draft, the Mila community for creating a stimulating research environment, Digital Research Alliance of Canada and Nvidia for computational resources. This work was partially supported by CIFAR, Samsung, Hitachi, Facebook AI Chair, Borealis, IVADO, and Gruppo Ermenegildo Zegna.

We acknowledge the Python community (Van Rossum et al., 1995; Oliphant, 2007) for developing the core set of tools that enabled this work, including JAX (Bradbury et al., 2018; Babuschkin et al., 2020), Jupyter (Kluyver et al., 2016), Matplotlib (Hunter, 2007), numpy (Oliphant, 2006; Van Der Walt et al., 2011), pandas (McKinney, 2012), and SciPy (Jones et al., 2014).

Chapter 6

Bigger, Better, Faster: Human-Level Atari with Human-Level Efficiency

This chapter reproduces, with some formatting changes, the following work:

- **Max Schwarzer**, Johan Obando-Ceron, Marc Bellemare, Aaron Courville, Rishabh Agarwal & Pablo Castro. *Bigger, Better, Faster: Human-Level Atari with Human-Level Efficiency.* ICML (2023).

This work advances the goals of this thesis by demonstrating that it is possible to leverage *model scaling* in sample-efficient RL through intelligent hyperparameter selection, allowing for much larger neural networks to be used in sample-efficient RL and ultimately leading to an algorithm, BBF, that possesses human-level sample-efficiency.

The accompanying supplemental materials for this work may be found in Appendix D. I led this project and performed the vast majority of experimental work, including all of the results in the main body of the paper.

6.1. Abstract

We introduce a value-based RL agent, which we call BBF, that achieves super-human performance in the Atari 100K benchmark. BBF relies on scaling the neural networks used for value estimation, as well as a number of other design choices that enable this scaling in a sample-efficient manner. We conduct extensive analyses of these design choices and provide insights for future work. We end with a discussion about updating the goalposts for sample-efficient RL research on the ALE. **We make our code and data publicly available.**

6.2. Introduction

Deep reinforcement learning (RL) has been central to a number of successes including playing complex games at a human or super-human level, such as OpenAI Five (OpenAI et al., 2019), AlphaGo (Silver et al., 2016b), and AlphaStar (Vinyals et al., 2019), controlling nuclear fusion plasma in a tokomak (Degrave et al., 2022b), and integrating human feedback for conversational agents (Ouyang et al., 2022). The success of these RL methods has relied on large neural networks and an enormous number of environment samples to learn from – a human player would require tens of thousands of years of game play to gather the same amount of experience as OpenAI Five or AlphaGo. It is plausible that such large networks are necessary for the agent's value estimation and/or policy to be expressive enough for the environment's complexity, while large number of samples might be needed to gather enough experience so as to determine the long-term effect of different action choices as well as train such large networks effectively. As such, obtaining human-level sample efficiency with deep RL remains an outstanding goal.

Although advances in modern hardware enable using large networks, in many environments it may be challenging to scale up the number of environment samples, especially for real-world domains such as healthcare or robotics. While approaches such as offline RL leverage existing datasets to reduce the need for environment samples (Agarwal et al., 2020), the learned policies may be unable to handle distribution shifts when interacting with the real environment (Levine et al., 2020) or may simply be limited in performance without online interactions (Ostrovski et al., 2021).

Thus, as RL continues to be used in increasingly challenging and sample-scarce scenarios, the need for scalable yet sample-efficient online RL methods becomes more pressing. Despite the variability in problem characteristics making a one-size-fits-all solution unrealistic, there are many insights that may transfer *across* problem domains. As such, methods that achieve "state-of-the-art" performance on established benchmarks can provide guidance and insights for others wishing to integrate their techniques.

In this vein, we focus on the Atari 100K benchmark (Kaiser et al., 2020), a well-known benchmark where agents are constrained to roughly 2 hours of game play, which is the amount of practice time the professional tester was given before human score evaluation. While human-level efficiency has been obtained by the model-based EfficientZero agent (Ye et al., 2021), it has remained elusive for model-free RL agents. To this end, we introduce BBF, a model-free RL agent that achieves super-human

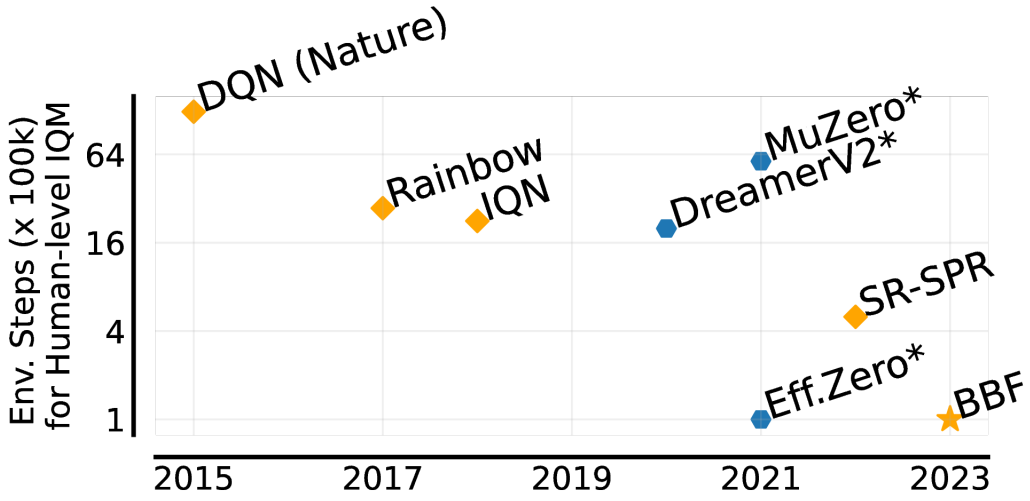


Figure 1. Environment samples to reach human-level performance, in terms of IQM (Agarwal et al., 2021b) over 26 games. Our proposed model-free agent, BBF, results in 5× improvement over SR-SPR (D'Oro et al., 2023) and at least 16× improvement over representative model-free RL methods, including DQN (Mnih et al., 2015), Rainbow (Hessel et al., 2018) and IQN (Dabney et al., 2018). To contrast with the sample-efficiency progress in model-based RL, we also include DreamerV2 (Hafner et al., 2020b), MuZero Reanalyse (Schrittwieser et al., 2021) and EfficientZero (Ye et al., 2021).

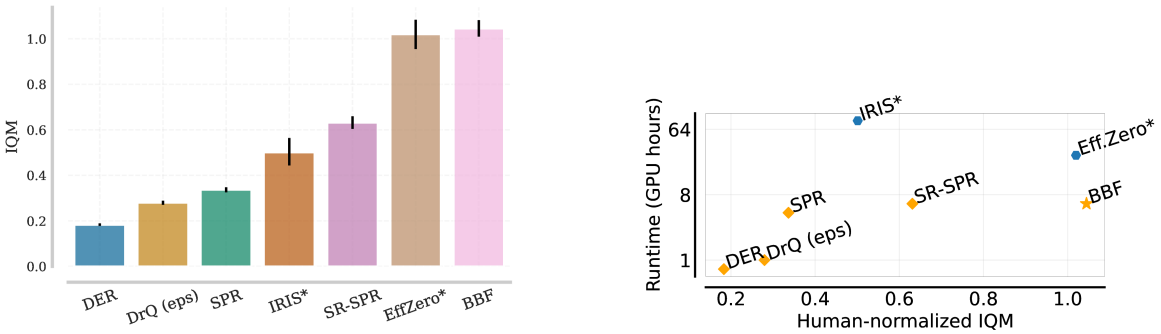


Figure 2. Comparing Atari 100K performance and computational cost of our model-free BBF agent to model-free SR-SPR (D'Oro et al., 2023), SPR (Schwarzer et al., 2021a), DrQ (eps) (Kostrikov* et al., 2021) and DER (Hasselt et al., 2019) as well as model-based* EfficientZero (Ye et al., 2021) and IRIS (Micheli et al., 2023). **(Left)** BBF achieves higher performance than all competitors as measured by interquartile mean human-normalized over 26 games. Error bars show 95% bootstrap CIs. **(Right)** Computational cost *vs.* Performance, in terms of human-normalized IQM over 26 games. BBF results in 2× improvement in performance over SR-SPR with nearly the same computational-cost, while results in similar performance to model-based EfficientZero with at least 4× reduction in runtime. For measuring runtime, we use the total number of A100 GPU hours spent per environment.

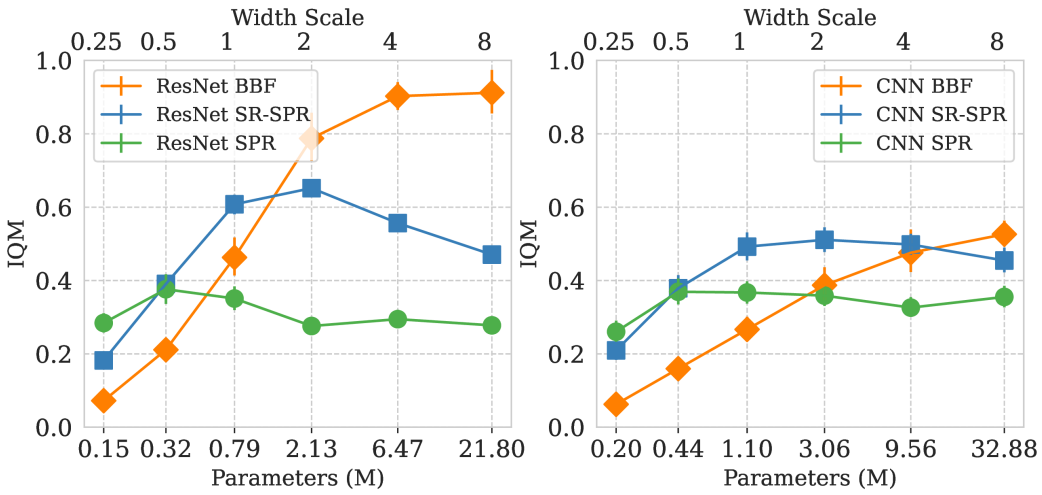


Figure 3. Scaling network widths for both ResNet and CNN architectures, for BBF, SR-SPR and SPR at replay ratio 2, with an Impala-based ResNet **(left)** and the standard 3-layer CNN (Mnih et al., 2015) **(right)**. We report interquantile mean performance with error bars indicating 95% confidence intervals. On the x-axis we report the approximate parameter count of each configuration as well as its width relative to the default (width scale = 1).

performance – interquartile mean (Agarwal et al., 2021b) human-normalized score above 1.0 – while being much more computationally efficient than EfficientZero (Figure 2). Achieving this level of performance required a larger network than the decade-old 3-layer CNN architecture (Mnih et al., 2013), but as we will discuss below, scaling network size is not sufficient on its own. We discuss and analyze the various techniques and components that are necessary to train BBF successfully and provide guidance for future work to build on our findings. Finally, we propose moving the goalpost for sample-efficient RL research on the ALE.

6.3. Background

The RL problem is generally described as a Markov Decision Proces (MDP) (Puterman, 2014), defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of available actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$[1] is the transition function, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. Agent behavior in RL can be formalized by a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, which maps states to a distribution of actions. The *value* of π when starting from $s \in \mathcal{S}$ is defined as the discounted sum of expected rewards: $V^\pi(s) := \mathbb{E}_{\pi,\mathcal{P}} \left[\sum_{t=0}^{\infty} \gamma^t r\left(s_t, a_t\right) \right]$, where $\gamma \in [0, 1)$ is a discount factor that encourages the agent to accumulate rewards

[1] $\Delta(S)$ denotes a distribution over the set S.

sooner rather than later. The goal of an RL agent is to find a policy π^* that maximizes this sum: $V^{\pi^*} \geq V^{\pi}$ for all π.

While there are a number of valid approaches (Sutton et al., 2018), in this paper we focus on model-free *value-based* methods. Common value-based algorithms approximate the Q^*-values, defined via the Bellman recurrence:
$Q^*(s,a) := R(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)}[\max_{a' \in \mathcal{A}} Q^*(s',a')]$. The optimal policy π^* can then be obtained from the optimal state-action value function Q^* as $\pi^*(x) := \max_{a \in \mathcal{A}} Q^*(s,a)$. A common approach for learning Q^* is the method of temporal differences, optimizing the Bellman temporal difference:

$$\left(r\left(s_t, a_t\right) + \gamma \max_{a_{t+1}} Q\left(s_{t+1}, a_{t+1}\right)\right) - Q\left(s_t, a_t\right).$$

We often refer to $\left(r\left(s_t, a_t\right) + \gamma \max_{a_{t+1}} Q\left(s_{t+1}, a_{t+1}\right)\right)$ as the *Bellman target*.

Mnih et al. (2015) introduced the agent DQN by combining temporal-difference learning with deep networks, and demonstrated its capabilities in achieving human-level performance on the Arcade Learning Environment (ALE) (Bellemare et al., 2013). They used a network consisting of 3 convolutional layers and 2 fully connected layers, parameterized by θ, to approximate Q (denoted as Q_θ). We will refer to this architecture as the CNN architecture. Most of the work in value-based agents is built on the original DQN agent, and we discuss a few of these advances below which are relevant to our work.

Hessel et al. (2018) combined six components into a single agent they called *Rainbow*: prioritized experience (Schaul et al., 2016), n-step learning (Sutton, 1988), distributional RL (Bellemare et al., 2017a), double Q-learning (Hasselt et al., 2016), dueling architecture (Wang et al., 2016b) and NoisyNets (Fortunato et al., 2018). Hessel et al. (2018) and Ceron et al. (2021) both showed that Multi-step learning is one of the most crucial components of Rainbow, in that removing it caused a large drop in performance.

In n-step learning, instead of computing the temporal difference error using a single-step transition, one can use n-step targets instead (Sutton, 1988), where for a trajectory $(s_0, a_0, r_0, s_1, a_1, \cdots)$ and update horizon n: $R_t^{(n)} := \sum_{k=0}^{n-1} \gamma^k r_{t+k+1}$, yielding the multi-step temporal difference: $R_t^{(n)} + \gamma^n \max_{a'} Q_\theta(s_{t+n}, a') - Q_\theta(s_t, a_t)$.

Most modern RL algorithms store past experiences in a *replay buffer* that increases sample efficiency by allowing the agent to use samples multiple times during learning, and to leverage modern hardware such as GPUs and TPUs by training on sampled mini-batches. An important design parameter is the **replay ratio**, the ratio of learning

updates to online experience collected (Fedus et al., 2020). It is worth noting that DQN uses a replay ratio of 0.25 (4 environment interactions for every learning update), while some sample-efficient agents based on Rainbow use a value of 1.

Nikishin et al. (2022) showed that the networks used by deep RL agents have a tendency to overfit to early experience, which can result in sub-optimal performance. They proposed a simple strategy consisting of periodically resetting the parameters of the final layers of DQN-based agents to counteract this. Building on this promising work, D'Oro et al. (2023) added a shrink-and-perturb technique for the parameters of the convolutional layers, and showed that this allowed them to scale the replay ratio to values as high as 16, with no performance degradation.

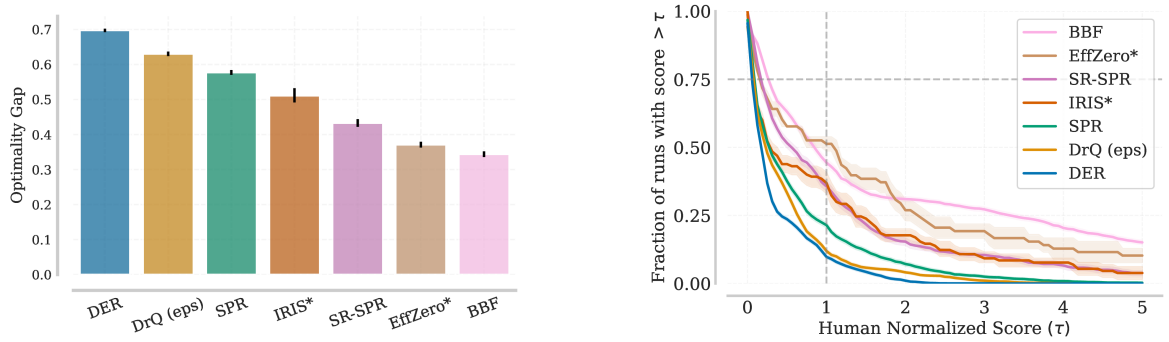


Figure 4. (**Left**). Optimality Gap (lower is better) for BBF at replay ratio 8 and competing methods on Atari 100K. Error bars show 95% CIs. BBF, has a lower optimality gap than any competing algorithm, indicating that it comes closer on average to achieving human-level performance across all tasks. **(Right)** Performance profiles showing the distribution of scores across all runs and 26 games at the end of training (higher is better). Area under an algorithm's profile is its mean performance while τ value where it intersects y = 0.75 shows its 25th percentile performance. BBF has better performance on challenging tasks that may not otherwise contribute to IQM or median performance.

6.4. Related Work

Sample-Efficient RL on ALE: Sample efficiency has always been an import aspect of evaluation in RL, as it can often be expensive to interact with an environment. Kaiser et al. (2020) introduced the Atari 100K benchmark, which has proven to be useful for evaluating sample-efficiency, and has led to a number of recent advances.

Kostrikov* et al. (2021) use data augmentation to design a sample-efficient RL method, DrQ, which outperformed prior methods on Atari 100K. Data-Efficient Rainbow (DER) (Hasselt et al., 2019) and DrQ(ϵ) (Agarwal et al., 2021b) simply modified the

hyperparameters of existing model-free algorithms to exceed the performance of existing methods without any algorithmic innovation.

Schwarzer et al. (2021a) introduced SPR, which builds on Rainbow (Hessel et al., 2018) and uses a self-supervised temporal consistency loss based on BYOL (Grill et al., 2020) combined with data augmentation. SR-SPR (Schwarzer et al., 2021a) combines SPR with periodic network resets to achieve state-of-the-art performance on the 100K benchmark. Ye et al. (2021) used a self-supervised consistency loss similar to SPR (Chen et al., 2021a).

EfficientZero (Ye et al., 2021), an efficient variant of MuZero (Schrittwieser et al., 2020), learns a discrete-action latent dynamics model from environment interactions, and selects actions via lookahead MCTS in the latent space of the model. Micheli et al. (2023) introduce IRIS, a data-efficient agent that learns in a world model composed of an autoencoder and an auto-regressive Transformer.

Scaling in Deep RL: Deep neural networks are useful for extracting features from data relevant for various downstream tasks. Recently, there has been interest in the scaling properties of neural network architectures, as scaling model size has led to commensurate performance gains in applications ranging from language modelling to computer vision.

Based on those promising gains, the deep RL community has begun to investigate the effect of increasing the model size of the function approximator. Sinha et al. (2020) and Ota et al. (2021) explore the interplay between the size, structure, and performance of deep RL agents to provide intuition and guidelines for using larger networks. Kumar et al. (2022) find that with ResNets (up to 80 million parameter networks) combined with distributional RL and feature normalization, offline RL can exhibit strong performance that scales with model capacity. Taiga et al. (2023) show that generalization capabilities on the ALE benefit from higher capacity networks, such as ResNets. Cobbe et al. (2020) and Farebrother et al. (2023) demonstrate benefits when scaling the number of features in each layer of the ResNet architecture used by Impala (Espeholt et al., 2018), which motivated the choice of feature width scaling in this work. Different from these works, our work focus on improving sample-efficiency in RL as opposed to offline RL or improving generalization in RL.

In the context of online RL, Hafner et al. (2023) demonstrate that increased dynamics model size, trained via supervised learning objectives, leads to monotonic improvements in the agent's final performance. Recently, AdA (Bauer et al., 2023)

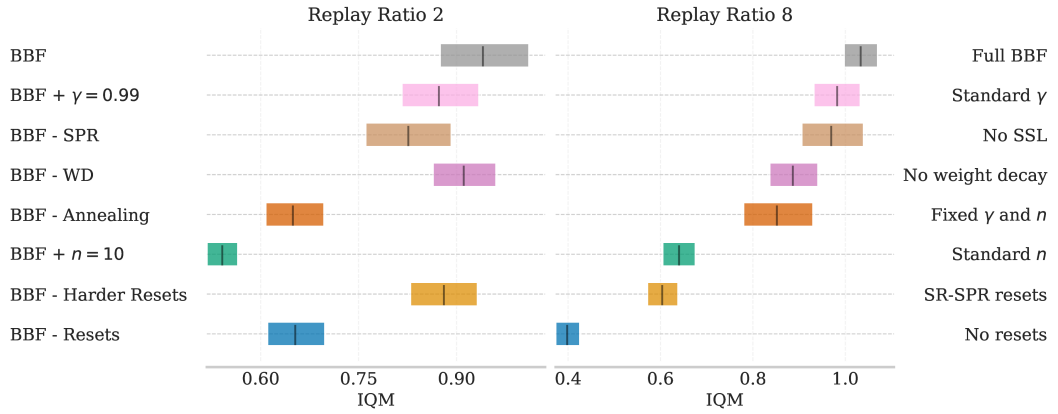


Figure 5. Evaluating the impact of removing the various components that make up BBF with RR=2 and RR=8. Reporting interquantile mean averaged over the 26 Atari 100k games, with 95% CIs over 15 independent runs.

scales transformer encoder for a Muesli agent up to 265M parameters. Interestingly, AdA required distillation from smaller models to bigger models to achieve this scaling, in the spirit of reincarnating RL (Agarwal et al., 2022). However, it is unclear whether findings from above papers generalize to scaling typical value-based deep RL methods in sample-constraint settings, which we study in this work.

6.5. Method

The question driving this work is: *How does one scale networks for deep RL when samples are scarce?* To investigate this, we focus on the well-known Atari 100K benchmark (Kaiser et al., 2020), which includes 26 Atari 2600 games of diverse characteristics, where the agent may perform only 100K environment steps, roughly equivalent to two hours of human gameplay[2]. As we will see, näively scaling networks can rarely maintain performance, let alone improve it.

The culmination of our investigation is the Bigger, Better, Faster agent, or BBF in short, which achieves super-human performance on Atari 100K with about 6 hours on single GPU. Figure 2 demonstrates the strong performance of BBF relative to some of the best-performing Atari 100K agents: EfficientZero (Ye et al., 2021), SR-SPR (D'Oro et al., 2023), and IRIS (Micheli et al., 2023). BBF consists of a number of components, which we discuss in detail below.

[2]100k steps (400k frames) at 60 FPS is 111 minutes.

Our implementation is based on the Dopamine framework (Castro et al., 2018) and uses mostly already previously-released components. For evaluation, we use rliable (Agarwal et al., 2021b) and in particular, the interquartile mean (IQM) metric, which is the average score of the middle 50% runs combined across all games and seeds.

Base agent. BBF uses a modified version of the recently introduced SR-SPR agent (D'Oro et al., 2023). Through the use of periodic network resets, SR-SPR is able to scale up its replay ratio (RR) to values as high as 16, yielding better sample efficiency. For BBF, we use RR=8 in order to balance the increased computation arising from our large network. Note that this is still very high relative to existing Atari agents – Rainbow and its data-efficient variant DER (Hasselt et al., 2019) use RR=0.25 and 1, respectively.

As we expect that many users will not wish to pay the computational costs of running at replay ratio 8, we also present results for BBF and ablations at replay ratio 2 (matching SPR). For all experiments we state which replay ratio is being used in the captions.

Harder resets. The original SR-SPR agent (D'Oro et al., 2023) used a shrink-and-perturb method for the convolutional layers where parameters were only perturbed 20% of the way towards a random target, while later layers were fully reset to a random initialization. An interesting result of our investigation is that using harder resets of the convolutional layers yields better performance. In our work, we move them 50% towards the random target, resulting in a stronger perturbation and improving results (see Figure 5). This may be because larger networks need more regularization, as we find that they reduce loss faster (Figure 1).

Larger network. Scaling network capacity is one of the motivating factors for our work. As such, we adopt the Impala-CNN (Espeholt et al., 2018) network, a 15-layer ResNet, which has previously led to substantial performance gains over the standard 3-layer CNN architecture in Atari tasks where large amounts of data are available (Agarwal et al., 2022; Schmidt et al., 2021). Additionally, BBF scales the width of each layer in Impala-CNN by 4×. In Figure 3, we examine how the performance of SPR, SR-SPR and BBF varies with different choices of scaling width, for both the ResNet and original CNN architectures. Interestingly, although the CNN has roughly 50% more parameters than the ResNet at each scale level, the ResNet yields better performance at all scaling levels for both SR-SPR and BBF.

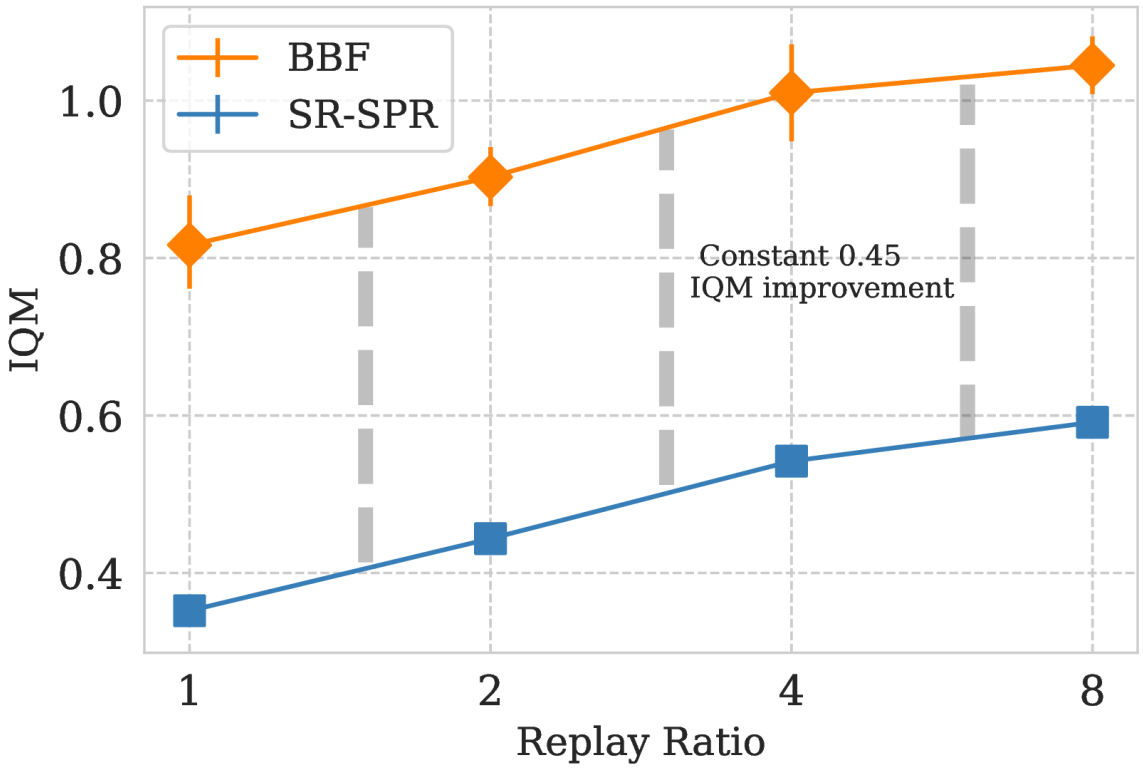


Figure 6. Comparison of BBF and SR-SPR across different replay ratios. We report IQM with 95% CIs for each point. BBF achieves an almost-constant 0.45 IQM improvement over SR-SPR at each replay ratio.

What stands out from Figure 3 is that BBF's performance continues to grow as width is increased, whereas SR-SPR seems to peak at 1-2× (for both architectures). Given that ResNet BBF performs comparably at 4× and 8×, we chose 4× to reduce the computational burden. While reducing widths beyond this could further reduce computational costs, this comes at the cost of increasingly sharp reductions in performance for all methods tested.

Receding update horizon. One of the surprising components of BBF is the use of an update horizon (n-step) that decreases exponentially from 10 to 3 over the first 10K gradient steps following each network reset. Given that we follow the schedule of D'Oro et al. (2023) and reset every 40k gradient steps, the annealing phase is always 25% of training, regardless of the replay ratio. As can be seen in Figure 5, this yields a much stronger agent than using a fixed value of $n = 3$, which is default for Rainbow, or $n = 10$, which is typically used by Atari 100K agents like SR-SPR.

