

Université de Montréal

Self-Supervision for Reinforcement Learning

par

Ankesh Anand

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 Discipline

March 29, 2024

© Ankesh Anand, 2023

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée

Self-Supervision for Reinforcement Learning

présentée par

Ankesh Anand

a été évaluée par un jury composé des personnes suivantes :

Yoshua Bengio

(président-rapporteur)

Aaron Courville

(directeur de recherche)

Devon Hjelm

(codirecteur)

Glen Besnerth

(membre du jury)

Animesh Garg

(examineur externe)

Yoshua Bengio

(representant du doyen de la FESP)

SOMMAIRE

Cette thèse tente de construire de meilleurs agents d'apprentissage par renforcement (RL) en tirant parti de l'apprentissage auto-supervisé. Il se présente sous la forme d'une thèse par article qui contient trois travaux.

Dans le premier article, nous construisons un benchmark basé sur les jeux Atari pour évaluer systématiquement les méthodes d'apprentissage auto-supervisé dans les environnements RL. Nous comparons un éventail de ces méthodes à travers une suite de tâches de sondage pour identifier leurs forces et leurs faiblesses. Nous montrons en outre qu'une nouvelle méthode contrastive ST-DIM excelle à capturer la plupart des facteurs génératifs dans les environnements étudiés, sans avoir besoin de s'appuyer sur des étiquettes ou des récompenses.

Dans le deuxième article, nous proposons des représentations auto-prédictives (SPR) qui apprennent un modèle latent auto-supervisé de la dynamique de l'environnement parallèlement à la résolution de la tâche RL en cours. Nous montrons que SPR réalise des améliorations spectaculaires dans l'état de l'art sur le benchmark Atari 100k difficile où les agents n'ont droit qu'à 2 heures d'expérience en temps réel.

Le troisième article étudie le rôle de la RL basée sur un modèle et de l'apprentissage auto-supervisé dans le contexte de la généralisation en RL. Grâce à des contrôles minutieux, nous montrons que la planification et l'apprentissage de représentation basé sur un modèle contribuent tous deux à une meilleure généralisation pour l'agent Muzero. Nous améliorons encore MuZero avec des objectifs d'apprentissage auto-supervisés auxiliaires, et montrons que cet agent MuZero++ obtient des résultats de pointe sur les benchmarks Procgen et Metaworld.

Mots-clés: Apprentissage en profondeur, Apprentissage auto-supervisé, Apprentissage par renforcement

SUMMARY

This thesis tries to build better Reinforcement Learning (RL) agents by leveraging self-supervised learning. It is presented as a thesis by article that contains three pieces of work.

In the first article, we construct a benchmark based on Atari games to systematically evaluate self-supervised learning methods in RL environments. We compare an array of such methods across a suite of probing tasks to identify their strengths and weaknesses. We further show that a novel contrastive method ST-DIM excels at capturing most generative factors in the studied environments, without needing to rely on labels or rewards.

In the second article, we propose Self-Predictive Representations (SPR) that learns a self-supervised latent model of the environment dynamics alongside solving the RL task at hand. We show that SPR achieves dramatic improvements in state-of-the-art on the challenging Atari 100k benchmark where agents are allowed only 2 hours of real-time experience.

The third article studies the role of model-based RL and self-supervised learning in the context of generalization in RL. Through careful controls, we show that planning and model-based representation learning both contribute towards better generalization for the Muzero agent. We further improve MuZero with auxiliary self-supervised learning objectives, and show that this MuZero++ agent achieves state-of-the-art results on the Procgen and Metaworld benchmarks.

Keywords: Deep Learning, Reinforcement Learning, Self-Supervised Learning

ACKNOWLEDGEMENTS

First and foremost, I am extremely thankful to my advisor, Aaron Courville for his constant mentorship, support and encouragement throughout my journey. Aaron has been the best advisor I could ask for, helping me ask the right research questions, providing a lot of intellectual freedom, and enabling and fostering collaborations during my PhD. I am also thankful to my co-advisor Devon Hjelm for who advised me on multiple projects, and highly influenced my foray into self-supervised learning. I'd also like to thank Hugo Larochelle, who guided me through multiple projects early in my PhD and helped me internalise valuable lessons on being a good researcher.

I am grateful to have been given the opportunity to pursue a PhD at Mila. Mila is a special place bursting with scientific curiosity and the collective goal of moving AI forward, and has been a constant source for me to find inspiring ideas and wonderful collaborators. Thanks to Aaron for giving me this opportunity, and Linda and Celine, for helping me navigate the administrative hurdles.

I am thankful to several of my wonderful friends throughout my PhD whose presence made this journey a lot more fulfilling. Konrad, Joseph, Chinwei, Michael, Rithesh, Sandeep, Vinayak, Hattie, Ethan and Nitarshan - you have all been a wonderful company! Thanks to my roommates - Sai, Nithin, Jae, Evan, and Vinayak for listening through my rants and providing a constant source of social support. Thanks to Sherjil who has been a great mentor, and helped me whenever I was stuck on a career or research question.

I've been very fortunate to have been mentored by several amazing researchers during my internships. I'd like to thank Phil Bachman and Marc-Alexandre Côté at Microsoft Research for shaping my thinking on difficult problems, and always providing detailed feedback. I'd also like to thank Jessica Hamrick at DeepMind who went out of her way in supporting me as a mentor and close collaborator during my internship, and helped me get better at communicating research.

I've had the utmost pleasure of working with the most wonderful collaborators I could ask for. Thank you Evan and Max for listening and believing in my ideas

(even the silly ones!), helping concretise them and driving multiple of our projects to completion. Thanks to Yoshua, Ethan, Simon, Florian, Eugene, Kyle, Sherjil, Phil, Jacob, Yazhe, Eszter, Julian and Theo for being great collaborators during my projects.

Finally I would like to thank my family back home for being constantly supportive and helping me succeed right from my childhood.

CONTENTS

1	Introduction	24
2	Background	26
2.1	Deep Learning	26
2.2	Reinforcement Learning	27
2.2.1	Model-free Reinforcement Learning	28
2.2.2	Model-based Reinforcement Learning	29
2.3	Representation Learning	30
2.4	Latent Representation Learning	30
2.4.1	Contrastive Methods	31
2.4.2	Bootstrapped Latent Methods	32
1	Article I	
3	Prologue to the First Article	34
3.1	Article Details	34
3.2	Context	34
3.3	Contributions	34
3.4	Research Impact	35
4	Unsupervised State Representation Learning in Atari	36
4.1	Introduction	36
4.2	Related Work	38
4.3	Spatiotemporal Deep Infomax	39
4.3.1	Maximizing mutual information across space and time	40
4.4	The <u>A</u> tari <u>A</u> nnotated <u>R</u> AM <u>I</u> nterface (AARI)	43
4.5	Experimental Setup	44
4.5.1	Data preprocessing and acquisition	45
4.5.2	Methods	45
4.5.3	Probing	46
4.6	Results	47
4.7	Discussion	48
4.8	Conclusion	51

ii Article II

5	Prologue to the Second Article	53
5.1	Article Details	53
5.2	Context	53
5.3	Contribution	53
5.4	Research Impact	54
6	Data-Efficient Reinforcement Learning with Self-Predictive Representations	55
6.1	Introduction	56
6.2	Method	58
6.2.1	Deep Q-Learning	58
6.2.2	Self-Predictive Representations	60
6.2.3	Transition Model Architecture	63
6.2.4	Data Augmentation	63
6.2.5	Implementation Details	63
6.3	Related Work	64
6.3.1	Data-Efficient RL:	64
6.3.2	Representation Learning in RL:	65
6.4	Results	65
6.4.1	Evaluation	67
6.5	Analysis	68
6.6	Future Work	70
6.7	Conclusion	71

iii Article III

7	Prologue to the Third Article	73
7.1	Article Details	73
7.2	Context	73
7.3	Contributions	73
7.4	Research Impact	74
8	Procedural Generalization by Planning with Self-Supervised World Models	75
8.1	Introduction	75
8.2	Motivation and Background	78
8.2.1	Generalization in RL	78
8.2.2	Factors of Generalization	79

8.2.3	MuZero	81
8.3	Experimental Design	82
8.3.1	Environments	82
8.3.2	Factors of Generalization	84
8.4	Results	86
8.4.1	Procedural Generalization	86
8.4.2	Task Generalization	89
8.5	Conclusion	91

iv Conclusion & Future Work

9	Conclusion	93
10	Future Work: Reinforcement Learning as a Fine-Tuning Paradigm	95
10.1	Why RL fine-tuning over other alternatives?	96

v Appendix

11	Appendix to the First Article	98
11.1	Architecture Details	98
11.2	Preprocessing and Hyperparameters	100
11.3	Results with Probes Trained on Data Collected By a Pretrained RL agent	101
11.4	More Detailed Ablation Results	101
11.5	Probing Pretrained RL Agents	101
11.5.1	Accuracy Metric	102
12	Appendix to the Second Article	110
12.1	Hyperparameters	110
12.2	Full Results	110
12.3	Controlled baselines	110
12.4	Comparison with a contrastive loss	111
12.5	The role of the target encoder in SPR	112
12.6	Wall Clock Times	114
13	Appendix to the Third Article	118
13.1	Agent Details	118
13.1.1	MuZero	118
13.2	Controlled Model-free Baseline	119
13.2.1	MuZero + Reconstruction	121
13.2.2	MuZero + Contrastive	121

13.2.3 MuZero + SPR	122
Bibliography	129

LIST OF FIGURES

Figure 1	Drawing of a one-dollar bill completely from memory.	31
Figure 2	Drawing subsequently made with a dollar bill present.	31
Figure 3	We use a collection of 22 Atari 2600 games to evaluate state representations. We leveraged the source code of the games to annotate the RAM states with important state variables such as the location of various objects in the game. We compare various unsupervised representation learning techniques based on how well the representations linearly-separate the state variables. Shown above are examples of state variables annotated for Montezuma’s Revenge and MsPacman.	37
Figure 4	A schematic overview of SpatioTemporal DeepInfoMax (ST-DIM). (a) shows the two different mutual information objectives: local infomax and global infomax. (b) shows a simplified version of the contrastive task we use to estimate mutual information. In practice, we use multiple negative samples.	40
Figure 5	InfoNCE vs JSD	49
Figure 6	Effect of Spatial Loss	49
Figure 7	Median and Mean Human-Normalized scores of different methods across 26 games in the Atari 100k benchmark (Kaiser et al., 2019), averaged over 10 random seeds for SPR, and 5 seeds for most other methods except CURL, which uses 20. Each method is allowed access to only 100k environment steps or 400k frames per game. (*) indicates that the method uses data augmentation. SPR achieves state-of-art results on both mean and median human-normalized scores. Note that, even without data augmentation, SPR still outperforms all prior methods on both metrics.	57

- Figure 8 An illustration of the full SPR method. Representations from the online encoder are used in the reinforcement learning task and for prediction of future representations from the target encoder via the transition model. The target encoder and projection head are defined as an exponential moving average of their online counterparts and are not updated via gradient descent. For brevity, we illustrate only the k^{th} step of future prediction, but in practice we compute the loss over all steps from 1 to K . Note: our implementation for this paper includes g_o in the Q-learning head. 59
- Figure 9 A boxplot of the distribution of human-normalized scores across the 26 Atari games under consideration, after 100k environment steps. The whiskers represent the interquartile range of human-normalized scores over the 26 games. Scores for each game are recorded at the end of training and averaged over 10 random seeds for SPR, 20 for CURL, and 5 for other methods. 67
- Figure 10 Performance of SPR with various prediction depths. Results are averaged across ten seeds per game, for all 26 games. To equalize the importance of games, we calculate an SPR-normalized score analogously to human-normalized scores, and show its mean and median across all 26 games. All other hyperparameters are identical to those used for SPR with augmentation. 69
- Figure 11 Two different kinds of generalization, using Procgen and Meta-World as examples. *Procedural* generalization involves evaluating on unseen environment configurations, whereas *task* generalization evaluates adaptability to unseen tasks (reward functions). 77

Figure 12

The impact of planning and self-supervision on procedural generalization in Procgen (hard difficulty, 500 train levels). We plot the zero-shot evaluation performance on unseen levels for each agent throughout training. The Q-Learning agent (QL) is a replica of the MuZero (MZ) with its model-based components removed. MZ+Contr is a MuZero agent augmented with a temporal contrastive self-supervised loss that is action-conditioned (we study other losses in Figure 13). We observe that both planning and self-supervision improve procedural generalization on Procgen. Comparing with existing state-of-the-art methods which were trained for 200M frames on the right (PPO (Schulman et al., 2017), PLR (Jiang et al., 2021), and UCB-DrAC+PLR (Raileanu et al., 2021b; Jiang et al., 2021), data from (Jiang et al., 2021)), we note that MuZero itself exceeds state-of-the-art performance after being trained on only 30M frames. For all plots, dark lines indicate median performance across 3 seeds and the shaded regions denote the min and max performance across seeds. For training curves see Figure 22, for additional metrics see Figure 20. 83

Figure 13

Evaluation performance on Procgen (hard, 500 train levels). On the left, we ablate the effectiveness of planning. The Q-Learning agent (QL) is a replica of MuZero (MZ) without model-based components. We then add a model to this agent (QL+Model) (see Section 13.2) to disentangle the effects of the model-based representation learning from planning in the full MuZero model (MZ). On the right, we ablate the effect of self-supervision with three different losses: Contrastive (Contr), Self-Predictive (SPR), and Image Reconstruction (Recon). We also include a Q-Learning+Model agent with reconstruction (QL+Model+Recon) as a baseline. For all plots, dark lines indicate median performance across 3 seeds (5 seeds for MZ and MZ+Recon) and the shaded regions denote the min and max performance across seeds. For corresponding training curves see Figure 23. 85

- Figure 14 Qualitative comparison of the information encoded in the embeddings learned by MuZero with and without the auxiliary pixel reconstruction loss. For MuZero, embeddings are visualized by learning a standalone pixel decoder trained with MSE. Visualized are environment frames (top row) and decoded frames (bottom row) for two games (Chaser and Climber), for embeddings at the current time step ($k = 0$) and 5 steps into the future ($k = 5$). Colored circles highlight important entities that are or are not well captured (blue=captured, yellow=so-so, red=missing). 87
- Figure 15 Interaction of self-supervision and data diversity on procedural generalization. Each plot shows generalization performance as a function of environment frames for different numbers of training levels. With only 10 levels, self-supervision does not bring much benefit over vanilla MuZero. Once the training set includes at least 100 levels, there is large improvement with self-supervised learning both in terms of data efficiency and final performance. For all plots, dark lines indicate median performance across seeds and shading indicates min/max seeds. 88
- Figure 16 Finetuning performance on ML-10 and ML-45, shown as cumulative regret over the success rate (**lower is better**). Both the pre-trained Q-Learning and MuZero agents have lower regret than corresponding agents trained from scratch. MuZero also achieves lower regret than Q-Learning, indicating a positive benefit of planning (though the difference is small on ML-45). Self-supervision (pixel reconstruction) does not provide any additional benefits. Solid lines indicate median performance across seeds, and shading indicates min/max seeds. 90
- Figure 17 Pre-training and RL fine-tuning, a two stage process. While pre-trained models are super general, RL fine-tuning can make them highly capable at individual tasks. 96
- Figure 18 The base encoder architecture used for all models in this work 99

- Figure 19 Performance on a subset of 10 Atari games for different values of the EMA parameter τ with augmentation (left) and without (right). Scores are averaged across 10 seeds per game for each value of τ . Self-normalized score is calculated separately for the augmentation and no-augmentation cases. [114](#)
- Figure 20 Additional metrics (proposed in [Agarwal et al. \(2021\)](#)) indicating zero-shot test performance of different methods on ProcGen. IQM corresponds to the Inter-Quartile Mean among all runs, and Optimality Gap refers to the amount by which the algorithm fails to meet a minimum score of 1.0. [125](#)
- Figure 21 Isolating the interaction between self-supervision and data diversity. Each plot shows the generalization performance as a function of training performance for different numbers of training levels. Also shown is the unity line, where training and testing performance are equivalent (note that with infinite training levels, all lines collapse to the unity line, as in this case there is no difference between train and test). Generally speaking, self-supervision results in stronger generalization than MuZero. Reporting median values over 1 seed on 10, 100 levels; on 500 levels 3 seeds for MZ+Contr, MZ+SPR and 5 seeds for MZ, MZ+Recon; on infinite levels 1 seed for MZ+Contr, MZ+SPR and 2 seeds for MZ, MZ+Recon. For visibility, min/max seeds are not shown. [126](#)
- Figure 22 Training performance on Procgen. See [Figure 12](#) in the main text for results on the test set. Reporting median values over 3 seeds (5 seeds for MZ), with shaded regions indicating min/max seeds. [126](#)
- Figure 23 Training performance on Procgen. See [Figure 13](#) in the main text for corresponding results on the test set. Left: the effect of planning. Right: the effect of self-supervision. Reporting median performance over 3 seeds (5 seeds for MZ and MZ+Recon), with shaded regions indicating min/max seeds. [127](#)

Figure 24 Zero-shot test success rates MuZero and Q-Learning on unseen goals on the ML-1 procedural generalization benchmark of Meta-World. While both MuZero and Q-Learning achieve near optimal performance, MuZero is more stable and a bit more data-efficient than Q-Learning. Shown are medians across three seeds (for visibility, min/max seeds are not shown). [127](#)

LIST OF TABLES

Table 1	Number of ground truth labels available in the benchmark for each game across each category. Localization is shortened for local. See next section for descriptions and examples for each category. 42
Table 2	Probe F1 scores averaged across categories for each game (data collected by random agents) 47
Table 3	Probe F1 scores for different methods averaged across all games for each category (data collected by random agents) 48
Table 4	Different ablations of ST-DIM. F1 scores for for each category averaged across all games (data collected by random agents) 50
Table 5	Breakdown of F1 Scores for every state variable in Boxing for ST-DIM, CPC, and Global-T-DIM, an ablation of ST-DIM that removes the spatial contrastive constraint, for the game Boxing 50
Table 6	Performance of different methods on the 26 Atari games considered by Kaiser et al., 2019 after 100k environment steps. Results are recorded at the end of training and averaged over 10 random seeds for SPR, 20 for CURL, and 5 for other methods. SPR outperforms prior methods on all aggregate metrics, and exceeds expert human performance on 7 out of 26 games. 66
Table 7	Scores on the 26 Atari games under consideration for ablated variants of SPR. All variants listed here use data augmentation. 68

Table 8	Train and various test success rates (zero-shot, few-shot, fine-tuning) on the ML-10 and ML-45 task generalization benchmarks of Meta-World. Shown are baseline results on MAML (Finn et al., 2017a) and RL ² (Duan et al., 2016) from the Meta-World paper (Yu et al., 2020a) as well as our results with Q-Learning and MuZero. Note that MAML and RL ² were trained for around 400M environment steps from states, whereas MuZero was trained from pixels for 50M steps on train tasks and 10M steps on test tasks for fine-tuning. 89
Table 9	Preprocessing steps and hyperparameters 100
Table 10	Probe F1 scores for all games for data collected by a pretrained PPO (50M steps) agent 103
Table 11	Probe F1 scores for different methods averaged across all games for each category (data collected by a pretrained PPO (50M steps) agent) 104
Table 12	Probe F1 scores for different ablations of ST-DIM for all games averaged across each category (data collected by random agents) 105
Table 13	Different ablations of ST-DIM. F1 scores for for each category averaged across all games (data collected by random agents) 106
Table 14	Probe results on features from a PPO agent trained on 50 million timesteps compared with a majority classifier and random-cnn baseline. The probes for all three methods are trained with data from the PPO agent that was trained for 50M frames 107
Table 15	Probe Accuracy scores averaged across categories for each game (data collected by random agents) 108
Table 16	Probe Accuracy scores for different methods averaged across all games for each category (data collected by random agents) 109

Table 19	Scores on the 26 Atari games under consideration for our controlled Rainbow implementation with and without augmentation, compared to previous methods. The high mean DQN-normalized score of our DQN without augmentation is due to an atypically high score on Private Eye, a hard exploration game on which the original DQN achieves a low score. 111
Table 20	Scores on the 26 Atari games under consideration for various contrastive alternatives to SPR implemented in our codebase. All variants listed here use data augmentation. 112
Table 21	Scores on the 26 Atari games under consideration for variants of SPR with different target encoder schemes, without augmentation. 113
Table 17	Hyperparameters for SPR on Atari, with and without augmentation. 115
Table 18	Mean episodic returns on the 26 Atari games considered by Kaiser et al., 2019 after 100k environment steps. The results are recorded at the end of training and averaged over 10 random seeds. SPR outperforms prior methods on all aggregate metrics, and exceeds expert human performance on 7 out of 26 games while using a similar amount of experience. 116
Table 22	Wall-clock runtimes for various algorithms for a complete training and evaluation run on a single Atari game using a P100 GPU. Rainbow (controlled) is roughly comparable to DrQ, although its runtime will differ due different DQN hyperparameters. Runtime for SimPLe is taken from its v3 version on Arxiv, although the latest version doesn't mention runtime. All runtimes are approximate, as exact running times vary from game to game. 117
Table 23	Hyper-parameters 119