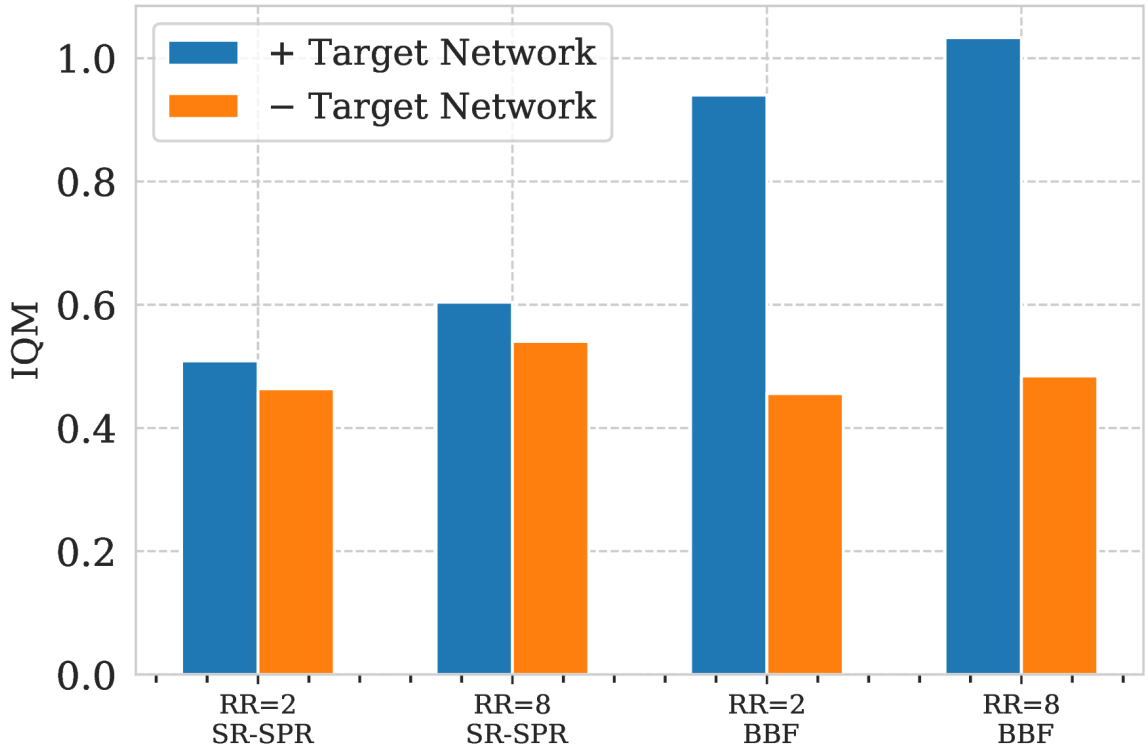


Figure 7. Comparison of BBF and SR-SPR at replay ratios 2 and 8 with and without EMA target networks. Human-normalized IQM on the 26 Atari 100k games.

Our n-step schedule is motivated by the theoretical results of Kearns et al. (2000) – larger values of n-step leads to faster convergence but to higher asymptotic errors with respect to the optimal value function. Thus, selecting a fixed value of n corresponds to a choice between having either rapid convergence to a worse asymptote, or slower convergence to a better asymptote. As such, our exponential annealing schedule closely resembles the optimal decreasing schedule for n-step derived by Kearns et al. (2000).

Increasing discount factor. Motivated by findings that increasing the discount factor γ during learning improves performance (François-Lavet et al., 2015), we increase γ from γ_1 to γ_2, following the same exponential schedule as for the update horizon. Note that increasing γ has the effect of progressively giving more weights to delayed rewards. We choose $\gamma_1 = 0.97$, slightly lower than the typical discount used for Atari, and $\gamma_2 = 0.997$ as it is used by MuZero (Schrittwieser et al., 2021) and EfficientZero (Ye et al., 2021). As with the update horizon, Figure 5 demonstrates that this strategy outperforms using a fixed value.

Weight decay. We incorporate weight decay in our agent to curb statistical overfitting, as BBF is likely to overfit with its high replay ratio. To do so, we use the AdamW

optimizer (Loshchilov et al., 2019) with a weight decay value of 0.1. Figure 5 suggests the gains from adding weight decay are significant and increase with replay ratio, indicating that the regularizing effects of weight decay enhance replay ratio scaling with large networks.

Removing noisy nets. Finally, we found that NoisyNets (Fortunato et al., 2018), used in the original SPR (Schwarzer et al., 2021a) and SR-SPR, did not improve performance. This could be due to NoisyNets causing over-exploration due to increased policy churn (Schaul et al., 2022) from added noise during training, or due to added variance in optimization, and we leave investigation to future work. Removing NoisyNets results in large computational and memory savings, as NoisyNets creates duplicate copies of the weight matrices for the final two linear layers in the network, which contain the vast majority of all parameters: turning on NoisyNets increases the FLOPs per forward pass and the memory footprint by a factor of 2.5× and 1.6×, respectively, which both increases runtime and reduces the number of training runs that can be run in parallel on a single GPU. Removing NoisyNets is thus critical to allowing BBF to achieve reasonable compute efficiency despite its larger networks. We found that this decision had no significant impact on task performance (see Figure 2 in appendix).

6.6. Analysis

In light of the importance of BBF's components, we discuss possible consequences of our findings for other algorithms.

The importance of self-supervision. One unifying aspect of the methods compared in Figure 2 is that they all use some form of self-supervised objective. In sample-constrained scenarios, like the one considered here, relying on more than the temporal-difference backups is likely to improve learning speed, provided the self-supervised losses are consistent with the task at hand. We test this by removing the SPR objective (inherited from SR-SPR) from BBF, and observe a substantial performance degredation (see fig. 5). It is worth noting that EfficientZero uses a self-supervised objective that is extremely similar to SPR, a striking commonality between BBF and EfficientZero.

Sample efficiency via more gradient steps. The original DQN agent (Mnih et al., 2015) has a replay ratio of 0.25, which means a gradient update is performed only after every 4 environment steps. In low-data regimes, it is more beneficial to perform more gradient steps, although many algorithms cannot benefit from this without additional

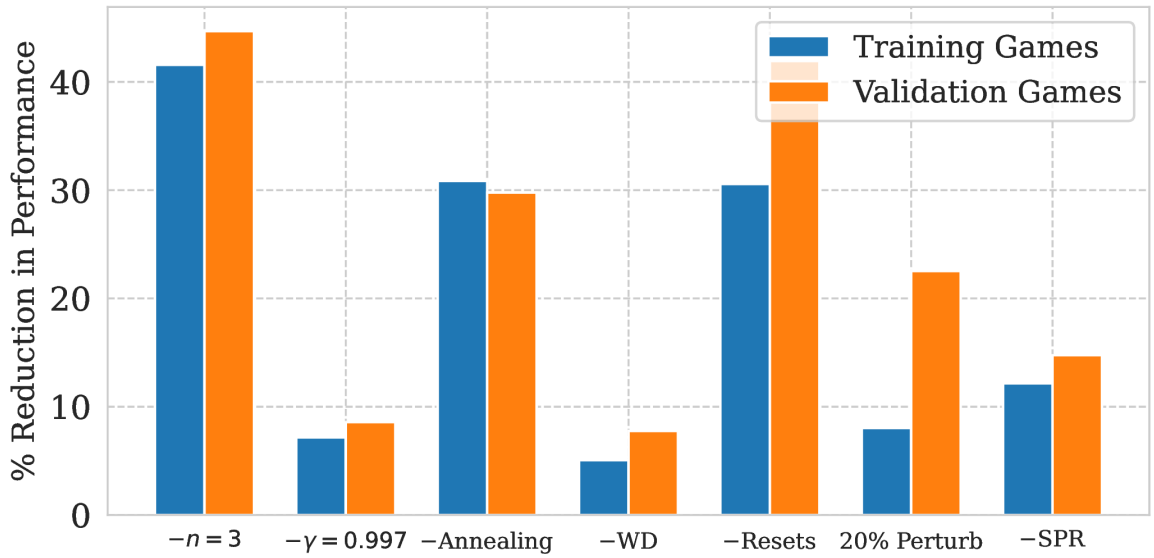


Figure 8. Validating BBF design choices at RR=2 on 29 unseen games. While Atari 100K training set consists of 26 games, we evaluate the performance of various components in BBF on 29 validation games in ALE that are not in Atari 100K. Interestingly, all BBF components lead to a large performance improvement on unseen games. Specifically, we measure the % decrease in human-normalized IQM performance relative to the full BBF agent at RR=2.

regularization (D'Oro et al., 2023). As Figure 6 confirms, performance of BBF grows with increasing replay ratio in the same manner as its base algorithm, SR-SPR. More strikingly, we observe a *linear* relationship between the performance of BBF and SR-SPR across all replay ratios, with BBF performing roughly 0.45 IQM above SR-SPR. While the direction of this relationship is intuitive given the network scaling introduced by BBF, its linearity is unexpected, and further investigation is needed to understand the nature of the interaction between replay ratio and network scaling.

One interesting comparison to note is that, although EfficientZero uses a replay ratio of 1.2, they train with a batch size that is 8 times larger than ours. Thus, their *effective* replay ratio is comparable to ours.

The surprising importance of target networks. Many prior works on Atari 100k, such as DrQ and SPR (Kostrikov* et al., 2021; Schwarzer et al., 2021a) chose not to use target networks, seeing them unnecessary or an impediment to sample efficiency. Later, D'Oro et al. (2023) re-introduced an exponential moving average target network, used both for training and action selection, and found that it improved performance somewhat, especially at high replay ratios. With network scaling, however, using a target network

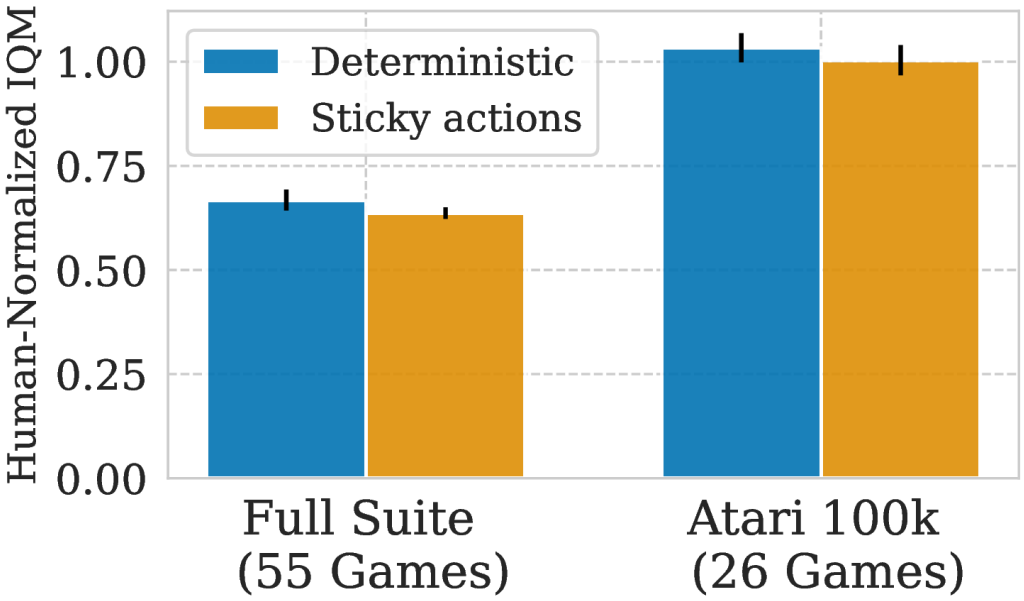


Figure 9. Evaluating BBF on ALE with and w/o sticky actions. We report IQM human-normalized performance at replay ratio 8 on 26 games in Atari 100K as well the full set of 55 games in ALE. While performance on the full set of 55 games is lower, neither setting has its performance significantly affected by sticky actions.

becomes a critical, but easy-to-overlook, component of the algorithm at all replay ratios (see Figure 7).

Reset Strength. Increasing the replay ratio is in general challenging, as explored by Fedus et al. (2020) and Kumar et al. (2020a). Periodic resetting, as suggested by Nikishin et al. (2022) and D'Oro et al. (2023), has proven effective to enable scaling to larger replay ratios, quite possibly a result of reduced overfitting. This is confirmed in Figure 5, where the importance of resets is clear. Further, Figure 5 and Figure 8 demonstrate the added benefit of more aggressive perturbations, relative to SR-SPR.

Scale is not enough on its own. The naïve approach of simply scaling the capacity of the CNN used by SR-SPR turns out to be insufficient to improve performance. Instead, as Figure 3 shows, the performance of SR-SPR collapses as network size increases. As discussed in section 6.5, it is interesting to observe that the smaller Impala-CNN ResNet (as measured by number of parameters and FLOPs) yields stronger performance at all width scales.

Computational efficiency. As machine learning methods become more sophisticated, an often overlooked metric is their computational efficiency. Although EfficientZero trains in around 8.5 hours, it requires about 512 CPU cores and 4 distributed GPUs. IRIS

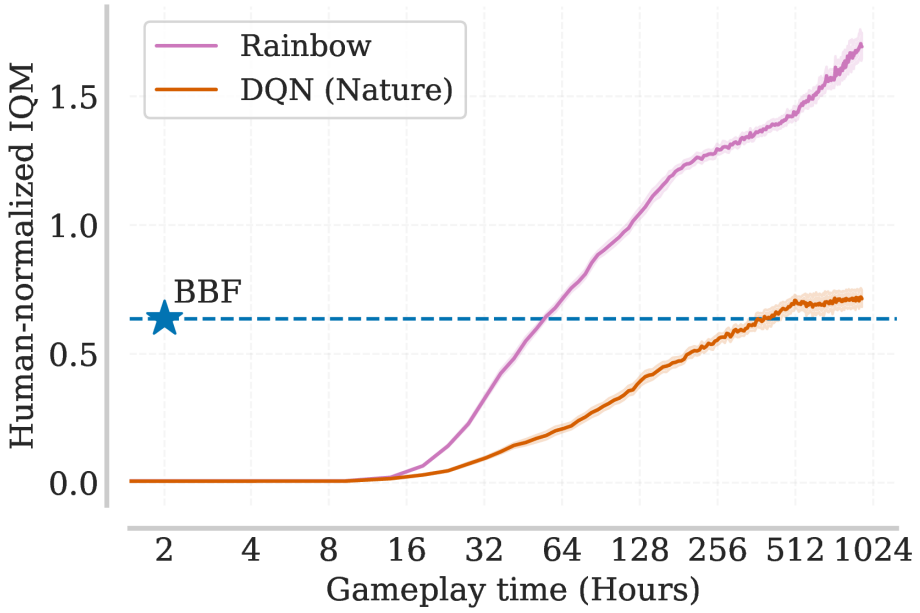


Figure 10. Sample efficiency progress on ALE, measured via human-normalized IQM over 55 Atari games with sticky actions, as a function of amount of human game play hours, with BBF at RR=8. Shaded regions show 95% CIs.

uses half of an A100 GPU for a week per run. SR-SPR, at its highest replay ratio of 16, uses 25% of an A100 GPU and a single CPU for roughly 24 hours. Our BBF agent at replay ratio 8 takes only 10 hours with a single CPU and half of an A100 GPU. Thus, measured by GPU-hours, BBF provides the best trade-off between performance and compute (see Figure 2).

6.7. Revisiting the Atari 100k benchmark

A natural question is whether there is any value in continuing to use the Atari 100K benchmark, given that both EfficientZero and BBF are able to achieve human-level performance (IQM ≥ 1.0) in just 100K steps. When considering this, it is important to remember that IQM is an *aggregate* measure. Indeed, in the left panel of Figure 4 we can see there is still room for improvement with regards to the *optimality gap*, which measures the amount by which each algorithm fails to meet a minimum score of 1.0 (Agarwal et al., 2021b). Specifically, despite monotonic progress over the years, no agent is yet able to achieve human-level performance on all 26 games, which would yield an optimality gap of zero, without using dramatically more than two hours of data (Kapturowski et al., 2022).

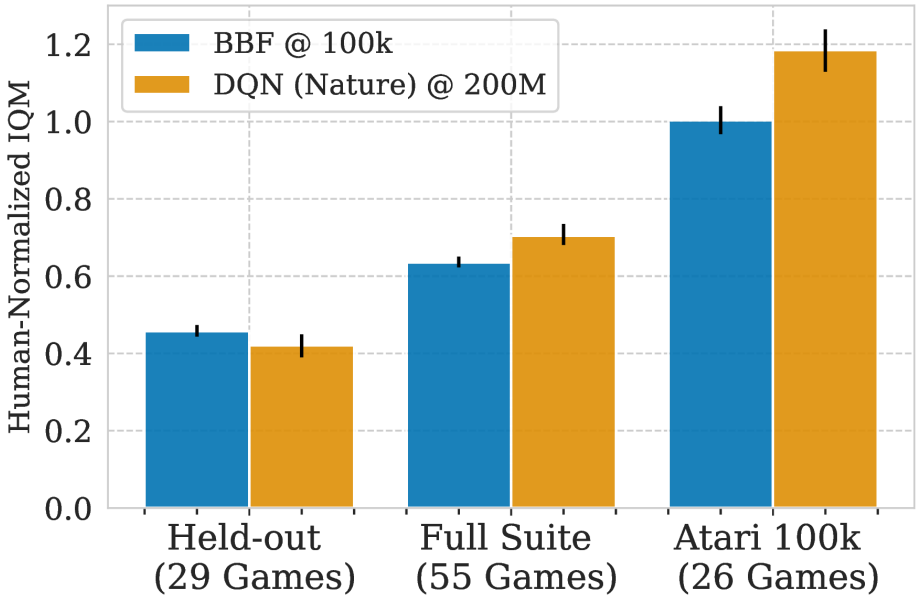


Figure 11. Comparing performance on the 29 unseen games to the 26 Atari 100k games. BBF trained with sticky actions at RR=8 for 100k steps approximately matches DQN (Nature) with 500 times more training data on each set. While we find that the 29 games not included in the Atari 100k setting are significantly harder than the 26 Atari 100k games, we see no evidence that BBF has overfitted to Atari 100k compared to DQN.

Overfitting on Atari 100K. Another important consideration is that the Atari 100K benchmark uses only 26 of the 55 games from the full ALE suite, and it does not include sticky actions[3] (Machado et al., 2018), which may make tasks significantly harder. Since we extensively benchmark BBF on Atari 100K, this raises the question of whether BBF works well on unseen Atari games and with sticky actions.

Fortunately, it does. In Figure 9, we compare the performance of BBF on all 55 games with sticky actions, and show that sticky action do not significantly harm performance. We do observe that the held-out games not included in the Atari 100k set are significantly more challenging than the 26 Atari 100k games (see Figure 11) – but this is even more true for baselines such as DQN (Nature) that did not use Atari 100k. Furthermore, as shown in Figure 8, we find that BBF's design choices generally provide even more benefit on these held-out games, possibly due to their increased difficulty.

New Frontiers. In fact, BBF works so well on the standard Atari setting that it is able to roughly match DQN's performance at 256 hours with only two hours of gameplay

[3]With 25% probability, the environment will execute the previous action again, instead of the agent's executed action.

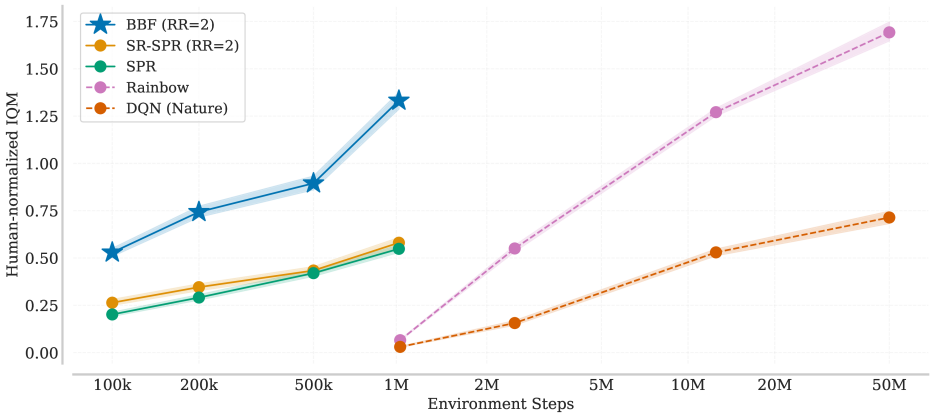


Figure 12. Learning curves for BBF, SR-SPR and SPR at replay ratio 2, measured via human-normalized IQM over 55 Atari games with sticky actions, as a function of number of environment interactions. Shaded regions show 95% CIs.

time (Figure 10). This suggests a clear new milestone for the community: can we match Rainbow's final performance with just two hours of gameplay? To facilitate future research toward this, we release scores on the set of 55 games with sticky actions, at various scales and replay ratios.

Data Scaling. Prior works have indicated that many sample-efficient RL algorithms plateau in performance when trained for longer than they were originally designed for (e.g., Agarwal et al., 2022). To examine this phenomenon, we train BBF, SPR and SR-SPR at replay ratio 2 out to one million environment steps (Figure 12), keeping all parameters unchanged (including conducting resets as normal past 100k steps). We observe that SPR and SR-SPR experience stagnating performance, with SR-SPR's advantage over SPR fading by 1M steps. BBF, however, remains consistently ahead of both, matching DQN's final performance before 200k environment steps and matching Rainbow's performance at 20M environment steps by 1M steps. We note that this experiment costs only 2.5 times more than training at replay ratio 8 to 100k steps, so we encourage other researchers to run similar experiments.

Additionally, we note in Figure 13 that it is possible to compare algorithms even with extremely small amounts of data, such as 20k or 50k steps, by which point BBF at replay ratio 2 (even with sticky actions enabled) outperforms most recently proposed algorithms (Robine et al., 2023; Micheli et al., 2023; Hafner et al., 2023), which did not use sticky actions. We thus suggest that compute-constrained groups consider this setting, as training BBF at replay ratio 2 for 40k environment steps takes only half of an A100 for 1 hour.

6.8. Discussion and Future Work

We introduced BBF, an algorithm that is able to achieve super-human level performance on the ALE with only 2-hours of gameplay. Although BBF is not the first to achieve this milestone, it is able to do so in a computationally efficient manner. Furthermore, BBF is able to better handle the scaling of networks and replay ratios, which are crucial for network expressivity and learning efficiency. Indeed, Figure 3 suggests that BFF is better-able to use over-parameterized networks than prior agents.

The techniques necessary to achieve this result invite a number of research questions for future work. Large replay ratios are a key element of BFF's performance, and the ability to scale them is due to the periodic resets incorporated into the algorithm. These resets are likely striking a favourable balance between catastrophic forgetting and network plasticity. An interesting avenue for future research is whether there are other mechanisms for striking this balance that perhaps are more targeted (e.g. not requiring resetting the full network, as was recently explored by Sokar et al. (2023)). We remarked on the fact that all the methods compared in Figure 2 use a form of self-supervision. Would other self-supervised losses (e.g. (Mazoure et al., 2020; Castro et al., 2021; Agarwal et al., 2021a)) produce similar results? Surprisingly, Li et al. (2022) argue that self-supervision from pixels does not improve performance; our results seem to contradict this finding.

Recent attention has shifted towards more realistic benchmarks (Fan et al., 2022) but such benchmarks exclude the majority of researchers outside certain resource-rich labs, and may require an alternative paradigm (Agarwal et al., 2022). One advantage of the Atari 100k benchmark is that, while still a challenging benchmark, it is relatively cheap compared to other benchmarks of similar complexity. However, despite its apparent saturation, scientific progress can still be made on this benchmark if we expand its scope. We hope our work provides a solid starting point for this.

Overall, we hope that our work inspires other researchers to continue pushing the frontier of sample efficiency in deep RL forward, to ultimately reach human-level performance across all tasks with human-level or superhuman efficiency.

Acknowledgements

Many thanks to Ross Goroshin, Georg Ostrovski, and Gopeshh Subbaraj for their feedback on an earlier draft of this paper. The authors would like to thank the

anonymous reviewers for useful discussions and feedback on this paper. We would also like to thank the Python community (Van Rossum et al., 1995; Oliphant, 2007) for developing tools that enabled this work, including NumPy (Harris et al., 2020), Matplotlib (Hunter, 2007) and JAX (Bradbury et al., 2018).

Societal impact

Although the work presented here is mostly academic, it aids in the development of more capable autonomous agents. While our contributions do not directly contribute to any negative societal impacts, we urge the community to consider these when building on our research.

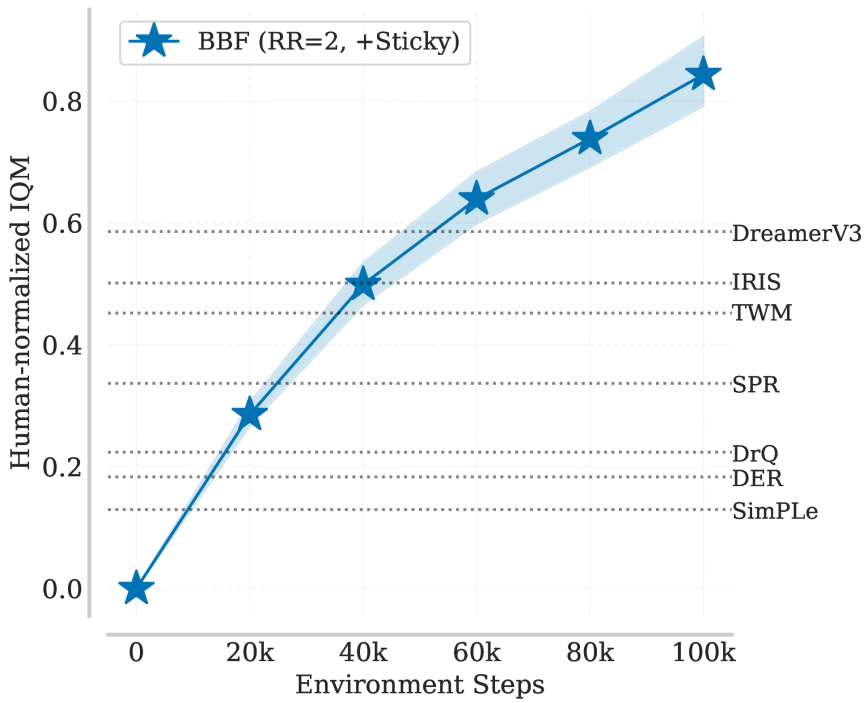


Figure 13. IQM Human-normalized learning curve for BBF at RR=2 with sticky actions on the 26 Atari 100k games, with final performances of many recent algorithms after they have trained for 100k steps. Even a weakened BBF outperforms all by 50k steps.

Chapter 7

Conclusion

In this thesis, we presented a series of works that more than doubled the state-of-the-art performance of sample-efficient RL agents. Two of these works, *Pretraining Representations for Data-Efficient Reinforcement Learning* and *The Primacy Bias in Deep Reinforcement Learning* were more exploratory, while *Sample-Efficient Reinforcement Learning by Breaking the Replay Ratio Barrier* and *Bigger, Better, Faster: Human-Level Atari with Human-Level Efficiency* represented a focused line of work on scaling up sample-efficient RL.

It is perhaps interesting to reflect on the state of RL when the work in this thesis began in 2020. The state-of-the-art on Atari 100k (which had just been introduced) was about 20% of human-normalized learning efficiency, which I was about to increase to 40% in my master's thesis (published academically as Schwarzer et al., 2021a). Within two years, through the works in this thesis, we were able to achieve fully super-human learning efficiency on these tasks, representing a surprising leap in capabilities.

This result is particularly remarkable given the (anecdotally widespread) perceptions in the ML community that (1) reinforcement learning essentially did not work and (2) exploration, pretraining and model-based learning were going to be key to fixing sample efficiency. While our work on the primacy bias did suggest a way in which RL was fundamentally "broken", it was surprisingly easy to fix with resets. Our works on scaling, meanwhile, represent proof that perception (2) was not correct, and suggest that the reason RL appeared not to work was simply that it was being deployed at too small of scales to show any particularly impressive learning behavior. In hindsight, I think it is clear that the field (to some extent also including myself) underestimated both the power of scaling and the surprisingly strong capabilities of pure model-free

RL, once properly scaled and tuned. If I were to re-do my PhD, I would have pivoted to thinking about scaling even earlier and even more exclusively than I actually did.

The other direction I would have wanted to investigate in more depth is fine-tuning pretrained models, even beyond the first work in this thesis, *Pretraining Representations for Data-Efficient Reinforcement Learning*. This is a vital topic for the future of sample-efficient RL, for it appears likely from progress in language modeling (Brown et al., 2020; OpenAI, 2023) that we are rapidly approaching the point where general-purpose models capable of flexible instruction following can be used for arbitrary tasks; it is now up to us to figure out how to fine-tune, adapt and leverage these models to advance the goals of reinforcement learning research. While *Bigger, Better, Faster* demonstrated that it is possible to stably perform value learning with larger models, and while *Breaking the Replay Ratio Barrier* showed that it is possible to leverage resets to efficiently fine-tune pretrained models, GPT-4 is (according to publicly-available non-OpenAI estimates) more than five orders of magnitude larger than the networks used by BBF. It is almost certain that the methods proposed here will not directly transfer to models of this size, but we may be optimistic that the ideas and research practices that gave rise to them may eventually guide us towards solutions that work with even AGI-scale models.

Références bibliographiques

[1] Alessandro Achille, Matteo Rovere, and Stefano Soatto. "Critical learning periods in deep networks." In: *International Conference on Learning Representations*. 2018.

[2] Rishabh Agarwal, Marlos C Machado, Pablo Samuel Castro, and Marc G Bellemare. "Contrastive behavioral similarity embeddings for generalization in reinforcement learning." In: *arXiv preprint arXiv:2101.05265* (2021).

[3] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. "An optimistic perspective on offline reinforcement learning." In: *International Conference on Machine Learning*. PMLR. 2020, pp. 104–114.

[4] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. "Deep reinforcement learning at the edge of the statistical precipice." In: *Advances in Neural Information Processing Systems* 34 (2021).

[5] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. "Reincarnating Reinforcement Learning: Reusing Prior Computation to Accelerate Progress." In: *Advances in Neural Information Processing Systems*. 2022.

[6] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. "Solving rubik's cube with a robot hand." In: *arXiv preprint arXiv:1910.07113* (2019).

[7] Ibrahim Alabdulmohsin, Hartmut Maennel, and Daniel Keysers. "The Impact of Reinitialization on Generalization in Convolutional Neural Networks." In: *arXiv preprint arXiv:2109.00267* (2021).

[8] Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. "Fixing a broken ELBO." In: *International Conference on Machine Learning*. 2018, pp. 159–168.

[9] Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. "Unsupervised state representation learning in atari." In: *NeurIPS*. 2019.

[10] Charles W Anderson et al. "Q-learning with hidden-unit restarting." In: *Advances in Neural Information Processing Systems* (1993), pp. 81–81.

[11] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. "Hindsight experience replay." In: *Advances in neural information processing systems*. 2017, pp. 5048–5058.

[12] Solomon E Asch. *Forming impressions of personality*. University of California Press, 1961.

[13] Jordan Ash and Ryan P Adams. "On warm-starting neural network training." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 3884–3894.

[14] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization." In: *arXiv preprint arXiv:1607.06450* (2016).

[15] Igor Babuschkin et al. *The DeepMind JAX Ecosystem*. 2020. URL: http://github.com/deepmind.

[16] Philip Bachman, R Devon Hjelm, and William Buchwalter. "Learning representations by maximizing mutual information across views." In: *NeurIPS*. 2019.

[17] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. "Agent57: Outperforming the atari human benchmark." In: *ICML*. 2020.

[18] Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. "Explaining neural scaling laws." In: *arXiv preprint arXiv:2102.06701* (2021).

[19] Jakob Bauer et al. "Human-Timescale Adaptation in an Open-Ended Task Space." In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, July 2023, pp. 1887–1935. URL: https://proceedings.mlr.press/v202/bauer23a.html.

[20] Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C Machado, Subhodeep Moitra, Sameera S Ponda, and Ziyu Wang. "Autonomous navigation of stratospheric balloons using reinforcement learning." In: *Nature* 588.7836 (2020), pp. 77–82.

[21] Marc G. Bellemare, Will Dabney, and R. Munos. "A Distributional Perspective on Reinforcement Learning." In: *ICML*. 2017.

[22] Marc G Bellemare, Will Dabney, and Rémi Munos. "A distributional perspective on reinforcement learning." In: *International Conference on Machine Learning*. PMLR. 2017, pp. 449–458.

[23] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. "The arcade learning environment: An evaluation platform for general agents." In: *Journal of Artificial Intelligence Research* 47 (2013).

[24] Emmanuel Bengio, Joelle Pineau, and Doina Precup. "Interference and Generalization in Temporal Difference Learning." In: *arXiv preprint arXiv:2003.06350* (2020).

[25] Yoshua Bengio. "Learning Deep Architectures for AI." In: *Machine Learning* 2.1 (2009), pp. 1–127.

[26] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. "Greedy layer-wise training of deep networks." In: *Advances in neural information processing systems*. 2007, pp. 153–160.

[27] Tudor Berariu, Wojciech Czarnecki, Soham De, Jorg Bornschein, Samuel Smith, Razvan Pascanu, and Claudia Clopath. "A study on the plasticity of neural networks." In: *arXiv preprint arXiv:2106.00042* (2021).

[28] Lilian Besson and Emilie Kaufmann. "What doubling tricks can and can't do for multi-armed bandits." In: *arXiv preprint arXiv:1803.06971* (2018).

[29] Johan Bjorck, Carla P Gomes, and Kilian Q Weinberger. "Is High Variance Unavoidable in RL? A Case Study in Continuous Control." In: *International Conference on Learning Representations*. 2022.

[30] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. "On the opportunities and risks of foundation models." In: *arXiv preprint arXiv:2108.07258* (2021).

[31] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.2.5. 2018. URL: http://github.com/google/jax.

[32] 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." In: *arXiv preprint arXiv:2005.14165* (2020).

[33] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A. Efros. *Large-Scale Study of Curiosity-Driven Learning*. 2018. arXiv: 1808.04355 [cs.LG].

[34] Víctor Campos, Pablo Sprechmann, Steven Stenberg Hansen, Andre Barreto, Charles Blundell, Alex Vitvitskyi, Steven Kapturowski, and Adria Puigdomenech Badia. *Coverage as a Principle for Discovering Transferable Behavior in Reinforcement Learning*. 2021. URL: https://openreview.net/forum?id=INhwJdJtxn6.

[35] Pablo Samuel Castro, Tyler Kastner, Prakash Panangaden, and Mark Rowland. "MICo: Improved representations via sampling-based state similarity for Markov decision processes." In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan. 2021. URL: https://openreview.net/forum?id=wFp6kmQELgu.

[36] Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G. Bellemare. "Dopamine: A Research Framework for Deep Reinforcement Learning." In: *CoRR* abs/1812.06110 (2018). arXiv: 1812.06110. URL: http://arxiv.org/abs/1812.06110.

[37] Johan Samir Obando Ceron and Pablo Samuel Castro. "Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research." In: *International Conference on Machine Learning*. PMLR. 2021, pp. 1373–1383.

[38] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. "Riemannian walk for incremental learning: Understanding forgetting and intransigence." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 532–547.

[39] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A simple framework for contrastive learning of visual representations." In: *ICML* (2020).

[40] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. "Big Self-Supervised Models are Strong Semi-Supervised Learners." In: *NeurIPS*. 2020.

[41] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. "Improved baselines with momentum contrastive learning." In: *arXiv preprint arXiv:2003.04297* (2020).

[42] Xinlei Chen and Kaiming He. "Exploring simple siamese representation learning." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 15750–15758.

[43] Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. "Randomized ensembled double q-learning: Learning fast without a model." In: *arXiv preprint arXiv:2101.05982* (2021).

[44] Brian Chu, Vashisht Madhavan, Oscar Beijbom, Judy Hoffman, and Trevor Darrell. "Best Practices for Fine-Tuning Visual Classifiers to New Domains." In: Nov. 2016, pp. 435–442. ISBN: 978-3-319-49408-1. DOI: 10.1007/978-3-319-49409-8_34.

[45] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. "Leveraging procedural generation to benchmark reinforcement learning." In: *International conference on machine learning*. PMLR. 2020, pp. 2048–2056.

[46] Pierluca D'Oro and Wojciech Jaśkowski. "How to Learn a Useful Critic? Model-based Action-Gradient-Estimator Policy Optimization." In: *Advances in Neural Information Processing Systems* 33 (2020).

[47] Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G. Bellemare, and Aaron Courville. "Sample-Efficient Reinforcement Learning by Breaking the Replay Ratio Barrier." In: *To appear in The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=OpC-9aBBVJe.

[48] 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." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 8. 2021, pp. 7160–7168.

[49] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. "Implicit quantile networks for distributional reinforcement learning." In: *International conference on machine learning*. PMLR. 2018, pp. 1096–1105.

[50] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. "Magnetic control of tokamak plasmas through deep reinforcement learning." In: *Nature* 602.7897 (2022), pp. 414–419.

[51] Jonas Degrave et al. “Magnetic control of tokamak plasmas through deep reinforcement learning.” In: *Nature* 602.7897 (2022), pp. 414–419. DOI: 10.1038/s41586-021-04301-9. URL: https://doi.org/10.1038/s41586-021-04301-9.

[52] Thomas Degris and Joseph Modayil. “Scaling-up knowledge for a cognizant robot.” In: *AAAI Spring Symposium on Designing Intelligent Robots: Reintegrating AI*. 2012.

[53] 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*. Ieee. 2009, pp. 248–255.

[54] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” In: *ACL*. 2019.

[55] Josip Djolonga, Jessica Yung, Michael Tschannen, Rob Romijnders, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Matthias Minderer, Alexander D’Amour, Dan Moldovan, et al. “On robustness and transferability of convolutional neural networks.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 16458–16468.

[56] Shibhansh Dohare, A Rupam Mahmood, and Richard S Sutton. “Continual Backprop: Stochastic Gradient Descent with Persistent Randomness.” In: *arXiv preprint arXiv:2108.06325* (2021).

[57] Shibhansh Dohare, Richard S. Sutton, and A. Rupam Mahmood. *Continual Backprop: Stochastic Gradient Descent with Persistent Randomness*. 2022. URL: https://openreview.net/forum?id=86sEVRfeGYS.

[58] Rachit Dubey, Pulkit Agrawal, Deepak Pathak, Tom Griffiths, and Alexei Efros. “Investigating Human Priors for Playing Video Games.” In: *International Conference on Machine Learning*. 2018.

[59] Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. “An empirical investigation of the challenges of real-world reinforcement learning.” In: (2020).

[60] Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. “Why does unsupervised pre-training help deep learning?” In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 201–208.

[61] Lasse Espeholt et al. “IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures.” In: *CoRR* abs/1802.01561 (2018). arXiv: 1802.01561. URL: http://arxiv.org/abs/1802.01561.

[62] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. “Diversity is All You Need: Learning Skills without a Reward Function.” In: *International Conference on Learning Representations*. 2018.

[63] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. “MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge.” In: *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2022.

[64] Amir Farahmand, Mohammad Ghavamzadeh, Shie Mannor, and Csaba Szepesvári. “Regularized policy iteration.” In: *Advances in Neural Information Processing Systems* 21 (2008).

[65] Jesse Farebrother, Joshua Greaves, Rishabh Agarwal, Charline Le Lan, Ross Goroshin, Pablo Samuel Castro, and Marc G Bellemare. “Proto-Value Networks: Scaling Representation Learning with Auxiliary Tasks.” In: *Submitted to The Eleventh International Conference on Learning Representations*. under review. 2023. URL: https://openreview.net/forum?id=oGDKSt9JrZi.

[66] William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. “Revisiting fundamentals of experience replay.” In: *International Conference on Machine Learning*. PMLR. 2020, pp. 3061–3071.

[67] Meire Fortunato et al. “Noisy Networks For Exploration.” In: *ICLR*. 2018. URL: https://openreview.net/forum?id=rywHCPkAW.

[68] Vincent François-Lavet, Raphael Fonteneau, and Damien Ernst. “How to discount deep reinforcement learning: Towards new dynamic strategies.” In: *arXiv preprint arXiv:1512.02011* (2015).

[69] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, and Joelle Pineau. “An introduction to deep reinforcement learning.” In: *arXiv preprint arXiv:1811.12560* (2018).

[70] Robert M French. “Catastrophic forgetting in connectionist networks.” In: *Trends in cognitive sciences* 3.4 (1999), pp. 128–135.

[71] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. "D4rl: Datasets for deep data-driven reinforcement learning." In: *arXiv preprint arXiv:2004.07219* (2020).

[72] Scott Fujimoto, Herke Hoof, and David Meger. "Addressing function approximation error in actor-critic methods." In: *International conference on machine learning*. PMLR. 2018, pp. 1587–1596.

[73] Scott Fujimoto, David Meger, and Doina Precup. "Off-policy deep reinforcement learning without exploration." In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2052–2062.

[74] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. "Deepmdp: Learning continuous latent space models for representation learning." In: *ICML* (2019).

[75] Mohammad Ghavamzadeh, Hilbert Kappen, Mohammad Azar, and Rémi Munos. "Speedy Q-Learning." In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger. Vol. 24. Curran Associates, Inc., 2011. URL: https://proceedings.neurips.cc/paper/2011/file/ab1a4d0dd4d48a2ba1077c4494791306-Paper.pdf.

[76] Raphael Gontijo-Lopes, Sylvia J Smullin, Ekin D Cubuk, and Ethan Dyer. "Affinity and Diversity: Quantifying Mechanisms of Data Augmentation." In: *arXiv preprint arXiv:2002.08973* (2020).

[77] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[78] 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." In: *NeurIPS*. 2020.

[79] 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." In: *ICML*. 2020.

[80] Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Bernardo A Pires, and Rémi Munos. "Neural predictive belief representations." In: *ICML* (2018).

[81] 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. 2010, pp. 297–304.

[82] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor." In: *ICML*. 2018.

[83] Raia Hadsell, Dushyant Rao, Andrei A. Rusu, and Razvan Pascanu. "Embracing Change: Continual Learning in Deep Neural Networks." In: *Trends in Cognitive Sciences* (2020). DOI: https://doi.org/10.1016/j.tics.2020.09.004. URL: http://www.sciencedirect.com/science/article/pii/S1364661320302199.

[84] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. "Dream to control: Learning behaviors by latent imagination." In: *ICLR* (2020).

[85] 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.

[86] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. "Mastering atari with discrete world models." In: *arXiv preprint arXiv:2010.02193* (2020).

[87] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. *Mastering Diverse Domains through World Models*. 2023. DOI: 10.48550/ARXIV.2301.04104. URL: https://arxiv.org/abs/2301.04104.

[88] Jessica B Hamrick, Abram L. Friesen, Feryal Behbahani, Arthur Guez, Fabio Viola, Sims Witherspoon, Thomas Anthony, Lars Holger Buesing, Petar Veličković, and Theophane Weber. "On the role of planning in model-based deep reinforcement learning." In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=IrM64DGB21.

[89] Steven Hansen, Will Dabney, Andre Barreto, David Warde-Farley, Tom Van de Wiele, and Volodymyr Mnih. "Fast Task Inference with Variational Intrinsic Successor Features." In: *International Conference on Learning Representations*. 2020.

[90] Charles R. Harris et al. "Array programming with NumPy." In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[91] Hado P van Hasselt, Matteo Hessel, and John Aslanides. "When to use parametric models in reinforcement learning?" In: *NeurIPS*. 2019.

[92] Hado van Hasselt, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-learning." In: *Proceedings of the Thirthieth AAAI Conference On Artificial Intelligence (AAAI), 2016*. cite arxiv:1509.06461Comment: AAAI 2016. 2016.

[93] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. "Momentum contrast for unsupervised visual representation learning." In: *CVPR*. 2020.

[94] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In: *CVPR*. 2016.

[95] 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." In: *arXiv preprint arXiv:1905.09272* (2019).

[96] Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. *Scaling Laws for Transfer*. 2021. arXiv: 2102.01293 [cs.LG].

[97] Matteo Hessel, Ivo Danihelka, Fabio Viola, Arthur Guez, Simon Schmitt, Laurent Sifre, Theophane Weber, David Silver, and Hado van Hasselt. "Muesli: Combining improvements in policy optimization." In: *Proceedings of the 38th International Conference on Machine Learning* (2021).

[98] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. "Rainbow: Combining improvements in deep reinforcement learning." In: *Thirty-second AAAI conference on artificial intelligence*. 2018.

[99] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory F. Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. "Deep Learning Scaling is Predictable, Empirically." In: *CoRR* abs/1712.00409 (2017). arXiv: 1712.00409. URL: http://arxiv.org/abs/1712.00409.

[100] 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." In: (2016).

[101] Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. "Dropout Q-Functions for Doubly Efficient Reinforcement Learning." In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=xCVJMsPv3RT.

[102] 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." In: *ICLR* (2019).

[103] Jordan Hoffmann et al. "Training Compute-Optimal Large Language Models." In: *ArXiv* abs/2203.15556 (2022).

[104] G Zacharias Holland, Erin J Talvitie, and Michael Bowling. "The effect of planning shape on dyna-style planning in high-dimensional state spaces." In: *arXiv preprint arXiv:1806.01825* (2018).

[105] John D Hunter. "Matplotlib: A 2D graphics environment." In: *IEEE Annals of the History of Computing* 9.03 (2007), pp. 90–95.

[106] Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. "Transient Non-stationarity and Generalisation in Deep Reinforcement Learning." In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=Qun8fv4qSby.

[107] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." In: *ICML*. 2015.

[108] Jacqueline S Johnson and Elissa L Newport. "Critical period effects in second language learning: The influence of maturational state on the acquisition of English as a second language." In: *Cognitive psychology* 21.1 (1989), pp. 60–99.

[109] Eric Jones, Travis Oliphant, and Pearu Peterson. *SciPy: Open source scientific tools for Python*. 2014.

[110] Lukasz 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.

[111] Lukasz Kaiser et al. "Model-Based Reinforcement Learning for Atari." In: *ArXiv* abs/1903.00374 (2020).

[112] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. "Scaling laws for neural language models." In: *arXiv preprint arXiv:2001.08361* (2020).

[113] Steven Kapturowski, Víctor Campos, Ray Jiang, Nemanja Rakićević, Hado van Hasselt, Charles Blundell, and Adrià Puigdomènech Badia. "Human-level Atari 200x faster." In: *arXiv preprint arXiv:2209.07550* (2022).

[114] Michael J Kearns and Satinder Singh. "Bias-Variance Error Bounds for Temporal Difference Updates." In: *COLT*. 2000, pp. 142–147.

[115] Kacper Piotr Kielak. *Do recent advancements in model-based deep reinforcement learning really improve data efficiency?* 2020. URL: https://openreview.net/forum?id=Bke9u1HFwB.

[116] Diederik P Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *ICLR (Poster)*. 2015.

[117] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes." In: (2013).

[118] Simon Kirby. "Spontaneous evolution of linguistic structure-an iterated learning model of the emergence of regularity and irregularity." In: *IEEE Transactions on Evolutionary Computation* 5.2 (2001), pp. 102–110.

[119] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. "A survey of generalisation in deep reinforcement learning." In: *arXiv preprint arXiv:2111.09794* (2021).

[120] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. *Jupyter Notebooks-a publishing format for reproducible computational workflows.* Vol. 2016. 2016.

[121] Christof Koch. *Biophysics of computation: information processing in single neurons*. Oxford university press, 2004.

[122] Ilya Kostrikov. *JAXRL: Implementations of Reinforcement Learning algorithms in JAX*. Oct. 2021. DOI: 10.5281/zenodo.5535154. URL: https://github.com/ikostrikov/jaxrl.

[123] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. "Offline Reinforcement Learning with Implicit Q-Learning." In: *ArXiv* abs/2110.06169 (2022).

[124] Ilya Kostrikov*, Denis Yarats*, and Rob Fergus. "Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels." In: *International Conference on Learning Representations*. 2021.

[125] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[126] Aviral Kumar, Rishabh Agarwal, Xinyang Geng, George Tucker, and Sergey Levine. *Offline Q-Learning on Diverse Multi-Task Data Both Scales And Generalizes*. 2022. DOI: 10.48550/ARXIV.2211.15144. URL: https://arxiv.org/abs/2211.15144.

[127] Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. "Implicit Under-Parameterization Inhibits Data-Efficient Deep Reinforcement Learning." In: *International Conference on Learning Representations*. 2020.

[128] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. "Conservative q-learning for offline reinforcement learning." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 1179–1191.

[129] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. "Building machines that learn and think like people." In: *Behavioral and brain sciences* 40 (2017).

[130] Romain Laroche and Remi Tachet Des Combes. "Beyond the Policy Gradient Theorem for Efficient Policy Updates in Actor-Critic Algorithms." In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 5658–5688.

[131] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. "Reinforcement Learning with Augmented Data." In: *NeurIPS*. 2020.

[132] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. "Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model." In: *arXiv preprint arXiv:1907.00953* (2019).

[133] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. "Learning quadrupedal locomotion over challenging terrain." In: *Science robotics* 5.47 (2020), eabc5986.

[134] Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. "Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning." In: *International Conference on Machine Learning* (2021).

[135] Kuang-Huei Lee et al. "Multi-Game Decision Transformers." In: *ArXiv* abs/2205.15241 (2022).

[136] Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Franois Goudou, and David Filliat. "State representation learning for control: An overview." In: *Neural Networks* 108 (2018).

[137] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems." In: *CoRR* abs/2005.01643 (2020). arXiv: 2005.01643. URL: https://arxiv.org/abs/2005.01643.

[138] Chris Junchi Li, Yaodong Yu, Nicolas Loizou, Gauthier Gidel, Yi Ma, Nicolas Le Roux, and Michael I Jordan. "On the convergence of stochastic extragradient for bilinear games with restarted iteration averaging." In: *arXiv preprint arXiv:2107.00464* (2021).

[139] Hao Li, Pratik Chaudhari, Hao Yang, Michael Lam, Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. "Rethinking the Hyperparameters for Fine-tuning." In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=B1g8VkHFPH.

[140] Xiang Li, Jinghuan Shang, Srijan Das, and Michael S Ryoo. "Does Self-supervised Learning Really Improve Reinforcement Learning from Pixels?" In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho. 2022. URL: https://openreview.net/forum?id=fVslVNBfjd8.

[141] 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.

[142] Long-Ji Lin. *Reinforcement learning for robots using neural networks*. Carnegie Mellon University, 1992.

[143] Hao Liu and Pieter Abbeel. "Aps: Active pretraining with successor features." In: *International Conference on Machine Learning*. PMLR. 2021, pp. 6736–6747.

[144] Hao Liu and Pieter Abbeel. *Unsupervised Active Pre-Training for Reinforcement Learning*. 2021. URL: https://openreview.net/forum?id=cvNYovr16SB.

[145] Zhuang Liu, Xuanlin Li, Bingyi Kang, and Trevor Darrell. "Regularization Matters in Policy Optimization - An Empirical Study on Continuous Control." In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=yr1mzrH3IC.

[146] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization." In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

[147] William Lotter, Gabriel Kreiman, and David Cox. “Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning.” In: (2016).

[148] Clare Lyle, Mark Rowland, and Will Dabney. “Understanding and Preventing Capacity Loss in Reinforcement Learning.” In: *arXiv preprint arXiv:2204.09560* (2022).

[149] Clare Lyle, Mark Rowland, Will Dabney, Marta Z. Kwiatkowska, and Yarin Gal. “Learning Dynamics and Generalization in Reinforcement Learning.” In: *ArXiv* abs/2206.02126 (2022).

[150] Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. “Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents.” In: *J. Artif. Int. Res.* 61.1 (2018), 523–562. ISSN: 1076-9757.

[151] Philip H Marshall and Pamela R Werder. “The effects of the elimination of rehearsal on primacy and recency.” In: *Journal of Verbal Learning and Verbal Behavior* 11.5 (1972), pp. 649–653.

[152] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. “Deployment-Efficient Reinforcement Learning via Model-Based Offline Optimization.” In: *International Conference on Learning Representations*. 2021.

[153] Bogdan Mazoure, Remi Tachet des Combes, Thang Long Doan, Philip Bachman, and R Devon Hjelm. “Deep Reinforcement and InfoMax Learning.” In: *Advances in Neural Information Processing Systems* 33 (2020).

[154] Wes McKinney. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O’Reilly Media, Inc.", 2012.

[155] Vincent Micheli, Eloi Alonso, and François Fleuret. “Transformers are Sample-Efficient World Models.” In: *ICLR*. 2023. URL: https://openreview.net/forum?id=vhFu1Acb0xb.

[156] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. “Playing Atari with deep reinforcement learning.” In: *arXiv:1312.5602* (2013).

[157] 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.” In: *Nature* 518.7540 (2015).

[158] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines." In: *ICML*. 2010.

[159] Andrew Y Ng, Daishi Harada, and Stuart Russell. "Policy invariance under reward transformations: Theory and application to reward shaping." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 1999.

[160] Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. "The Primacy Bias in Deep Reinforcement Learning." In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 16828–16847.

[161] Travis E Oliphant. *A guide to NumPy*. Vol. 1. Trelgol Publishing USA, 2006.

[162] Travis E Oliphant. "Python for scientific computing." In: *Computing in Science & Engineering* 9.3 (2007), pp. 10–20.

[163] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding." In: *arXiv preprint arXiv:1807.03748* (2018).

[164] OpenAI et al. "Dota 2 with Large Scale Deep Reinforcement Learning." In: *arXiv preprint arXiv:1912.06680* (2019). URL: https://arxiv.org/abs/1912.06680.

[165] R OpenAI. "GPT-4 technical report." In: *arXiv* (2023), pp. 2303–08774.

[166] Georg Ostrovski, Pablo Samuel Castro, and Will Dabney. "The Difficulty of Passive Learning in Deep Reinforcement Learning." In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan. 2021. URL: https://openreview.net/forum?id=nPHA8fGicZk.

[167] Kei Ota, Devesh K Jha, and Asako Kanezaki. "Training larger networks for deep reinforcement learning." In: *arXiv preprint arXiv:2102.07920* (2021).

[168] Long Ouyang et al. "Training language models to follow instructions with human feedback." In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho. 2022. URL: https://openreview.net/forum?id=TG8KACxEON.

[169] 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.

[170] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. "Curiosity-driven exploration by self-supervised prediction." In: *ICML*. PMLR. 2017.

[171] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. "Film: Visual reasoning with a general conditioning layer." In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[172] Boris T Polyak and Anatoli B Juditsky. "Acceleration of stochastic approximation by averaging." In: *SIAM journal on control and optimization* 30.4 (1992), pp. 838–855.

[173] Mariya Popova, Olexandr Isayev, and Alexander Tropsha. "Deep reinforcement learning for de novo drug design." In: *Science advances* 4.7 (2018), eaap7885.

[174] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[175] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. *Improving language understanding by generative pre-training*.

[176] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. "Semi-supervised learning with ladder networks." In: *Advances in neural information processing systems*. 2015, pp. 3546–3554.

[177] Yi Ren, Shangmin Guo, Matthieu Labeau, Shay B Cohen, and Simon Kirby. "Compositional languages emerge in a neural iterated learning model." In: *International Conference on Learning Representations*. 2019.

[178] Martin A. Riedmiller, Jost Tobias Springenberg, Roland Hafner, and Nicolas Manfred Otto Heess. "Collect & Infer - a fresh look at data-efficient Reinforcement Learning." In: *CoRL*. 2021.

[179] Martin Riedmiller. "Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method." In: *European Conference on Machine Learning*. 2005.

[180] Jan Robine, Marc Höftmann, Tobias Uelwer, and Stefan Harmeling. "Transformer-based World Models Are Happy With 100k Interactions." In: *arXiv preprint arXiv:2303.07109* (2023).

[181] Anthony Robins. "Catastrophic forgetting in neural networks: the role of rehearsal mechanisms." In: *Proceedings 1993 The First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*. IEEE. 1993, pp. 65–68.