Table 24	Final normalised training and test scores on Procgen when training on 500 levels for 30M environment frames, reporting the median across 3 seeds (for MZ, MZ+Recon across 5 seeds). Min/max over seeds are shown in parentheses. Final scores for each seed were computed as means over data points from the last 1M frames. A subset of the experiments on Maze have terminated before 30M frames therefore we computed final scores at 27M for all agents on this game. 123
Table 25	MuZero with self-supervised losses on Procgen. Final mean normalised training and test scores for different number of training levels. Reporting median values over 1 seed on 10, 100 levels; on 500 levels 3 seeds for MZ+Contr, MZ+SPR and 5 seeds for MZ, MZ+Recon; on infinite levels 1 seed for MZ+Contr, MZ+SPR and 2 seeds for MZ, MZ+Recon. Final scores for each seed were computed as means over data points from the last 1M frames. 124
Table 26	Zero-shot test performance on the ML-1 procedural generalization benchmark of Meta-World. Shown are baseline results on RL ² (Duan et al., 2016) from the Meta-World paper (Yu et al., 2020a) as well as our results on MuZero. Note that RL ² is a meta-learning method and was trained for 300M environment steps, while MuZero and Q-Learning do not require meta-training were only trained for 50M steps and fine-tune for 10M. We average the success rate of the last 1M environment steps for each seed and report medians across three seeds for MuZero and Q-Learning (RL ² results from (Duan et al., 2016)). 125
Table 27	Success rates for MuZero (with and without self-supervision) and Q-Learning on ML-10 and ML-45. Training results are shown at 50M frames and fine-tuning results are shown at 10M frames. Reporting average of the last 1M frames and then medians across 3 seeds (2 for MZ and MZ+Recon on ML10). 128

LIST OF ACRONYMS AND ABBREVIATIONS

ALE	Arcade Learning Environment
AtariARI	Atari Annotated RAM Interface
BYOL	Bootstrap Your Own Latent
CNN	Convolutional Neural Networks
CPC	Contrastive Predictive Coding
DIM	Deep InfoMax
DQN	Deep Q-Networks
EMA	Exponential Moving Average
IQM	Inter-Quartile Mean
JSD	Jensen-Shannon Divergence
MAML	Model-Agnostic Meta Learning
MCTS	Monte-Carlo Tree Search
MDP	Markov Decision Process
MLP	Multi-Layer Perceptron
NCE	Noise Contrastive Estimation
NLP	Natural Language Processing
RAM	Random Access Memory
RL	Reinforcement Learning
PPO	Proximal Policy Optimization
SPR	Self-Predictive Representations
SSL	Self-Supervised Learning
TD	Temporal Difference

INTRODUCTION

Most tasks humans perform involve taking a sequence of decisions or actions by interacting with our environment. Some are short-term spanning a few seconds, like a child picking up a toy from the floor, while others can take years like learning and mastering a music instrument. The ability to learn from interactions and respond to stimulus or feedback in a wide range of scenarios is thus central to most theories about intelligence (Minsky, 1961; Turing, 2004), or one may argue the very definition of intelligence itself (Legg et al., 2007). Emulating this capability in artificial agents is therefore crucial to making them as intelligent as humans themselves.

Reinforcement Learning (Sutton et al., 2018) is a formulation that directly takes a stab at this ambitious moonshot. It is a computational framework that entails letting an agent learn through interaction with an environment. The first attempts to formalize Reinforcement Learning were made in the 1950s (Bellman, 1957a), and because of its generality it has since been studied in a variety of fields such as optimal control, neuroscience, and artificial intelligence.

Early attempts to use Reinforcement Learning focused on “tabular methods” which required a large amount of memory to store every possible instantiation of the problem space, and thus were hard to scale up: a problem often termed as “the curse of dimensionality” (Bellman, 1957b). In the 1990s, Gerry Tesauro (Tesauro et al., 1995) showed that it’s possible to use Reinforcement Learning methods without enumerating the whole problem space, and instead approximating it with artificial neural networks to play Backgammon better than any human players.

These networks were however still small in size, and thus limited in their approximation power. In early 2010s, the machine learning community started to gain success in training deep neural networks practically, and this propelled the next generation of RL systems. The mid-to-late 2010s saw RL agents leverage deep neural networks

to outperform the strongest humans in long-standing board games like Go (Silver et al., 2016a) and Chess (Silver et al., 2018); results which are now widely acclaimed. Similar feats were later achieved in more complex games that require co-ordination such as StarCraft (Vinyals et al., 2017) and Dota (OpenAI et al., 2019).

These successes have however remained siloed in the narrow game-like domains in which they were originally developed, and failed to translate to material real-world successes. Why is this the case? There are a few reasons. First, these algorithms are quite data-inefficient, often requiring the equivalent of decades of experience to master a narrow task. This tends to work okay in simulators where we can collect virtually infinite amounts of data, but does not translate to real-world tasks where we don't have that luxury. Secondly, these algorithms are focused on a narrow goal at a time in a narrow environment, making them brittle and un-adaptive to environment or task changes. Solving these issues is critical to making reinforcement learning practical (Dulac-Arnold et al., 2019), and ultimately understanding and building general intelligence itself. This leads us to the key question underlying this thesis: How can we design agents that learn efficiently, and generalize well, given only sensory information and a scalar reward signal?

In my work, I have postulated that self-supervised learning might be the answer. The key idea behind self-supervised learning is to design machine learning systems in such a way that they can learn without explicit labels provided by a human annotator. Typically, self-supervised learning methods generate their own labels or "supervision" signal based on the structure or properties of the input data itself. In the context of reinforcement learning, this involves enabling an agent to learn without explicit rewards, by exploiting the inherent structure present in the unlabelled interactions to learn meaningful representations or dynamics for the task.

The paradigm has been remarkably successful elsewhere in other areas of Deep Learning such as Computer Vision and Natural Language Processing, in what would only be termed as a revolution when the history books are written. The most capable systems in these fields now rely purely on tasks that are focused on modelling the raw unlabelled inputs (Brown et al., 2020; He et al., 2022; Radford et al., 2021a) and these capabilities improve predictably with scale (Sutton, 2019; Kaplan et al., 2020; Branwen, 2021a). Reinforcement Learning is due for a similar revolution, and this thesis takes baby steps in building RL agents that leverage self-supervised learning to make them more general and data-efficient.

BACKGROUND

2.1 DEEP LEARNING

Deep Learning ([Goodfellow et al., 2016](#)) colloquially refers to the set of machine learning algorithms where the prediction function is parameterized by an artificial neural network (ANN), which uses multiple layers to progressively extract higher-level features from the raw input. In its simplest form, deep learning involves training a multi-layer perceptron (a feedforward ANN) with a single hidden layer using backpropagation ([Linnainmaa, 1970](#); [Rumelhart et al., 1986](#)).

Convolutional Neural Networks ([Fukushima et al., 1982](#); [LeCun et al., 1989](#))[CNNs] are a specialized form of deep learning architecture that are most commonly applied to processing visual data. CNNs are characterized by their use of shared-weight kernels or features that provide translation equivariance with respect to the inputs. Modern CNNs also leverage several techniques such as skip connections ([He et al., 2016](#)) and normalization layers ([Ioffe et al., 2015b](#); [Ba et al., 2016](#)) to scale to large number of layers.

Recurrent Neural Networks ([Hochreiter et al., 1997](#); [Chung et al., 2014](#)) and its variants are another popular form of deep learning architecture that are most commonly applied to processing sequential data.

More recently, Transformers ([Vaswani et al., 2017](#)) - an architecture based on self-attention ([Bahdanau et al., 2015](#)) have been showed to scale well across several modalities ([Kaplan et al., 2020](#); [Dosovitskiy et al., 2021](#)) and are fast becoming the default neural network architecture.

2.2 REINFORCEMENT LEARNING

Reinforcement Learning is a field of machine learning that is concerned with sequential decision making. In a typical reinforcement learning problem, an agent interacts with an environment by taking actions and receiving feedback in the form of positive or negative rewards. The goal of the agent is to learn a policy that will maximize the total reward it receives over time.

The standard reinforcement learning loop can be broken down into the following steps:

- **Observation:** At each timestep t , the agent receives an observation o_t of the environment which is a snapshot of the current complete or partial environment state s_t .
- **Action:** The agent selects an action a_t to take based on its current state and the policy π it has learned.
- **Transition:** The environment transitions to a new state s_{t+1} based on the action and the dynamics of the environment.
- **Reward:** The agent receives a reward r_t on the action it took and the resulting state of the environment.
- **Update:** The agent updates its policy based on the reward it received and the new state of the environment.

For simplicity, consider that the environment is Markovian, ie the future of the process only depends on the current observation and not the entire history. We can then formalize the Reinforcement Learning problem as a Markov decision process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma, \rho_0)$. Here \mathcal{S} and \mathcal{A} are the state and action spaces, respectively, and $\gamma \in (0, 1)$ is the discount factor. The dynamics or transition distribution are denoted as $p(s'|s, a)$, the initial state distribution as $\rho_0(s)$, and the reward function as $r(s, a)$.

The goal of reinforcement learning is to find the optimal policy π^* that maximizes the expected sum of discounted rewards, denoted by η :

$$\pi^* = \operatorname{argmax}_{\pi} \eta[\pi] = \operatorname{argmax}_{\pi} E_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right].$$

Reinforcement Learning algorithms typically fall into two major categories, model-based and model-free methods.

2.2.1 Model-free Reinforcement Learning

Model-free Reinforcement Learning algorithms learn directly from experience, without relying on an explicit or learned model of the environment dynamics. Model-free methods can be further categorized into two major types: Value-based methods and Policy-based methods.

Value-based Methods: Value-based methods aim to learn a value function that provides a prediction of how good each state or each state/action pair is. The value function can be either a lookup-table for small problems, or parameterized by an artificial neural network when the number of states / actions is too large. The policy is usually directly derived from the value function, for example by greedily selecting the action which maximises the value function.

Value-based methods typically use Temporal-Difference (TD) Learning (Sutton, 1988) to update the value function. The basic idea behind TD learning is that the agent can learn from the difference between the current value estimate $V(s_t)$, the discounted value estimate of $V(s_{t+1})$ and the actual reward r_{t+1} gained from transitioning between s_t and s_{t+1} . This difference, called the TD error, is used to update the agent's value function as follows:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \quad (1)$$

Sometimes, it is more convenient to estimate the state-value function $Q(s, a)$ (also commonly referred to as Q function) since it allows us to derive a policy by simply selecting the action that maximizes the state-value function at a particular state. This algorithm is commonly called Q-learning, we can write its update rule as:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]. \quad (2)$$

Deep Q-Networks (Mnih et al., 2015, DQNs) showed that learning Q-values with neural networks and using a replay buffer of past trajectories (Lin, 1992) can lead to human-level performance in a suite of Atari games. Over the years, several improvements have been made to DQN including prioritized experience replay (Schaul et al., 2016), distributional DQN (Bellemare et al., 2017), multi-step learning (Watkins, 1989), etc which were later combined in a single agent called Rainbow (Hessel et al., 2018).

Policy-based Methods: Policy-based methods directly optimize an objective such as the expected cumulative reward by computing the gradient of the objective with respect to the policy, and thus broadly referred to as policy gradient methods. The most direct / naive version of computing the policy gradient is using REINFORCE (Williams, 1992). There are several methods that improve over reinforce by trying to reduce the variance of the policy gradient using a learned critic gradient. These are called actor-critic methods, and popular examples include A2C (Mnih et al., 2016) and PPO (Schulman et al., 2017).

In this thesis, we will primarily focus on value-based methods.

2.2.2 *Model-based Reinforcement Learning*

Model-based Reinforcement Learning agents use an explicit or learned model of the environment. Using the model allows the agent to "look ahead" and predict the outcome of its actions in the form of rewards or estimated values multiple steps into the future, before taking an action (often termed as planning). Model-based RL methods are distinguished by how they use the model, and how they learn the model.

How to use the model? (Planning) There are several ways in which an RL agent use predictions from its mode to plan over possible future outcomes. In domains with discrete actions, its common to use tree search methods such as A^* search or Monte-Carlo Tree Search (Coulom, 2006, MCTS). MCTS in particular has been combined with neural networks and shown to be effective in quite a varied range of environments such as Go (Silver et al., 2016b), Chess, Shogi (Silver et al., 2018) and Atari games (Schrittwieser et al., 2020). In environments with continuous actions, it is more common to use the Cross-Entropy Method (Rubinstein et al., 2004) for planning, although there have been extensions of MCTS to continuous actions (Hubert et al., 2021).

How to learn the model? Choosing the objective for learning the dynamics model (sometimes called the world model) is a critical design choice for model-based RL methods. A common theme is to use future prediction error to guide learning of the model. Value equivalent models such as MuZero (Schrittwieser et al., 2020) use prediction errors of task-specific quantities to learn the model. It's also common to

directly predict the future state / observation and use the corresponding prediction error to learn the model (Finn et al., 2016; Ha et al., 2018; Hafner et al., 2020).

2.3 REPRESENTATION LEARNING

Representation Learning (Bengio et al., 2013) is concerned with learning useful (and often compact) representations that capture the underlying explanatory factors of the data, such that these representations are useful in a subsequent learning problem. This ability to represent data into useful and concise descriptions is considered a fundamental cognitive capability in humans (Marr, 1982; Gordon et al., 1996), and allows us to seamlessly transfer knowledge across different tasks.

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

Self-Supervised learning is a paradigm designed to learn representations by optimizing an objective that rewards an algorithm for learning about the data itself, without a particular task in mind. A key motivation for self-supervised learning is that the underlying data has a much richer structure than what sparse labels or rewards could provide, and thus leveraging that structure should lead to better representations.

Contemporary methods for self-supervised representation learning fall into two modes: **generative**, and **latent** representation learning. Generative methods aim to learn a representation that can be decoded back to the original data itself. Contrastive methods on the other hand, learn representations by leveraging differences across similar and dissimilar data points.

2.4 LATENT REPRESENTATION LEARNING

In context of visual observations, generative methods often rely on the reconstruction error in the pixel-space as the learning objective. This leads to such methods being

overly focused on pixel-level details, and failing to capture more abstract latent factors.

Consider this illustrative experiment conducted by [Epstein \(2016\)](#), where subjects were asked to draw a picture of the dollar bill as detailed as possible. Figure 1 shows the drawing a subject made by recalling what a dollar bill looks like from memory. Figure 2 is their drawing subsequently made with a dollar bill present. As is evident, the drawing made in the absence of the dollar bill is quite different compared with the drawing made from an exemplar. Thus, even though a dollar bill is something we have seen countless number of times, we don't retain a full visual representation of it, and only enough features which can distinguish it from another object.



Figure 1: Drawing of a one-dollar bill Figure 2: Drawing subsequently made completely from memory. with a dollar bill present.

2.4.1 Contrastive Methods

Contrastive Learning is one of the pre-dominant methods to learn representations of high dimensional data like images or videos. In contrastive learning, the goal is to learn an embedding function f such that:

$$\langle f(x), f(x^+) \rangle \gg \langle f(x), f(x^-) \rangle$$

That is, given similar data points x, x^+ , we want the inner product ¹ of $f(x)$ and $f(x^+)$ to be much higher than the inner product of $f(x)$ and $f(x^-)$, where x^- is a random data point (and thus presumably dissimilar to x). x^- is also typically referred to as a negative sample.

To achieve this, we could construct a loss function of the form:

¹ It is common to use other score functions, such as a bilinear critic in place of the inner product.

$$L_{cont}(f) := \mathbb{E}_{x, x^+, x^-} \left[-\log \left(\frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + e^{f(x)^T f(x^-)}} \right) \right], \quad (3)$$

This loss function is reminiscent of many past ideas including kernel learning, metric learning and co-training (Gutmann et al., 2010; Cortes et al., 2010; Bellet et al., 2013).

If we use multiple negative samples, we get a lower bound on Mutual Information:

$$I(X, X^+) \geq \sum_{i=1}^N \log \frac{\exp f(x_i, x_i^+)}{\sum_{j=1}^N \exp f(x_i, x_j^-)} \triangleq \mathcal{I}_{NCE}(\{(x_i, x_i^+)\}_{i=1}^N) \quad (4)$$

This bound is refer to as the InfoNCE objective, see Oord et al. (2018) and Poole et al. (2019) for a derivation and more details on this bound.

2.4.2 Bootstrapped Latent Methods

Although contrastive learning has shown remarkably effective, it’s still limited by the fact that it requires careful treatment of negative examples (which can manifest into a task design problem itself) to work. Recent works, such as Bootstrap Your Own Latent (BYOL, Grill et al., 2020b) have worked around this limitation by iteratively learning from the outputs of the network itself.

Using the same notation as before, we can write the BYOL loss as:

$$L_{BYOL}(f) := \mathbb{E}_{x, x^+} \left\| \overline{g(f(x))} - \overline{g(f'(x^+))} \right\|_2^2 \quad (5)$$

where g is a small prediction network on top of the encoder, and f' is a slower moving copy (target network) of the encoder, $\overline{(\cdot)}$ denotes that the representations are L2 normalized.

ARTICLE I

PROLOGUE TO THE FIRST ARTICLE

3.1 ARTICLE DETAILS

Title: Unsupervised State Representation Learning in Atari

Authors: Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, R Devon Hjelm

Presented At: Neural Information Processing Systems (NeurIPS), 2019.

3.2 CONTEXT

Around late 2018, latent self-supervised learning methods like DIM (Hjelm et al., 2019a) and CPC (Oord et al., 2018) started to show quite a bit of promise in computer vision, and were shown to scale to large-scale datasets like ImageNet. Around the same time issues related to brittleness and data-inefficiency of deep RL agents started to surface up. We wondered if self-supervised learning methods could help RL agents in visually rich environments, and if we could find a principled way to show their efficacy.

3.3 CONTRIBUTIONS

I had discussions with Devon Hjelm and Marc-Alexandre on the role of self-supervised learning in RL, and prototyped a codebase to experiment with different representation learning methods. Probing learnt representations to verify if they had captured salient factors of the environment seemed like a principled way to go compare these methods. Sherjil suggested that we focus on probing Atari games by

looking at the evolving RAM state. Evan and I found source code of a bunch of Atari games, and Evan contributed majority of the labels with help from me and Sherjil. I implemented several contrastive learning methods, while Evan implemented the reconstruction based methods. Evan and I launched majority of the experiments to evaluate different representation learning methods on our probing benchmark with help from Marc-Alexandre. Evan, Devon and I wrote majority of the paper with help from Sherjil and Yoshua. Devon and Sherjil also provided weekly feedback and advise on project, and helped scope our experiments.

3.4 RESEARCH IMPACT

The paper was quite well received, and is often cited as one of the first works to show how to leverage latent self-supervised learning in RL environments. The AtariARI benchmark we proposed has been used numerous times to evaluate agents or representations (Tupper et al., 2020; Khan et al., 2021; Veličković et al., 2021). The work also inspired follow-up papers which leveraged contrastive or other latent methods to build better RL agents (Srinivas et al., 2020; Mazouze et al., 2020; Schwarzer et al., 2021a).

UNSUPERVISED STATE REPRESENTATION LEARNING IN ATARI

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

4.1 INTRODUCTION

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

Recently, deep representation learning has led to tremendous progress in a variety of machine learning problems across numerous domains (Krizhevsky et al., 2012; Amodei et al., 2016; Wu et al., 2016; Mnih et al., 2015; Silver et al., 2016a). Typically, such representations are often learned via end-to-end learning using the signal from

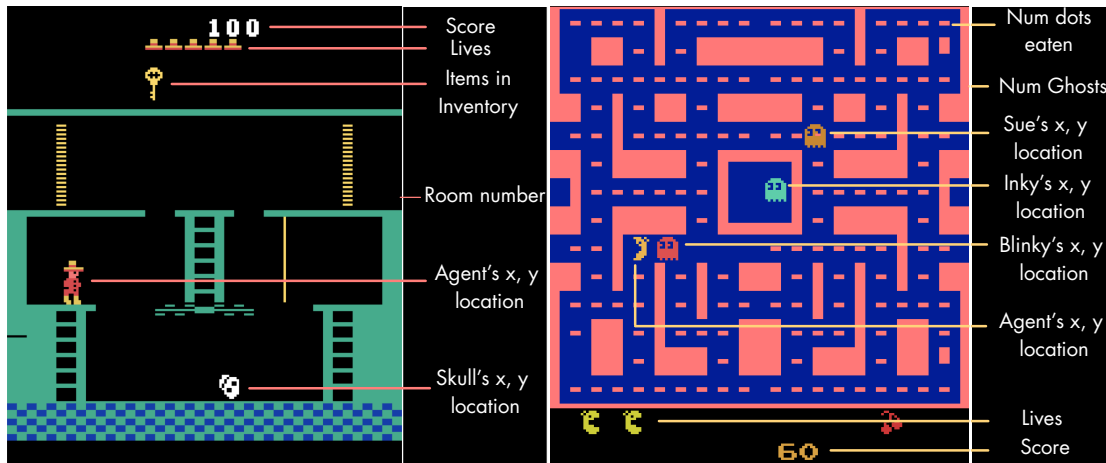


Figure 3: We use a collection of 22 Atari 2600 games to evaluate state representations. We leveraged the source code of the games to annotate the RAM states with important state variables such as the location of various objects in the game. We compare various unsupervised representation learning techniques based on how well the representations linearly-separate the state variables. Shown above are examples of state variables annotated for Montezuma’s Revenge and MsPacman.

labels or rewards, which makes such techniques often very sample-inefficient. In contrast, human learning in the natural world appears to require little to no explicit supervision for perception (Gross, 1968).

Unsupervised (Dumoulin et al., 2017; Kingma et al., 2014; Dinh et al., 2017) and self-supervised representation learning (Pathak et al., 2016; Doersch et al., 2017; Kolesnikov et al., 2019) have emerged as an alternative to supervised versions which can yield useful representations with reduced sample complexity. In the context of learning state representations (Lesort et al., 2018), current unsupervised methods rely on generative decoding of the data using either VAEs (Watter et al., 2015; Higgins et al., 2017; Ha et al., 2018; Duan, 2017) or prediction in pixel-space (Oh et al., 2015; Finn et al., 2016). Since these objectives are based on reconstruction error in the pixel space, they are not incentivized to capture abstract latent factors and often default to capturing pixel level details.

In this work, we leverage recent advances in self-supervision that rely on scalable estimation of mutual information (Belghazi et al., 2018; Oord et al., 2018; Hjelm et al., 2019b; Veličković et al., 2019), and propose a new contrastive state representation

learning method named Spatiotemporal DeepInfomax (ST-DIM), which maximizes the mutual information across both the spatial and temporal axes.

To systematically evaluate the ability of different representation learning methods at capturing the true underlying factors of variation, we propose a benchmark based on Atari 2600 games using the Arcade Learning Environment (ALE, [Bellemare et al., 2013](#)). A simulated environment provides access to the underlying generative factors of the data, which we extract using the source code of the games. These factors include variables such as the location of the player character, location of various items of interest (keys, doors, etc.), and various non-player characters, such as enemies (see figure 3). Performance of a representation learning technique in the Atari representation learning benchmark is then evaluated using *linear probing* ([Alain et al., 2017](#)), i.e. the accuracy of linear classifiers trained to predict the latent generative factors from the learned representations.

Our contributions are the following

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

4.2 RELATED WORK

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

(Gutmann et al., 2010; Ma et al., 2018). See (Poole et al., 2019) for a review of different variational bounds for mutual information.

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

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

4.3 SPATIOTEMPORAL DEEP INFOMAX

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

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

4.3.1 Maximizing mutual information across space and time

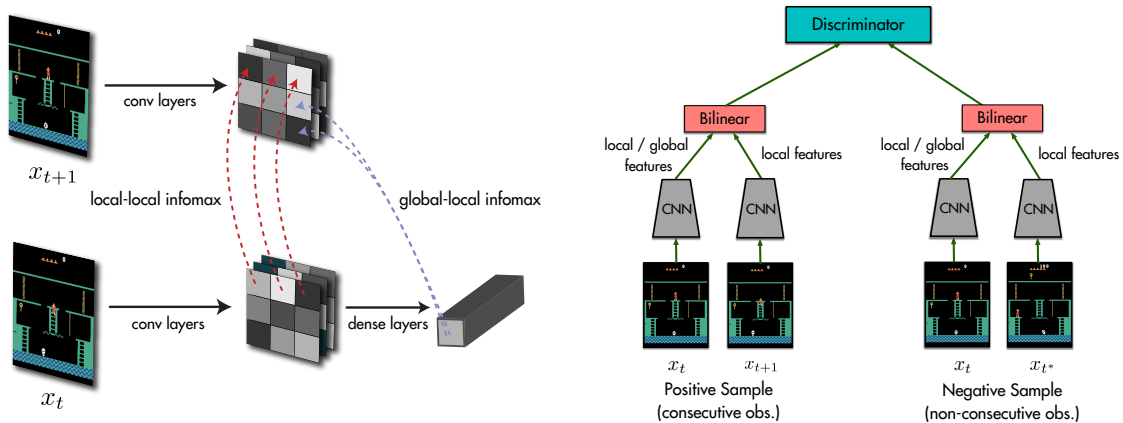


Figure 4: A schematic overview of SpatioTemporal DeepInfoMax (ST-DIM). (a) shows the two different mutual information objectives: local infomax and global infomax. (b) shows a simplified version of the contrastive task we use to estimate mutual information. In practice, we use multiple negative samples.

Given a mutual information estimator, we follow DIM (Hjelm et al., 2019b) and maximize a sum of patch-level mutual information objectives. The global objectives maximize the mutual information between the full observation at time t with small patches of the observation at time $t + 1$. The representations of the small image patches are taken to be the hidden activations of the convolutional encoder applied to the full observation. The layer is picked appropriately to ensure that the hidden activations only have a limited receptive field corresponding to $1/16^{th}$ the size of the

full observations. The local objective maximizes the mutual information between the local feature at time t with the corresponding local feature at time $t + 1$. Figure 4 is a visual depiction of our model which we call Spatiotemporal Deep Infomax (ST-DIM).

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

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

$$\mathcal{I}_{NCE}(\{(x_i, y_i)\}_{i=1}^N) = \sum_{i=1}^N \log \frac{\exp f(x_i, y_i)}{\sum_{j=1}^N \exp f(x_i, y_j)} \quad (6)$$

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

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

¹ For convenience, ignoring those that are in the support of the joint.

$f_g(x_t, x_{t+1}) = \phi(x_t)^T W_g \phi_{l,m,n}(x_{t+1})$ and the score function of the local objective is $f_l(x_t, x_{t+1}) = \phi_{l,m,n}(x_t)^T W_l \phi_{l,m,n}(x_{t+1})$, where $\phi_{l,m,n}$ is the feature map at the l^{th} layer at the (m, n) spatial location.

Table 1: Number of ground truth labels available in the benchmark for each game across each category. Localization is shortened for local. See next section for descriptions and examples for each category.

game	agent	small object	other	score/clock/lives/		overall
	local.	local.	local.	display	misc	
asteroids	2	4	30	3	3	41
berzerk	2	4	19	4	5	34
bowling	2	2	0	2	10	16
boxing	2	0	2	3	0	7
breakout	1	2	0	1	31	35
demonattack	1	1	6	1	1	10
freeway	1	0	10	1	0	12
frostbite	2	0	9	4	2	17
hero	2	0	0	3	3	8
montezumarevenge	2	0	4	4	5	15
mspacman	2	0	10	2	3	17
pitfall	2	0	3	0	0	5
pong	1	2	1	2	0	6
privateeye	2	0	2	4	2	10
qbert	3	0	2	0	0	5
riverraid	1	2	0	2	0	5
seaquest	2	1	8	4	3	18
spaceinvaders	1	1	2	2	1	7
tennis	2	2	2	2	0	8
venture	2	0	12	3	1	18
videopinball	2	2	0	2	0	6
yarsrevenge	2	4	2	0	0	8
total	39	27	124	49	70	308

4.4 THE ATARI ANNOTATED RAM INTERFACE (AARI)

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

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

Annotating Atari RAM: ALE does not explicitly expose any ground truth state information. However, ALE does expose the RAM state (128 bytes per timestep) which are used by the game programmer to store important state information such as the location of sprites, the state of the clock, or the current room the agent is in. To extract these variables, we consulted commented disassemblies ([Whalen et al., 2008](#)) (or source code) of Atari 2600 games which were made available by [Engelhardt \(2019\)](#) and [Jentzsch et al. \(2019\)](#). We were able to find and verify important state variables for a total of 22 games. Once this information is acquired, combining it with the ALE interface produces a wrapper that can automatically output a state label for every example frame generated from the game. We make this available with an easy-to-use *gym* wrapper, which returns this information with no change to existing code using *gym* interfaces. Table 1 lists the 22 games along with the categories of variables for each game. We describe the meaning of each category in the next section.

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

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

4.5 EXPERIMENTAL SETUP

We evaluate the performance of different representation learning methods on our benchmark. Our experimental pipeline consists of first training an encoder, then freezing its weights and evaluating its performance on linear probing tasks. For each identified generative factor in each game, we construct a linear probing task where the representation is trained to predict the ground truth value of that factor.

Note that the gradients are not backpropagated through the encoder network, and only used to train the linear classifier on top of the representation.

4.5.1 *Data preprocessing and acquisition*

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

4.5.2 *Methods*

In our evaluations, we compare the following methods:

1. Randomly-initialized CNN encoder (RANDOM-CNN).
2. Variational autoencoder (VAE) (Kingma et al., 2014) on raw observations.
3. Next-step pixel prediction model (PIXEL-PRED) inspired by the "No-action Feedforward" model from Oh et al., 2015.
4. Contrastive Predictive Coding (CPC) (Oord et al., 2018), which maximizes the mutual information between current latents and latents at a future timestep.
5. SUPERVISED model which learns the encoder and the linear probe using the labels. The gradients are backpropagated through the encoder in this case, so this provides a base-case performance bound.

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

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

4.5.3 *Probing*

We train a different 256-way³ linear classifier with the representation under consideration as input. We ensure the distribution of realizations of each state variable has high entropy by pruning any variable with entropy less than 0.6. We also ensure there are no duplicates between the train and test set. We train each linear probe with 35,000 frames and use 5,000 and 10,000 frames each for validation and test respectively. We use early stopping and a learning rate scheduler based on plateaus in the validation loss.

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

4.6 RESULTS

Table 2: Probe F1 scores averaged across categories for each game (data collected by random agents)

Game	maj-clf	random-cnn	vae	pixel-pred	cpc	st-dim	supervised
asteroids	0.28	0.34	0.36	0.34	0.42	0.49	N/A
berzerk	0.18	0.43	0.45	0.55	0.56	0.53	0.68
bowling	0.33	0.48	0.50	0.81	0.90	0.96	0.95
boxing	0.01	0.19	0.20	0.44	0.29	0.58	0.83
breakout	0.17	0.51	0.57	0.70	0.74	0.88	0.94
demonattack	0.16	0.26	0.25	0.32	0.57	0.69	0.83
freeway	0.01	0.50	0.26	0.81	0.47	0.81	0.98
frostbite	0.08	0.57	0.01	0.72	0.76	0.75	0.85
hero	0.22	0.75	0.51	0.74	0.90	0.93	0.98
montezumarevenge	0.08	0.68	0.69	0.74	0.75	0.78	0.87
mspacman	0.10	0.48	0.38	0.74	0.65	0.70	0.87
pitfall	0.07	0.34	0.56	0.44	0.46	0.60	0.83
pong	0.10	0.17	0.09	0.70	0.71	0.81	0.87
privateeye	0.23	0.70	0.71	0.83	0.81	0.91	0.97
qbert	0.29	0.49	0.49	0.52	0.65	0.73	0.76
riverraid	0.04	0.34	0.26	0.41	0.40	0.36	0.57
seaquest	0.29	0.57	0.56	0.62	0.66	0.67	0.85
spaceinvaders	0.14	0.41	0.52	0.57	0.54	0.57	0.75
tennis	0.09	0.41	0.29	0.57	0.60	0.60	0.81
venture	0.09	0.36	0.38	0.46	0.51	0.58	0.68
videopinball	0.09	0.37	0.45	0.57	0.58	0.61	0.82
yarsrevenge	0.01	0.22	0.08	0.19	0.39	0.42	0.74
mean	0.14	0.44	0.39	0.58	0.60	0.68	0.83

Table 3: Probe F1 scores for different methods averaged across all games for each category (data collected by random agents)

Category	maj-clf	random-cnn	vae	pixel-pred	cpc	st-dim	supervised
Small Loc.	0.14	0.19	0.17	0.31	0.42	0.51	0.69
Agent Loc.	0.12	0.31	0.30	0.48	0.43	0.58	0.83
Other Loc.	0.14	0.50	0.36	0.61	0.66	0.69	0.81
Score/Clock/Lives/Display	0.13	0.58	0.53	0.76	0.83	0.86	0.93
Misc.	0.26	0.59	0.65	0.70	0.71	0.74	0.86

We report the F1 averaged across all categories for each method and for each game in Table 2 for data collected by random agent. In addition, we provide a breakdown of probe results in each category, such as small object localization or score/lives classification in Table 3 for the random agent. We include the corresponding tables for these results with data collected by a pretrained PPO agent in tables 10 and 11. The results in table 2 show that ST-DIM largely outperforms other methods in terms of mean F1 score. In general, contrastive methods (ST-DIM and CPC) methods seem to perform better than generative methods (VAE and PIXEL-PRED) at these probing tasks. We find that RandomCNN is a strong prior in Atari games as has been observed before (Burda et al., 2019), possibly due to the inductive bias captured by the CNN architecture empirically observed in (Ulyanov et al., 2018). We find similar trends to hold on results with data collected by a PPO agent. Despite contrastive methods performing well, there is still a sizable gap between ST-DIM and the fully supervised approach, leaving room for improvement from new unsupervised representation learning techniques for the benchmark.

4.7 DISCUSSION

Ablations: We investigate two ablations of our ST-DIM model: Global-T-DIM, which only maximizes the mutual information between the global representations and JSD-ST-DIM, which uses the NCE loss (Hyvärinen et al., 1999) instead of the InfoNCE loss, which is equivalent to maximizing the Jensen Shannon Divergence between representations. We report results from these ablations in Figure 5 and 6. We see from the results in that 1) the InfoNCE loss performs better than the JSD loss

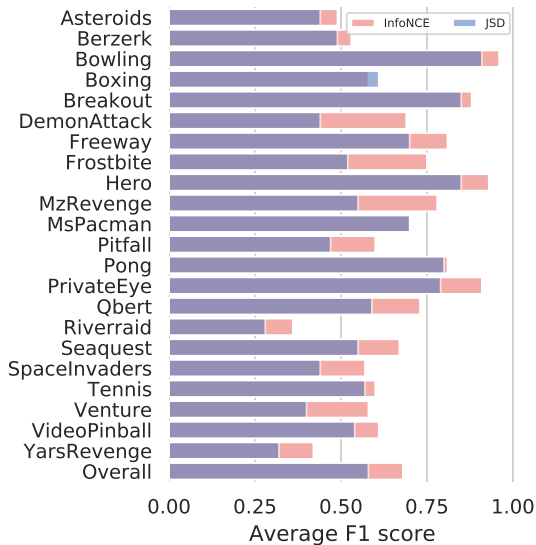


Figure 5: InfoNCE vs JSD

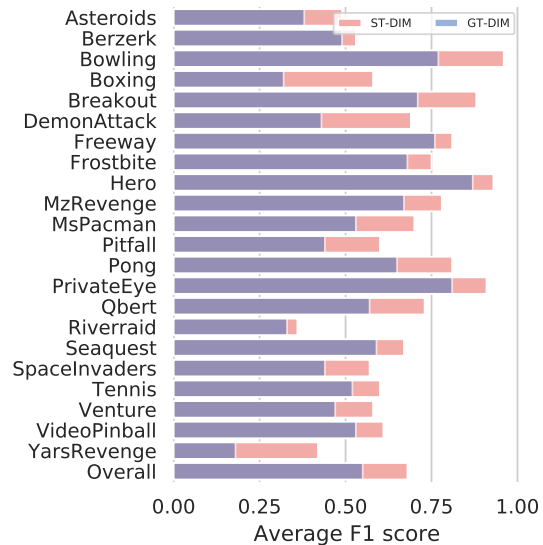


Figure 6: Effect of Spatial Loss

and 2) contrasting spatiotemporally (and not just temporally) is important across the board for capturing all categories of latent factors.

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

Capturing small objects: As we can see in Table 3, ST-DIM performs better at capturing small objects than other methods, especially generative models like VAE and pixel prediction methods. This is likely because generative models try to model every pixel, so they are not penalized much if they fail to model the few pixels that make up a small object. Similarly, ST-DIM holds this same advantage over Global-T-DIM (see Table 13), which is likely due to the fact that Global-T-DIM is not penalized if its global representation fails to capture features from some patches of the frame.

Table 4: Different ablations of ST-DIM. F1 scores for for each category averaged across all games (data collected by random agents)

	jsd-st-dim	global-t-dim	st-dim
Small Loc.	0.44	0.37	0.51
Agent Loc.	0.47	0.43	0.58
Other Loc.	0.64	0.53	0.69
Score/Clock/Lives/Display	0.69	0.76	0.86
Misc.	0.64	0.66	0.74

Table 5: Breakdown of F1 Scores for every state variable in Boxing for ST-DIM, CPC, and Global-T-DIM, an ablation of ST-DIM that removes the spatial contrastive constraint, for the game Boxing

method	vae	pixel-pred	cpc	global-t-dim	st-dim
clock	0.03	0.27	0.79	0.81	0.92
enemy_score	0.19	0.58	0.59	0.74	0.70
enemy_x	0.32	0.49	0.15	0.17	0.51
enemy_y	0.22	0.42	0.04	0.16	0.38
player_score	0.08	0.32	0.56	0.45	0.88
player_x	0.33	0.54	0.19	0.13	0.56
player_y	0.16	0.43	0.04	0.14	0.37

Robust to presence of easy-to-exploit features: Representation learning with mutual information or contrastive losses often fail to capture all salient features if a few easy-to-learn features are sufficient to saturate the objective. This phenomenon has been linked to the looseness of mutual information lower bounds (McAllester et al., 2018; Ozair et al., 2019) and *gradient starvation* (Combes et al., 2018). We see the most prominent example of this phenomenon in Boxing. The observations in Boxing have a clock showing the time remaining in the round. A representation which

encodes the shown time can perform near-perfect predictions without learning any other salient features in the observation. Table 5 shows that CPC, Global T-DIM, and ST-DIM perform well at predicting the clock variable. However only ST-DIM does well on encoding the other variables such as the score and the position of the boxers.

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

4.8 CONCLUSION

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

ARTICLE II

PROLOGUE TO THE SECOND ARTICLE

5.1 ARTICLE DETAILS

Title: Data-Efficient Reinforcement Learning with Self-Predictive Representations.

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

Presented At: International Conference on Learning Representations (ICLR), 2021.

5.2 CONTEXT

Encouraged by the promising results of self-supervised learning methods on probing tasks ([Anand et al., 2019](#)), we thought it would be great to explore their role in control. At that time, the Atari 100k benchmark ([Kaiser et al., 2020](#)) (2hrs of realtime play per game) was seen as a significantly challenging testbed for RL algorithms, and a domain where SotA agents performed quite poorly compared to humans. Our intuition was that a dynamics-aware representation learning method ought to improve the data-efficiency of RL agents.

5.3 CONTRIBUTION

I proposed the initial idea for abstract world models in my predoc report, and refined the idea after feedback from Aaron Courville and Phil Bachman. I set up the initial codebase with RL infrastructure and ran initial experiments testing out a few contrastive model-based RL ideas. Max and I tried a self-predictive instead of a contrastive model and that ended up giving great initial results. I ported our

codebase to rlpyt (Stooke et al., 2019) which significantly improved our wall-clock training time. Max added data augmentations on the suggestion of Phil which further improved performance. Max and I ran most experiments on the Mila and Microsoft Research clusters. Max and I wrote the initial draft of the paper with significant feedback and improvements from Devon, Phil and Aaron. Aaron, Devon and Phil provided weekly advice and guidance on the project, and helped concretize the scope of the project.

5.4 RESEARCH IMPACT

The SPR paper was very well received and surprised a lot of folks due to the large improvements in the SoTA on the challenging Atari 100k benchmark. The method has since then been validated in a number of domains, including Procgen (Anand et al., 2021) and DeepMind Control (McInroe et al., 2021).