[182] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. "Mobilenetv2: Inverted residuals and linear bottlenecks." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[183] Tom Schaul, André Barreto, John Quan, and Georg Ostrovski. "The Phenomenon of Policy Churn." In: *Advances in Neural Information Processing Systems* (2022).

[184] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. "Prioritized Experience Replay." In: *International Conference on Learning Representations*. 2016.

[185] Dominik Schmidt and Thomas Schmied. "Fast and data-efficient training of rainbow: an experimental study on atari." In: *arXiv preprint arXiv:2111.10247* (2021).

[186] 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." In: *Nature* 588.7839 (2020), pp. 604–609.

[187] Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. "Online and offline reinforcement learning by planning with a learned model." In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 27580–27591.

[188] Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. "Data-Efficient Reinforcement Learning with Self-Predictive Representations." In: *International Conference on Learning Representations*. 2021.

[189] Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron Courville. "Pretraining Representations for Data-Efficient Reinforcement Learning." In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan. 2021. URL: https://openreview.net/forum?id=XpSAvlvnMa.

[190] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. "Planning to explore via self-supervised world models." In: *ICML*. 2020.

[191] Noel E Sharkey and Amanda JC Sharkey. "An analysis of catastrophic interference." In: *Connection Science* (1995).

[192] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. "Dynamics-Aware Unsupervised Discovery of Skills." In: *International Conference on Learning Representations*. 2019.

[193] Hanan Shteingart, Tal Neiman, and Yonatan Loewenstein. "The role of first impression in operant learning." In: *Journal of Experimental Psychology: General* 142.2 (2013), p. 476.

[194] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. "Mastering the game of Go with deep neural networks and tree search." In: *nature* 529.7587 (2016), pp. 484–489.

[195] David Silver et al. "Mastering the game of Go with deep neural networks and tree search." In: *Nature* 529 (2016), pp. 484–503. URL: http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

[196] Patrice Y Simard, David Steinkraus, John C Platt, et al. "Best practices for convolutional neural networks applied to visual document analysis." In: *Icdar*. Vol. 3. 2003.

[197] Samarth Sinha, Homanga Bharadhwaj, Aravind Srinivas, and Animesh Garg. "D2RL: Deep Dense Architectures in Reinforcement Learning." In: *arXiv preprint arXiv:2010.09163* (2020).

[198] Laura Smith, Ilya Kostrikov, and Sergey Levine. "A Walk in the Park: Learning to Walk in 20 Minutes With Model-Free Reinforcement Learning." In: *ArXiv* abs/2208.07860 (2022).

[199] 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." In: *arXiv preprint arXiv:2001.07685* (2020).

[200] Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. "The Dormant Neuron Phenomenon in Deep Reinforcement Learning." In: *ICML*. 2023.

[201] Xingyou Song, Yiding Jiang, Yilun Du, and Behnam Neyshabur. "Observational Overfitting in Reinforcement Learning." In: *The International Conference on Learning Representations (ICLR)*. 2019.

[202] Aravind Srinivas, Michael Laskin, and Pieter Abbeel. "Curl: Contrastive unsupervised representations for reinforcement learning." In: *ICML*. 2020.

[203] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." In: *jmlr* (2014).

[204] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Highway networks." In: *arXiv preprint arXiv:1505.00387* (2015).

[205] Adam Stooke and Pieter Abbeel. "rlpyt: A research code base for deep reinforcement learning in pytorch." In: *arXiv preprint arXiv:1909.01500* (2019).

[206] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. "Decoupling Representation Learning from Reinforcement Learning." In: *arXiv preprint arXiv:2009.08319* (2020).

[207] Richard S Sutton. "Learning to predict by the methods of temporal differences." In: *Machine learning* 3.1 (1988), pp. 9–44.

[208] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. 2nd. MIT Press, 2018.

[209] Ahmed Taha, Abhinav Shrivastava, and Larry S Davis. "Knowledge Evolution in Neural Networks." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 12843–12852.

[210] Adrien Ali Taiga, Rishabh Agarwal, Jesse Farebrother, Aaron Courville, and Marc G Bellemare. "Investigating Multi-task Pretraining and Generalization in Reinforcement Learning." In: *Submitted to The Eleventh International Conference on Learning Representations*. under review. 2023. URL: https://openreview.net/forum?id=sSt9fROSZRO.

[211] Mingxing Tan and Quoc Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." In: *International Conference on Machine Learning*. 2019, pp. 6105–6114.

[212] Antti Tarvainen and Harri Valpola. "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results." In: *NeurIPS*. 2017.

[213] Yuval Tassa, Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, and Nicolas Heess. *dm_control: Software and Tasks for Continuous Control*. 2020. arXiv: 2006.12983 [cs.RO].

[214] Yee Whye Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. "Distral: Robust multitask reinforcement learning." In: *NIPS*. 2017.

[215] Gerald Tesauro. "Practical issues in temporal difference learning." In: *Advances in neural information processing systems*. 1992, pp. 259–266.

[216] 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*. 2012, pp. 5026–5033.

[217] 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.

[218] Michael Tschannen, Josip Djolonga, Paul K Rubenstein, Sylvain Gelly, and Mario Lucic. "On mutual information maximization for representation learning." In: *arXiv preprint arXiv:1907.13625* (2019).

[219] Pedro Tsividis, Thomas Pouncy, Jaqueline L. Xu, Joshua B. Tenenbaum, and Samuel J. Gershman. "Human Learning in Atari." In: *AAAI Spring Symposia*. 2017.

[220] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. "The NumPy array: a structure for efficient numerical computation." In: *Computing in science & engineering* 13.2 (2011), pp. 22–30.

[221] Hado Van Hasselt, Arthur Guez, and David Silver. "Deep reinforcement learning with double Q-learning." In: *aaai* (2016).

[222] Hado Van Hasselt, Arthur Guez, and David Silver. "Deep reinforcement learning with double q-learning." In: *AAAI*. 2016.

[223] Guido Van Rossum and Fred L Drake Jr. *Python tutorial*. Vol. 620. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[224] Harm Vanseijen and Rich Sutton. "A Deeper Look at Planning as Learning from Replay." In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 2314–2322. URL: https://proceedings.mlr.press/v37/vanseijen15.html.

[225] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[226] 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*. 2019, pp. 6428–6437.

[227] 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*. 2008, pp. 1096–1103.

[228] 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." In: *Nature* (2019).

[229] Che Wang, Yanqiu Wu, Quan Vuong, and Keith Ross. "Striving for simplicity and performance in off-policy DRL: Output normalization and non-uniform sampling." In: *International Conference on Machine Learning*. PMLR. 2020, pp. 10070–10080.

[230] Tongzhou Wang and Phillip Isola. *Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere*. 2020. arXiv: 2005.10242 [cs.LG].

[231] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. "Sample efficient actor-critic with experience replay." In: *arXiv preprint arXiv:1611.01224* (2016).

[232] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. "Dueling Network Architectures for Deep Reinforcement Learning." In: *Proceedings of the 33rd International Conference on Machine Learning*. Vol. 48. 2016, pp. 1995–2003.

[233] Yanqiu Wu, Xinyue Chen, Che Wang, Yiming Zhang, Zijian Zhou, and Keith W. Ross. *Aggressive Q-Learning with Ensembles: Achieving Both High Sample Efficiency and High Asymptotic Performance*. 2022. URL: https://openreview.net/forum?id=NOApNZTiTNU.

[234] 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*. 2020, pp. 10687–10698.

[235] 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*. 1997, pp. 807–816.

[236] Annik Yalnizyan-Carson and Blake A Richards. "Forgetting Enhances Episodic Control with Structured Memories." In: *bioRxiv* (2021).

[237] Mengjiao Yang and Ofir Nachum. "Representation matters: Offline pretraining for sequential decision making." In: *arXiv preprint arXiv:2102.05815* (2021).

[238] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. "Reinforcement learning with prototypical representations." In: *arXiv preprint arXiv:2102.11271* (2021).

[239] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. "Improving Sample Efficiency in Model-Free Reinforcement Learning from Images." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 12. 2021, pp. 10674–10681.

[240] Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. "Mastering atari games with limited data." In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 25476–25488.

[241] Friedemann Zenke, Ben Poole, and Surya Ganguli. "Continual learning through synaptic intelligence." In: *ICML*. 2017.

[242] Albert Zhan, Philip Zhao, Lerrel Pinto, Pieter Abbeel, and Michael Laskin. *A Framework for Efficient Robotic Manipulation*. 2020. arXiv: 2012.07975 [cs.RO].

[243] Chiyuan Zhang, Samy Bengio, and Yoram Singer. "Are all layers created equal?" In: *arXiv preprint arXiv:1902.01996* (2019).

[244] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. "A study on overfitting in deep reinforcement learning." In: *arXiv preprint arXiv:1804.06893* (2018).

[245] Hattie Zhou, Ankit Vani, Hugo Larochelle, and Aaron Courville. "Fortuitous Forgetting in Connectionist Networks." In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=ei3SY1_zYsE.

Annexe A

Appendix for Chapter 3

A.1. Implementation Details

We base our work on the code released for SPR (Schwarzer et al., 2021a), which in turn is based on rlpyt (Stooke et al., 2019), and makes use of NumPy (Harris et al., 2020) and PyTorch (Paszke et al., 2019).

A.1.1. Training

We set $\lambda^{\text{SPR}} = 2$ and $\lambda^{\text{IM}} = 1$ during pre-training. Unless otherwise noted, all settings match SPR during fine-tuning, including batch size, replay ratio, target network update period, and λ^{SPR}. We use a batch size of 256 during pre-training to maximize throughput, and update both the SPR and goal-conditioned RL target network target networks with an exponential moving average with $\tau = 0.99$. We pre-train for a number of gradient steps equivalent to 10 epochs over 6M samples, no matter the amount of data used. Due to the cost of pretraining, we pre-train a single encoder per game for each configuration tested. However, we use 10 random seeds at fine-tuning time, allowing us to average over variance due to exploration and data order. Finally, we reduce fine-tuning learning rates for pretrained encoders and dynamics models by a factor of 100, and by a factor of 3 for other pretrained weights. We find this crucial to SGI's performance, and discuss it in detail in Section 3.6.6.

We trained SGI on standard GPUs, including V100s and P100s. We found that pretraining took roughly one to three days and finetuning between four and 12 hours per run on a single GPU, depending on the size of the network used and type of GPU.

A.1.2. Goal-Conditioned Reinforcement Learning

We generate goals in a three-stage process: a goal g for state s_t is initially chosen to be the target representation of a state sampled uniformly from the near future, $g \leftarrow \tilde{z}_{t+i}, i \sim \text{Uniform}(50)$, before being combined with a normalized vector of isotropic Gaussian noise n as $g \leftarrow \alpha n + (1-\alpha)g$, where $\alpha \sim \text{Uniform}(0, 0.5)$. Finally, we exchange goal vectors between states in the minibatch with probability 0.2, to ensure that some goals correspond to states reached in entirely different trajectories.

In defining our synthetic goal-conditioned rewards, we take inspiration from potential-based reward shaping (Ng et al., 1999). Using the target representations $\tilde{z}_t \triangleq f_m(s_t)$

and $\tilde{z}_{t+1} \triangleq f_m(s_{t+1})$, we define the reward as follows:

$$R(\tilde{z}_t, \tilde{z}_{t+1}, g) = d(\tilde{z}_t, g) - d(\tilde{z}_{t+1}, g) \tag{A.1.1}$$

$$d(\tilde{z}_t, g) = \exp\left(2\frac{\tilde{z}_t \cdot g}{||\tilde{z}||_2 \cdot ||g||_2} - 2\right). \tag{A.1.2}$$

As this reward function depends on the target encoder f_m, it changes throughout training, although using the slower-moving f_m rather than the online encoder f_o may provide some measure of stability. Like SPR, however, this objective is technically vulnerable to collapse. If all representations $\tilde{z}_t$ collapse to a single constant vector then all rewards will be 0, allowing the task to be trivially solved.

We estimate $Q(s_t, a_t, g)$ using FiLM (Perez et al., 2018) to condition the DQN on the goal g, which we found to be more robust than simple concatenation. A FiLM generator j produces per-channel biases β_c and scales γ_c, which then modulate features through a per-channel affine transformation:

$$\text{FiLM}(F_c|\gamma_c, \beta_c) = \gamma_c F_c + \beta_c \tag{A.1.3}$$

We use these parameters to replace the learned per-channel affine transformation in a layer norm layer (Ba et al., 2016), which we insert immediately prior to the final linear layer in the DQN head.

We apply FiLM after the first layer in the DQN's MLP head. We parameterize our FiLM generator j as a small convolutional network, which takes the goal g (viewed as a $64 \times 7 \times 7$ spatial feature map) as input and applies two 128-channel convolutions followed by a flatten and linear layer to produce the FiLM parameters γ and β.

A.1.3. Model Architectures

In addition to the standard three-layer CNN encoder introduced by (Mnih et al., 2015), we experiment with larger residual networks (He et al., 2016). We use the design proposed by Espeholt et al. (2018) as a starting point, while still adopting innovations used in more modern architectures such as EfficientNets (Tan et al., 2019) and MobileNetv2 (Sandler et al., 2018). In particular, we use inverted residual blocks with an expansion ratio of 2, and batch normalization (Ioffe et al., 2015) after each convolutional layer. We use three groups of three residual blocks with 32, 64 and 64 channels each, downscaling by a factor of three in the first group and two in each successive group. This yields a final representation of shape $64 \times 7 \times 7$ when applied to 84×84-dimensional Atari frames, identical to that of the standard CNN encoder.

In our scaling experiment with a larger network, we increase to five blocks per group, with 48, 96 and 96 channels in each group, as well as using a larger expansion ratio of 4, producing a representation of shape $96 \times 7 \times 7$. This enlargement increases the number of parameters by roughly a factor of 5. Finally, our DQN head has 512 hidden units, as opposed to 256 in SPR.

A.1.4. Image Augmentation

We use the same image augmentations as used in SPR (Schwarzer et al., 2021a), which itself used the augmentations used in DrQ (Kostrikov* et al., 2021), in all experiments, including during both pretraining and fine-tuning. Specifically, we employ random crops (4 pixel padding and 84x84 crops) in combination with image intensity jittering.

A.1.5. Experiments with ATC

As ATC (Stooke et al., 2020) was not tested on the Atari100k setting, and as its hyperparameters (including network size and fine-tuning scheme) are very different from those used by SGI, we modify its code[1] to allow it to be fairly compared to SGI. We replace the convolutional encoder with that used by SGI, and use the same optimizer settings, image augmentation, pre-training data, and number of pre-training epochs as in SGI. However, we retain ATC's mini-batch structure (i.e., sampling 32 subsequences of eight consecutive time steps, for a total batch size of 512), as this structure defines the negative samples used by ATC's InfoNCE loss. During fine-tuning, we transfer the ATC projection head to the first layer of the DQN MLP head, as in SPR; we otherwise fine-tune identically to SGI, including using SPR.

[1] https://github.com/astooke/rlpyt/tree/master/rlpyt/ul

A.2. Pseudocode

Algorithm 1: Pre-Training with SGI

Denote parameters of online encoder f_o, projection p_o and Q-learning head as θ_o;
Denote parameters of target encoder f_m, projection p_m and Q-learning target head as θ_m;
Denote parameters of transition model h, predictor q, inverse model I as ϕ;
Denote the maximum prediction depth as K, batch size as N;
Denote distance function in goal RL reward as d;
initialize offline dataset D;
while *Training* **do**
  sample a minibatch of sequences of $(s_t, a, s+t+1) \sim D$; `// sample unlabeled data`
  `/* sample goals */`
  for i *in range*$(0, N)$ **do**
    $s^i \leftarrow \text{augment}(s^i); s'^i \leftarrow \text{augment}(s'^i)$; `// augment input images`
    $j \sim$ Discrete Uniform$(1, 50)$; `// sample hindsight goal states`
    $g^i \leftarrow f_m(s^n_j)$; `// encode goal states`
    $\alpha \sim \text{Uniform}(0, 0.5)$, $n \sim \text{Normal}(0, 1)$; `// sample noise parameters`
    $g^i \leftarrow \alpha g^i + (1-\alpha)n$; `// apply noise`
    `/* Permute to make some goals very challenging to reach */`
    permute $\sim$ Bernoulli(0.2)
    if *permute* **then**
      $j \sim$ Discrete Uniform(N)
      $g^i \leftarrow g^j$; `// permute goal`
    end
  end
  `/* compute SGI loss */`
  for i *in range*$(0, N)$ **do**
    $\hat{z}^i_0 \leftarrow f_\theta(s^i_0)$; `// compute online representations`
    $l^i \leftarrow 0$;
    `/* compute SPR loss */`
    for k *in (1, . . . , K)* **do**
      $\hat{z}^i_k \leftarrow h(\hat{z}^i_{k-1}, a^i_{k-1})$; `// latent states via transition model`
      $\tilde{z}^i_k \leftarrow f_m(s^i_k)$; `// target representations`
      $\hat{y}^i_k \leftarrow q(p_o(\hat{z}^i_k))$, $\tilde{y}^i_k \leftarrow g_m(\tilde{z}^i_k)$; `// projections`
      $l^i \leftarrow l^i - \lambda^{\text{SPR}} \left(\frac{\tilde{y}^i_k}{||\tilde{y}^i_k||_2}\right)^\top \left(\frac{\hat{y}^i_k}{||\hat{y}^i_k||_2}\right)$; `// SGI loss at step k`
    end
    `/* compute inverse modeling loss */`
    for k *in (1, . . . , K)* **do**
      $l^i \leftarrow \lambda^{\text{IM}} \cdot$ Cross-entropy loss$(a^i_{k-1}, I(\hat{y}_{k-1}, \tilde{y}_k))$
    end
    `/* compute goal RL loss */`
    $r^i \leftarrow d(g^i, \tilde{z}_t) - d(g^i, \tilde{z}_{t+1})$; `// Calculate goal RL reward`
    $l^i \leftarrow l^i +$ RL loss(s^i, a^i, r^i, s'^i) ; `// Add goal RL loss for batch`
  end
  $l \leftarrow \frac{1}{N} \sum_{i=0}^{N} l^i$; `// average loss over minibatch`
  $\theta_o, \phi \leftarrow \text{optimize}((\theta_o, \phi), l)$; `// update online parameters`
  $\theta_m \leftarrow \tau\theta_o + (1-\tau)\theta_m$; `// update target parameters`
end
return (θ_o, ϕ) ; `// return parameters for fine-tuning`

A.3. Full Results on Atari100k

We report full scores for SGI agents across all 26 games in Table 1. We do not reproduce the per-game scores for APT and VISR provided by Liu et al. (2021b), as we believe that the scores in the currently-available version of their paper may contain errors.[2]

Tableau 1. Mean return per episode for the 26 Atari100k games (Kaiser et al., 2019) after 100k steps. Agents are evaluated at the end of training, and scores for all methods are averaged over 10 random seeds. We reproduce scores for SPR from Schwarzer et al. (2021a), whereas ATC scores are from our implementation.

	Random	Human	SPR	ATC-M	SGI-R	SGI-E	SGI-W	SGI-M/S	SGI-M	SGI-M/L
Alien	227.8	7127.7	801.5	699.0	1034.5	857.6	1043.8	1070.5	1101.7	**1184.0**
Amidar	5.8	1719.5	176.3	95.4	154.8	166.8	**206.7**	185.9	168.2	171.2
Assault	222.4	742.0	571.0	509.8	446.6	583.1	759.5	632.4	905.1	**1326.5**
Asterix	210.0	8503.3	977.8	454.1	754.6	953.6	**1539.1**	651.8	835.6	567.2
Bank Heist	14.2	753.1	380.9	534.9	397.4	514.8	426.3	547.4	**608.4**	567.8
Battle Zone	2360.0	37187.5	**16651.0**	13683.8	4439.0	16417.0	7103.0	12107.0	13170.0	14462.0
Boxing	0.1	12.1	35.8	16.8	57.7	33.6	50.2	40.0	36.9	**73.9**
Breakout	1.7	30.5	17.1	16.9	23.4	17.8	35.4	23.8	42.8	**251.9**
Chopper Command	811.0	7387.8	974.8	870.8	784.7	1136.2	1040.1	1042.7	**1404.0**	1037.9
Crazy Climber	10780.5	35829.4	42923.6	74215.5	50561.2	76356.3	81057.4	75542.1	88561.2	**94602.2**
Demon Attack	152.1	1971.0	545.2	524.6	2198.7	357.5	1408.5	1135.5	968.1	**5634.8**
Freeway	0.0	29.6	24.4	5.7	2.1	15.1	26.5	12.5	**30.0**	28.6
Frostbite	65.2	4334.7	**1821.5**	222.6	349.3	981.4	247.7	861.1	741.3	927.8
Gopher	257.6	2412.5	715.2	946.2	1033.9	964.9	1846.0	1172.4	1660.4	**2035.8**
Hero	1027.0	30826.4	7019.2	6119.4	7875.2	6863.7	7503.9	7090.4	7474.0	**9975.9**
Jamesbond	29.0	302.8	365.4	272.6	263.9	383.8	**425.1**	413.2	366.4	394.8
Kangaroo	52.0	3035.0	**3276.4**	603.1	923.8	1588.9	598.6	1236.8	2172.8	1887.5
Krull	1598.0	2665.5	3688.9	4494.7	5672.6	4070.7	5583.2	**6161.3**	5734.0	5862.6
Kung Fu Master	258.5	22736.3	13192.7	11648.2	13349.2	11802.1	14199.7	16781.8	16137.8	**17340.7**
Ms Pacman	307.3	6951.6	1313.2	848.9	411.0	1278.3	1970.8	1519.5	1520.0	**2218.0**
Pong	-20.7	14.6	-5.9	-13.5	-3.9	4.2	4.7	**9.7**	7.6	7.7
Private Eye	24.9	69571.3	**124.0**	95.0	95.3	100.0	100.0	84.7	90.0	83.8
Qbert	163.9	13455.0	669.1	572.2	595.0	717.6	**855.6**	804.7	709.8	702.6
Road Runner	11.5	7845.0	14220.5	7989.3	5476.0	9195.2	18011.9	12083.5	**18370.2**	18306.8
Seaquest	68.4	42054.7	583.1	415.7	735.3	615.2	656.1	728.2	728.4	**1979.3**
Up N Down	533.4	11693.2	28138.5	84361.2	67968.1	63612.9	**84551.4**	42165.6	79228.8	46083.3
Median HNS	0.000	1.000	0.415	0.204	0.326	0.456	0.589	0.423	0.679	**0.755**
Mean HNS	0.000	1.000	0.704	0.780	0.888	0.838	1.144	0.914	1.149	**1.590**
#Games > Human	0	0	7	5	5	6	8	6	**9**	**9**
#Games > 0	0	26	**26**	**26**	25	**26**	**26**	**26**	**26**	**26**

[2]In particular, we observed that VISR claimed to have a score below -21 on Pong, which is impossible with standard settings.

Tableau 2. Mean return per episode for the 26 Atari100k games (Kaiser et al., 2019) after 100k steps for versions of SGI with modified fine-tuning, as discussed in Section 6.7. Agents are evaluated at the end of training, and scores for all methods are averaged over 10 random seeds. We reproduce scores for SPR from Schwarzer et al. (2021a).

	Random	Human	SGI-None	Naive	Frozen	No SPR	Full SSL	SGI-M
Alien	227.8	7127.7	835.9	1049.3	**1242.8**	1060.7	1117.6	1101.7
Amidar	5.8	1719.5	107.6	133.6	147.7	154.2	**206.0**	168.2
Assault	222.4	742.0	657.7	752.1	869.2	756.3	**1145.2**	905.1
Asterix	210.0	8503.3	832.9	**1029.3**	433.1	575.5	603.1	835.6
Bank Heist	14.2	753.1	613.2	**726.5**	273.6	365.8	323.4	608.4
Battle Zone	2360.0	37187.5	13490.0	**15708.0**	11754.0	13692.0	11689.8	13170.0
Boxing	0.1	12.1	6.6	24.0	**61.5**	34.7	42.7	36.9
Breakout	1.7	30.5	12.1	29.3	34.0	43.0	**62.6**	42.8
Chopper Command	811.0	7387.8	1085.2	1081.2	916.5	925.5	965.8	**1404.0**
Crazy Climber	10780.5	35829.4	19707.6	55002.4	65220.0	69505.6	69052.0	**88561.2**
Demon Attack	152.1	1971.0	778.8	850.0	1329.4	981.7	**1783.8**	968.1
Freeway	0.0	29.6	17.2	28.1	24.4	13.2	10.9	**30.0**
Frostbite	65.2	4334.7	1475.8	662.1	1045.4	482.1	**1664.9**	741.3
Gopher	257.6	2412.5	438.2	626.1	**2214.1**	1561.7	1998.7	1660.4
Hero	1027.0	30826.4	6472.0	5538.3	6353.3	5249.6	**8715.4**	7474.0
Jamesbond	29.0	302.8	157.4	324.2	358.2	346.8	**407.6**	366.4
Kangaroo	52.0	3035.0	**3802.8**	3091.6	800.0	685.6	999.5	2172.8
Krull	1598.0	2665.5	3954.0	5202.7	**6073.7**	5722.8	5323.9	5734.0
Kung Fu Master	258.5	22736.3	7929.4	11952.2	**19374.6**	15039.8	18123.2	16137.8
Ms Pacman	307.3	6951.6	990.2	1276.4	1663.3	1753.3	**1779.3**	1520.0
Pong	-20.7	14.6	-4.4	-4.2	3.8	3.9	-0.1	**7.6**
Private Eye	24.9	69571.3	62.8	**385.9**	96.7	90.5	90.0	90.0
Qbert	163.9	13455.0	720.0	664.8	587.6	681.3	**3015.8**	709.8
Road Runner	11.5	7845.0	5428.4	14629.7	14311.9	17036.5	13998.2	**18370.2**
Seaquest	68.4	42054.7	577.8	509.0	1054.4	**1397.8**	989.4	728.4
Up N Down	533.4	11693.2	46042.6	48856.6	29938.4	**105466.9**	45023.5	79228.8
Median HNS	0.000	1.000	0.343	0.425	0.499	0.452	0.397	**0.679**
Mean HNS	0.000	1.000	0.565	0.849	0.971	1.114	1.011	**1.149**
#Games > Human	0	0	3	8	8	8	8	**9**
#Games > SPR	0	19	10	14	15	14	17	**20**

A.4. Transferring Representations between Games

One advantage of pretraining representations is the possibility of representations being useful across games. Intuitively, we expect better transfer between similar games so we chose five "cliques" of games with similar semantics and visual elements. The cliques are shown in Table 3. We pretrain on a dataset of 750k frames from each game in a clique (i.e. 3M frames for a clique of 4) and finetune on a single game. To show whether pretraining on other games is beneficial, we compare to a baseline of pretraining on just the 750k frames from the single Atari 100k game we use for finetuning.

Our results in Table 4 show that pretraining with the extra frames from the clique games is mostly unhelpful to finetune performance. Only Kangaroo shows a modest improvement, a few games show no difference in performance, and most games show a decrease in performance when pretraining with other games. We believe that Atari may not be as suitable to transferring representations as other domains, and previous work using Atari to learn transferable representations has also had negative results (Stooke et al., 2020). Though game semantics can be similar, we note that even small differences in rule sets and visual cues can make transfer difficult.

Tableau 3. Cliques of semantically similar games

Clique	Games
space	Space Invaders, Assault, Demon Attack, Phoenix
pacman	MsPacman, Alien, Bank Heist, Wizard Of Wor
platformer	Montezuma Revenge, Hero, Kangaroo, Tutankham
top scroller	Crazy Climber, Up N Down, Skiing, Journey Escape
side scroller	Chopper Command, James Bond, Kung Fu Master, Private Eye

Tableau 4. Mean return per episode for clique games in Atari100k (Kaiser et al., 2019) after 100k steps. Agents are evaluated at the end of training, and scores for all methods are averaged over 10 random seeds. Games in the same clique are placed together.