In a follow-up work, we also showed its efficacy as a pre-training objective (Schwarzer et al., 2021b), while another group showed its efficacy as an exploration objective (Guo et al., 2022).

SPR was further improved by using the model to also perform planning (something we postulated as future work). EfficientZero (Ye et al., 2021) combined SPR with MuZero to achieve new SoTAs on Atari 100k (comparable to human scores), and MuZero++ (Anand et al., 2021) combined SPR with MuZero to achieve new SoTA results on Procgen.

DATA-EFFICIENT REINFORCEMENT LEARNING WITH SELF-PREDICTIVE REPRESENTATIONS

Abstract: While deep reinforcement learning excels at solving tasks where large amounts of data can be collected through virtually unlimited interaction with the environment, learning from limited interaction remains a key challenge. We posit that an agent can learn more efficiently if we augment reward maximization with self-supervised objectives based on structure in its visual input and sequential interaction with the environment. Our method, Self-Predictive Representations (SPR), trains an agent to predict its own latent state representations multiple steps into the future. We compute *target* representations for future states using an encoder which is an exponential moving average of the agent’s parameters and we make predictions using a learned transition model. On its own, this future prediction objective outperforms prior methods for sample-efficient deep RL from pixels. We further improve performance by adding data augmentation to the future prediction loss, which forces the agent’s representations to be consistent across multiple views of an observation. Our full self-supervised objective, which combines future prediction and data augmentation, achieves a median human-normalized score of 0.415 on Atari in a setting limited to 100k steps of environment interaction, which represents a 55% relative improvement over the previous state-of-the-art. Notably, even in this limited data regime, SPR exceeds expert human scores on 7 out of 26 games. We’ve made the code associated with this work available at <https://github.com/mila-iqia/spr>.

6.1 INTRODUCTION

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

Deep RL involves training agents based on large neural networks using large amounts of data ([Sutton, 2019](#)), a trend evident across both model-based ([Schrittwieser et al., 2020](#)) and model-free ([Badia et al., 2020](#)) learning. The sample complexity of such state-of-the-art agents is often incredibly high: MuZero ([Schrittwieser et al., 2020](#)) and Agent-57 ([Badia et al., 2020](#)) use 10-50 years of experience per Atari game, and OpenAI Five ([OpenAI et al., 2019](#)) uses *45,000 years* of experience to accomplish its remarkable performance. This is clearly impractical: unlike easily-simulated environments such as video games, collecting interaction data for many real-world tasks is costly, making improved *data efficiency* a prerequisite for successful use of deep RL in these settings ([Dulac-Arnold et al., 2019](#)).

Meanwhile, new self-supervised representation learning methods have significantly improved data efficiency when learning new vision and language tasks, particularly in low data regimes or semi-supervised learning ([Xie et al., 2020](#); [Hénaff et al., 2019](#); [Chen et al., 2020b](#)). Self-supervised methods improve data efficiency by leveraging a nearly limitless supply of training signal from tasks generated on-the-fly, based on “views” drawn from the natural structure of the data (e.g., image patches, data augmentation or temporal proximity, see [Noroozi et al., 2016](#); [Oord et al., 2018](#); [Hjelm et al., 2019a](#); [Tian et al., 2019](#); [Bachman et al., 2019](#); [He et al., 2020](#); [Chen et al., 2020a](#)).

Motivated by successes in semi-supervised and self-supervised learning ([Tarvainen et al., 2017](#); [Xie et al., 2020](#); [Grill et al., 2020b](#)), we train better state representations for RL by forcing representations to be temporally predictive and consistent when subject to data augmentation. Specifically, we extend a strong model-free agent by adding a dynamics model which predicts future latent representations provided by a parameter-wise exponential moving average of the agent itself. We also add data augmentation to the future prediction task, which enforces consistency across different views of each observation. Contrary to some methods ([Kaiser et al., 2019](#);

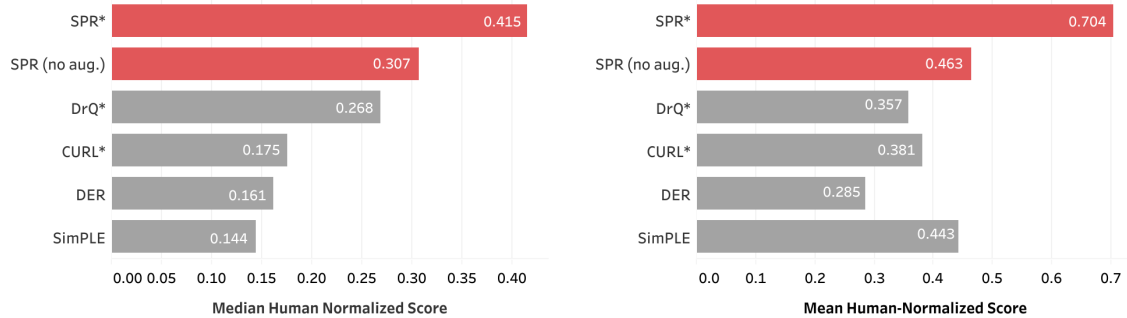


Figure 7: Median and Mean Human-Normalized scores of different methods across 26 games in the Atari 100k benchmark (Kaiser et al., 2019), averaged over 10 random seeds for SPR, and 5 seeds for most other methods except CURL, which uses 20. Each method is allowed access to only 100k environment steps or 400k frames per game. (*) indicates that the method uses data augmentation. SPR achieves state-of-art results on both mean and median human-normalized scores. Note that, even without data augmentation, SPR still outperforms all prior methods on both metrics.

Hafner et al., 2019), our dynamics model operates entirely in the latent space and does not rely on reconstructing raw states.

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

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

6.2 METHOD

We consider reinforcement learning (RL) in the standard Markov Decision Process (MDP) setting where an agent interacts with its environment in *episodes*, each consisting of sequences of observations, actions and rewards. We use s_t , a_t and r_t to denote the state, the action taken by the agent and the reward received at timestep t . We seek to train an agent whose expected cumulative reward in each episode is maximized. To do this, we combine a strong model-free RL algorithm, Rainbow (Hessel et al., 2018), with Self-Predictive Representations as an auxiliary loss to improve sample efficiency. We now describe our overall approach in detail.

6.2.1 Deep Q-Learning

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

$$\mathcal{L}_\theta^{DQN} = \left(Q_\theta(s_t, a_t) - (r_t + \gamma \max_a Q_\xi(s_{t+1}, a)) \right)^2. \quad (7)$$

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

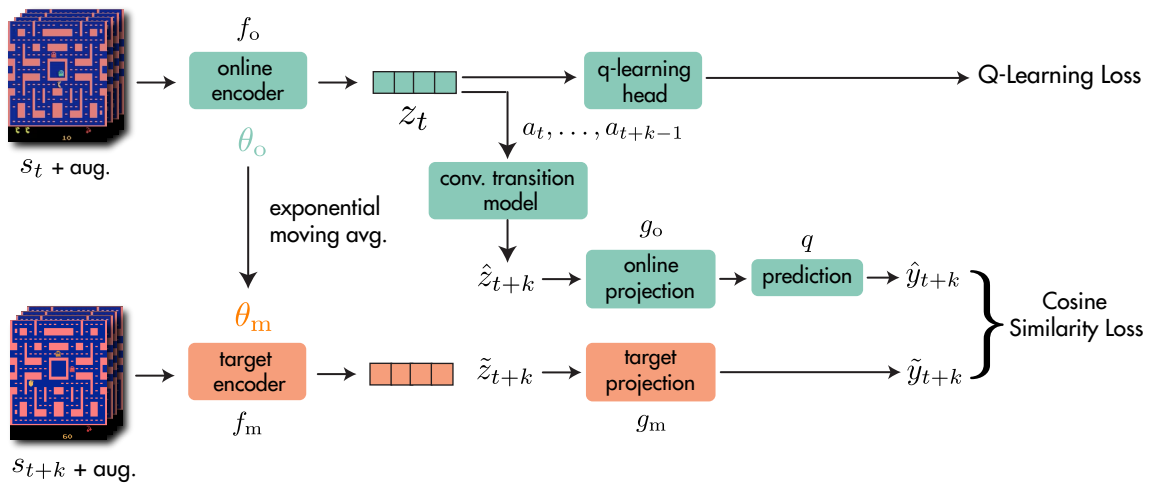


Figure 8: An illustration of the full SPR method. Representations from the online encoder are used in the reinforcement learning task and for prediction of future representations from the target encoder via the transition model. The target encoder and projection head are defined as an exponential moving average of their online counterparts and are not updated via gradient descent. For brevity, we illustrate only the k^{th} step of future prediction, but in practice we compute the loss over all steps from 1 to K . Note: our implementation for this paper includes g_o in the Q-learning head.

6.2.2 Self-Predictive Representations

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

- **Online and Target networks:** We use an *online encoder* f_o to transform observed states s_t into representations $z_t \triangleq f_o(s_t)$. We use these representations in an objective that encourages them to be *predictive* of future observations up to some fixed temporal offset K , given a sequence of K actions to perform. We augment each observation s_t independently when using data augmentation. Rather than predicting representations produced by the online encoder, we follow prior work (Tarvainen et al., 2017; Grill et al., 2020b) by computing target representations for future states using a *target encoder* f_m , whose parameters are an exponential moving average (EMA) of the online encoder parameters. Denoting the parameters of f_o as θ_o , those of f_m as θ_m , and the EMA coefficient as $\tau \in [0, 1)$, the update rule for θ_m is:

$$\theta_m \leftarrow \tau \theta_m + (1 - \tau) \theta_o. \quad (8)$$

The target encoder is not updated via gradient descent. The special case $\tau = 0$, $\theta_m = \theta_o$ is noteworthy, as it performs well when regularization is already provided by data augmentation.

- **Transition Model:** For the prediction objective, we generate a sequence of K predictions $\hat{z}_{t+1:t+K}$ of future state representations $\tilde{z}_{t+1:t+K}$ using an action-conditioned *transition model* h . We compute $\hat{z}_{t+1:t+K}$ iteratively: $\hat{z}_{t+k+1} \triangleq h(\hat{z}_{t+k}, a_{t+k})$, starting from $\hat{z}_t \triangleq z_t \triangleq f_o(s_t)$. We compute $\tilde{z}_{t+1:t+K}$ by applying the target encoder f_m to the observed future states $s_{t+1:t+K}$: $\tilde{z}_{t+k} \triangleq f_m(s_{t+k})$. The transition model and prediction loss operate in the latent space, thus avoiding pixel-based reconstruction objectives. We describe the architecture of h in section 6.2.3.
- **Projection Heads:** We use online and target projection heads g_o and g_m (Chen et al., 2020a) to project online and target representations to a smaller latent

space, and apply an additional *prediction head* q (Grill et al., 2020b) to the online projections to predict the target projections:

$$\hat{y}_{t+k} \triangleq q(g_o(\hat{z}_{t+k})), \forall \hat{z}_{t+k} \in \hat{z}_{t+1:t+K}; \quad \tilde{y}_{t+k} \triangleq g_m(\tilde{z}_{t+k}), \forall \tilde{z}_{t+k} \in \tilde{z}_{t+1:t+K}. \quad (9)$$

The target projection head parameters are given by an EMA of the online projection head parameters, using the same update as the online and target encoders.

- **Prediction Loss:** We compute the future prediction loss for SPR by summing over cosine similarities¹ between the predicted and observed representations at timesteps $t+k$ for $1 \leq k \leq K$:

$$\mathcal{L}_\theta^{\text{SPR}}(s_{t:t+K}, a_{t:t+K}) = - \sum_{k=1}^K \left(\frac{\tilde{y}_{t+k}}{\|\tilde{y}_{t+k}\|_2} \right)^\top \left(\frac{\hat{y}_{t+k}}{\|\hat{y}_{t+k}\|_2} \right), \quad (10)$$

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

We call our method Self-Predictive Representations (SPR), following the predictive nature of the objective and the use of an exponential moving average target network similar to (Tarvainen et al., 2017; He et al., 2020). During training, we combine the SPR loss with the Q-learning loss for Rainbow. The SPR loss affects f_o , g_o , q and h . The Q-learning loss affects f_o and the Q-learning head, which contains additional layers specific to Rainbow. Denoting the Q-learning loss from Rainbow as $\mathcal{L}_\theta^{\text{RL}}$, our full optimization objective is: $\mathcal{L}_\theta^{\text{total}} = \mathcal{L}_\theta^{\text{RL}} + \lambda \mathcal{L}_\theta^{\text{SPR}}$.

Unlike some other proposed methods for representation learning in reinforcement learning (Srinivas et al., 2020), SPR can be used with or without data augmentation, including in contexts where data augmentation is unavailable or counterproductive. Moreover, compared to related work on contrastive representation learning, SPR does not use negative samples, which may require careful design of contrastive tasks, large batch sizes (Chen et al., 2020a), or the use of a buffer to emulate large batch sizes (He et al., 2020)

¹ Cosine similarity is linearly related to the “normalized L2” loss used in BYOL (Grill et al., 2020b).

Algorithm 1: Self-Predictive Representations

```
1 Denote parameters of online encoder  $f_o$  and projection  $g_o$  as  $\theta_o$ 
2 Denote parameters of target encoder  $f_m$  and projection  $g_m$  as  $\theta_m$ 
3 Denote parameters of transition model  $h$ , predictor  $q$  and Q-learning head as
   $\phi$ 
4 Denote the maximum prediction depth as  $K$ , batch size as  $N$ 
5 initialize replay buffer  $B$ 
6 while Training do
7   collect experience  $(s, a, r, s')$  with  $(\theta_o, \phi)$  and add to buffer  $B$ 
8   sample a minibatch of sequences of  $(s, a, r, s') \sim B$ 
9   for  $i$  in  $\text{range}(0, N)$  do
10    if augmentation then
11       $s^i \leftarrow \text{augment}(s^i); s'^i \leftarrow \text{augment}(s'^i)$ 
12    end
13     $z_0^i \leftarrow f_\theta(s_0^i)$  // online representations
14     $l^i \leftarrow 0$ 
15    for  $k$  in  $(1, \dots, K)$  do
16       $\hat{z}_k^i \leftarrow h(\hat{z}_{k-1}^i, a_{k-1}^i)$  // latent states via transition model
17       $\tilde{z}_k^i \leftarrow f_m(s_k^i)$  // target representations
18       $\hat{y}_k^i \leftarrow q(g_o(\hat{z}_k^i)), \tilde{y}_k^i \leftarrow g_m(\tilde{z}_k^i)$  // projections
19       $l^i \leftarrow l^i - \left( \frac{\hat{y}_k^i}{\|\hat{y}_k^i\|_2} \right)^\top \left( \frac{\tilde{y}_k^i}{\|\tilde{y}_k^i\|_2} \right)$  // SPR loss at step  $k$ 
20    end
21     $l^i \leftarrow \lambda l^i + \text{RL loss}(s^i, a^i, r^i, s'^i; \theta_o)$  // Add RL loss for batch with
     $\theta_o$ 
22  end
23   $l \leftarrow \frac{1}{N} \sum_{i=0}^N l^i$  // average loss over minibatch
24   $\theta_o, \phi \leftarrow \text{optimize}((\theta_o, \phi), l)$  // update online parameters
25   $\theta_m \leftarrow \tau \theta_o + (1 - \tau) \theta_m$  // update target parameters
26 end
```

6.2.3 Transition Model Architecture

For the transition model h , we apply a convolutional network directly to the $64 \times 7 \times 7$ spatial output of the convolutional encoder f_o . The network comprises two 64-channel convolutional layers with 3×3 filters, with batch normalization (Ioffe et al., 2015a) after the first convolution and ReLU nonlinearities after each convolution. We append a one-hot vector representing the action taken to each location in the input to the first convolutional layer, similar to Schrittwieser et al. (2020). We use a maximum prediction depth of $K = 5$, and we truncate calculation of the SPR loss at episode boundaries to avoid encoding environment reset dynamics into the model.

6.2.4 Data Augmentation

When using augmentation, we use the same set of image augmentations as in DrQ from Yarats et al. (2021b), consisting of small random shifts and color jitter. We normalize activations to lie in $[0, 1]$ at the output of the convolutional encoder and transition model, as in Schrittwieser et al., 2020. We use Kornia (Riba et al., 2020) for efficient GPU-based data augmentations.

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

6.2.5 Implementation Details

For our Atari experiments, we largely follow Hasselt et al., 2019 for DQN hyperparameters, with four exceptions. We follow DrQ (Yarats et al., 2021b) by: using the 3-layer convolutional encoder from Mnih et al., 2015, using 10-step returns instead of 20-step returns for Q-learning, and not using a separate DQN target network when using augmentation. We also perform two gradient steps per environment step instead of one. We show results for this configuration with and without augmentation in Table 19, and confirm that these changes are not themselves responsible for our performance. We reuse the first layer of the DQN MLP head as the SPR

projection head g_o . When using dueling DQN (Wang et al., 2016b), g_o concatenates the outputs of the first layers of the value and advantage heads. When these layers are noisy (Fortunato et al., 2018), g_o does not use the noisy parameters. Finally, we parameterize the predictor q as a linear layer. We use $\tau = 0.99$ when augmentation is disabled and $\tau = 0$ when enabled. For $\mathcal{L}_\theta^{\text{total}} = \mathcal{L}_\theta^{\text{RL}} + \lambda\mathcal{L}_\theta^{\text{SPR}}$, we use $\lambda = 2$. Hyperparameters were tuned over a subset of games (following Mnih et al., 2015; Machado et al., 2018). We list the complete hyperparameters in Table 17.

Our implementation uses rlpvt (Stooke et al., 2019) and PyTorch (Paszke et al., 2019). We find that SPR modestly increases the time required for training, which we discuss in more detail in Appendix 12.6.

6.3 RELATED WORK

6.3.1 Data-Efficient RL:

A number of works have sought to improve sample efficiency in deep RL. SiMPLe (Kaiser et al., 2019) learns a pixel-level transition model for Atari to generate simulated training data, achieving strong results on several games in the 100k frame setting, at the cost of requiring several weeks for training. However, Hasselt et al., 2019 and Kielak, 2020 introduce variants of Rainbow (Hessel et al., 2018) tuned for sample efficiency, Data-Efficient Rainbow (DER) and OTRainbow, which achieve comparable or superior performance with far less computation.

In the context of continuous control, several works propose to leverage a latent-space model trained on reconstruction loss to improve sample efficiency (Hafner et al., 2019; Lee et al., 2020; Hafner et al., 2020). Most recently, DrQ (Yarats et al., 2021b) and RAD (Laskin et al., 2020) have found that applying modest image augmentation can substantially improve sample efficiency in reinforcement learning, yielding better results than prior model-based methods. Data augmentation has also been found to improve generalization of reinforcement learning methods (Combes et al., 2018; Laskin et al., 2020) in multi-task and transfer settings. We show that data augmentation can be more effectively leveraged in reinforcement learning by forcing representations to be consistent between different augmented views of an observation while also predicting future latent states.

6.3.2 Representation Learning in RL:

Representation learning has a long history of use in RL – see [Lesort et al., 2018](#). For example, CURL ([Srinivas et al., 2020](#)) proposed a combination of image augmentation and a contrastive loss to perform representation learning for RL. However, follow-up results from RAD ([Laskin et al., 2020](#)) suggest that most of the benefits of CURL come from image augmentation, not its contrastive loss.

CPC ([Oord et al., 2018](#)), CPC | Action ([Guo et al., 2018](#)), ST-DIM ([Anand et al., 2019](#)) and DRIML ([Mazouze et al., 2020](#)) propose to optimize various temporal contrastive losses in reinforcement learning environments. We perform an ablation comparing such temporal contrastive losses to our method in Section 6.5. [Kipf et al., 2019](#) propose to learn object-oriented contrastive representations by training a structured transition model based on a graph neural network.

SPR bears some resemblance to DeepMDP ([Gelada et al., 2019](#)), which trains a transition model with an unnormalized L2 loss to predict representations of future states, along with a reward prediction objective. However, DeepMDP uses its online encoder to generate prediction targets rather than employing a target encoder, and is thus prone to representational collapse (sec. C.5 in [Gelada et al., 2019](#)). To mitigate this issue, DeepMDP relies on an additional observation reconstruction objective. In contrast, our model is self-supervised, trained entirely in the latent space, and uses a normalized loss. Our ablations (sec. 6.5) demonstrate that using a target encoder has a large impact on our method.