Game	Single	Clique
Assault	738.5	554.1
Demon Attack	1171.8	695.0
Alien	1183.9	830.2
Bank Heist	448.8	303.0
Ms Pacman	1595.8	1352.1
Kangaroo	489.2	994.0
Crazy Climber	52036.0	21829.8
Up N Down	18974.7	13493.9
James Bond	397.6	325.4
Kung Fu Master	16402.6	16499.0
Chopper Command	933.6	854.6

A.5. Uncertainty-aware comparisons

Concurrent work (Agarwal et al., 2021b) has found that many prior comparisons in deep reinforcement learning are not robust and may be entirely incorrect, particularly in the Atari 100K setting. They demonstrate that these misleading comparisons are partially due to undesirable properties of the per-game median and mean normalized scores, the most commonly-used aggregate metrics, and propose using the inter-quartile mean (IQM) normalized score, calculated over *runs* rather than *tasks*. Moreover, they suggest providing percentile bootstrap confidence intervals to quantify uncertainty, to avoid misleading comparisons based on highly-variable point estimates.

As raw per-run data is required for this, which was not reported for prior work, we do so only for experiments conducted ourselves. In the interests of improving practices in the community moving forward, we also commit to making this data for our experiments available to other researchers in the future.

In Figure 1 through Figure 5 we show estimated uncertainty via bootstrapping for the various comparisons drawn throughout Section 6.7, while Table 5 gives IQM human-normalized scores and 95% bootstrap confidence intervals for the same results. All comparisons in Figure 1 through Figure 5 are statistically significant ($p < 0.05$) except for:

- ATC-M vs SGI-None in Figure 1 ($p \gg 0.05$)
- SGI-M vs SGI-W in Figure 2 ($p \approx 0.05$)
- SGI-M vs SGI-M w/ SGI FT in Figure 4 ($p \approx 0.4$)
- SGI-M vs G+I and S+I in Figure 5 ($p \approx 0.1$)

Tableau 5. Interquartile mean, median and mean human-normalized scores for variants of SGI and controls, evaluated after finetuning over all 10 runs for each of the 26 Atari 100k games. Confidence intervals computed by percentile bootstrap with 5000 resamples.

Method	IQM	95% CI	Median	95% CI	Mean	95% CI
SGI-M/L	**0.745**	(0.687, 0.805)	**0.753**	(0.625, 0.850)	**1.598**	(1.486, 1.676)
SGI-M	0.567	(0.524, 0.612)	0.679	(0.473, 0.739)	1.149	(0.974, 1.347)
SGI-M/S	0.444	(0.404, 0.487)	0.423	(0.341, 0.577)	0.914	(0.822, 1.031)
SGI-W	0.510	(0.476, 0.547)	0.589	(0.434, 0.675)	1.144	(0.981, 1.345)
SGI-E	0.363	(0.326, 0.404)	0.456	(0.309, 0.482)	0.838	(0.692, 1.008)
SGI-R	0.302	(0.275, 0.331)	0.326	(0.253, 0.385)	0.888	(0.776, 1.004)
SGI-None	0.242	(0.212, 0.274)	0.343	(0.268, 0.401)	0.565	(0.440, 0.711)
Baselines						
ATC-M	0.235	(0.210, 0.262)	0.204	(0.182, 0.291)	0.780	(0.601, 0.971)
ATC-W	0.221	(0.199, 0.244)	0.219	(0.170, 0.290)	0.587	(0.504, 0.673)
ATC-E	0.214	(0.193, 0.236)	0.237	(0.169, 0.266)	0.462	(0.420, 0.504)
ATC-R	0.187	(0.174, 0.202)	0.191	(0.139, 0.202)	0.472	(0.454, 0.491)
BC-M	0.481	(0.438, 0.524)	0.548	(0.390, 0.685)	0.858	(0.795, 0.924)
Pretraining Ablations						
S+I	0.522	(0.488, 0.559)	0.629	(0.494, 0.664)	0.978	(0.900, 1.061)
G+I	0.521	(0.486, 0.558)	0.512	(0.386, 0.582)	1.004	(0.892, 1.129)
S+G	0.032	(0.027, 0.039)	0.029	(0.025, 0.044)	0.098	(0.061, 0.146)
I	0.435	(0.404, 0.470)	0.411	(0.334, 0.489)	0.943	(0.783, 1.126)
G	0.060	(0.048, 0.072)	0.060	(0.037, 0.081)	0.181	(0.145, 0.218)
S	0.007	(0.002, 0.011)	0.009	(0.002, 0.014)	-0.054	(-0.082, -0.026)
Finetuning Ablations						
SGI-M (No S)	0.448	(0.412, 0.484)	0.419	(0.335, 0.524)	1.114	(0.921, 1.321)
SGI-None (No S)	0.139	(0.118, 0.162)	0.161	(0.123, 0.225)	0.315	(0.274, 0.356)
SGI-M (All SGI)	0.541	(0.498, 0.585)	0.397	(0.330, 0.503)	1.011	(0.909, 1.071)
SGI-M (Frozen)	0.510	(0.476, 0.543)	0.499	(0.406, 0.554)	0.971	(0.871, 1.088)
SGI-M (Naive)	0.453	(0.422, 0.485)	0.429	(0.380, 0.500)	0.845	(0.754, 0.952)

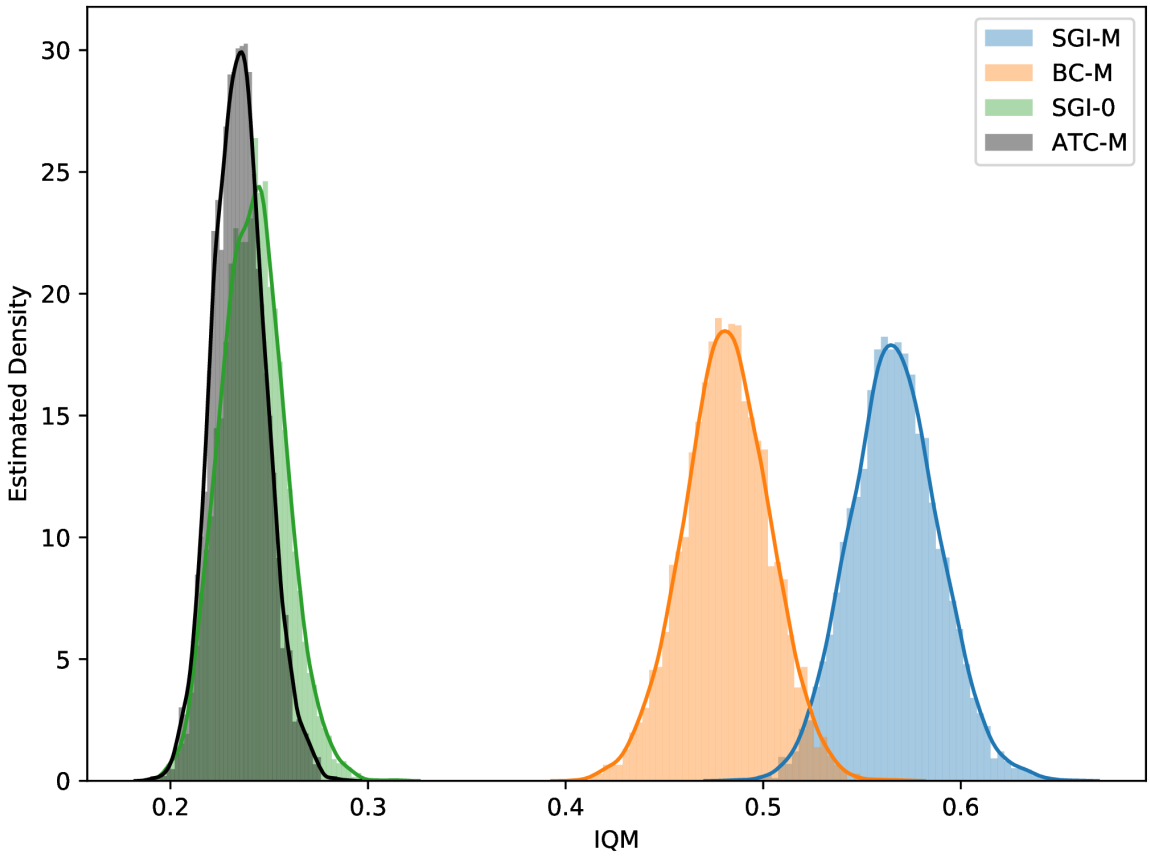


Figure 1. Comparisons to behavioral cloning (BC) and ATC.

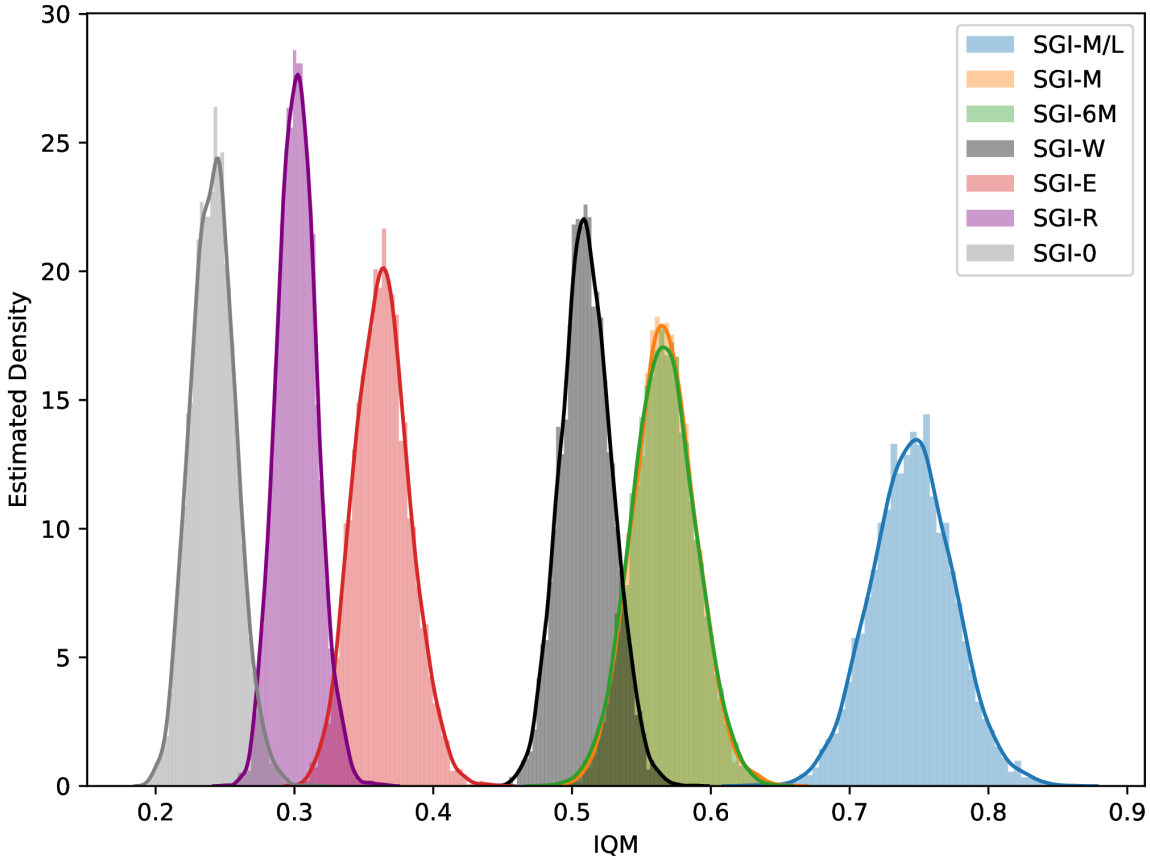


Figure 2. Ablations over different pretraining datasets.

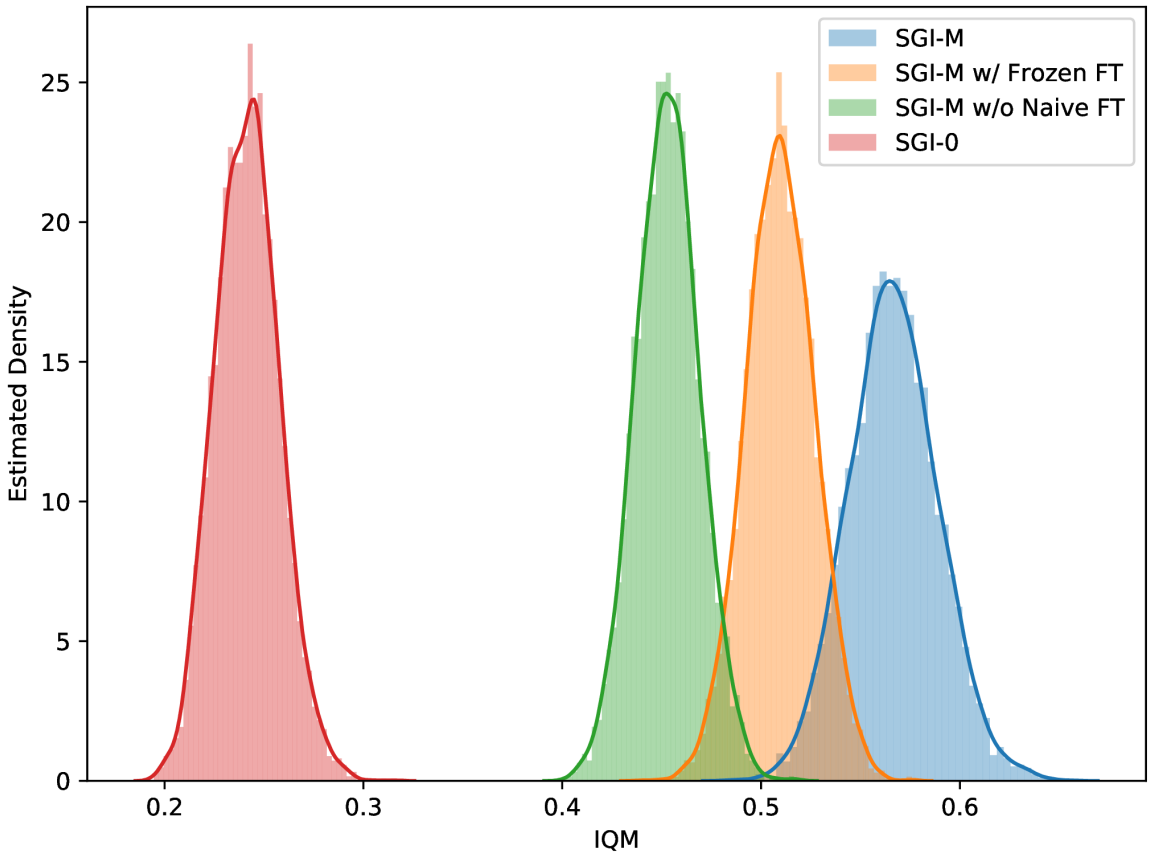


Figure 3. Ablations over various fine-tuning configurations.

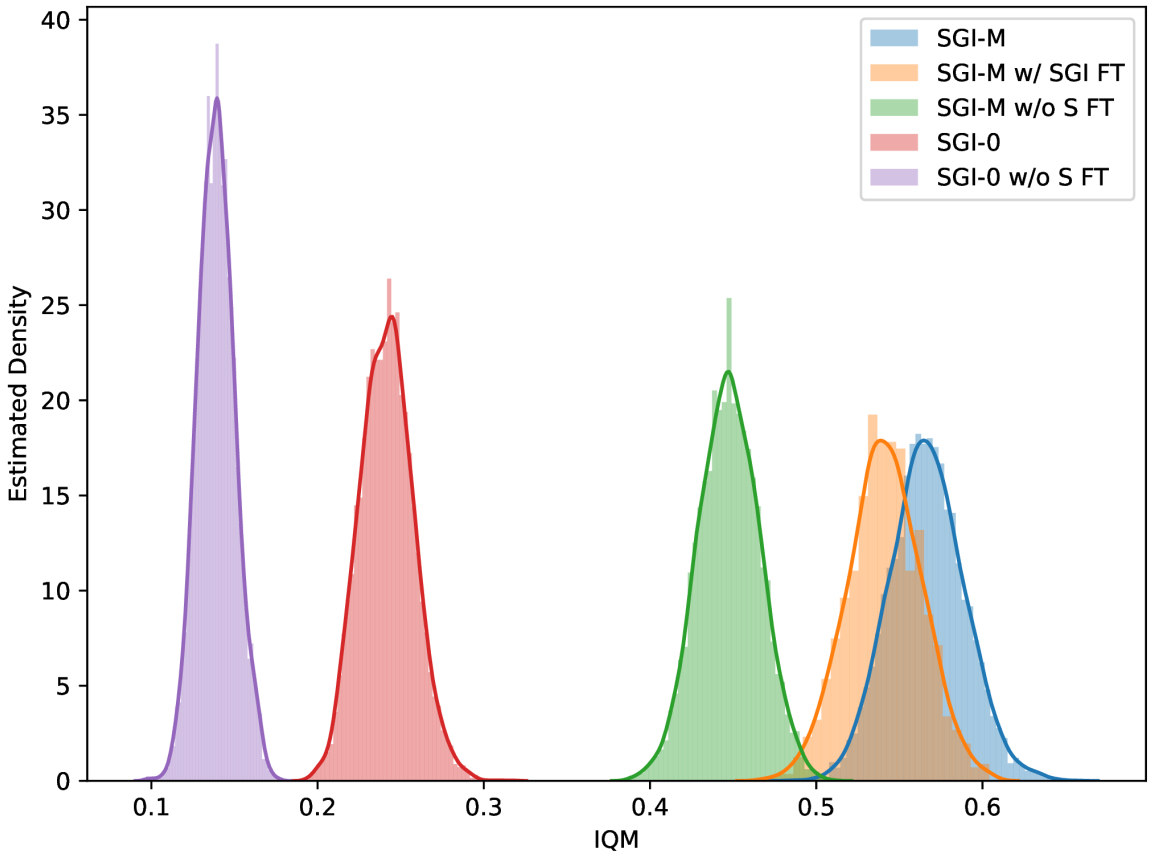


Figure 4. Ablations over SSL objectives during fine-tuning.

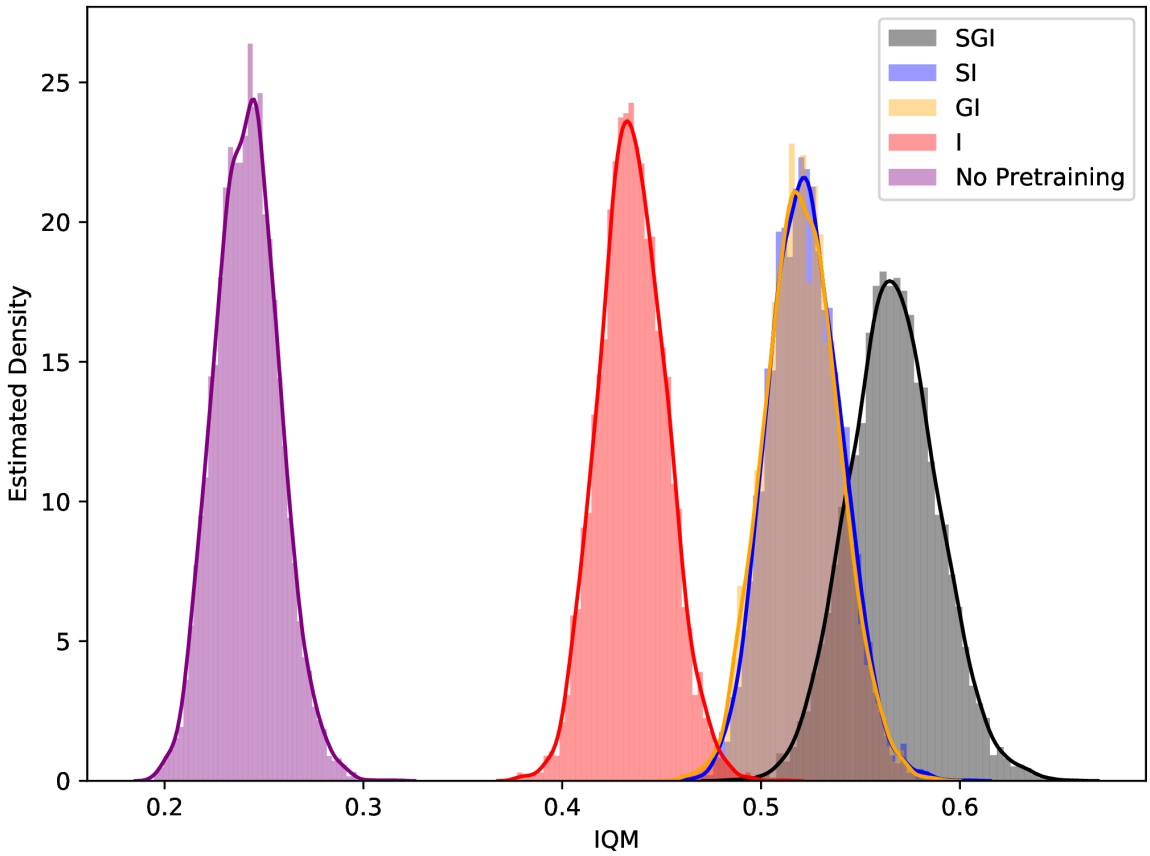


Figure 5. Ablations over pretraining SSL objectives.

Annexe B

Appendix for Chapter 4

Task	Steps
walker-run	1×10^6
cheetah-run	1×10^6
acrobot-swingup	1×10^6
finger-turn_hard	1×10^6
fish-swim	1×10^6
humanoid-stand	1×10^6
humanoid-run	1×10^6
quadruped-run	1×10^6
swimmer-swimmer15	1×10^6
hopper-hop	1×10^6

Tableau 1. Tasks for SAC experiments and a number of training steps. Many of DMC tasks are solved by SAC in a matter of several thousand steps; we chose environments where SAC requires a substantial amount of training according to the reported results from https://github.com/denisyarats/pytorch_sac#results.

B.1. Experimental Details and Additional Results

We use an open-source JAX implementation (Kostrikov, 2021) of the SAC and DrQ algorithms and an open-source JAX implementation[1] of SPR. This SPR implementation exhibit a slightly higher aggregate performance than the scores in Schwarzer et al. (2021a) based on a PyTorch implementation. All algorithms use default hyperparameters unless specified otherwise (for example, in experiments with replay ratio and n-step targets). SAC and DrQ experiments use 10 random seeds for evaluating the performance, SPR uses 20 random seeds.

Tables 1 and 2 report the sets of DeepMind Control Suite tasks for testing SAC and DrQ algorithms respectively. Note that SAC learns from dense states, while DrQ learns from raw pixel observations. SPR trains on a standard set of 26 tasks from the Atari 100k benchmark used by Kaiser et al. (2019), Hasselt et al. (2019), and Schwarzer et al. (2021a). The list of all Atari environments is available in Table 5.

B.1.1. Ablations

Section 4.6.4 outlines the interaction of different moving parts of RL algorithms and our proposed reset strategy. This appendix elaborates on our observations and provides

[1] https://github.com/MaxASchwarzer/dopamine/tree/atari100k_spr

Task	Steps
walker-stand	5×10^5
finger-spin	5×10^5
cartpole-balance	5×10^5
cartpole-swingup	5×10^5
walker-walk	5×10^5
cartpole-balance_sparse	5×10^5
pendulum-swingup	1×10^6
hopper-stand	1×10^6
quadruped-walk	2×10^6
walker-run	2×10^6
finger-turn_easy	2×10^6
cheetah-run	2×10^6
acrobot-swingup	2×10^6
finger-turn_hard	2×10^6
cartpole-swingup_sparse	2×10^6
quadruped-run	2×10^6
reacher-easy	2×10^6
reacher-hard	2×10^6
hopper-hop	2×10^6

Tableau 2. Tasks for DrQ experiments and a number of training steps.

supporting figures to help understanding the effect of resets. We now show, one by one, how resetting behaves under a diverse set of controlled settings.

B.1.1.1. Replay buffer. The reset mechanism we propose is a form of forgetting based on retaining all of the collected knowledge, stored in the replay buffer, and not retaining a part of the learned behavior, stored in the parameters of an agent's function approximators. How critical is it to preserve the knowledge in the replay buffer? We test its importance in DrQ by periodically resetting the replay buffer in addition to the last layers. Figure 1 shows that keeping the buffer is essential; resetting it amounts to learning almost from scratch. These results suggests that knowledge retention is significantly more important than behavior retention for preventing the negative effects of the primacy bias and simultaneously being able to recover from resets.

B.1.1.2. Initialization. After each reset, the new parameters are sampled from a canonical initialization distribution followed by the algorithms. To understand whether the susceptibility of an agent to the primacy bias is just a consequence of an unlucky

initialization of one of the layers of its neural networks, we run DrQ with resets, but set the value of the re-initialized parameters to the one they had at the beginning of training (i.e., by initializing them with the same seed). The results in Figure 1 show that the performance of this variant of our reset strategy is almost identical to the original version: the problem alleviated by resets is not one of pathological initializations, in line with findings of other works (Bjorck et al., 2022), but instead one resulting from the peculiar interactions happening when learning from a growing dataset of interactions.

B.1.1.3. Optimizer state. As highlighted in the main text, resetting the optimizer's moment estimates together with the corresponding neural network parameters have almost no effect. We show results demonstrating this for DrQ (see fig. 2). We believe this is mostly due to the momentum-based optimizers: with Adam with default parameters, the first moment ($\beta_1 = 0.9$) will vanish in about 10 updates, the second moment ($\beta_2 = 0.999$) after 1000 updates. On the scale of our tasks, it is indeed a quite rapid recovery time.

B.1.1.4. Reset depth. One of the two hyperparameters introduced by our reset strategy on top of any backbone algorithm is the number of layers of the agent's neural networks to be re-initialized. We investigated the impact of this choice for both SPR and DrQ, while sticking for SAC to the default choice of re-initializing all networks. Results for DrQ in Figure 3 demonstrate that resetting the last layer yields slightly inferior performance to resetting the entire Q-learning head (3 layers). For SPR, as shown in Figure 6, we found the reverse to be true. Thus, how many layers to reset is a hyperparameter that may need tuning, and the choice can be informed by the difficulty of representation learning for each domain. We recommend starting the exploration of this hyperparameter from resetting the last 1-3 layers.

B.1.1.5. Which networks to reset. In our experiments, we reset a subset of the value function parameters in SPR and a subset of the parameters of all the trained neural networks in DrQ and SAC. The latter two algorithms use three groups of function approximators: an actor, a critic, and a target critic. We investigate the impact of resetting each one of these modules in DrQ. Results in fig. 4 show that a simultaneous reset of all the neural networks is generally the most robust technique to improve performance over the backbone algorithm, while resetting the critic had the most impact on the performance. We have also tried a version of resets where each weight

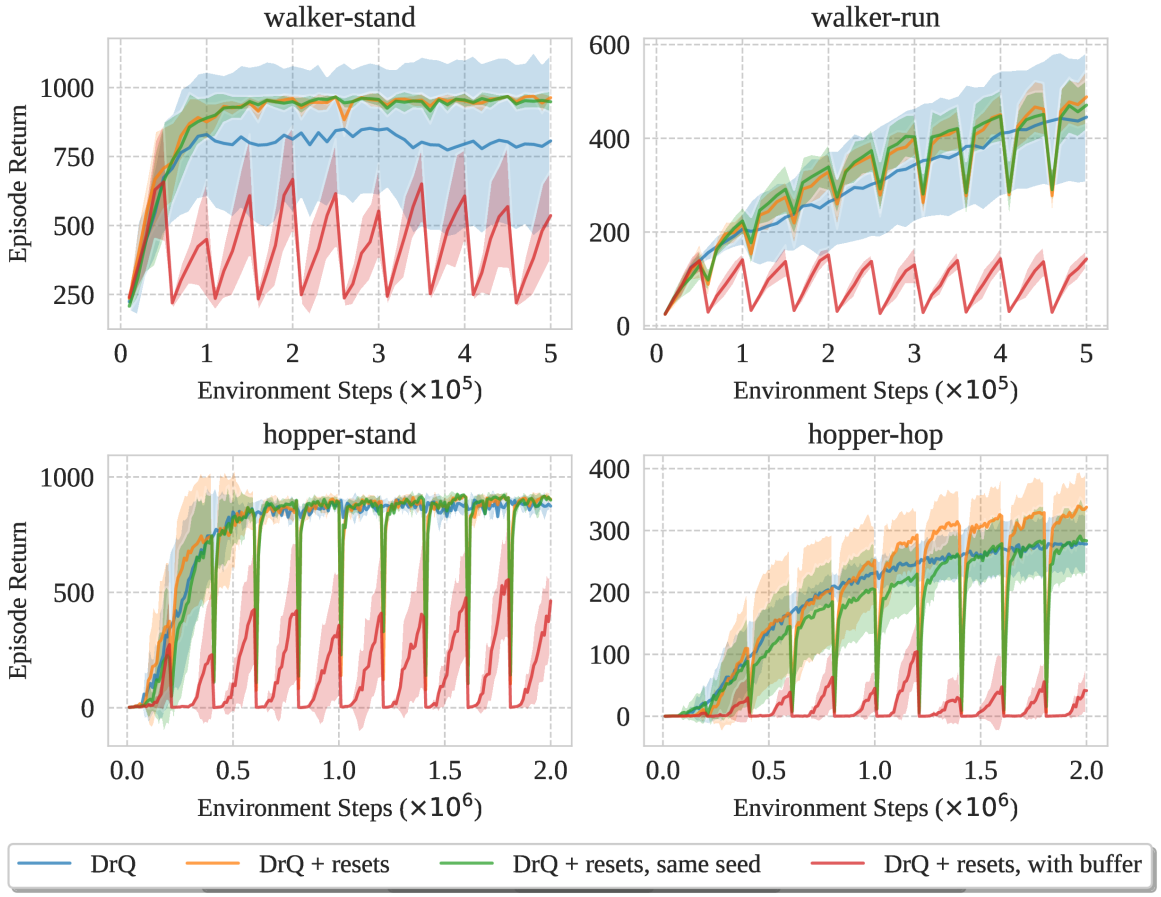


Figure 1. Performance of the DrQ agent with standard resets, with same-seed resets, and with both buffer and last layer resetting (10 resets during training) on four DMC tasks. Resetting the replay buffer in addition to the last layers nullifies the learning progress, while preserving the random seed for drawing re-initialized parameters delivers almost the same results as standard resets.

is re-initialized with probability 0.5. Figure 3 shows that such a random subnetwork resetting was either on par or worse than the standard scheme.