SPR is also similar to PBL ([Guo et al., 2020](#)), which directly predicts representations of future states. However, PBL uses two separate target networks trained via gradient descent, whereas SPR uses a single target encoder, updated without backpropagation. Moreover, PBL studies multi-task generalization in the asymptotic limits of data, whereas SPR is concerned with single-task performance in low data regimes, using 0.01% as much data as PBL. Unlike PBL, SPR additionally enforces consistency across augmentations, which empirically provides a large boost in performance.

6.4 RESULTS

We test SPR on the sample-efficient Atari setting introduced by [Kaiser et al., 2019](#) and [Hasselt et al., 2019](#). In this task, only 100,000 environment steps of training data

are available – equivalent to 400,000 frames, or just under two hours – compared to the typical standard of 50,000,000 environment steps, or roughly 39 days of experience. When used without data augmentation, SPR demonstrates scores comparable to the previous best result from [Yarats et al., 2021b](#). When combined with data augmentation, SPR achieves a median human-normalized score of 0.415, which is a new state-of-the-art result on this task. SPR achieves super-human performance on seven games in this data-limited setting: Boxing, Krull, Kangaroo, Road Runner, James Bond and Crazy Climber, compared to a maximum of two for any previous methods, and achieves scores higher than DrQ (the previous state-of-the-art method) on 23 out of 26 games. See [Table 6](#) for aggregate metrics and [Figure 9](#) for a visualization of results. A full list of scores is presented in [Table 18](#). For consistency with previous works, we report human and random scores from [Wang et al., 2016b](#).

Table 6: Performance of different methods on the 26 Atari games considered by [Kaiser et al., 2019](#) after 100k environment steps. Results are recorded at the end of training and averaged over 10 random seeds for SPR, 20 for CURL, and 5 for other methods. SPR outperforms prior methods on all aggregate metrics, and exceeds expert human performance on 7 out of 26 games.

Metric	Random	Human	SimPLe	DER	OTRainbow	CURL	DrQ	SPR (no Aug)	SPR
Mean Human-Norm'd	0.000	1.000	0.443	0.285	0.264	0.381	0.357	0.463	0.704
Median Human-Norm'd	0.000	1.000	0.144	0.161	0.204	0.175	0.268	0.307	0.415
Mean DQN@50M-Norm'd	0.000	23.382	0.232	0.239	0.197	0.325	0.171	0.336	0.510
Median DQN@50M-Norm'd	0.000	0.994	0.118	0.142	0.103	0.142	0.131	0.225	0.361
# Games Superhuman	0	N/A	2	2	1	2	2	5	7

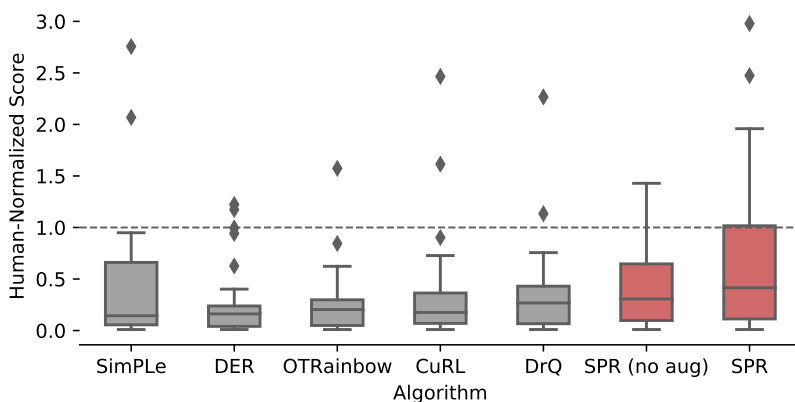


Figure 9: A boxplot of the distribution of human-normalized scores across the 26 Atari games under consideration, after 100k environment steps. The whiskers represent the interquartile range of human-normalized scores over the 26 games. Scores for each game are recorded at the end of training and averaged over 10 random seeds for SPR, 20 for CURL, and 5 for other methods.

6.4.1 Evaluation

We evaluate the performance of different methods by computing the average episodic return at the end of training. We normalize scores with respect to expert human scores to account for different scales of scores in each game, as done in previous works. The human-normalized score of an agent on a game is calculated as $\frac{\text{agent score} - \text{random score}}{\text{human score} - \text{random score}}$ and aggregated across the 26 games by mean or median.

We find that human scores on some games are so high that differences between methods are washed out by normalization, making it hard for these games to influence aggregate metrics. Moreover, we find that the median score is typically only influenced by a handful of games. Both these factors compound together to make the median human-normalized score an unreliable metric for judging overall performance. To address this, we also report DQN-normalized scores, defined analogously to human-normalized scores and calculated using scores from DQN agents (Mnih et al., 2015) trained over 50 million steps, and report both mean and median of those metrics in all results and ablations, and plot the distribution of scores over all the games in Figure 9.

Additionally, we note that the standard evaluation protocol of evaluating over only 500,000 frames per game is problematic, as the quantity we are trying to measure

Table 7: Scores on the 26 Atari games under consideration for ablated variants of SPR. All variants listed here use data augmentation.

Variant	Human-Normalized Score		DQN@50M-Normalized Score	
	mean	median	mean	median
SPR	0.704	0.415	0.510	0.361
1-step SPR	0.570	0.301	0.337	0.346
Non-temporal SPR	0.507	0.271	0.326	0.295
Quadratic SPR	0.047	0.040	-0.016	0.031
SPR without projections	0.437	0.171	0.247	0.174
Rainbow (controlled, w/ aug.)	0.480	0.346	0.284	0.278

is expected return over episodes. Because episodes may last up to up to 108,000 frames, this method may collect as few as four complete episodes. As variance of results is already a concern in deep RL see [Henderson et al., 2018](#), we recommend evaluating over 100 episodes irrespective of their length. Moreover, to address findings from [Henderson et al., 2018](#) that comparisons based on small numbers of random seeds are unreliable, we average our results over ten random seeds, twice as many as most previous works.

6.5 ANALYSIS

The target encoder We find that using a separate target encoder is vital in all cases. A variant of SPR in which target representations are generated by the online encoder without a stopgradient (as done by e.g., [Gelada et al., 2019](#)) exhibits catastrophically reduced performance, with median human-normalized score of 0.278 with augmentation versus 0.415 for SPR. However, there is more flexibility in the EMA constant used for the target encoder. When using augmentation, a value of $\tau = 0$ performs best, while without augmentation we use $\tau = 0.99$. The success of $\tau = 0$ is interesting, since the related method BYOL reports very poor representation learning performance in this case. We hypothesize that optimizing a reinforcement learning objective in parallel with the SPR loss explains this difference, as it provides an additional gradient which discourages representational collapse. Full results for these experiments are presented in Section [12.5](#).

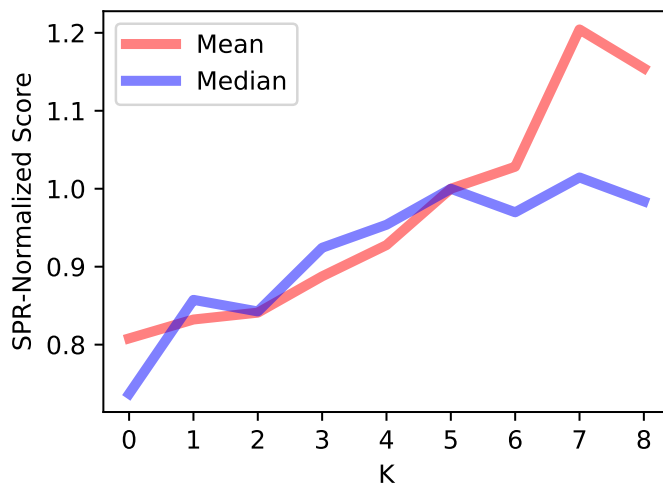


Figure 10: Performance of SPR with various prediction depths. Results are averaged across ten seeds per game, for all 26 games. To equalize the importance of games, we calculate an SPR-normalized score analogously to human-normalized scores, and show its mean and median across all 26 games. All other hyperparameters are identical to those used for SPR with augmentation.

Dynamics modeling is key A key distinction between SPR and other recent approaches leveraging representation learning for reinforcement learning, such as CURL (Srinivas et al., 2020) and DRIML (Mazouze et al., 2020), is our use of an explicit multi-step dynamics model. To illustrate the impact of dynamics modeling, we test SPR with a variety of prediction depths K . Two of these ablations, one with no dynamics modeling and one that models only a single step of dynamics, are presented in Table 7 (as *Non-temporal SPR* and *1-step SPR*), and all are visualized in Figure 10. We find that extended dynamics modeling consistently improves performance up to roughly $K = 5$. Moving beyond this continues to improve performance on a subset of games, at the cost of increased computation. Note that the non-temporal ablation we test is similar to using BYOL (Grill et al., 2020b) as an auxiliary task, with particular architecture choices made for the projection layer and predictor.

Comparison with contrastive losses Though many recent works in representation learning employ contrastive learning, we find that SPR consistently outperforms both temporal and non-temporal variants of contrastive losses (see Table 20), including CURL (Srinivas et al., 2020).

Using a quadratic loss causes collapse SPR’s use of a cosine similarity objective (or a normalized L2 loss) sets it in contrast to some previous works, such as DeepMDP (Gelada et al., 2019), which have learned latent dynamics models by minimizing an un-normalized L2 loss over predictions of future latents. To examine the importance of this objective, we test a variant of SPR that minimizes un-normalized L2 loss (*Quadratic SPR* in Table 7), and find that it performs only slightly better than random. This is consistent with results from Gelada et al., 2019, who find that DeepMDP’s representations are prone to collapse, and use an auxiliary reconstruction objective to prevent this.

Projections are critical Another distinguishing feature of SPR is its use of projection and prediction networks. We test a variant of SPR that uses neither, instead computing the SPR loss directly over the $64 \times 7 \times 7$ convolutional feature map used by the transition model (*SPR without projections* in Table 7). We find that this variant has inferior performance, and suggest two possible explanations. First, the convolutional network represents only a small fraction of the capacity of SPR’s network, containing only some 80,000 parameters out of a total of three to four million. Employing the first layer of the DQN head as a projection thus allows the SPR objective to affect far more of the network, while in this variant its impact is limited. Second, the effects of SPR in forcing invariance to augmentation may be undesirable at this level; as the convolutional feature map is the product of only three layers, it may be challenging to learn features that are simultaneously rich and invariant.

6.6 FUTURE WORK

Recent work in both visual (Chen et al., 2020b) and language representation learning (Brown et al., 2020) has suggested that self-supervised models trained on large datasets perform exceedingly well on downstream problems with limited data, often outperforming methods trained using only task-specific data. Future works could similarly exploit large corpora of unlabelled data, perhaps from multiple MDPs or raw videos, to further improve the performance of RL methods in low-data regimes. As the SPR objective is unsupervised, it could be directly applied in such settings.

Another interesting direction is to use the transition model learned by SPR for planning. MuZero (Schrittwieser et al., 2020) has demonstrated that planning with a model supervised via reward and value prediction can work extremely well given sufficient (massive) amounts of data. It remains unclear whether such models can work well in low-data regimes, and whether augmenting such models with self-supervised objectives such as SPR can improve their data efficiency.

It would also be interesting to examine whether self-supervised methods like SPR can improve generalization to unseen tasks or changes in environment, similar to how unsupervised pretraining on ImageNet can generalize to other datasets (He et al., 2020; Grill et al., 2020b).

6.7 CONCLUSION

In this paper we introduced Self-Predictive Representations (SPR), a self-supervised representation learning algorithm designed to improve the data efficiency of deep reinforcement learning agents. SPR learns representations that are both temporally predictive and consistent across different views of environment observations, by directly predicting representations of future states produced by a target encoder. SPR achieves state-of-the-art performance on the 100k steps Atari benchmark, demonstrating significant improvements over prior work. Our experiments show that SPR is highly robust, and is able to outperform the previous state of the art when either data augmentation or temporal prediction is disabled. We identify important directions for future work, and hope continued research at the intersection of self-supervised learning and reinforcement learning leads to algorithms which rival the efficiency and robustness of humans.

ARTICLE III

PROLOGUE TO THE THIRD ARTICLE

7.1 ARTICLE DETAILS

Title: Procedural Generalization by Planning with Self-Supervised World Models

Authors: Ankesh Anand, Jacob Walker, Yazhe Li, Eszter Vértés, Julian Schrittwieser, Sherjil Ozair, Théophane Weber, Jessica B. Hamrick

Presented At: International Conference on Learning Representations (ICLR), 2022

7.2 CONTEXT

Having made headways on data-efficiency in our prior work, I was interested in investigating if world models and self-supervised learning could similarly help in generalization of RL algorithms. MuZero (Schrittwieser et al., 2020) was the leading model-based method, so we sought to investigate its generalization capability and whether self-supervised learning provides any benefits there.

7.3 CONTRIBUTIONS

I proposed the research idea to Jess Hamrick, who provided very detailed and concrete feedback which helped solidify the research plan. Jess helped me navigate the DeepMind technical infrastructure and I wrote the code to run MuZero on Procgen and Metaworld. I also implemented the SPR and Pixel-prediction methods and ran experiments with them. Jacob Walker implemented the contrastive methods. Jacob also ran some important ablations. Julian and Sherjil helped me resolve questions about the MuZero infrastructure and codebase at DeepMind. Eszter, Jess

and Theo provided detailed feedback during research meetings. Yazhe implemented the Q-learning baseline, and ran experiments on more seeds along with Eszter. Me and Jess wrote most of the initial draft of the paper which was refined with feedback from everyone. Jess also provided very frequent advice on the project and helped prioritize experiments.

7.4 RESEARCH IMPACT

This work remains the best performing method on the Procgen benchmark, and represented a significant improvement over existing methods. It was also the first model-based method shown to work on generalization-first benchmarks of Procgen and MetaWorld. The work was cited in the 2021 AI Index Report ([Zhang et al., 2022](#)) as one of the key indicators of progress in Reinforcement Learning. Moreover, the work identified the strengths and weakness of MuZero, a staple RL algorithm these days which and is helping guide its further usage and algorithm improvements. Lastly, it has paved way for more model-based methods to be used to exhibit generalization in a way that model-free methods lack. More recently, we built on this work and conducted further analysis on the benefits model-based methods in exploration and transfer ([Walker et al., 2023](#)).

PROCEDURAL GENERALIZATION BY PLANNING WITH SELF-SUPERVISED WORLD MODELS

Abstract: One of the key promises of model-based reinforcement learning is the ability to generalize using an internal model of the world to make predictions in novel environments and tasks. However, the generalization ability of model-based agents is not well understood because existing work has focused on model-free agents when benchmarking generalization. Here, we explicitly measure the generalization ability of model-based agents in comparison to their model-free counterparts. We focus our analysis on MuZero (Schrittwieser et al., 2020), a powerful model-based agent, and evaluate its performance on both procedural and task generalization. We identify three factors of procedural generalization—planning, self-supervised representation learning, and procedural data diversity—and show that by combining these techniques, we achieve state-of-the-art generalization performance and data efficiency on Procgen (Cobbe et al., 2020). However, we find that these factors do not always provide the same benefits for the task generalization benchmarks in Meta-World (Yu et al., 2020b), indicating that transfer remains a challenge and may require different approaches than procedural generalization. Overall, we suggest that building generalizable agents requires moving beyond the single-task, model-free paradigm and towards self-supervised model-based agents that are trained in rich, procedural, multi-task environments.

8.1 INTRODUCTION

The ability to generalize to previously unseen situations or tasks using an internal model of the world is a hallmark capability of human general intelligence (Craig, 1952; Lake et al., 2017) and is thought by many to be of central importance in

machine intelligence as well (Dayan et al., 1995; Ha et al., 2018; Schmidhuber, 1991; Sutton, 1991). Although significant strides have been made in model-based systems in recent years (Hamrick, 2019), the most popular model-based benchmarks consist of identical training and testing environments (e.g. Hafner et al., 2021; Wang et al., 2019b) and do not measure or optimize for generalization at all. While plenty of other work in model-based RL does measure generalization (e.g. Finn et al., 2017b; Nagabandi et al., 2019; Weber et al., 2017; Zhang et al., 2018a), each approach is typically evaluated on a bespoke task, making it difficult to ascertain the state of generalization in model-based RL more broadly.

Model-free RL, like model-based RL, has also suffered from both the “train=test” paradigm and a lack of standardization around how to measure generalization. In response, recent papers have discussed what generalization in RL means and how to measure it (Chollet, 2019; Cobbe et al., 2019; Justesen et al., 2019; Nichol et al., 2018; Witty et al., 2018), and others have proposed new environments such as Procgen (Cobbe et al., 2020) and Meta-World (Yu et al., 2020b) as benchmarks focusing on measuring generalization. While popular in the model-free community (e.g. Mohanty et al., 2021; Yu et al., 2020a; Zhang et al., 2021), these benchmarks have not yet been widely adopted in the model-based setting. It is therefore unclear whether model-based methods outperform model-free approaches when it comes to generalization, how well model-based methods perform on standardized benchmarks, and whether popular model-free algorithmic improvements such as self-supervision (Dittadi et al., 2021; Mazoure et al., 2022) or procedural data diversity (Tobin et al., 2017; Zhang et al., 2018b) yield the same benefits for generalization in model-based agents.

In this paper, we investigate three factors of generalization in model-based RL: planning, self-supervised representation learning, and procedural data diversity. We analyze these methods through a variety of modifications and ablations to MuZero Reanalyse (Schrittwieser et al., 2020; Schrittwieser et al., 2021), a state-of-the-art model-based algorithm. To assess generalization performance, we test our variations of MuZero on two types of generalization (See Figure 11): *procedural* and *task*. Procedural generalization involves evaluating agents on unseen configurations of an environment (e.g., changes in observation rendering, map or terrain changes, or new goal locations) while keeping the reward function largely the same. Task generalization, in contrast, involves evaluating agents’ adaptability to unseen reward functions (“tasks”) within the same environment. We focus on two benchmarks

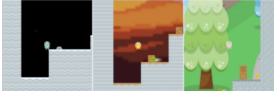



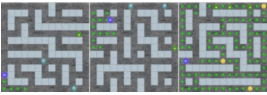




	Train	Test
Procedural Generalization	  	  
Task Generalization	 	

Figure 11: Two different kinds of generalization, using Procgen and Meta-World as examples. *Procedural* generalization involves evaluating on unseen environment configurations, whereas *task* generalization evaluates adaptability to unseen tasks (reward functions).

designed for both types of generalization, Procgen (Cobbe et al., 2020) and Meta-World (Yu et al., 2020b).

Our results broadly indicate that self-supervised, model-based agents hold promise in making progress towards better generalization. We find that (1) MuZero achieves state-of-the-art performance on Procgen and the procedural and multi-task Meta-World benchmarks (ML-1 and ML-45 train), outperforming a controlled model-free baseline; (2) MuZero’s performance and data efficiency can be improved with the incorporation of self-supervised representation learning; and (3) that with self-supervision, less data diversity is required to achieve good performance. However, (4) these ideas help less for task generalization on the ML-45 test set from Meta-World, suggesting that different forms of generalization may require different approaches.

8.2 MOTIVATION AND BACKGROUND

8.2.1 Generalization in RL

We are interested in the setting where an agent is trained on a set of n_{train} MDPs drawn IID from the same distribution, $\mathbb{M}_{\text{train}} = \{\mathcal{M}_i\}_{i=1}^{n_{\text{train}}}$, where $\mathcal{M}_i \sim p(\mathcal{M})$. The agent is then tested on another set of n_{test} MDPs drawn IID¹ from p , but disjoint from the training set: $\mathbb{M}_{\text{test}} = \{\mathcal{M}_j\}_{j=1}^{n_{\text{test}}}$ where $\mathcal{M}_j \sim p(\mathcal{M})$ such that $\mathcal{M}_j \notin \mathbb{M}_{\text{train}} \forall \mathcal{M}_j$. *Generalization*, then, is how well the agent performs in expectation on the test MDPs after training (analogous to how well a supervised model performs on the test set). Qualitatively, generalization difficulty can be roughly seen as a function of the number (n_{train}) of distinct MDPs seen during training (what we will refer to as *data diversity*), as well as the breadth or amount of variation in $p(\mathcal{M})$ itself. Intuitively, if the amount of diversity in the training set is small (i.e. low values of n_{train}) relative to the breadth of $p(\mathcal{M})$, then this poses a more difficult generalization challenge. As $n_{\text{train}} \rightarrow \infty$, we will eventually enter a regime where $p(\mathcal{M})$ is densely sampled, and generalization should become much easier.