B.1.1.6. Number of resets. Intuitively, the primacy bias affects the agent in a progressively milder way after each reset. It is natural to ask whether a limited number of resets, or even a single one, is sufficient to overcome the effects of overfitting to initial experiences. We test this hypothesis using DrQ, showing in fig. 5 that, despite the first reset contributing the most to mitigating the primacy bias, it is not always sufficient to reach the same performance of the standard continual resetting strategy. As a default choice, we recommend using the reset periodicity resulting in 3-10 resets over the course of training.

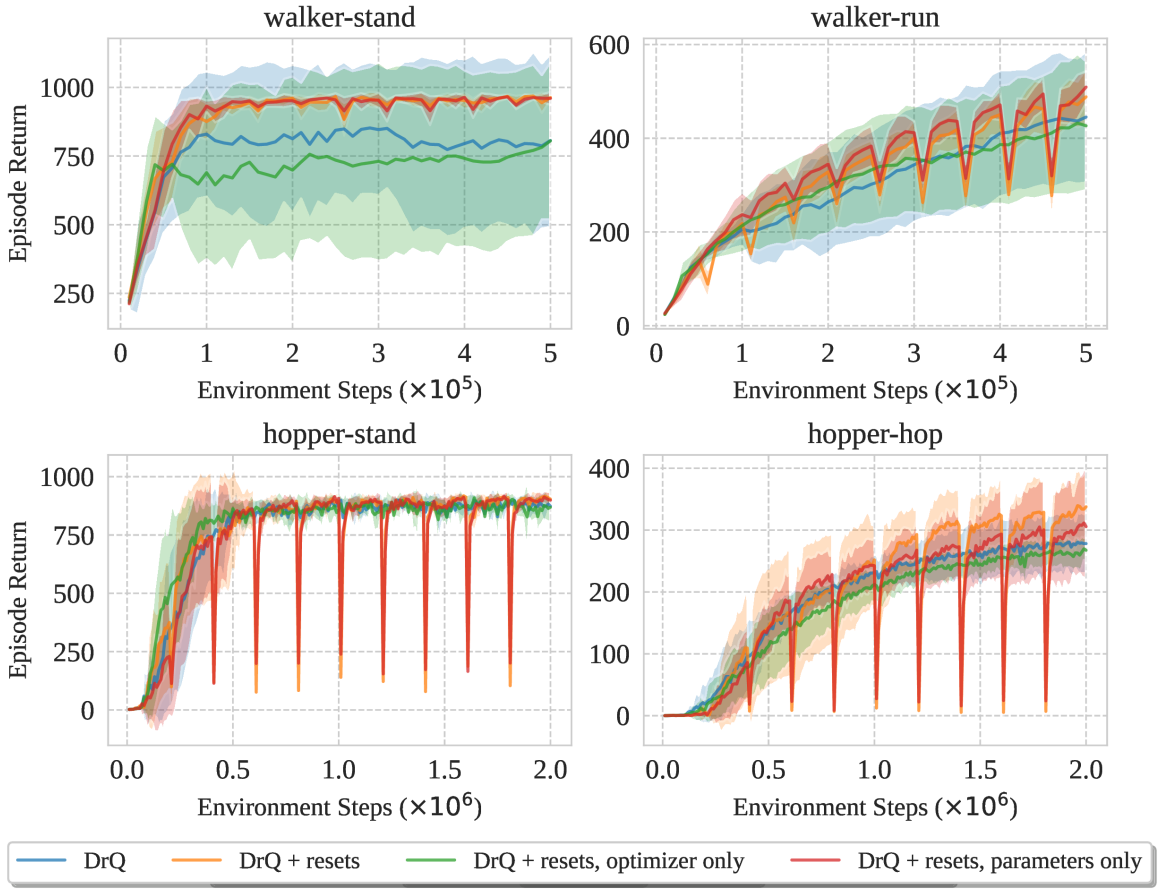


Figure 2. Performance of the DrQ agent with standard resets, with optimizer-only resets, and with parameter-only resets (10 resets during training) on four DMC tasks. Resetting the optimizer statistics does not alter training if the weights are preserved, while keeping the optimizer for re-initialized parameters delivers almost the same results as standard resets.

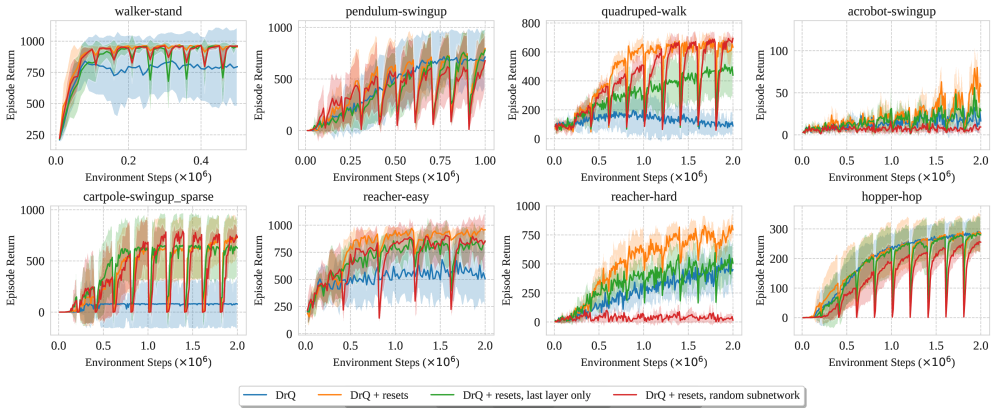


Figure 3. Performance of DrQ with resetting the 3-layer heads (i.e. standard resets), only the last layers, and random subnetworks. The overall performance of the latter two versions is either comparable or worse.

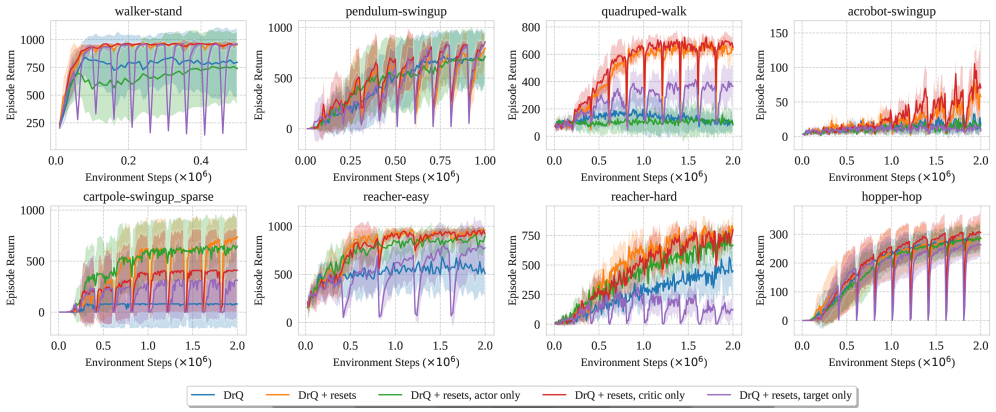


Figure 4. Performance of DrQ with resets of actor, critic, and target critic networks simultaneously (i.e. standard resets) and individually. Resetting the critic yields the predominant effect in most environments, but resetting all networks proved to be the most robust option.

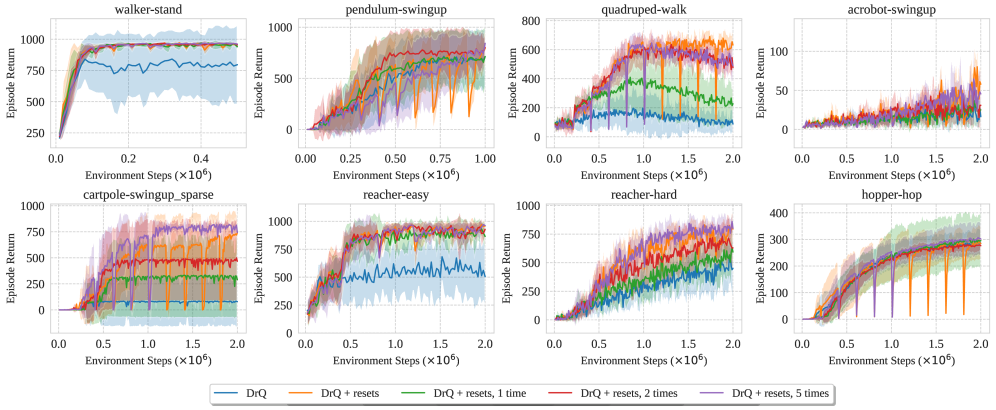


Figure 5. Performance of DrQ when using a limited number of resets. The number of resets for reaching the best performance varies: in some environments, a single reset suffices to overcome the primacy bias, in other environments, keeping resetting continually is required.

B.1.1.7. Other regularizers. Resets can be seen as a form of regularization because they implicitly constrain the final solutions to be not too far from the initial parameters. However, they specifically tackle the primacy bias better than other common forms of regularization. To test this conjecture, we repeat the heavy priming experiment on the `quadruped-run` task with standard L2 regularization of both critic and actor weights of SAC. We find that no value of regularization coefficient among the set $[10^{-5}, 3 \cdot 10^{-5}, 10^{-4}, 3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}, 10^{-2}, 3 \cdot 10^{-2}, 10^{-1}, 3 \cdot 10^{-1}, 1.0]$ can overcome heavy priming, obtaining results almost identical to the ones reported on fig. 1. The heavy priming setting artificially creates the conditions for the effect of the primacy bias to be particularly highlighted. To test whether resets offer superior performance also in

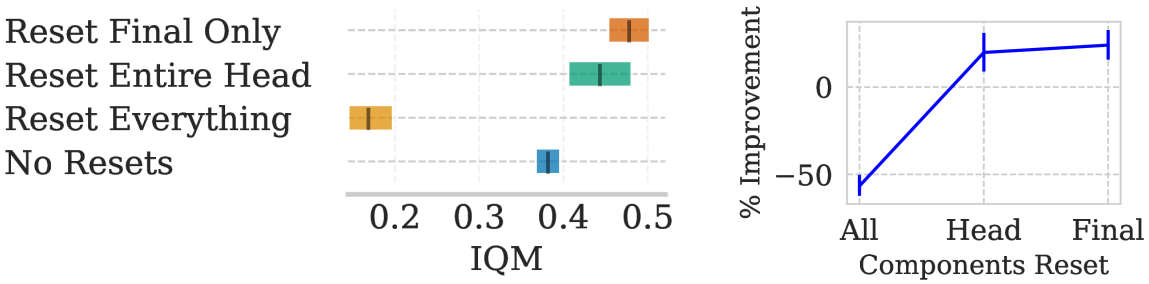


Figure 6. Performance of SPR when resetting different groups of parameters. The right plot visualizes the percentage of improvement gained by resetting a certain group of parameters compared to no resets at all. Resetting the last layer only delivers a slightly higher IQM than resetting the 2-layer head, while resetting the whole network severely damages the performance.

the context of standard training of reinforcement learning algorithms, we compare the performance of SAC and DrQ enriched with standard regularization methods to that of SAC and DrQ augmented with resets. In particular, we leverage L2 regularization of both the actor and the critic, as well as dropout (Srivastava et al., 2014). We report in table 3 the best value over the grid $[10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, 5 \cdot 10^{-3}]$ suggested by (Liu et al., 2021c) for L2 and the standard grid $[0.5, 0.1]$ for dropout. For SAC-based approaches, we also sweep over the replay ratios $[1, 9, 32]$ and report the best result for other regularizers. The table shows that not only standard regularization methods are not better than resets in the context of these RL algorithms, but that they do not provide any benefit, on aggregate performance, compared to the baselines.

Method	IQM	Median	Mean
SAC	501 (389, 609)	475 (407, 563)	484 (420, 548)
SAC + resets	**656** (549, 753)	**617** (538, 681)	**607** (547, 667)
SAC + dropout	219 (160, 285)	254 (204, 307)	258 (216, 300)
SAC + L2	412 (299, 524)	415 (337, 495)	416 (351, 481)
DrQ	569 (475, 662)	521 (470, 600)	535 (481, 589)
DrQ + resets	**762** (704, 815)	**680** (625, 731)	**677** (632, 720)
DrQ + dropout	492 (414, 567)	480 (420, 541)	479 (431, 527)
DrQ + L2	463 (362, 566)	473 (403, 541)	472 (415, 529)

Tableau 3. Comparison of the performance of SAC and DrQ when augmented with standard regularization techniques and resets. We leverage 10 runs and the same set of evaluation tasks reported in table 1 and table 2.

B.2. Per-Environment and Additional Results

The remainder of the appendix presents results for each task and supplementary plots for training with varying replay ratios and n-step targets.

Figure 7 demonstrates learning curves for SPR. We note that the low loss and high parameter norm for high n and replay ratios might indicate the symptoms of overfitting. Whilst resets implicitly control the weight norm, doing so explicitly through L2 regularization proved to be less effective for mitigating heavy priming.

Table 4 presents the aggregate metrics for the combinations of n and replay ratios in SAC. We additionally probe extreme replay ratios of 128 and 256 and observe that, even in these cases, learning with resets delivers meaningful performance, while the no-reset agent achieves near-zero returns.

Lastly, per-environment training curves for SAC as well as the results for varying replay ratios and n-step targets are available in Figures 9 and 11 respectively. Per-environment training curves for DrQ are available in Figure 12. Table 5 provides scores for SPR in all Atari 100k tasks.

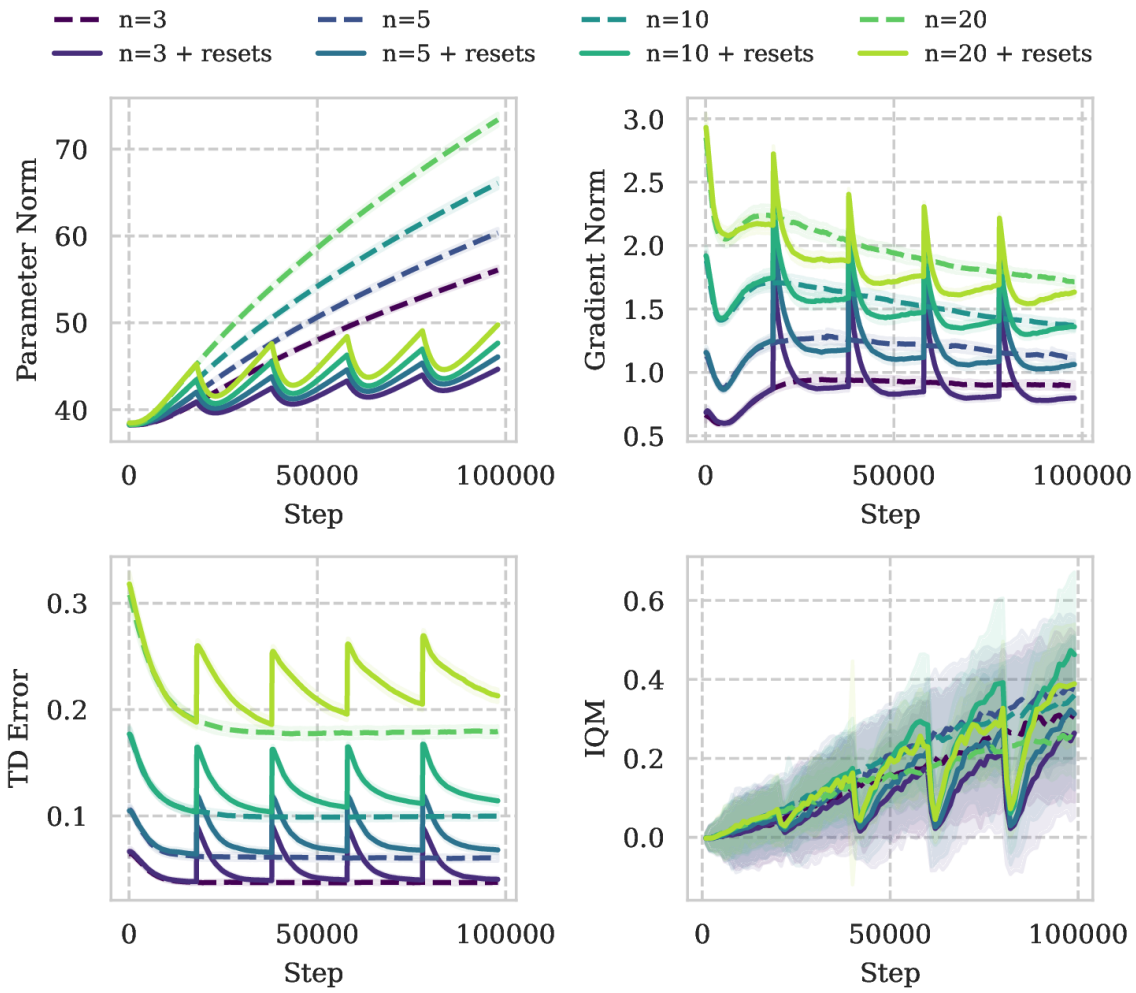


Figure 7. Training curves for SPR with resets on Atari 100k with different n-step targets. Resets increase TD errors and temporarily increase gradient norms for all values of n, while implicitly regularizing parameter norms.

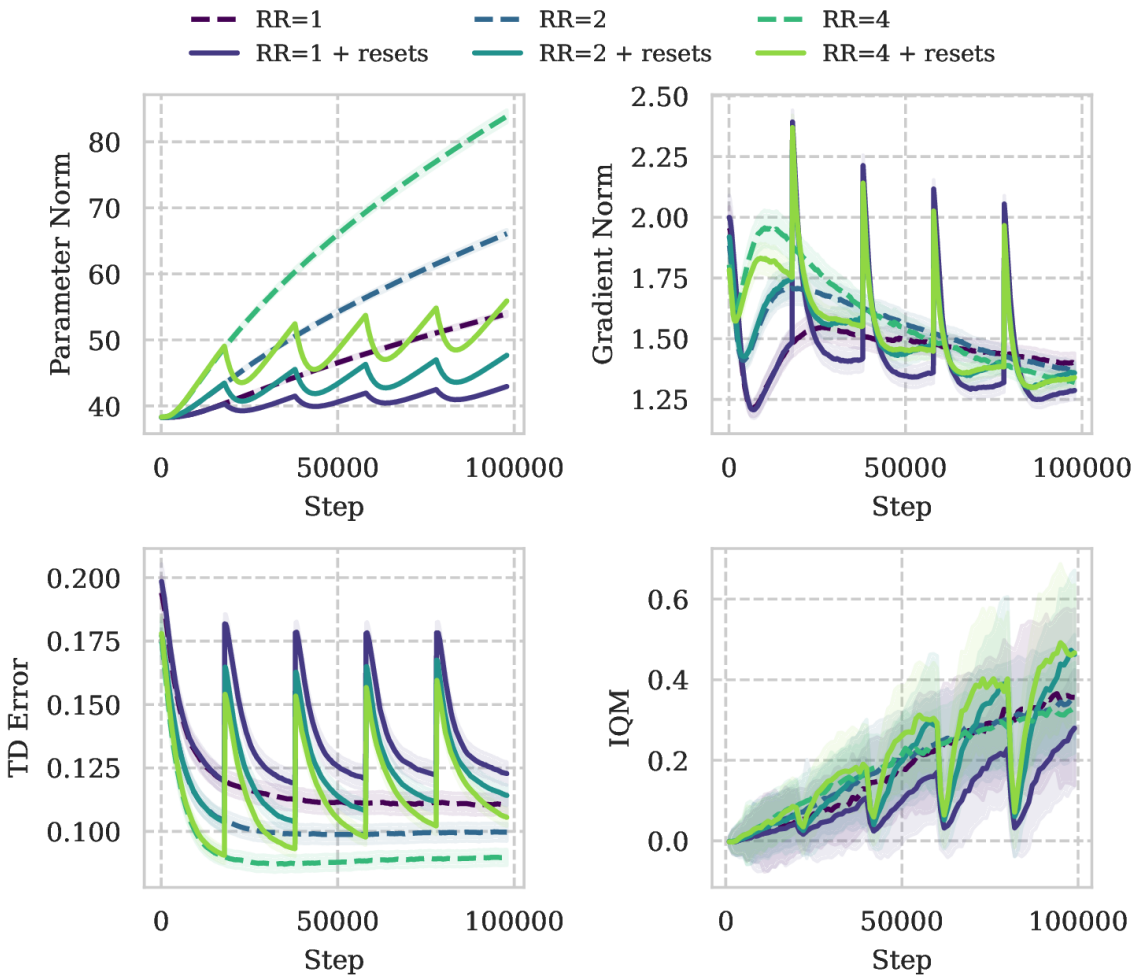


Figure 8. Training curves for SPR with resets on Atari 100k with different replay ratios. Resets increase TD errors and temporarily increase gradient norms for all values of RR, while implicitly regularizing parameter norms.

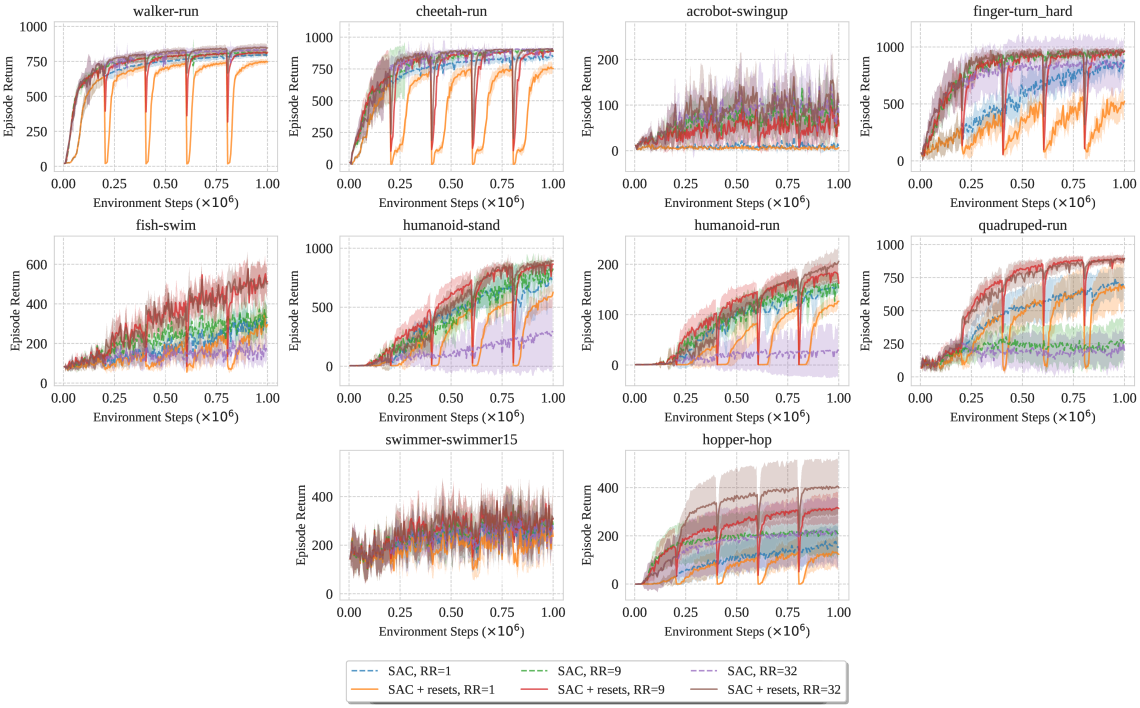


Figure 9. Per-environment training curves for SAC for various replay ratio (RR) values and a fixed value of $n = 1$.

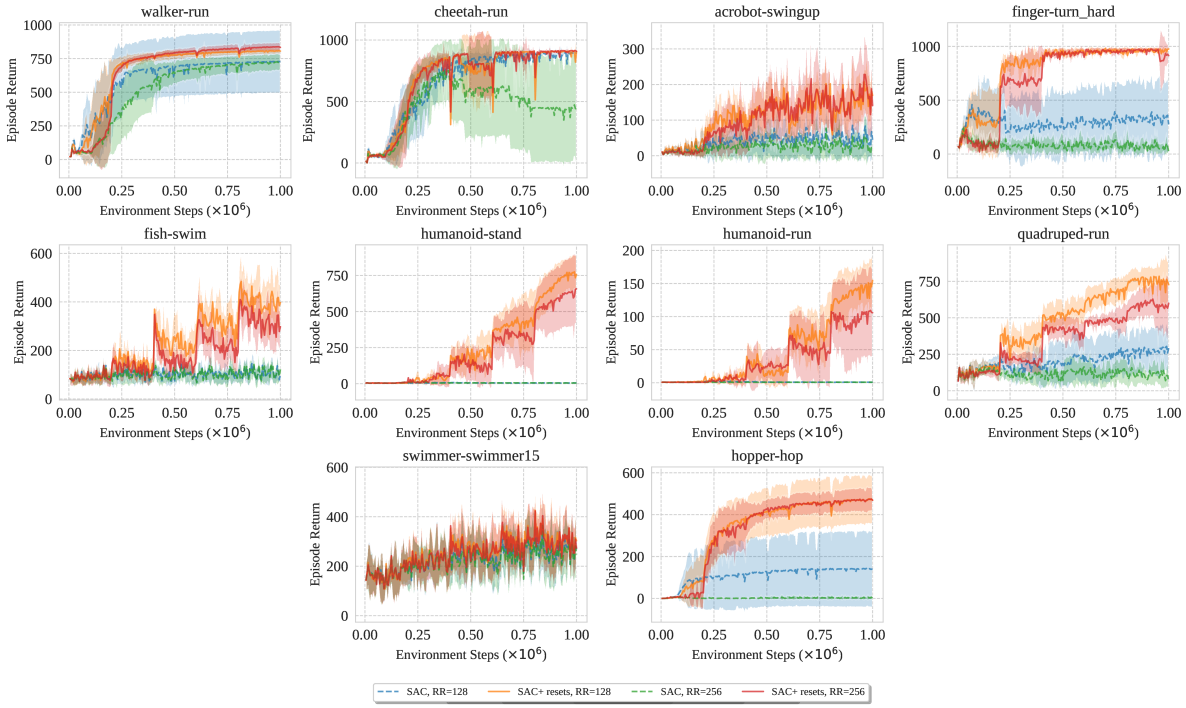


Figure 10. Per-environment training curves for SAC for the extreme replay ratio (RR) values of 128 and 256 and a fixed value of $n = 1$. While a standard learning algorithm struggles to make any progress, resets allow to achieve reasonable performance in this regime.