Given the above definition of generalization, we distinguish between two qualitative types of distributions over MDPs. In *procedural* generalization, all MDPs with non-zero probability under $p(\mathcal{M})$ share the same underlying logic to their dynamics (e.g., that walls are impassable) and rewards (e.g., that coins are rewarding), but differ in how the environment is laid out (e.g., different mazes) and in how it is rendered (e.g., color or background). In *task* generalization, MDPs under $p(\mathcal{M})$ share the same dynamics and rendering, but differ in the reward function (e.g., picking an object up vs. pushing it), which may be parameterized (e.g. specifying a goal location).

Procedural generalization Many recent works attempt to improve procedural generalization in different ways. For example, techniques that have been successful in supervised learning have also been shown to help procedural generalization in RL, including regularization (Cobbe et al., 2020; Igl et al., 2019; Laskin et al., 2020) and auxiliary self-supervised objectives (Mazouze et al., 2020; Mazouze et al.,

¹ Technically, the different tasks contained in Meta-World (see Section 8.3) are not quite drawn IID as they were designed by hand. However, they qualitatively involve the same types of objects and level of complexity, so we feel this fits approximately into this regime.

2022). Other approaches including better hyperparameter tuning (Mohanty et al., 2021), curriculum learning strategies (Jiang et al., 2021), and stronger architectural inductive biases (Bapst et al., 2019; Guez et al., 2019; Zambaldi et al., 2018) may also be effective. However, these various ideas are often only explored in the model-free setting, and are not usually evaluated in combination, making it challenging to know which are the most beneficial for model-based RL. Here we focus explicitly on model-based agents, and evaluate three factors in combination: planning, self-supervision, and data diversity.

Task generalization Generalizing to new tasks or reward functions has been a major focus of meta-RL (Finn et al., 2017a; Rakelly et al., 2019), in which an agent is trained on a dense distribution of tasks during training and a meta-objective encourages few-shot generalization to new tasks. However, meta-RL works have primarily focused on model-free algorithms (though see Nagabandi et al., 2019), and the distribution from which train/test MDPs are drawn is typically quite narrow and low dimensional. Another approach to task generalization has been to first train task-agnostic representations (Yarats et al., 2021a) or dynamics (Sekar et al., 2020) in an exploratory pre-training phase, and then use them for transfer. Among these, Sekar et al. (2020) focus on the model-based setting but require access to the reward function during evaluation. In Section 8.4.2, we take a similar approach of pre-training models and using them for task transfer, with the goal of evaluating whether planning and self-supervised learning might assist in this.

8.2.2 Factors of Generalization

Planning Model-based RL is an active area of research (Hamrick, 2019; Moerland et al., 2020) with the majority of work focusing on gains in data efficiency (Ha et al., 2018; Hafner et al., 2021; Janner et al., 2019; Kaiser et al., 2020; Schrittwieser et al., 2021), though it is often motivated by a desire for better zero- or few-shot generalization as well (Finn et al., 2017b; Hamrick et al., 2021; Nagabandi et al., 2019; Sekar et al., 2020; Weber et al., 2017). In particular, model-based techniques may benefit data efficiency and generalization in three distinct ways. First, model learning can act as an auxiliary task and thus aid in learning representations that better capture the structure of the environment and enable faster learning (Gregor et al., 2019). Second, the learned model can be used to select actions on-the-fly via

MPC (Finn et al., 2017b), enabling faster adaptation in the face of novelty. Third, the model can also be used to train a policy or value function by simulating training data (Sutton, 1991) or constructing more informative losses (Grill et al., 2020a; Heess et al., 2015), again enabling faster learning (possibly entirely in simulation). A recent state-of-the-art agent, MuZero (Schrittwieser et al., 2020), combines all of these techniques: model learning, MPC, simulated training data, and model-based losses.² We focus our analysis on MuZero for this reason, and compare it to baselines that do not incorporate model-based components.

Self-supervision Model learning is itself a form of self-supervision, leveraging an assumption about the structure of MDPs in order to extract further learning signal from collected data than is possible via rewards alone. However, recent work has argued that this is unnecessary for model-based RL: all that should be required is for models to learn the task dynamics, not necessarily environment dynamics (Grimm et al., 2020). Yet even in the context of model-free RL, exploiting the structure of the dynamics has been shown to manifest in better learning efficiency, generalization, and representation learning (Mazouze et al., 2022; Schwarzer et al., 2021a; Yarats et al., 2021a). Here, we aim to test the impact of self-supervision in model-based agents by focusing on three popular classes of self-supervised losses: reconstruction, contrastive, and self-predictive. Reconstruction losses involve directly predicting future observations (e.g. Finn et al., 2017b; Hafner et al., 2021; Kaiser et al., 2020; Weber et al., 2017). Contrastive objectives set up a classification task to determine whether a future frame could result from the current observation (Banino et al., 2021; Kipf et al., 2019; Oord et al., 2018). Finally, self-predictive losses involve having agents predict their own future latent states (Guo et al., 2020; Schwarzer et al., 2021a).

Data diversity Several works have shown that exposing a deep neural network to a diverse training distribution can help its representations better generalize to unseen situations (Djolonga et al., 2021; Radford et al., 2021b). A dense sampling over a diverse training distribution can also act as a way to sidestep the out-of-

² In this paper, we refer to *planning* as some combination of MPC, simulated training data, and model-based losses (excluding model learning itself). We are agnostic to the question of how deep or far into the future the model must be used; indeed, it may be the case that in the environments we test, very shallow or even single-step planning may be sufficient, as in Hamrick et al. (2021) and Hessel et al. (2021).

distribution generalization problem (Radford et al., 2021b). In robotics, collecting and training over a diverse range of data has been found to be critical particularly in the sim2real literature where the goal is to transfer a policy learned in simulation to the real world (Andrychowicz et al., 2020; Tobin et al., 2017). Guez et al. (2019) also showed that in the context of zero-shot generalization, the amount of data diversity can have interesting interactions with other architectural choices such as model size. Here, we take a cue from these findings and explore how important procedural data diversity is in the context of self-supervised, model-based agents.

8.2.3 *MuZero*

We evaluate generalization with respect to a state-of-the-art model-based agent, MuZero (Schrittwieser et al., 2020). MuZero is an appealing candidate for investigating generalization because it already incorporates many ideas that are thought to be useful for generalization and transfer, including replay (Schrittwieser et al., 2021), planning (Silver et al., 2018), and model-based representation learning (Grimm et al., 2020; Oh et al., 2017). However, while these components contribute to strong performance across a wide range of domains, other work has suggested that MuZero does not necessarily achieve perfect generalization on its own (Hamrick et al., 2021). It therefore serves as a strong baseline but with clear room for improvement.

MuZero learns an implicit (or value-equivalent, see Grimm et al. (2020)) world model by simply learning to predict future rewards, values and actions. It then plans with Monte-Carlo Tree Search (MCTS) (Kocsis et al., 2006; Coulom, 2006) over this learned model to select actions for the environment. Data collected from the environment, as well as the results of planning, are then further used to improve the learned reward function, value function, and policy.

More specifically, MuZero optimizes the following loss at every time-step t , applied to a model that is unrolled $0 \dots K$ steps into the future: $l_t(\theta) = \sum_{k=0}^K (l_{\pi}^k + l_v^k + l_r^k) = \sum_{k=0}^K (\text{CE}(\hat{\pi}^k, \pi^k) + \text{CE}(\hat{v}^k, v^k) + \text{CE}(\hat{r}^k, r^k))$, where $\hat{\pi}^k$, \hat{v}^k and \hat{r}^k are respectively the policy, value and reward prediction produced by the k -step unrolled model. The targets for these predictions are drawn from the corresponding time-step $t + k$ of the real trajectory: π^k is the improved policy generated by the search tree, v^k is an n -step return bootstrapped by a target network, and r^k is the true reward. As MuZero uses a distributional approach for training the value and reward functions (Dabney et al., 2018), their losses involve computing the cross-entropy (CE); this

also typically results in better representation learning. For all our experiments, we specifically parameterize v and r as categorical distributions similar to the Atari experiments in [Schrittwieser et al. \(2020\)](#).

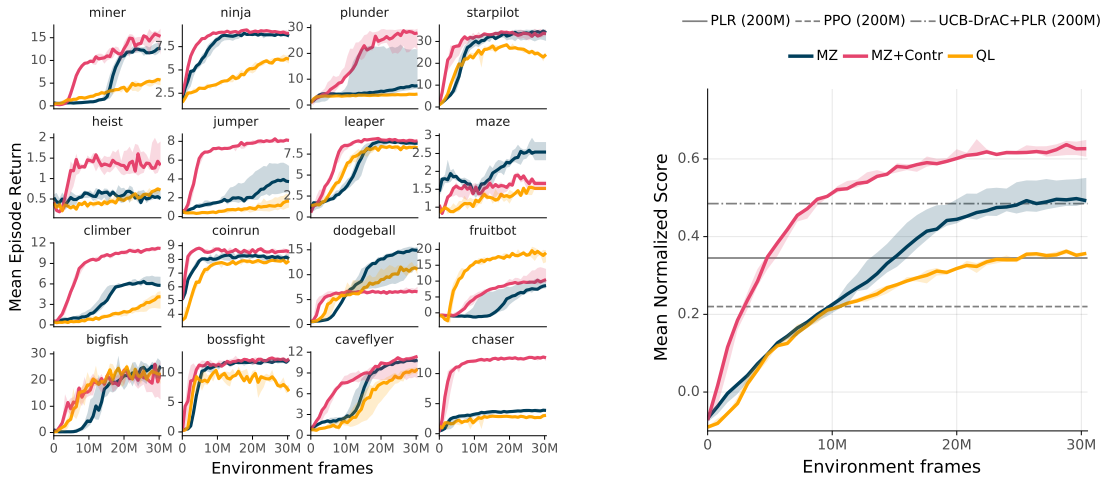
To enable better sample reuse and improved data efficiency, we use the *Reanalyse* version of MuZero ([Schrittwieser et al., 2021](#)). Reanalyse works by continuously re-running MCTS on existing data points, thus computing new improved training targets for the policy and value function. It does not change the loss function described in [Section 8.2.3](#). In domains that use continuous actions (such as Meta-World), we use Sampled MuZero ([Hubert et al., 2021](#)) that modifies MuZero to plan over sampled actions instead. See [Section 13.1.1](#) for more details on MuZero.

8.3 EXPERIMENTAL DESIGN

We analyze three potential drivers of generalization (planning, self-supervision, and data diversity) across two different environments. For each algorithmic choice, we ask: to what extent does it improve procedural generalization, and to what extent does it improve task generalization?

8.3.1 Environments

Procgen Procgen ([Cobbe et al., 2020](#)) is a suite of 16 different Atari-like games with procedural environments (e.g. game maps, terrain, and backgrounds), and was explicitly designed as a setting in which to test procedural generalization. It has also been extensively benchmarked ([Cobbe et al., 2020](#); [Cobbe et al., 2021](#); [Jiang et al., 2021](#); [Raileanu et al., 2021a](#)). For each game, Procgen allows choosing between two difficulty settings (easy or hard) as well as the number of levels seen during training. In our experiments, we use the “hard” difficulty setting and vary the numbers of training levels to test data diversity (see below). For each Procgen game, we train an agent for 30M environment frames. We use the same network architecture and hyper-parameters that [Schrittwieser et al. \(2021\)](#) used for Atari and perform no environment specific tuning. We report mean normalized scores across all games as in [Cobbe et al. \(2020\)](#). Following the recommendation of [Agarwal et al. \(2021\)](#), we report the min-max normalized scores across all games instead of PPO-normalized scores. Thus, the normalized score for each game is computed as $\frac{(score-min)}{(max-min)}$, where



(a) Individual scores across all 16 Procgen games

(b) Mean normalized Score

Figure 12: The impact of planning and self-supervision on procedural generalization in Procgen (hard difficulty, 500 train levels). We plot the zero-shot evaluation performance on unseen levels for each agent throughout training. The Q-Learning agent (QL) is a replica of the MuZero (MZ) with its model-based components removed. MZ+Contr is a MuZero agent augmented with a temporal contrastive self-supervised loss that is action-conditioned (we study other losses in Figure 13). We observe that both planning and self-supervision improve procedural generalization on Procgen. Comparing with existing state-of-the-art methods which were trained for 200M frames on the right (PPO (Schulman et al., 2017), PLR (Jiang et al., 2021), and UCB-DrAC+PLR (Raileanu et al., 2021b; Jiang et al., 2021), data from (Jiang et al., 2021)), we note that MuZero itself exceeds state-of-the-art performance after being trained on only 30M frames. For all plots, dark lines indicate median performance across 3 seeds and the shaded regions denote the min and max performance across seeds. For training curves see Figure 22, for additional metrics see Figure 20.

min and *max* are the minimum and maximum scores possible per game as reported in (Cobbe et al., 2020). We then average the normalized scores across all games to report the mean normalized score.

Meta-World Meta-World (Yu et al., 2020b) is a suite of 50 different tasks on a robotic SAWYER arm, making it more suitable to test task generalization. Meta-World has three benchmarks focused on generalization: ML-1, ML-10, and ML-45. ML-1 consists of 3 goal-conditioned tasks where the objective is to generalize

to unseen goals during test time. ML-10 and ML-45 require generalization to completely new tasks (reward functions) at test time after training on either 10 or 45 tasks, respectively. Meta-World exposes both state and pixel observations, as well as dense and sparse versions of the reward functions. In our experiments, we use the v2 version of Meta-World, dense rewards, and the corner3 camera angle for pixel observations. For Meta-World, we trained Sampled MuZero (Hubert et al., 2021) for 50M environment frames. We measure performance in terms of average episodic success rate across task(s). Note that this measure is different from task rewards, which are dense.

8.3.2 Factors of Generalization

Planning To evaluate the contribution of planning (see paragraph 8.2.2), we compared the performance of a vanilla MuZero Reanalyse agent with a Q-Learning agent. We designed the Q-Learning agent to be as similar to MuZero as possible: for example, it shares the same codebase, network architecture, and replay strategy. The primary difference is that the Q-Learning agent uses a Q-learning loss instead of the MuZero loss in Section 8.2.3 to train the agent, and uses ϵ -greedy instead of MCTS to act in the environment. See Section 13.2 for further details.

Self-supervision We looked at three self-supervised methods as auxiliary losses on top of MuZero: image reconstruction, contrastive learning (Guo et al., 2018), and self-predictive representations (Schwarzer et al., 2021a). We do not leverage any domain-specific data-augmentations for these self-supervised methods.

Reconstruction. Our approach for image reconstruction differs slightly from the typical use of mean reconstruction error over all pixels. In this typical setting, the use of averaging implies that the decoder can focus on the easy to model parts of the observation and still perform well. To encourage the decoder to model all including the hardest to reconstruct pixels, we add an additional loss term which corresponds to the max reconstruction error over all pixels. See Section 13.2.1 for details.

Contrastive. We also experiment with a temporal contrastive objective which treats pairs of observations close in time as positive examples and un-correlated times-tamps as negative examples (Anand et al., 2019; Oord et al., 2018; Guez et al., 2019). Our implementation of the contrastive objective is action-conditioned and uses the MuZero dynamics model to predict future embeddings. Similar to ReLIC (Mitrovic

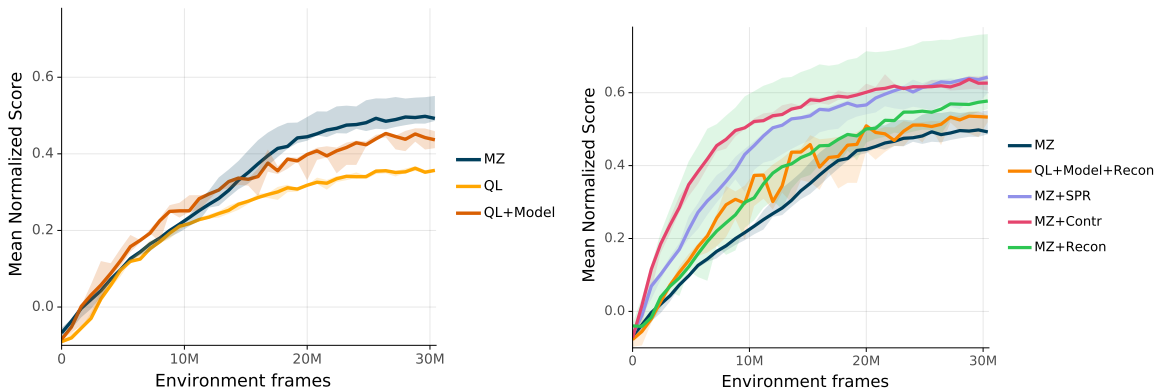


Figure 13: Evaluation performance on Procgen (hard, 500 train levels). On the left, we ablate the effectiveness of planning. The Q-Learning agent (QL) is a replica of MuZero (MZ) without model-based components. We then add a model to this agent (QL+Model) (see Section 13.2) to disentangle the effects of the model-based representation learning from planning in the full MuZero model (MZ). On the right, we ablate the effect of self-supervision with three different losses: Contrastive (Contr), Self-Predictive (SPR), and Image Reconstruction (Recon). We also include a Q-Learning+Model agent with reconstruction (QL+Model+Recon) as a baseline. For all plots, dark lines indicate median performance across 3 seeds (5 seeds for MZ and MZ+Recon) and the shaded regions denote the min and max performance across seeds. For corresponding training curves see Figure 23.

et al., 2021), we also use a KL regularization term in addition to the contrastive loss. The contrastive loss operates entirely in the latent space and does not need decoding to the pixel space at each prediction step, unlike the reconstruction loss. See Section 13.2.2 for details.

Self-predictive. Finally, we experiment with self-predictive representations (SPR) (Schwarzer et al., 2021a) which are trained by predicting future representations of the agent itself from a target network using the MuZero dynamics model. Similar to the contrastive loss, this objective operates in the latent space but does not need negative examples. See Section 13.2.3 for details.

Data diversity To evaluate the contribution of data diversity (see Section 8.2.1), we ran experiments on Procgen in which we varied the number of levels seen during training (either 10, 100, 500, or ∞). In all cases, we evaluated on the infinite test split. Meta-World does not expose a way to modify the amount of data diversity, therefore we did not analyze this factor in our Meta-World experiments.

8.4 RESULTS

We ran all experiments according to the design described in [Section 8.3](#). Unless otherwise specified, all reported results are on the test environments of Procgen and Meta-World and are computed as medians across seeds.

8.4.1 Procedural Generalization

Overall results on Procgen are shown in [Figure 12](#) and [Table 24](#). MuZero achieves a mean-normalized test score of 0.50, which is slightly higher than earlier state-of-the-art methods like UCB-DrAC+PLR which was specifically designed for procedural generalization tasks ([Jiang et al., 2021](#)), while also being much more data efficient (30M frames vs. 200M frames). MuZero also outperforms our Q-Learning baseline which gets a score of 0.36. Performance is further improved by the addition of self-supervision, indicating that both planning and self-supervised model learning are important.

Effect of planning While not reaching the same level of performance of MuZero, the Q-Learning baseline performs quite well and achieves a mean normalized score of 0.36, matching performance of other specialized model-free methods such as PLR ([Jiang et al., 2021](#)). By modifying the Q-Learning baseline to learn a 5-step value equivalent model (similar to MuZero), we find its performance further improves to 0.45, though does not quite catch up to MuZero itself (see [Figure 13](#), left). This suggests that while simply learning a value-equivalent model can bring representational benefits, the best results come from also using this model for action selection and/or policy optimization.

We also tested the effect of planning in the Meta-World ML-1 benchmark from states. [Table 26](#) and [Figure 24](#) show the results. We find that both the Q-Learning and MuZero agents achieve perfect or near-perfect generalization performance on this task, although MuZero is somewhat more stable and data-efficient. The improvement of MuZero in stability or data efficiency again suggests that model-learning and planning can play a role in improving performance.

Effect of self-supervision Next, we looked at how well various self-supervised auxiliary losses improve over MuZero’s value equivalent model on Procgen ([Fig-](#)

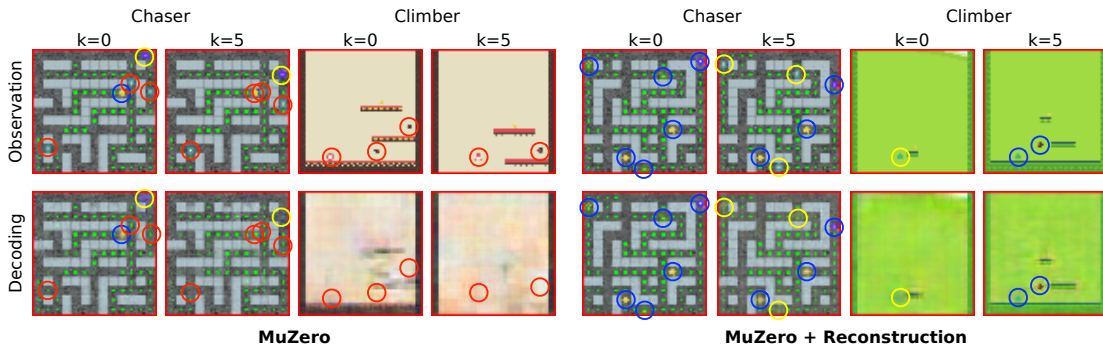


Figure 14: Qualitative comparison of the information encoded in the embeddings learned by MuZero with and without the auxiliary pixel reconstruction loss. For MuZero, embeddings are visualized by learning a standalone pixel decoder trained with MSE. Visualized are environment frames (top row) and decoded frames (bottom row) for two games (Chaser and Climber), for embeddings at the current time step ($k = 0$) and 5 steps into the future ($k = 5$). Colored circles highlight important entities that are or are not well captured (blue=captured, yellow=so-so, red=missing).

Figure 13, right). We find that contrastive learning and self-predictive representations both substantially improve over MuZero’s normalized score of 0.50 to 0.63 (contrastive) or 0.64 (SPR), which are new state-of-the-art scores on Procgen. The reconstruction loss also provides benefits but of a lesser magnitude, improving MuZero’s performance from 0.50 to 0.57. All three self-supervised losses also improve the data efficiency of generalization. Of note is the fact that a MuZero agent with the contrastive loss can match the final performance of the baseline MuZero agent using only a third of the data (10M environment frames).

To further tease apart the difference between MuZero with and without self-supervision, we performed a qualitative comparison between reconstructed observations of MuZero with and without image reconstruction. The vanilla MuZero agent was modified to include image reconstruction, but with a stop gradient on the embedding so that the reconstructions could not influence the learned representations. We trained each agent on two games, Chaser and Climber. As can be seen in Figure 14, it appears that the primary benefit brought by self-supervision is in learning a more accurate world model and capturing more fine-grained details such as the position of the characters and enemies. In Chaser, for example, the model augmented with a reconstruction loss is able to predict the position of

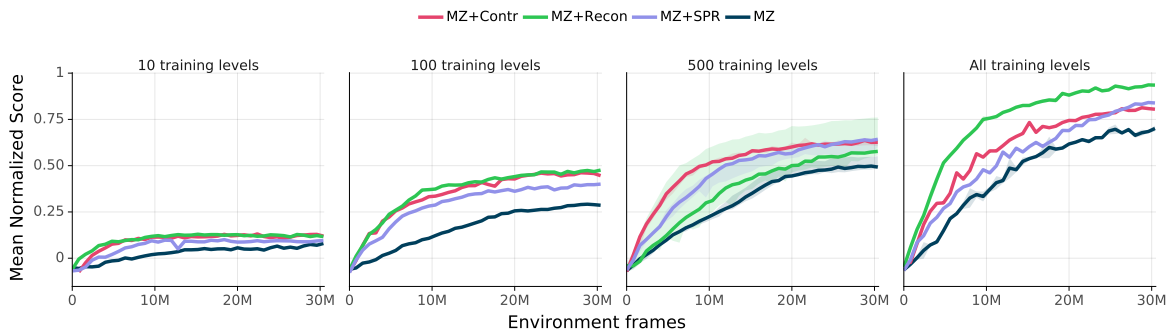


Figure 15: Interaction of self-supervision and data diversity on procedural generalization. Each plot shows generalization performance as a function of environment frames for different numbers of training levels. With only 10 levels, self-supervision does not bring much benefit over vanilla MuZero. Once the training set includes at least 100 levels, there is large improvement with self-supervised learning both in terms of data efficiency and final performance. For all plots, dark lines indicate median performance across seeds and shading indicates min/max seeds.

the character across multiple time steps, while MuZero’s embedding only retains precise information about the character at the current time step. This is somewhat surprising in light of the arguments made for value-equivalent models (Grimm et al., 2020), where the idea is that world models do not need to capture full environment dynamics—only what is needed for the task. Our results suggest that in more complex, procedurally generated environments, it may be challenging to learn even the task dynamics from reward alone without leveraging environment dynamics, too. Further gains in model quality might be gained by also properly handling stochasticity (Ozair et al., 2021) and causality (Rezende et al., 2020).

Effect of data diversity Overall, we find that increased data diversity improves procedural generalization in Procgen, as shown in Figure 15. Specifically, at 10 levels, self-supervision barely improves the test performance on Procgen over MuZero. But as we increase the number of levels to 100, 500, or ∞ , we observe a substantial improvement when using self-supervision. This is an interesting finding which suggests that methods that improve generalization might show promise only when we evaluate them on environments with a lot of inherent diversity, such as procedurally generated environments. Consequently, evaluating methods solely on single task settings (as is common in RL) might lead us to overlook innovations which might have a big impact on generalization.

		MAML	RL ²	QL	MZ	MZ+Recon
ML-10	train	44.4%	86.9%	85.2%	97.6%	97.8%
	zero-shot	-	-	6.8%	26.5%	25.0%
	few-shot	31.6%	35.8%	-	-	-
	finetune	-	-	80.1%	94.1%	97.5%
ML-45	train	40.7%	70%	55.9%	77.2%	74.9%
	zero-shot	-	-	10.8%	17.7%	18.5%
	few-shot	39.9%	33.3%	-	-	-
	finetune	-	-	78.1%	76.7%	81.7%

Table 8: Train and various test success rates (zero-shot, few-shot, finetuning) on the ML-10 and ML-45 task generalization benchmarks of Meta-World. Shown are baseline results on MAML (Finn et al., 2017a) and RL² (Duan et al., 2016) from the Meta-World paper (Yu et al., 2020a) as well as our results with Q-Learning and MuZero. Note that MAML and RL² were trained for around 400M environment steps from states, whereas MuZero was trained from pixels for 50M steps on train tasks and 10M steps on test tasks for fine-tuning.

Not only do we find that MuZero with self-supervision achieves better final generalization performance than the baselines, we also observe that self-supervision improves generalization performance *even when controlling for training performance*. To see this, we visualized generalization performance as a function of training performance (see Figure 21). The figure shows that even when two agents are equally strong (according to the rewards achieved during training), they differ at test time, with those trained with self-supervision generally achieving stronger generalization performance. This suggests that in addition to improving data efficiency, self-supervision leads to more robust representations—a feature that again might be overlooked if not measuring generalization.

8.4.2 Task Generalization

As we have shown, planning, self-supervision, and data diversity all play an important role in procedural generalization. We next investigated whether this trend holds for task generalization, too. To test this, we pre-trained the Q-Learning and MuZero agents with and without self-supervision on the ML-10 and ML-45 training sets of Meta-World and then evaluated zero-shot test performance. In all experiments,

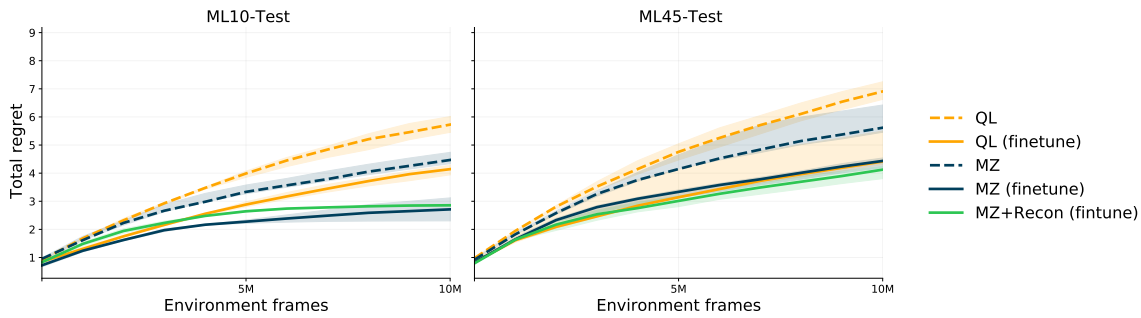


Figure 16: Finetuning performance on ML-10 and ML-45, shown as cumulative regret over the success rate (**lower is better**). Both the pre-trained Q-Learning and MuZero agents have lower regret than corresponding agents trained from scratch. MuZero also achieves lower regret than Q-Learning, indicating a positive benefit of planning (though the difference is small on ML-45). Self-supervision (pixel reconstruction) does not provide any additional benefits. Solid lines indicate median performance across seeds, and shading indicates min/max seeds.

we trained agents from pixels using the dense setting of the reward functions. We report the agent’s success rate computed the same way as in [Yu et al. \(2020b\)](#).

As shown in [Table 8](#), we can observe that MuZero reaches better training performance on ML-10 (97.6%) and ML-45 (77.2%) compared to existing state-of-the-art agents ([Yu et al., 2020b](#)). These existing approaches are meta-learning methods, and are therefore not directly comparable in terms of test performance due to different data budgets allowed at test time; although MuZero does not reach the same level of performance as these agents at test, we find it compelling that it succeeds as often as it does, especially after being trained for only 50M environment steps (compared to 400M for the baselines). MuZero also outperforms Q-Learning in terms of zero-shot test performance, indicating a positive benefit of planning for task generalization. However, reconstruction and other forms of self-supervision do not improve performance and may even decrease it.

We also looked at whether the pre-trained representations or dynamics would assist in more data-efficient task transfer by fine-tuning the agents on the test tasks for 10M environment steps. [Figure 16](#) shows the results, measured as cumulative regret. Using pre-trained representations or dynamics does enable both Q-Learning and MuZero to outperform corresponding agents trained from scratch—evidence for weak positive transfer to unseen tasks for these agents. Additionally, MuZero exhibits better data efficiency than Q-Learning, again showing a benefit for planning. However, self-supervision again does not yield improvements in fine-tuning, and as

before may hurt in some cases. This indicates that while MuZero (with or without self-supervision) excels at representing variation across tasks seen during training, there is room for improvement to better transfer this knowledge to unseen reward functions.

We hypothesize that MuZero only exhibits weak positive transfer to unseen tasks due to a combination of factors. First, the data that its model is trained on is biased towards the training tasks, and thus may not be sufficient for learning a globally-accurate world model that is suitable for planning in different tasks. Incorporating a more exploratory pre-training phase (e.g. [Sekar et al., 2020](#)) might help to alleviate this problem. Second, because the model relies on task-specific gradients, the model may over-represent features that are important to the training tasks (perhaps suggesting that value-equivalent models ([Grimm et al., 2020](#)) may be poorly suited to task generalization). Third, during finetuning, the agent must still discover what the reward function is, even if it already knows the dynamics. It is possible that the exploration required to do this is the primary bottleneck for task transfer, rather than the model representation itself.

8.5 CONCLUSION

In this paper, we systematically investigate how well modern model-based methods perform at hard generalization problems, and whether self-supervision can improve the generalization performance of such methods. We find that in the case of procedural generalization in Procgen, both model-based learning and self-supervision have additive benefits and result in state-of-the-art performance on test levels with remarkable data efficiency. In the case of task generalization in Meta-World, we find that while a model-based agent does exhibit weak positive transfer to unseen tasks, auxiliary self-supervision does not provide any additional benefit, suggesting that having access to a good world model is not always sufficient for good generalization ([Hamrick et al., 2021](#)). Indeed, we suspect that to succeed at task generalization, model-based methods must be supplemented with more sophisticated online exploration strategies, such as those learned via meta-learning ([Duan et al., 2016](#); [Kirsch et al., 2020](#); [Wang et al., 2016a](#)) or by leveraging world models in other ways ([Sekar et al., 2020](#)). Overall, we conclude that self-supervised model-based methods are a promising starting point for developing agents that generalize better, particularly when trained in rich, procedural, and multi-task environments.

CONCLUSION & FUTURE WORK

CONCLUSION

This thesis studies a challenging scientific and engineering problem: How do we build AI agents that learn efficiently and generalize well when learning largely only interacting with an environment, just like humans do?

We explore building agents that learn using self-supervised objectives in an interactive environment using Reinforcement Learning (RL) as the core problem formulation. We develop novel ways of building and studying RL agents equipped with self-supervised representations and world models, and find significant evidence that such agents dramatically improve learning efficiency and generalization of RL agents. The methods and findings in the following articles have had longevity, and inspired several follow-up works that pushed these frontiers even further.

Here's a summary of the contributions of the individual articles:

1. **What do self-supervised representations learn?** (Chapter 4): In [Anand et al., 2019](#) we perform a comprehensive study of what different representation learning methods learn in a visual RL environment like the Atari games. We build a benchmark that allows us to quantitatively compare an array of such methods across a suite of probing tasks to identify their strengths and weaknesses. Findings from this work influenced design decisions in our future work, and more importantly convinced us of the feasibility of leveraging self-supervised learning in RL environments.
2. **Building extremely data-efficient agents with self-supervised learning** (Chapter 6): In [Schwarzer et al., 2021a](#), we introduced Self-Predictive Representations (SPR) where an agent learns a latent world model by predicting future latent vectors of the agent itself. SPR demonstrates dramatic improvement over the state-of-art in terms of data efficiency on the challenging Atari 100k benchmark where agents are allowed only 2 hours of real-time experience. SPR has

since been a key ingredient in several follow-up works that also show further improvements in data efficiency (Nikishin et al., 2022; Schwarzer et al., 2023). In (Schwarzer et al., 2021b) we further show that SPR can also be used as a pre-training objective for RL agents.

3. **Building self-supervised world models that generalize well** (Chapter 8): Anand et al., 2021 studies how learning world models in conjunction with self-supervised learning can improving the generalization abilities of RL agents. Therein, we augment MuZero (Schrittwieser et al., 2020) with auxiliary self-supervised learning objectives, and show that this MuZero++ agent achieves state-of-the-art results on the Procgen and Metaworld benchmarks, a result that still stands. We also perform very careful ablations, and find that planning and model-based representation learning both contribute towards better generalization. In Walker et al., 2023 we further extend this study towards an exploration and transfer learning setting.

FUTURE WORK: REINFORCEMENT LEARNING AS A FINE-TUNING PARADIGM

TL;DR: Reinforcement Learning (RL) should be better seen as a “fine-tuning” paradigm that can add capabilities to general-purpose pretrained models, rather than a paradigm that can bootstrap intelligence from scratch.

Most contemporary reinforcement learning works involve training agents “tabula-rasa”, without relying on any sort of knowledge about the world. So when solving a task(s), the agent not only has to optimize the reward function at hand, but in the process also discover how to see, how physics works, what consequences its actions have, how language works, and so forth. This tends to work okay in simulations where we can collect infinite data, and don’t need to discover a lot of common-sense knowledge because the simulators themselves are quite niche. But it breaks down when solving tasks in the real world which has fractals of complexity and practical limits on how much data we can collect.

To train agents on real-world data, why don’t we simply endow them with knowledge about the real world, and let the RL algorithms focus on what they are good at: black-box optimization of a reward function.

Thanks to large-scale self-supervised models trained on the internet — ones that soak up enormous amounts of physical and cultural knowledge, we now have a way of doing so. After training general-purpose pre-trained models, reinforcement learning (and/or search) can be used to fine-tune them to amplify their capabilities — making them experts at a particular tasks (goal-directedness), providing them agency, learning from feedback, aligning them with human values, and many more. RL fine-tuning provides a way out of the “simulation trap”, and lets us train agents on real world data and environments directly.

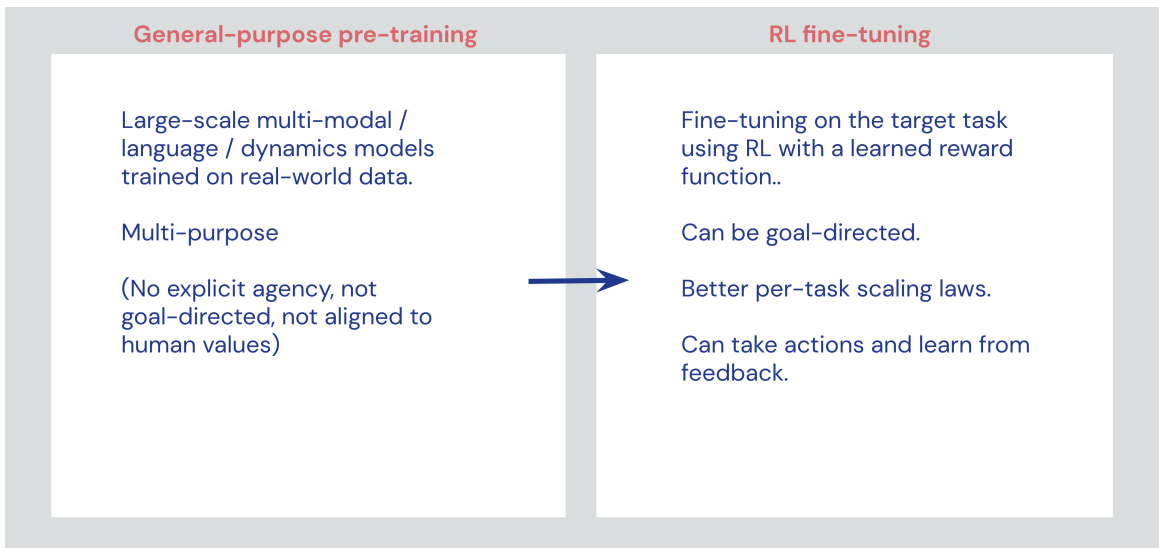


Figure 17: Pre-training and RL fine-tuning, a two stage process. While pre-trained models are super general, RL fine-tuning can make them highly capable at individual tasks.

10.1 WHY RL FINE-TUNING OVER OTHER ALTERNATIVES?

Now, it's natural to ask why don't we just use prompts to discover capabilities (Just ask for Generalization (Jang, 2021)), or use supervised learning to fine-tune. RL fine-tuning stands out in a few ways:

- It can **directly optimize** a non-differentiable objective, instead of merely trying to mimic existing data, and thus does not have performance ceilings. "Other learning paradigms are about minimization; reinforcement learning is about maximization" (Lu et al., 2021).
- It should have (and has shown to have) **better scaling laws** compared to prompts or supervised learning.
- Meta-capabilities (such as having **agency**) just fit better in the RL fine-tuning paradigm. A model having agency will be better than a model without it (Why Tool AIs Want to Be Agent AIs, Branwen, 2021b).

APPENDIX

APPENDIX TO THE FIRST ARTICLE

11.1 ARCHITECTURE DETAILS

All architectures below use the same encoder architecture as a base, which is the one used in [Mnih et al., 2013](#) adapted to work for the full 160x210 frame size as shown in figure 18.

- **Linear Probe:**

The linear probe is a linear layer of width 256 with a softmax activation and trained with a cross-entropy loss.

- **Majority Classifier (maj-clsf):**

The majority classifier is parameterless and just uses the mode of the distribution of classes from the training set for each state variable and guesses that mode for every example on the test set at test time.

- **Random-CNN:**

The Random-CNN is the base encoder with randomly initialized weights and no training

- **VAE and Pixel-Pred:**

The VAE and Pixel Prediction model use the base encoder plus each have an extra 256 wide fully connected layer to parameterize the log variance for the VAE and to more closely resemble the *No Action Feed Forward* model from [Oh et al., 2015](#). In addition both models have a deconvolutional network as a decoder, which is the exact transpose of the base encoder in figure 18.

- **CPC:**

CPC uses the same architecture as described in [Oord et al., 2018](#) with our base encoder from figure 18 being used as the image encoder g_{enc} .

- **ST-DIM (and its ablations):**
ST-DIM and the two ablations, JSD-ST-DIM and Global-T-DIM, all use the same architecture which is the base encoder plus a 1x256x256 bilinear layer.
- **Supervised:**
The supervised model is our base encoder plus our linear probe trained end-to-end with the ground truth labels.
- **PPO Features (section 11.5):**
The PPO model is our base encoder plus two linear layers for the policy and the value function, respectively.