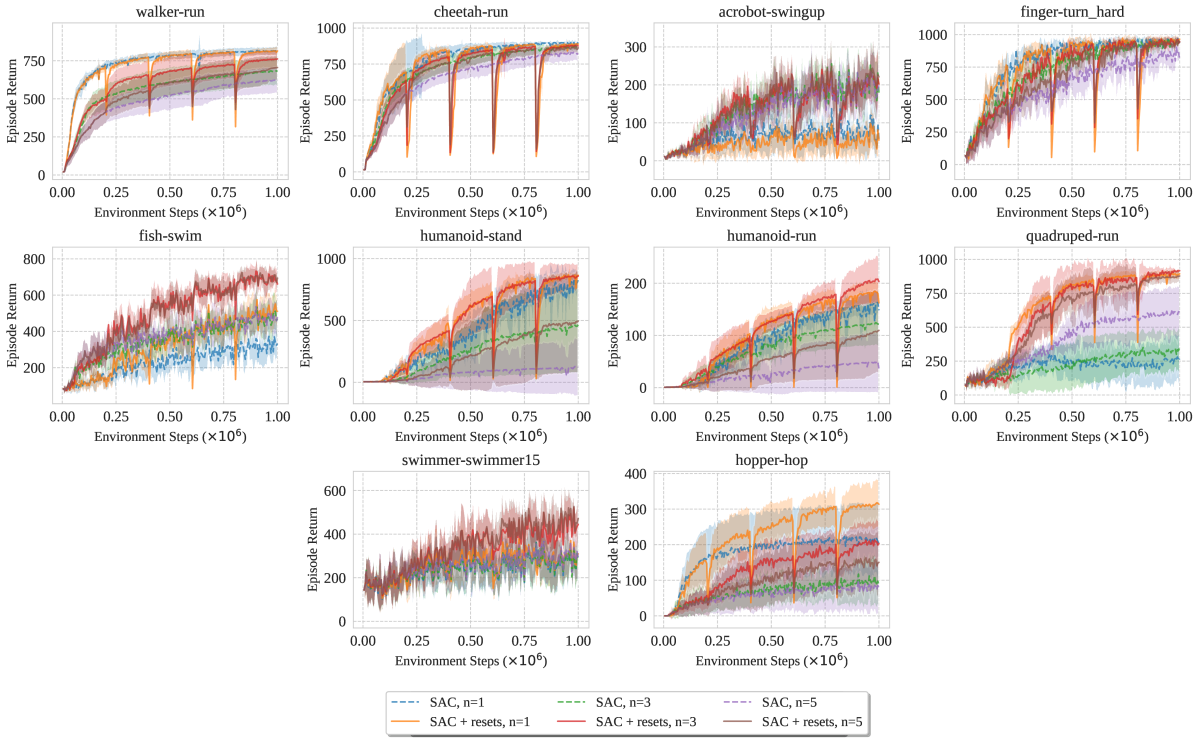


Figure 11. Per-environment training curves for SAC for various n and a fixed value of replay ratio RR $= 9$.

RR \ n	1		3		5	
	resets	no resets	resets	no resets	resets	no resets
1	401 (308,497),	**500** (389,609)	431 (366,499),	**452** (347,556)	**434** (333,533),	399 (304,493)
9	**628** (514,730),	459 (341,576)	**656** (549,752),	413 (319,517)	**597** (490,699),	392 (298,488)
32	**651** (542,750),	301 (193,430)	**642** (541,737),	342 (241,452)	**588** (482,684),	291 (200,394)
128	**584** (487,679),	149 (86,242)	—		—	
256	**520** (418,619),	39 (18,79)	—		—	

RR \ n	1		3		5	
	resets	no resets	resets	no resets	resets	no resets
1	415 (342,478),	**474** (406,562)	425 (382,481),	**448** (375,527)	**409** (346,492),	398 (328,475)
9	**575** (500,656),	473 (395,557)	**616** (537,680),	436 (365,519)	**547** (476,632),	403 (333,480)
32	**602** (527,677),	372 (297,469)	**599** (532,674),	398 (317,476)	**554** (475,626),	352 (279,433)
128	**568** (496,640),	268 (192,349)	—		—	
256	**518** (444,595),	152 (92,221)	—		—	

RR \ n	1		3		5	
	resets	no resets	resets	no resets	resets	no resets
1	410 (354,467),	**484** (419,549)	433 (393,473),	**451** (387,515)	**419** (360,478),	407 (348,467)
9	**577** (511,642),	476 (409,543)	**607** (547,666),	443 (381,504)	**553** (488,617),	407 (346,469)
32	**600** (537,662),	384 (314,456)	**601** (543,659),	397 (332,464)	**549** (486,611),	358 (294,422)
128	**566** (507,626),	275 (212,341)	—		—	
256	**520** (457,581),	165 (115,220)	—		—	

Tableau 4. Full results for SAC in terms of IQM (top), median (middle), and mean (bottom) performance across tasks.

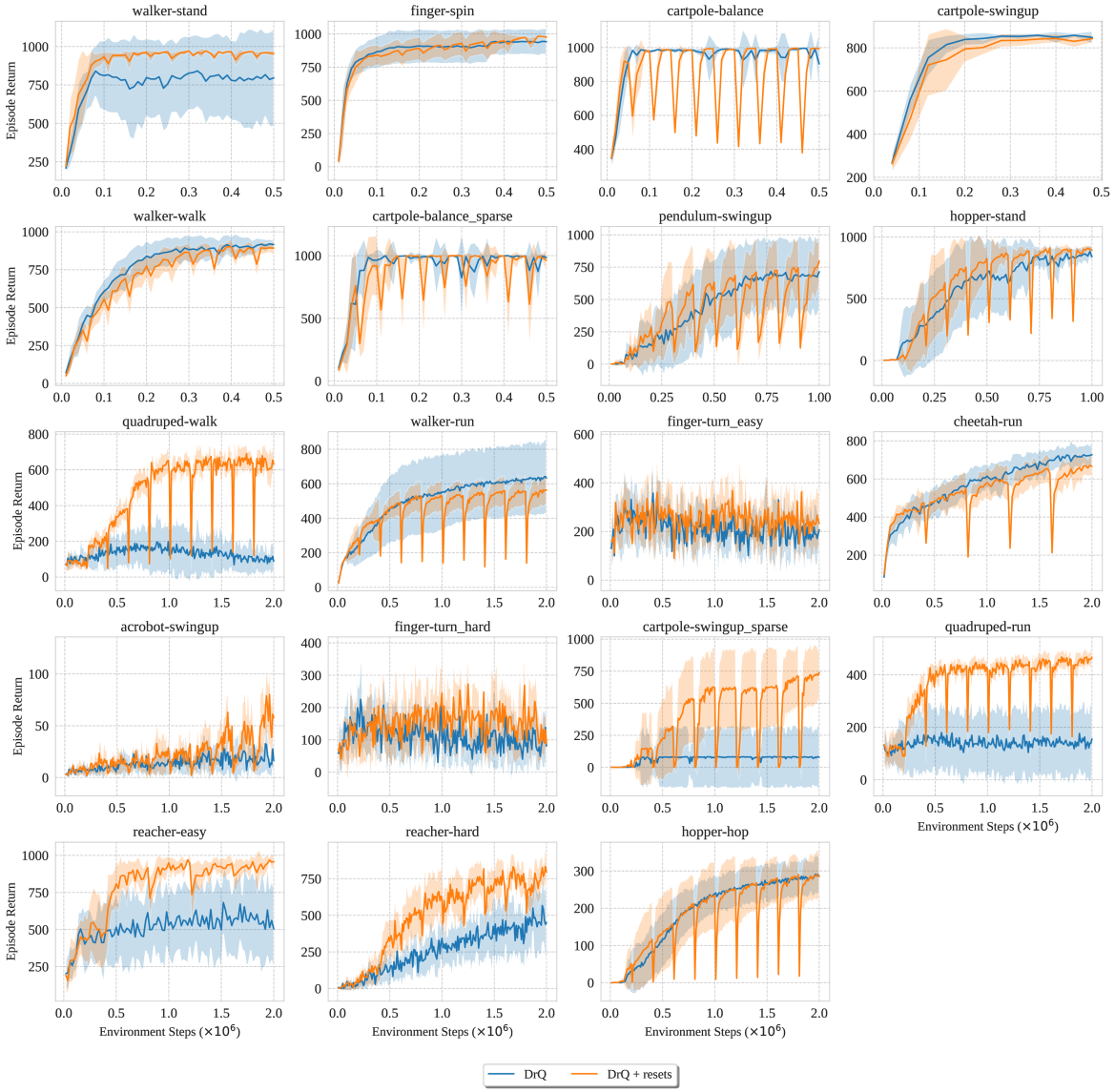


Figure 12. Per-environment training curves for DrQ with and without resets on DMC.

	Random	Human	DER	DrQ(ϵ)	SPR	SPR+resets
Alien	227.8	7127.7	802.3	865.2	901.2	**911.2**
Amidar	5.8	1719.5	125.9	137.8	**225.4**	201.7
Assault	222.4	742.0	561.5	579.6	658.6	**953.0**
Asterix	210.0	8503.3	535.4	763.6	**1095.0**	1005.8
Bank Heist	14.2	753.1	185.5	232.9	484.7	**547.0**
Battle Zone	2360.0	37187.5	8977.0	10165.3	**10873.5**	8821.2
Boxing	0.1	12.1	-0.3	9.0	27.6	**32.2**
Breakout	1.7	30.5	9.2	19.8	16.9	**23.4**
Chopper Command	811.0	7387.8	925.9	844.6	1454.0	**1680.6**
Crazy Climber	10780.5	35829.4	**34508.6**	21539.0	23596.9	28936.2
Demon Attack	152.1	1971.0	627.6	1321.5	1291.7	**2778.0**
Freeway	0.0	29.6	**20.9**	20.3	9.7	18.0
Frostbite	65.2	4334.7	871.0	1014.2	1746.2	**1834.3**
Gopher	257.6	2412.5	467.0	621.6	642.4	**930.4**
Hero	1027.0	30826.4	6226.0	4167.9	**7554.5**	6735.6
Jamesbond	29.0	302.8	275.7	349.1	383.2	**415.7**
Kangaroo	52.0	3035.0	581.7	1088.4	1674.8	**2190.6**
Krull	1598.0	2665.5	3256.9	4402.1	3412.1	**4772.4**
Kung Fu Master	258.5	22736.3	6580.1	11467.4	**16688.6**	14682.1
Ms Pacman	307.3	6951.6	1187.4	1218.1	**1334.1**	1324.6
Pong	-20.7	14.6	-9.7	-9.1	**2.1**	-9.0
Private Eye	24.9	69571.3	72.8	3.5	76.1	**82.2**
Qbert	163.9	13455.0	1773.5	1810.7	3816.2	**3955.3**
Road Runner	11.5	7845.0	11843.4	11211.4	**13588.5**	13088.2
Seaquest	68.4	42054.7	304.6	352.3	519.7	**655.6**
Up N Down	533.4	11693.2	3075.0	4324.5	8873.4	**60185.0**
Median HNS	0.000	1.000	0.189	0.313	0.453	**0.512**
Mean HNS	0.000	1.000	0.350	0.465	0.579	**0.901**
#Games > Human	0	0	2	3	4	**7**
#Games > 0	0	28	25	25	**26**	**26**

Tableau 5. Raw per-game scores and aggregate human-normalized scores (HNS) for SPR with resets and other methods on all 26 games in the Atari 100k benchmark. We report performance for SPR and SPR + resets from our codebase, averaged over 20 random seeds per game; other scores are taken from Agarwal et al. (2021b) and use 100 random seeds.

Annexe C

Appendix for Chapter 5

C.1. Definitions from Related Works

Tableau 1. Definitions of coinciding and related phenomena from previous work justifying the effectiveness of our strategy for replay ratio scaling.

Expression	Definition	Used In
Damage from Warm-Starting	[Phenomenon for which] "a warm-started network performs worse on test samples than a network trained on the same data but with a new random initialization"	Ash et al. (2020)
Damage from Non-Stationarity	"A memory effect where these transient non-stationarities can permanently impact the latent representation and adversely affect generalisation performance"	Igl et al. (2021)
Capacity Loss	"Reduced ability to fit new targets in deep neural networks"	Lyle et al. (2022b), Lyle et al. (2022a)
Loss of Plasticity	"Loss of the ability of the model to keep learning"	Berariu et al. (2021), Dohare et al. (2022)
Primacy Bias	"A tendency to overfit initial experiences that damages the rest of the learning process"	Nikishin et al. (2022)

To further clarify our description of previous work from the related work section, we report in Table 1 definitions of the different terms used to refer to the loss of the ability to learn and generalize in neural networks. Each definition is directly taken from one of the papers corresponding to it. Note that, despite their overlap, they reflect slightly different perspectives on the nature of this phenomenon, and it can be worth for future investigations to pin down which one of these is more relevant for replay ratio scaling or reinforcement learning as a whole.

C.2. Additional Experimental Results

C.2.1. Additional Studies

Reset Interval. An important hyperparameter for both SR-SAC and SR-SPR is the interval at which resets are performed, as denominated in terms of number of agent updates. In Figure 1, we study how performance is impacted by this choice, at different replay ratios. Overall, both SR-SAC and SR-SPR perform well for a vast range of reset intervals, with favorable replay ratio scaling and generally smooth performance

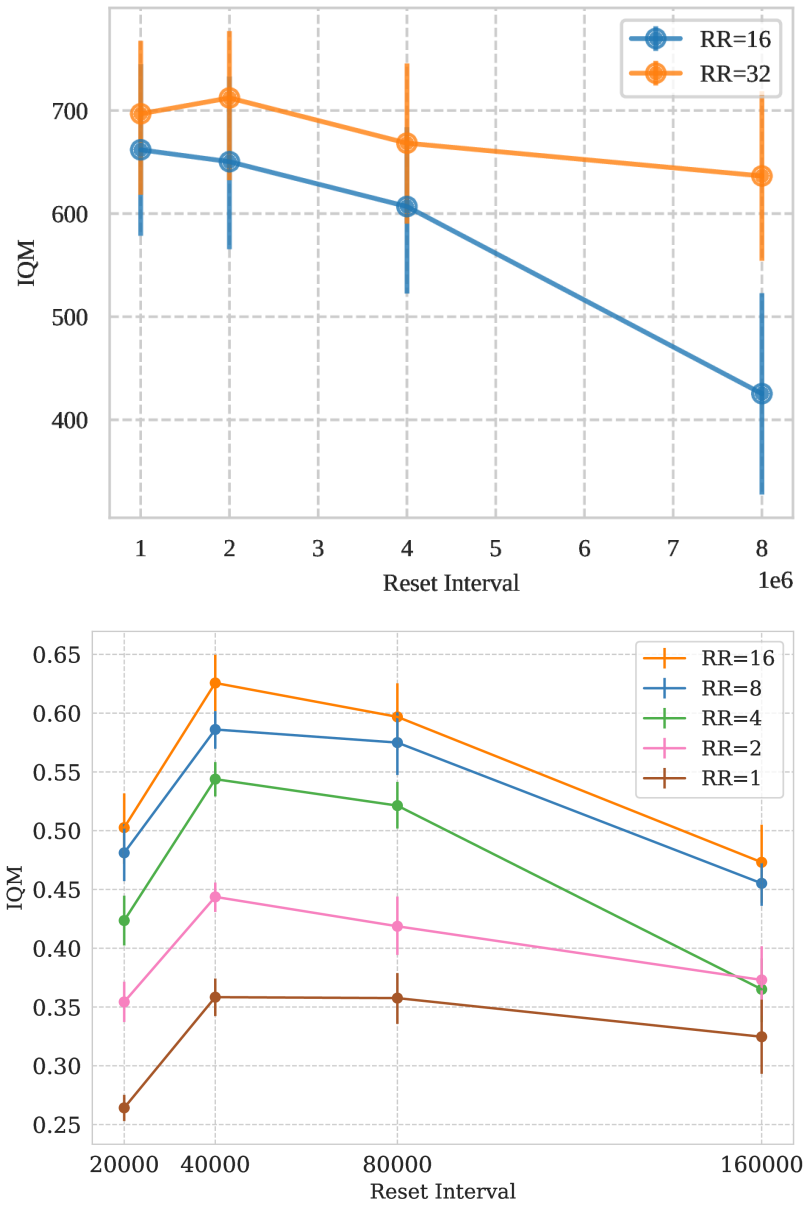


Figure 1. Sensitivity of the IQM to varying reset intervals (in terms of gradient updates) of SR-SAC on the DeepMind Control Suite (DMC15-500k) benchmark, and of SR-SPR on the Atari 100k benchmark. (10 seeds, 95% bootstrapped C.I.).

degradation. Note that, for large intervals (e.g., the last point on the right for SR-SAC with $RR = 16$, and last two points in the bottom right for SR-SPR), this is equivalent to actually performing no resets, just running the unmodified baseline algorithms. Thus, performance experiences non-smooth drops only in these easily avoidable cases.

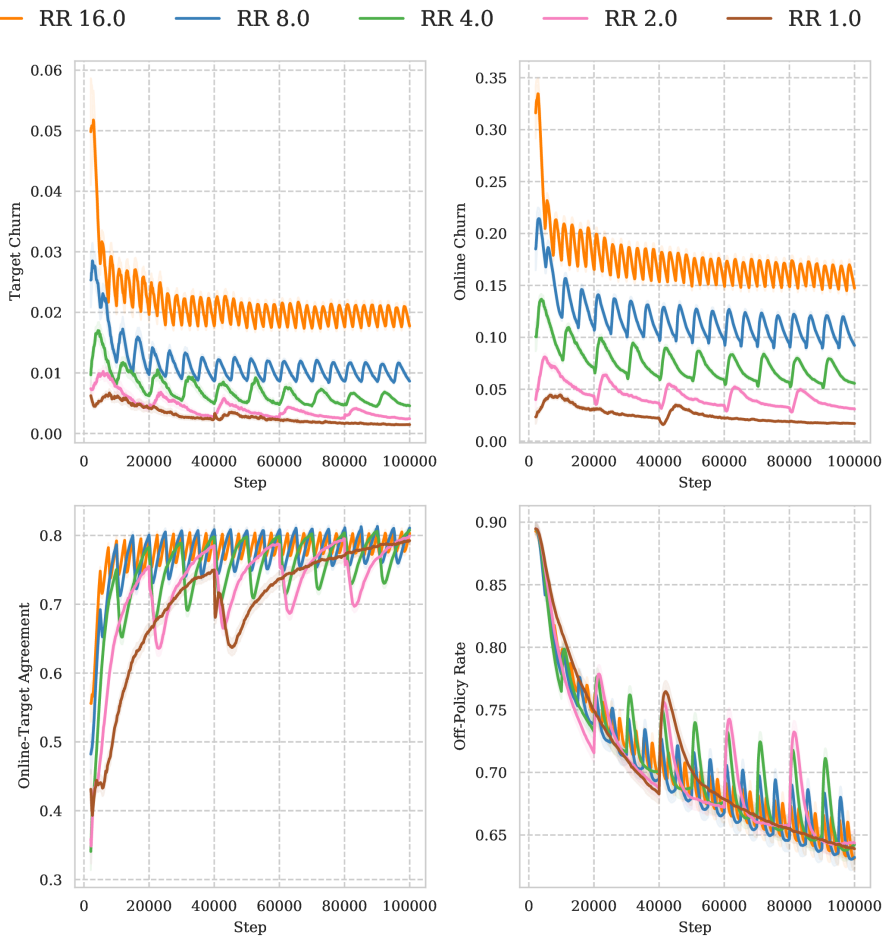


Figure 2. Churn-related diagnostics (based on the policy churn definition from Schaul et al. (2022)) for the online and target networks. Different colors and RR denote different values of replay ratio.

Counteracting Policy Churn by Acting with the Target Network. Schaul et al. (2022) defined the policy churn as the change in the agent's policy due to optimization. It was shown that a certain amount of policy churn can be beneficial for exploration in the absence of external noise (e.g., coming from ϵ-greedy exploration); however, it is intuitive that excessive churn can actually hurt performance, for instance by breaking the the consistency of trajectories. We hypothesize that mitigating excessive policy churn is a major reason why SR-SPR performs better when actions are selected with the target network rather than the online network. Figure 2 shows that, if we measure churn before each interaction with the environment, increasing the replay ratio will naturally increase it, while acting with the target network will decrease it. When the replay ratio is too high, it is likely that the benefits coming from additional offline computations might be nullified by the inconsistency in the exploration data; acting

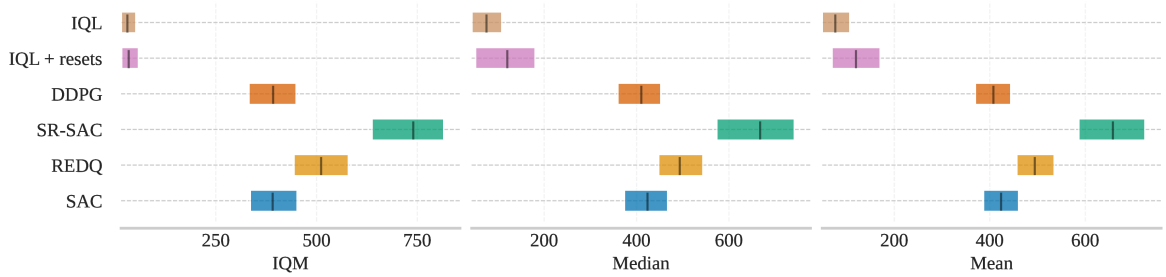


Figure 3. Performance on DMC15-500k of running IQL (RR=32) online, with and without resets.

with the target network reduces these inconsistencies without giving up on the more efficient optimization, at the cost of introducing minimal delays in the improvement of the data-collecting policy.

Using an offline RL algorithm online. In Section 5.6.1, we conducted a set of experiments with the goal of highlighting the importance of online interactions, concluding that a consistent stream of online interaction data and neural networks able to learn and generalize from a dataset of experiences are key factors behind effective replay ratio scaling. Online RL algorithms such as SAC are naturally reliant on the online stream of interactions; but it is natural to ask whether algorithms created for offline RL setting, where no interaction with the environment is assumed to be possible during training, can make the most out of the computational budget granted through high replay ratios in the online setting. To test this hypothesis, we run the Implicit Q-learning (IQL) (Kostrikov et al., 2022) algorithm as an online RL algorithm on DMC15-500k with a replay ratio of 32. Results in Figure 3 show that, despite being by design more robust to more aggressive training, the conservative nature of offline RL algorithms makes them not amenable to effective online learning, regardless of the presence of resets. This shows the effectiveness of online interactions as a strong supervision mechanism, able, when supported by resets, to make online RL algorithms robust to high replay ratios without the need of overly conservative behaviors.

C.2.2. Finetuning Pretrained Representations

Finetuning pretrained representations has become increasing common in reinforcement learning and elsewhere (Schwarzer et al., 2021b; Liu et al., 2021a; Bommasani et al., 2021). One obvious question to ask is whether or not replay scaling as demonstrated in the tabula rasa setting here can also be used to make this finetuning more sample efficient. In Figure 4 we answer this question in the affirmative. We initialize SPR

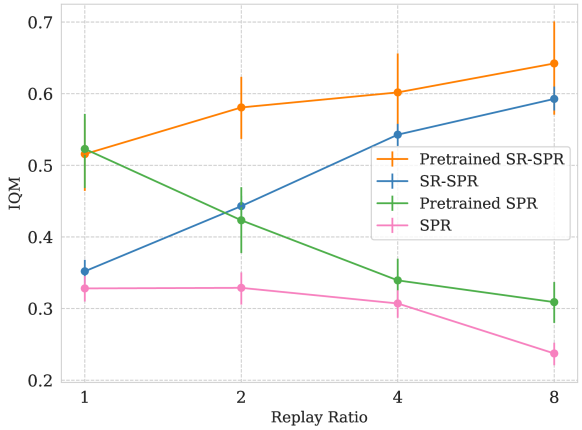


Figure 4. The performance of SR-SPR and SPR from scratch and when fine-tuning a pre-trained encoder on Atari 100k (10 seeds. 95% bootstrapped C.I.

and SR-SPR with pretrained encoders (taken for experimental convenience from SPR agents trained at replay ratio 1 for one million steps), and initialize all other parameters randomly. We then train at a range of replay ratios for 100k steps. For SR-SPR, we apply shrink and perturb towards the pretrained encoder weights rather than random parameters, but otherwise train as normal.

We find that while both SPR and SR-SPR benefit from the pretrained representations at low replay ratios, only SR-SPR is able to improve fine-tuning performance by replay scaling. Standard SPR with pretrained representations rapidly degrades in performance as the replay ratio is increased, while the performance of SR-SPR steadily increases at higher replay ratios. Although the gap between SR-SPR with and without pretraining closes somewhat at higher replay ratios, this is to be expected, as higher replay-ratio agents have more opportunities to improve their own representations even without pretraining.

C.2.3. Finetuning After Offline Training

The efficiency of SR-SAC and SR-SPR makes the general approach behind their design potentially appealing for the setting of offline RL with an additional fine tuning phase. In Figure 5, we provide preliminary evidence that the paradigm we advocated for in this paper may indeed be particularly beneficial in this setting. We test IQL with the same pretraining scheme presented in Kostrikov et al. (2022), consisting in a million offline training steps followed by a million interactions with the environment for fine tuning. We implement SR-IQL by using a replay ratio of 10 and resetting, every two

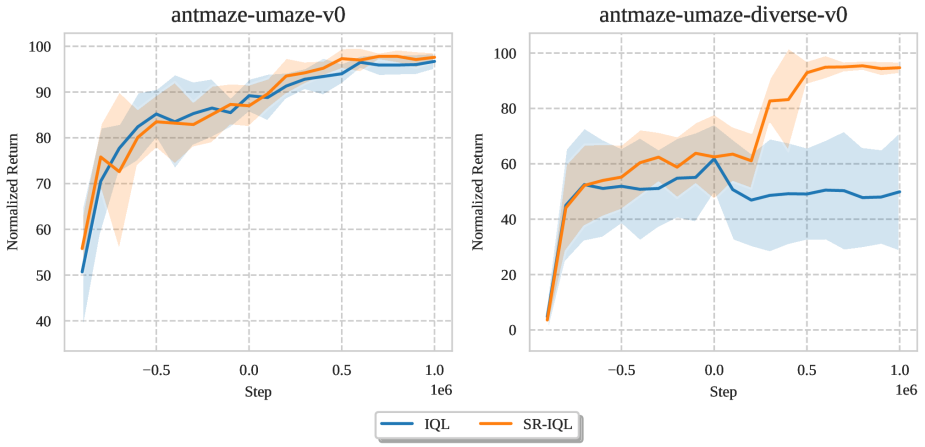


Figure 5. Performance of SR-IQL and IQL in two tasks from D4RL. Negative steps denotes the pretraining phase (10 seeds, ± std).

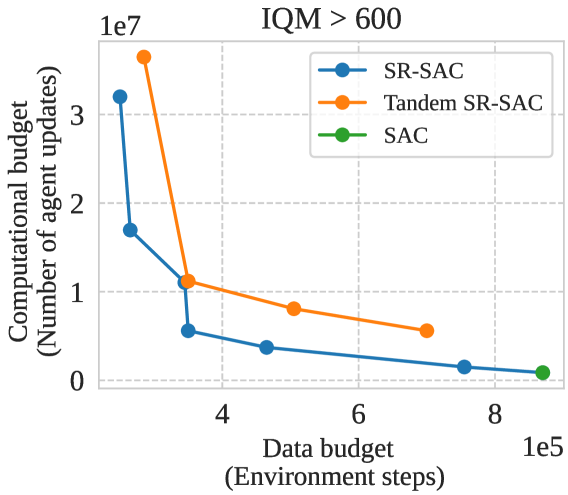


Figure 6. Pareto fronts for SR-SAC and Tandem SR-SAC on DMC15 (5 seeds).

million updates, all of the parameters of its neural networks during the fine tuning phase. We compare SR-IQL to IQL on two tasks from the D4RL benchmark (Fu et al., 2020). As shown in Figure 5, SR-IQL is roughly on par with IQL in the `antmaze-umaze-v0` task, but reaches superior performance during fine tuning in `antmaze-umaze-diverse-v0`. We believe this sets the stage to experimenting with our replay ratio scaling paradigm in this setting as a promising research direction for future work.

C.2.4. Pareto Fronts Comparison

With the same approach used for studying the data/computation tradeoffs of SR-SAC, it also becomes possible to directly compare the performance of different replay ratio-scalable algorithms. As a simple example, we compare SR-SAC, its tandem version and SAC in Figure 6. The different lines are Pareto curves, obtained by retaining the points that are dominating the other ones in terms of either data or computational budget, to reach an IQM of at least 600. On this plot, SAC simply appears as a point because, not allowing for effective replay ratio scaling, it can only reach the prescribed performance by using more data and a relatively small amount of computational resources.

C.2.5. Comparison with Neural Fitted Q-Iteration

The approach we demonstrated for replay ratio scaling, for its relationship with offline RL and its use of resets, could resemble the classic NFQI algorithm (Riedmiller, 2005), which train from scratch, after each large batch of transitions, a Q-function. Our approach propagates information across resets mainly through the use of the replay buffer, having a fast target network updated alongside the regular agent training; NFQI instead propagates information primarily through a target network, which is updated once per reset. We implement an-friendly variant of NFQI on SR-SPR at replay ratio 16, performing one target network update upon each reset. However, we find that this leads to very poor performance (IQM 0.350), achieving barely half that of standard SR-SPR. Although we hypothesized that 2,500 environment steps (the standard reset interval for SR-SPR at replay ratio 16) might be too infrequent for target network updates, making this interval shorter did not improve performance. Although we cannot rule out the possibility that NFQI might be competitive at dramatically higher replay ratios, its inherent slowness in propagating information is likely to lead it to lag in data-efficient settings; any reset interval that is sufficiently long to allow for accurate estimation of the value function may lead to insufficiently rapid value propagation via target updates, and vice versa.

C.3. Computational Considerations

For DMC, the running time depends on the individual environment, due to differences in dimensionality of the observation as well as physics simulation time. On an

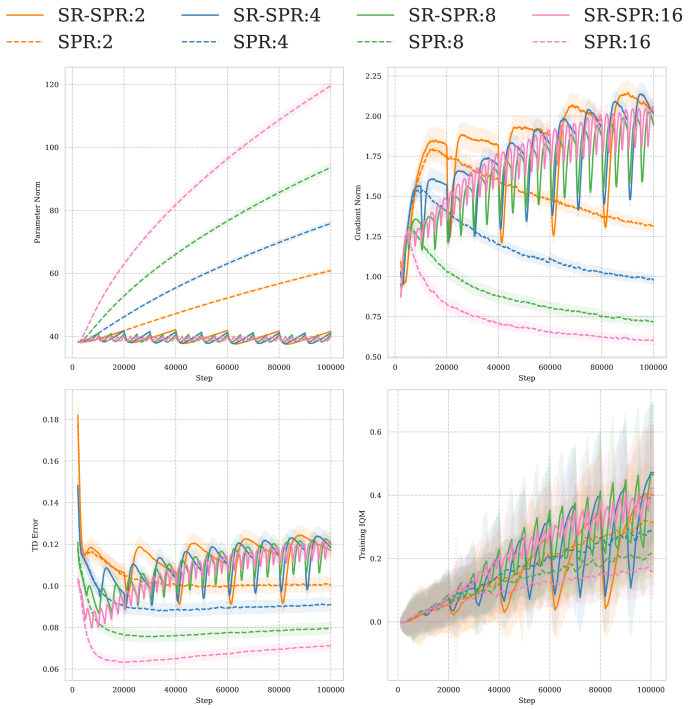


Figure 7. Learning curves for SR-SPR (solid) and SPR (dashed) at various replay ratios. Note that all SR-SPR runs converge to similar TD errors, gradient norms and parameter norms, while these metrics greatly differ for SPR at different replay ratios. IQM training performance does not match evaluation performance, as ongoing training episodes are often disrupted by the reset procedure.

NVIDIA V100 GPU, at this highest replay ratio of RR=128, our code takes about 10.5 hours on `acrobot-swingup` and about 15 hours on `humanoid-run` to complete 500k environment steps. For a replay ratio of RR=32, which yields remarkable, even if not best, performance, the time goes down to just about 3 hours and about 4 hours respectively. which is well-below the typical demands of modern model-based RL methods. With careful seed parallelization, running SR-SAC with RR=32 for 5 seeds for all tasks in the DMC15-500k benchmark takes less than 4 GPU/days on an NVIDIA V100. For Atari 100k, running time depends primarily on the replay ratio chosen. At the highest replay ratio used (16) and with five seeds running in parallel, our code takes roughly 25 hours to complete 100k steps on an NVIDIA A100, yielding a cost of roughly 5 GPU/hours per training run.

Tableau 2. Scores for all games in Atari 100k for SR-SPR and competing algorithms at various replay ratios.

Game	Random	Human	IRIS	SR-SPR:2	SR-SPR:4	SR-SPR:8	SR-SPR:16
Alien	227.8	7127.7	420.0	877.9	964.4	1015.5	**1107.8**
Amidar	5.8	1719.5	143.0	189.2	**211.8**	203.1	203.4
Assault	222.4	742.0	**1524.4**	891.9	987.3	1069.5	1088.9
Asterix	210.0	8503.3	853.6	836.7	894.2	**916.5**	903.1
Bank Heist	14.2	753.1	53.1	253.6	460.0	472.3	**531.7**
Battle Zone	2360.0	37187.5	13074.0	14493.5	17800.6	**19398.4**	17671.0
Boxing	0.1	12.1	**70.1**	36.1	42.0	46.7	45.8
Breakout	1.7	30.5	**83.7**	24.5	26.1	28.8	25.5
Chopper Command	811.0	7387.8	1565.0	1609.4	1933.7	2201.0	**2362.1**
Crazy Climber	10780.5	35829.4	**59324.2**	28004.7	38341.7	43122.3	45544.1
Demon Attack	152.1	1971.0	2034.4	2969.0	**3016.2**	2898.1	2814.4
Freeway	0.0	29.6	**31.1**	24.1	24.5	24.9	25.4
Frostbite	65.2	4334.7	259.1	1450.4	1809.9	1752.8	**2584.8**
Gopher	257.6	2412.5	**2236.1**	735.3	717.5	711.2	712.4
Hero	1027.0	30826.4	7037.4	6832.1	7195.7	7679.6	**8524.0**
Jamesbond	29.0	302.8	**462.7**	412.9	408.8	392.8	389.1
Kangaroo	52.0	3035.0	838.2	1651.2	2024.1	3254.9	**3631.7**
Krull	1598.0	2665.5	**6616.4**	5206.4	5364.3	5824.8	5914.4
Kung Fu Master	258.5	22736.3	**21759.8**	14165.6	17656.5	17095.6	18649.4
Ms Pacman	307.3	6951.6	999.1	1472.6	1544.7	1522.6	**1574.1**
Pong	-20.7	14.6	**14.6**	-10.5	-5.5	-3.0	2.9
Private Eye	24.9	69571.3	**100.0**	98.8	95.8	95.8	97.9
Qbert	163.9	13455.0	745.7	3431.7	3699.8	3850.6	**4044.1**
Road Runner	11.5	7845.0	9614.6	12199.0	**14287.3**	13623.5	13463.4
Seaquest	68.4	42054.7	661.3	714.7	766.6	800.5	**819.0**
Up N Down	533.4	11693.2	3546.2	61851.2	91435.2	95501.1	**112450.3**
Games > Human	0	0	**9**	7	8	**9**	**9**
IQM (↑)	0.000	1.000	0.501	0.444	0.544	0.589	**0.632**
Optimality Gap (↓)	1.000	0.000	0.512	0.516	0.470	0.452	**0.433**
Median (↑)	0.000	1.000	0.289	0.336	0.523	0.560	**0.685**
Mean (↑)	0.000	1.000	1.046	0.910	1.111	1.188	**1.272**

C.4. Experimental Details

We report in Table 5 the full list of tasks for the DMC15 benchmark. Our implementation of continuous control algorithms is based on the jaxrl codebase (Kostrikov, 2021). For REDQ, we use the best hyperparameters, as recommended by Chen et al. (2021b), as well as a replay ratio of 20. For discrete control, we use a version of SPR

Tableau 3. Hyperparameters for SR-SPR. The ones introduced by this work are at the bottom.

Parameter	Setting
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	Variable (1, 2, 4, 8, 16)
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	512
Non-linearity	ReLU
Target network update period	1
λ (SPR loss coefficient)	2
K (SPR prediction depth)	5
Data Augmentation	Shifts ($\pm$4 pixels)
	Intensity(scale=0.05)
τ (EMA coefficient)	0.995
Reset Interval (gradient steps)	40,000
Layers getting hard reset	Final 2
Shrink and Perturb α	0.8
Action selection	Target network

implemented in Jax (Bradbury et al., 2018) in Dopamine (Castro et al., 2018). See Table 3 for a full list of the employed hyperparameters.

Tableau 4. Hyperparameters for SR-SAC. The ones introduced by this work are at the bottom.

Parameter	Setting
Discount factor	0.99
Minibatch size	256
Optimizer (all)	Adam
Optimizer (all): learning rate	0.0003
Optimizer (all): β_1	0.9
Optimizer (all): β_2	0.999
Optimizer (all): ϵ	0.00015
Networks (all): activation	ReLU
Networks (all): # hidden layers	2
Networks (all): hidden units	256
Initial Temperature	1
Replay Buffer Size	10^6
Updates per step	Variable (1 to 128)
Target network update period	1
τ (EMA coefficient)	0.995
Reset Interval (gradient steps)	2560000
Layers getting hard reset	All

Tableau 5. The tasks from the DMC15 benchmark. We chose commonly-employed DMC tasks for which the optimal policy is not immediately found by SAC according to https://github.com/denisyarats/pytorch_sac#results.

Environment	Tasks
`walker`	`run`
`quadruped`	`run`, `walk`
`reacher`	`hard`
`humanoid`	`run`, `walk`, `stand`
`swimmer`	`swimmer6`
`cheetah`	`run`
`hopper`	`hop`, `stand`
`acrobot`	`swingup`
`pendulum`	`swingup`
`finger`	`turn_hard`
`fish`	`swim`

C.4.1. Full Experimental Results

In Figure 8, we show the scaling curve for DMC15-1M.

We report in Table 2 the full per-game results for SR-SPR and in Figure 9,10,11,12,13 full experimental results for SR-SAC. For completeness, we also report the performance of the modified settings and of REDQ at the same replay ratios.

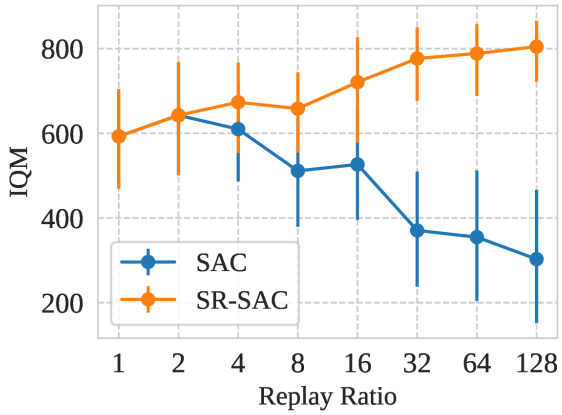


Figure 8. Scaling curve for SR-SAC and SAC on DMC15-1M.

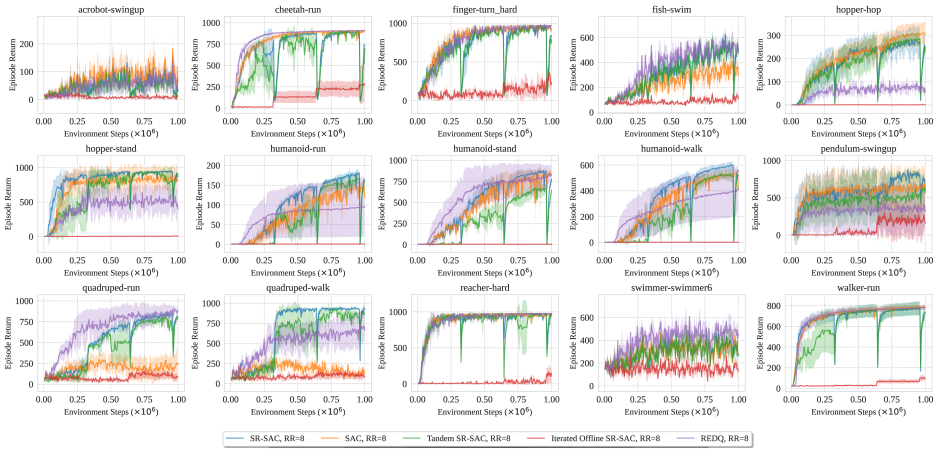


Figure 9. Evaluation Returns on individual DMC15 environments for replay ratio 8.

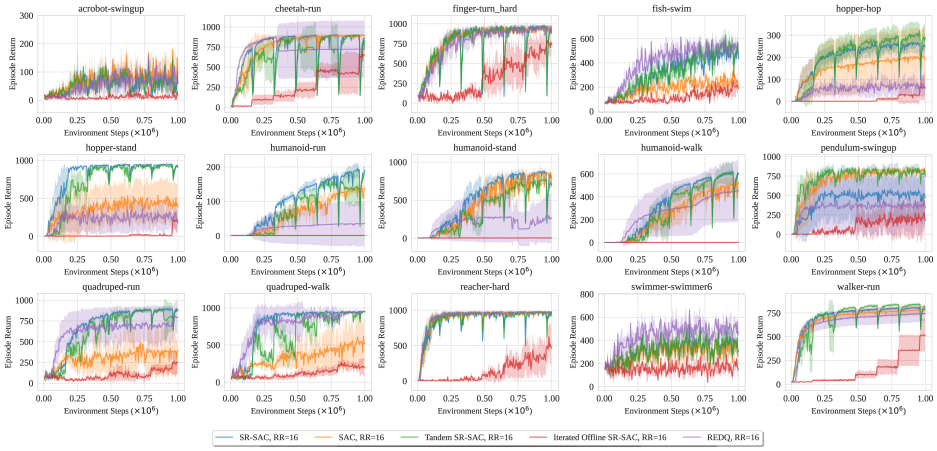


Figure 10. Evaluation Returns on individual DMC15 environments for replay ratio 16.

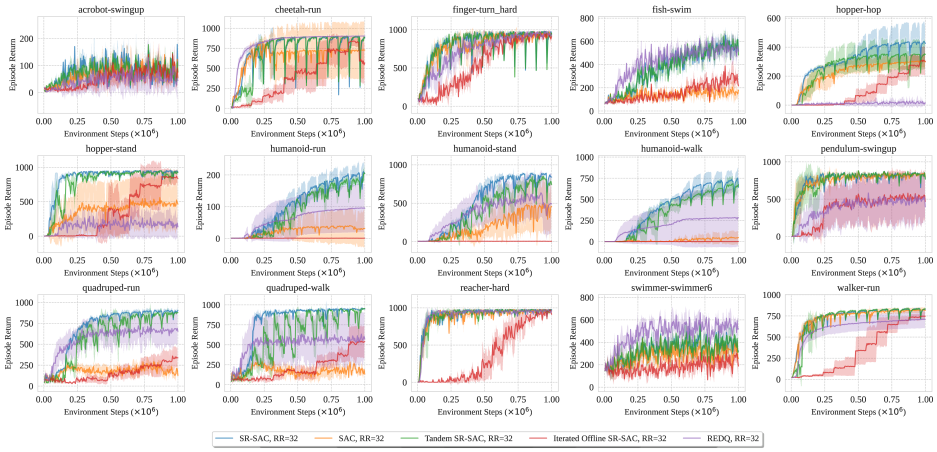


Figure 11. Evaluation Returns on individual DMC15 environments for replay ratio 32.

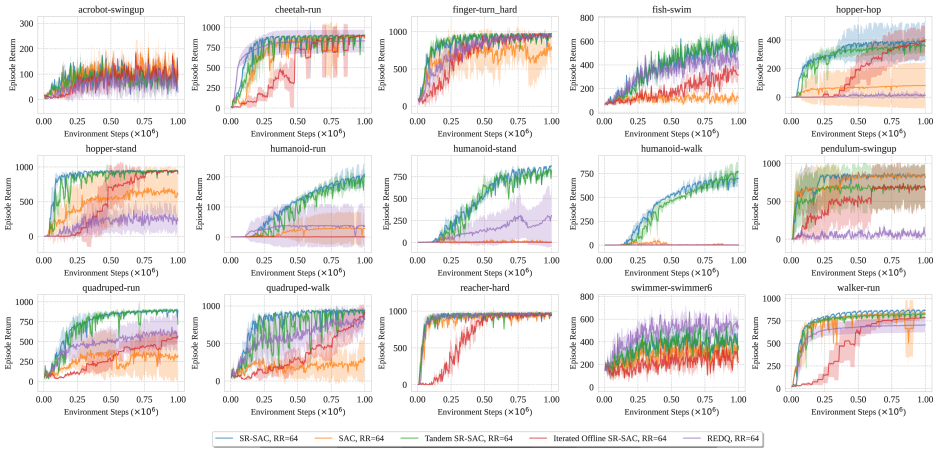


Figure 12. Evaluation Returns on individual DMC15 environments for replay ratio 64.

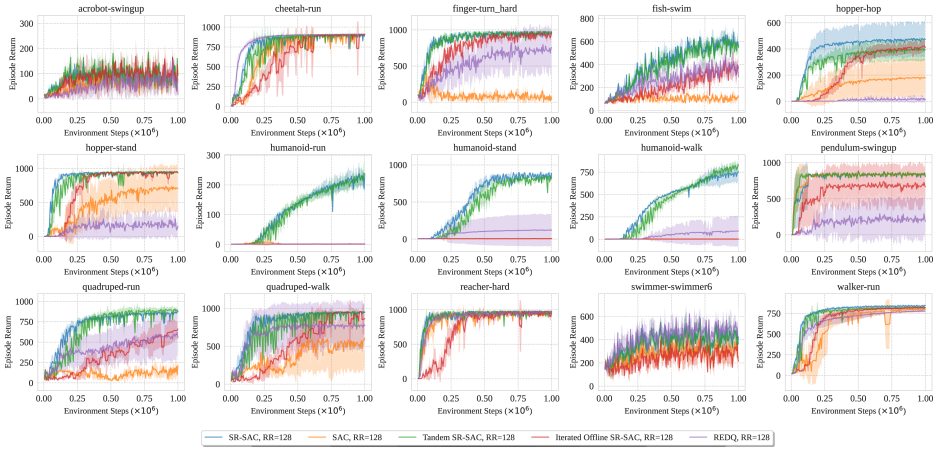


Figure 13. Evaluation Returns on individual DMC15 environments for replay ratio 128.

Annexe D

Appendix for Chapter 6

	Random	Human	DER	DrQ(ϵ)	SPR	IRIS	SR-SPR	EfficientZero	BBF
Alien	227.8	7127.7	802.3	865.2	841.9	420.0	1107.8	808.5	**1173.2**
Amidar	5.8	1719.5	125.9	137.8	179.7	143.0	203.4	148.6	**244.6**
Assault	222.4	742.0	561.5	579.6	565.6	1524.4	1088.9	1263.1	**2098.5**
Asterix	210.0	8503.3	535.4	763.6	962.5	853.6	903.1	**25557.8**	3946.1
BankHeist	14.2	753.1	185.5	232.9	345.4	53.1	531.7	351.0	**732.9**
BattleZone	2360.0	37187.5	8977.0	10165.3	14834.1	13074.0	17671.0	13871.2	**24459.8**
Boxing	0.1	12.1	-0.3	9.0	35.7	70.1	45.8	52.7	**85.8**
Breakout	1.7	30.5	9.2	19.8	19.6	83.7	25.5	**414.1**	370.6
ChopperCommand	811.0	7387.8	925.9	844.6	946.3	1565.0	2362.1	1117.3	**7549.3**
CrazyClimber	10780.5	35829.4	34508.6	21539.0	36700.5	59324.2	45544.1	**83940.2**	58431.8
DemonAttack	152.1	1971.0	627.6	1321.5	517.6	2034.4	2814.4	13003.9	**13341.4**
Freeway	0.0	29.6	20.9	20.3	19.3	**31.1**	25.4	21.8	25.5
Frostbite	65.2	4334.7	871.0	1014.2	1170.7	259.1	**2584.8**	296.3	2384.8
Gopher	257.6	2412.5	467.0	621.6	660.6	2236.1	712.4	**3260.3**	1331.2
Hero	1027.0	30826.4	6226.0	4167.9	5858.6	7037.4	8524.0	**9315.9**	7818.6
Jamesbond	29.0	302.8	275.7	349.1	366.5	462.7	389.1	517.0	**1129.6**
Kangaroo	52.0	3035.0	581.7	1088.4	3617.4	838.2	3631.7	724.1	**6614.7**
Krull	1598.0	2665.5	3256.9	4402.1	3681.6	6616.4	5911.8	5663.3	**8223.4**
KungFuMaster	258.5	22736.3	6580.1	11467.4	14783.2	21759.8	18649.4	**30944.8**	18991.7
MsPacman	307.3	6951.6	1187.4	1218.1	1318.4	999.1	1574.1	1281.2	**2008.3**
Pong	-20.7	14.6	-9.7	-9.1	-5.4	14.6	2.9	**20.1**	16.7
PrivateEye	24.9	69571.3	72.8	3.5	86.0	**100.0**	97.9	96.7	40.5
Qbert	163.9	13455.0	1773.5	1810.7	866.3	745.7	4044.1	**14448.5**	4447.1
Roadrunner	11.5	7845.0	11843.4	11211.4	12213.1	9614.6	13463.4	17751.3	**33426.8**
Seaquest	68.4	42054.7	304.6	352.3	558.1	661.3	819.0	1100.2	**1232.5**
UpNDown	533.4	11693.2	3075.0	4324.5	10859.2	3546.2	**112450.3**	17264.2	12101.7
Games > Human	0	0	2	3	6	9	9	**14**	12
IQM (↑)	0.000	1.000	0.183	0.280	0.337	0.501	0.631	1.020	**1.045**
Optimality Gap (↓)	1.000	0.000	0.698	0.631	0.577	0.512	0.433	0.371	**0.344**
Median (↑)	0.000	1.000	0.189	0.313	0.396	0.289	0.685	**1.116**	0.917
Mean (↑)	0.000	1.000	0.350	0.465	0.616	1.046	1.272	1.945	**2.247**

Tableau 1. Scores and aggregate metrics for BBF and competing methods across the 26 Atari 100k games. Scores are averaged across 50 seeds per game for BBF, 30 for SR-SPR, 5 for IRIS, 3 for EfficientZero, and 100 for others.

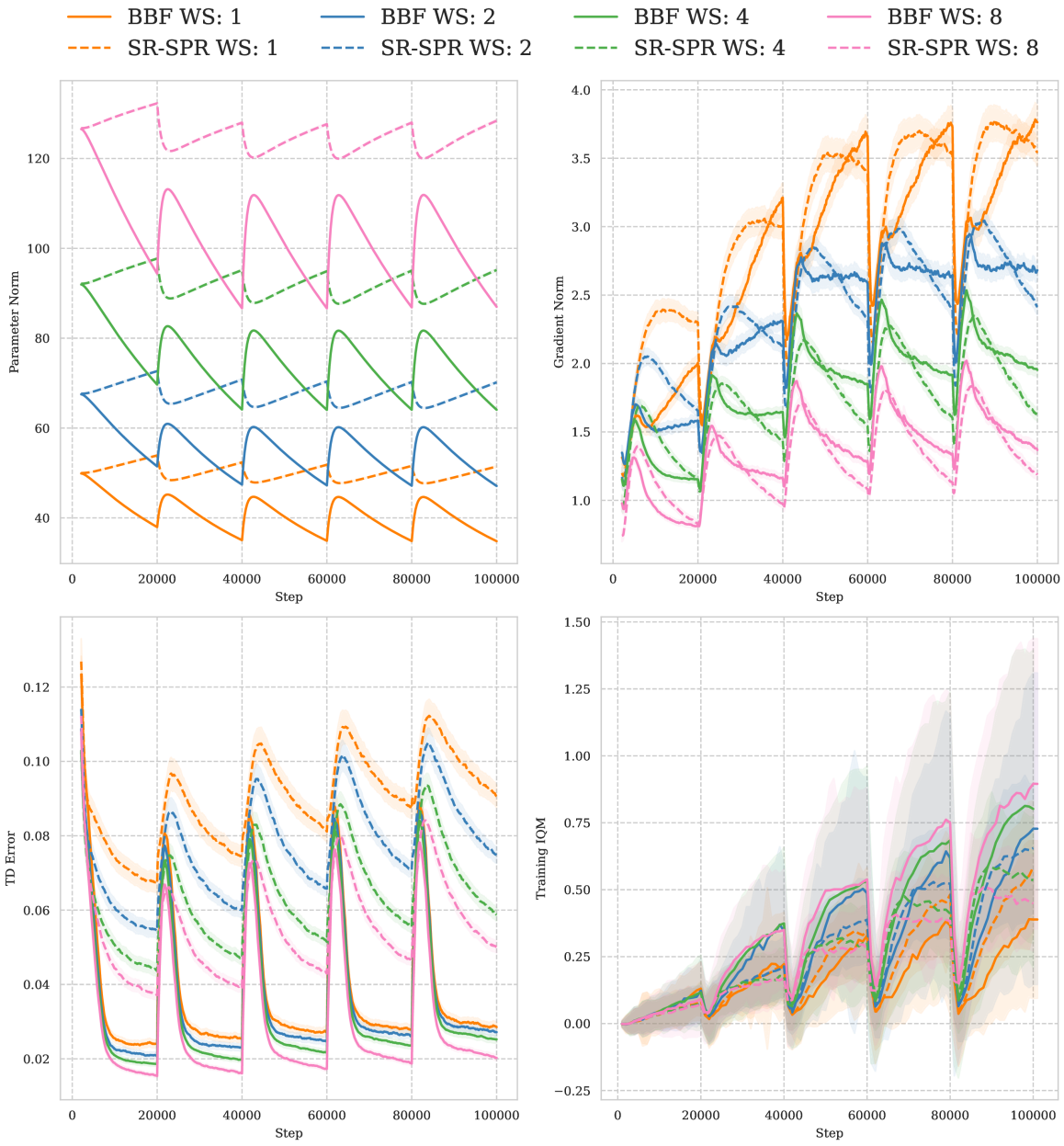


Figure 1. Learning curves for BBF and SR-SPR at RR=2 with a ResNet encoder at various width scales, on the 26 Atari 100k games. Larger networks consistently have lower TD errors and higher gradient norms, and higher parameter norms, but only BBF translates this to higher environment returns. The large, systematic difference in TD error between BBF and SR-SPR is due to BBF's use of a shorter update horizon, which makes each step of the TD backup easier to predict.

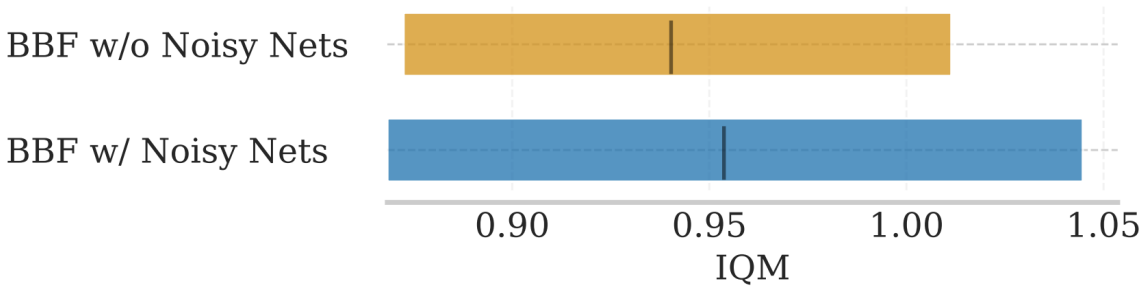


Figure 2. BBF at RR=2 on the 26 Atari 100k tasks, with and without Noisy Nets.