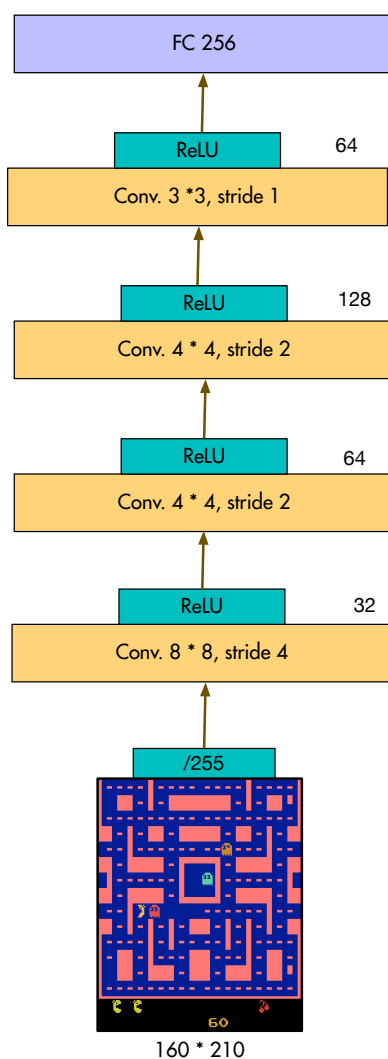


Figure 18: The base encoder architecture used for all models in this work

11.2 PREPROCESSING AND HYPERPARAMETERS

We preprocess frames primarily in the same way as described in [Mnih et al., 2013](#), with the key difference being we use the full 210x160 images for all our experiments instead of downsampling to 84x84. Table 9 lists the hyper-parameters we use across all games. For all our experiments, we use a learning rate scheduler based on plateaus in the validation loss (for both contrastive training and probing).

Table 9: Preprocessing steps and hyperparameters

Parameter	Value
Image Width	160
Image Height	210
Grayscale	Yes
Action Repetitions	4
Max-pool over last N action repeat frames	2
Frame Stacking	None
End of episode when life lost	Yes
No-Op action reset	Yes
Batch size	64
Sequence Length (CPC)	100
Learning Rate (Training)	3e-4
Learning Rate (Probing)	3e-4
Entropy Threshold	0.6
Encoder training steps	80000
Probe training steps	35000
Probe test steps	10000

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

11.3 RESULTS WITH PROBES TRAINED ON DATA COLLECTED BY A PRETRAINED RL AGENT

In addition to evaluating on data collected by a random agent, we also evaluate different representation learning methods on data collected by a pretrained PPO (Schulman et al., 2017) agent. Specifically, we use a PPO agent trained for 50M steps on each game. We choose actions stochastically by sampling from the PPO agent’s action distribution at every time step, and inject additional stochasticity by using an ϵ -greedy mechanism with $\epsilon = 0.2$. Table 10 shows the game-by-game breakdown of mean F1 probe scores obtained by each method in this evaluation setting. Table 11 additionally shows the category-wise breakdown of results for each method. We observe a similar trend in performance as observed earlier with a random agent.

11.4 MORE DETAILED ABLATION RESULTS

We expand on the results reported on different ablations (JSD-ST-DIM and Global-T-DIM) of STDIM in the main text, and provide a game by game breakdown of results in Table 12, and a category-wise breakdown in Table 13. We also include an additional Static-DIM ablation which gets rid of any temporal context in the contrastive task by sampling negatives from a different game.

11.5 PROBING PRETRAINED RL AGENTS

We make a first attempt at examining the features that RL agents learn. Specifically, we train linear probes on the representations from PPO agents that were trained for 50 million frames. The architecture of the PPO agent is described in section 11.1. As we see from table 14, the features perform poorly in the probing tasks compared to the baselines. Kansky et al. (2017) and Zhang et al. (2018b) have also argued that model-free agents have trouble encoding high level state information. However, we note that these are preliminary results and require thorough investigation over different policies and models.

11.5.1 *Accuracy Metric*

In tables 15 and 16, we report the game by game and categorical probe results for each method, but we use a standard percent accuracy metric instead of the F1 score that was used in tables 2 and 3.

Table 10: Probe F1 scores for all games for data collected by a pretrained PPO (50M steps) agent

game	mean agent rewards	maj-clf	random-cnn	vae	pixel-pred	cpc	st-dim	supervised
asteroids	489862.00	0.23	0.31	0.35	0.31	0.38	0.40	0.56
berzerk	1913.00	0.13	0.33	0.35	0.39	0.38	0.43	0.61
bowling	29.80	0.23	0.61	0.51	0.81	0.90	0.98	0.98
boxing	93.30	0.05	0.30	0.32	0.57	0.32	0.66	0.87
breakout	580.40	0.09	0.34	0.59	0.47	0.55	0.66	0.87
demonattack	428165.00	0.03	0.19	0.18	0.26	0.43	0.58	0.76
freeway	33.50	0.01	0.36	0.02	0.60	0.38	0.60	0.76
frostbite	3561.00	0.13	0.57	0.46	0.70	0.74	0.69	0.85
hero	44999.00	0.12	0.54	0.60	0.68	0.86	0.77	0.96
mzrevenge	0.00	0.08	0.68	0.58	0.72	0.77	0.76	0.88
mspacman	4588.00	0.07	0.34	0.36	0.52	0.45	0.49	0.71
pitfall	0.00	0.16	0.39	0.37	0.53	0.69	0.74	0.92
pong	21.00	0.02	0.10	0.24	0.67	0.63	0.79	0.87
privateeye	-10.00	0.24	0.71	0.69	0.87	0.83	0.91	0.99
qbert	30590.00	0.06	0.36	0.38	0.39	0.51	0.48	0.65
riverraid	20632.00	0.04	0.25	0.21	0.34	0.31	0.22	0.59
seaquest	1620.00	0.29	0.64	0.58	0.75	0.69	0.75	0.90
spaceinvaders	2892.50	0.02	0.28	0.30	0.41	0.32	0.41	0.65
tennis	-4.30	0.15	0.25	0.13	0.65	0.63	0.65	0.61
venture	0.00	0.05	0.32	0.36	0.37	0.50	0.59	0.69
videopinball	356362.00	0.13	0.36	0.42	0.56	0.57	0.54	0.79
yarsrevenge	5520.00	0.03	0.14	0.26	0.23	0.38	0.43	0.74
mean	-	0.11	0.38	0.38	0.54	0.56	0.62	0.78

Table 11: Probe F1 scores for different methods averaged across all games for each category (data collected by a pretrained PPO (50M steps) agent)

CATEGORY	MAJ-CLF	RANDOM-CNN	VAE	PIXEL-PRED	CPC	ST-DIM	SUPERVISED
SMALL LOC.	0.10	0.13	0.14	0.27	0.31	0.41	0.65
AGENT LOC.	0.11	0.34	0.34	0.48	0.45	0.54	0.83
OTHER LOC.	0.14	0.47	0.38	0.56	0.58	0.61	0.74
SCORE/CLOCK/LIVES/DISPLAY	0.05	0.44	0.50	0.71	0.74	0.80	0.90
MISC.	0.19	0.53	0.57	0.62	0.65	0.67	0.83

Table 12: Probe F1 scores for different ablations of ST-DIM for all games averaged across each category (data collected by random agents)

GAME	STATIC-DIM	JSD-ST-DIM	GLOBAL-T-DIM	ST-DIM
ASTEROIDS	0.37	0.44	0.38	0.49
BERZERK	0.41	0.49	0.49	0.53
BOWLING	0.34	0.91	0.77	0.96
BOXING	0.09	0.61	0.32	0.58
BREAKOUT	0.19	0.85	0.71	0.88
DEMONATTACK	0.30	0.44	0.43	0.69
FREEWAY	0.02	0.70	0.76	0.81
FROSTBITE	0.27	0.52	0.68	0.75
HERO	0.59	0.85	0.87	0.93
MONTEZUMAREVENGE	0.17	0.55	0.67	0.78
MSPACMAN	0.17	0.70	0.53	0.72
PITFALL	0.22	0.47	0.44	0.60
PONG	0.13	0.80	0.65	0.81
PRIVATEEYE	0.25	0.79	0.81	0.91
QBERT	0.41	0.59	0.57	0.73
RIVERRAID	0.16	0.28	0.33	0.36
SEAQUEST	0.41	0.55	0.59	0.67
SPACEINVADERS	0.40	0.44	0.44	0.57
TENNIS	0.17	0.57	0.52	0.60
VENTURE	0.25	0.40	0.47	0.58
VIDEOPINBALL	0.21	0.54	0.53	0.61
YARSREVENGE	0.12	0.32	0.18	0.42
MEAN	0.26	0.58	0.55	0.68

Table 13: Different ablations of ST-DIM. F1 scores for for each category averaged across all games (data collected by random agents)

	STATIC-DIM	JSD-ST-DIM	GLOBAL-T-DIM	ST-DIM
SMALL LOC.	0.18	0.44	0.37	0.51
AGENT LOC.	0.19	0.47	0.43	0.58
OTHER LOC.	0.27	0.64	0.53	0.69
SCORE/CLOCK/LIVES/DISPLAY	0.33	0.69	0.76	0.87
MISC.	0.41	0.64	0.66	0.75

Table 14: Probe results on features from a PPO agent trained on 50 million timesteps compared with a majority classifier and random-cnn baseline. The probes for all three methods are trained with data from the PPO agent that was trained for 50M frames

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

Table 15: Probe Accuracy scores averaged across categories for each game (data collected by random agents)

GAME	MAJ-CLF	RANDOM-CNN	VAE	PIXEL-PRED	CPC	ST-DIM	SUPERVISED
ASTEROIDS	0.37	0.42	0.41	0.43	0.48	0.52	0.53
BERZERK	0.30	0.48	0.46	0.56	0.57	0.54	0.69
BOWLING	0.43	0.54	0.56	0.83	0.90	0.96	0.95
BOXING	0.05	0.22	0.23	0.45	0.32	0.59	0.83
BREAKOUT	0.28	0.55	0.61	0.71	0.75	0.89	0.94
DEMONATTACK	0.26	0.30	0.31	0.35	0.58	0.70	0.83
FREEWAY	0.06	0.53	0.07	0.85	0.49	0.82	0.99
FROSTBITE	0.19	0.59	0.54	0.72	0.76	0.75	0.85
HERO	0.34	0.78	0.72	0.75	0.90	0.93	0.98
MONTEZUMAREVENGE	0.16	0.70	0.41	0.74	0.76	0.78	0.87
MSPACMAN	0.22	0.54	0.60	0.75	0.67	0.73	0.87
PITFALL	0.20	0.42	0.35	0.47	0.49	0.61	0.83
PONG	0.20	0.26	0.19	0.72	0.73	0.82	0.88
PRIVATEEYE	0.35	0.72	0.72	0.83	0.81	0.91	0.97
QBERT	0.42	0.52	0.53	0.54	0.66	0.74	0.76
RIVERRAID	0.13	0.40	0.31	0.43	0.41	0.37	0.58
SEAQUEST	0.43	0.63	0.61	0.65	0.69	0.69	0.85
SPACEINVADERS	0.24	0.46	0.57	0.61	0.57	0.59	0.76
TENNIS	0.22	0.49	0.37	0.59	0.61	0.61	0.81
VENTURE	0.19	0.40	0.43	0.48	0.52	0.59	0.68
VIDEOPINBALL	0.16	0.39	0.47	0.58	0.59	0.61	0.82
YARSREVENGE	0.05	0.25	0.11	0.20	0.41	0.43	0.74
MEAN	0.24	0.48	0.44	0.60	0.62	0.69	0.82

Table 16: Probe Accuracy scores for different methods averaged across all games for each category (data collected by random agents)

CATEGORY	RANDOM						SUPERVISED
	MAJ-CLF	CNN	VAE	PIXEL-PRED	CPC	ST-DIM	
SMALL LOC.	0.23	0.29	0.26	0.36	0.46	0.53	0.67
AGENT LOC.	0.21	0.37	0.37	0.51	0.46	0.59	0.81
OTHER LOC.	0.22	0.54	0.42	0.63	0.67	0.70	0.80
SCORE/CLOCK/LIVES/DISPLAY	0.24	0.61	0.56	0.77	0.84	0.87	0.91
MISC.	0.38	0.61	0.65	0.71	0.72	0.75	0.83

APPENDIX TO THE SECOND ARTICLE

12.1 HYPERPARAMETERS

We provide a full set of hyperparameters used in both the augmentation and no-augmentation cases in Table 17, including new hyperparameters for SPR.

12.2 FULL RESULTS

We provide full results across all 26 games for the methods considered, including SPR with and without augmentation, in Table 18. Methods are ordered in rough order of their date of release or publication.

12.3 CONTROLLED BASELINES

To ensure that the minor hyper-parameter changes we make to the DER baseline are not solely responsible for our improved performance, we perform controlled experiments using the same hyper-parameters and same random seeds for baselines. We find that our controlled Rainbow implementation without augmentation is slightly stronger than Data-Efficient Rainbow but comparable to Overtrained Rainbow (Kielak, 2020), while with augmentation enabled our results are somewhat stronger than DrQ.¹ None of these methods, however, are close to the performance of SPR.

¹ This is perhaps not surprising, given that the model used by DrQ omits many of the components of Rainbow.

Table 19: Scores on the 26 Atari games under consideration for our controlled Rainbow implementation with and without augmentation, compared to previous methods. The high mean DQN-normalized score of our DQN without augmentation is due to an atypically high score on Private Eye, a hard exploration game on which the original DQN achieves a low score.

Variant	Human-Normalized Score		DQN@50M-Normalized Score	
	mean	median	mean	median
Rainbow (controlled, no aug)	0.240	0.204	0.374	0.149
OTRainbow	0.264	0.204	0.197	0.103
DER	0.285	0.161	0.239	0.142
Rainbow (controlled, w/ aug)	0.480	0.346	0.284	0.278
DrQ	0.357	0.268	0.171	0.131

12.4 COMPARISON WITH A CONTRASTIVE LOSS

To compare SPR with alternative methods drawn from contrastive learning, we examine several variants of a contrastive losses based on InfoNCE (Oord et al., 2018):

- A contrastive loss based solely on different views of the same state, similar to CURL (Srinivas et al., 2020).
- A temporal contrastive loss with both augmentation and where targets are drawn one step in the future, equivalent to single-step CPC (Oord et al., 2018).
- A temporal contrastive loss with an explicit dynamics model, similar to CPC | Action (Guo et al., 2018). Predictions are made up to five steps in the future, and encodings of every state except s_{t+k} are used as negative samples for s_{t+k} .
- A soft contrastive approach inspired by Wang et al., 2020, who propose to decouple the repulsive and attractive effects of contrastive learning into two separate losses, one of which is similar to the SPR objective and encourages representations to be invariant to augmentation or noise, and one of which encourages representations to be uniformly distributed on the unit hypersphere. We optimize this uniformity objective jointly with the SPR loss, which takes the role of the “invariance” objective proposed by (Wang et al., 2020). We use

$t = 2$ in the uniformity loss, and give it a weight equal to that given to the SPR loss, based on hyperparameters used by Wang et al., 2020.

To create as fair a comparison as possible, we use the same augmentation (random shifts and intensity) and the same Rainbow hyperparameters as in SPR with augmentation. As in SPR, we calculate contrastive losses using the output of the first layer of the Q-head MLP, with a bilinear classifier (as in Oord et al., 2018). Following Chen et al., 2020a, we use annealed cosine similarities with a temperature of 0.1 in the contrastive loss. We present results in Table 20.

Although all of these variants outperform the previous contrastive result on this task, CURL, none of them substantially improve performance over the controlled Rainbow they use as a baseline. We consider these results broadly consistent with those of CURL, which observes a relatively small performance boost over their baseline, Data-Efficient Rainbow (Hasselt et al., 2019).

Table 20: Scores on the 26 Atari games under consideration for various contrastive alternatives to SPR implemented in our codebase. All variants listed here use data augmentation.

Variant	Human-Normalized Score		DQN@50M-Normalized Score	
	mean	median	mean	median
SPR	0.704	0.415	0.510	0.361
Rainbow (controlled)	0.480	0.346	0.284	0.278
Non-temporal contrastive	0.379	0.200	0.268	0.179
1-step contrastive	0.473	0.231	0.280	0.213
5-step contrastive	0.506	0.172	0.239	0.142
Uniformity loss	0.422	0.176	0.271	0.144

12.5 THE ROLE OF THE TARGET ENCODER IN SPR

We consider several variants of SPR with the target network modified, and present aggregate metrics for these experiments in Table 21. We first evaluate a variant of SPR in which target representations are drawn from the online encoder and gradients allowed to propagate into the online encoder through them, effectively allowing the encoder to learn to make its representations more predictable. We

find that this leads to drastic reductions in performance both with and without augmentation, which we attribute to representational collapse.

Table 21: Scores on the 26 Atari games under consideration for variants of SPR with different target encoder schemes, without augmentation.

Variant	Human-Normalized Score		DQN@50M-Normalized Score	
	mean	median	mean	median
<hr/>				
Without Augmentation				
SPR	0.463	0.307	0.336	0.225
No Stopgradient	0.375	0.208	0.301	0.233
<hr/>				
With Augmentation				
<hr/>				
SPR	0.704	0.415	0.510	0.361
No Stopgradient	0.515	0.278	0.344	0.231

To illustrate the influence of the EMA constant τ , we evaluate τ at 9 values logarithmically interpolated² between 0.999 and 0 on a subset of 10 Atari games.³ We use 10 seeds per game, and evaluate SPR both with and without augmentation; parameters other than τ are identical to those listed in Table 17. To equalize the importance of games in this analysis, we normalize by the average score across all tested values of τ for each game to calculate a *self-normalized* score, as $\text{score}_{\text{sns}} \triangleq \frac{\text{agent score} - \text{random score}}{\text{average score} - \text{random score}}$.

We test SPR both with and without augmentation, and calculate the self-normalized score separately between these cases. Results are shown in Figure 19. With augmentation, we observe a clear peak in performance at $\tau = 0$, equivalent to a target encoder with no EMA-based smoothing. Without augmentation, however, the story is less clear, and the method appears less sensitive to τ (note y-axis scales). We use $\tau = 0.99$ in this case, based on its reasonable performance and consistency with prior work (e.g., Grill et al., 2020b). Overall, however, we note that SPR does not appear overly sensitive to τ , unlike purely unsupervised methods such as BYOL; in no case does SPR fail to train.

We hypothesize that the difference between the augmentation and no-augmentation cases is partially due to augmentation rendering the stabilizing effect of using an

² $\tau \in \{0.999, 0.9976, 0.9944, 0.9867, 0.9684, 0.925, 0.8222, 0.5783, 0\}$.

³ Pong, Breakout, Up N Down, Kangaroo, Bank Heist, Assault, Boxing, BattleZone, Frostbite and Crazy Climber

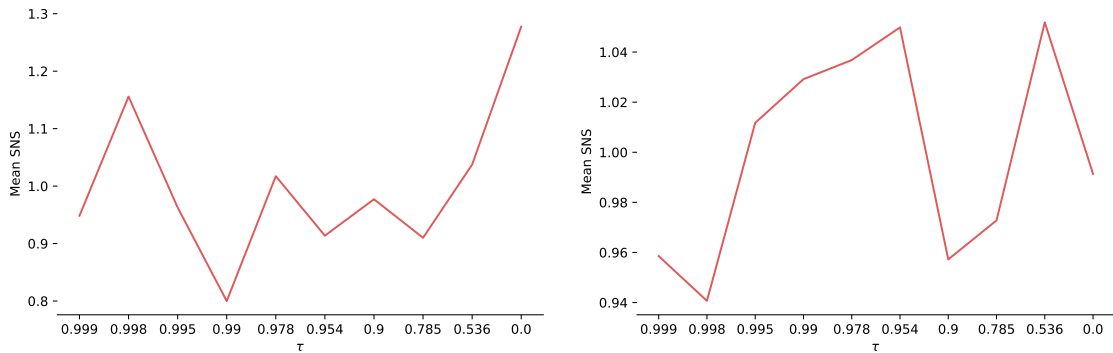


Figure 19: Performance on a subset of 10 Atari games for different values of the EMA parameter τ with augmentation (left) and without (right). Scores are averaged across 10 seeds per game for each value of τ . Self-normalized score is calculated separately for the augmentation and no-augmentation cases.

EMA target network (e.g., as observed by Grill et al., 2020b; Tarvainen et al., 2017) redundant. Prior work has already noted that using an EMA target network can slow down learning early in training (Tarvainen et al., 2017); in our context, where a limited number of environment samples are taken in parallel with optimization, this may “waste” environment samples by collecting them with an inferior policy. To resolve this, Tarvainen et al., 2017 proposed to increase τ over the course of training, slowing down changes to the target network later in training. It is possible that doing so here could allow SPR to achieve the best of both worlds, but it would require tuning an additional hyperparameter, the schedule by which τ is increased, and we thus leave this topic for future work.

12.6 WALL CLOCK TIMES

We report wall-clock runtimes for a selection of methods in Table 22. SPR with augmentation for a 100K steps on Atari takes around 4 and a half to finish a complete training and evaluation run on a single game. We find that using data augmentation adds an overhead, and SPR without augmentation can run in just 3 hours.

SPR’s wall-clock run-time compares very favorably to previous works such as SimPLe (Kaiser et al., 2019), which requires roughly three weeks to train on a GPU comparable to those used for SPR.

Table 17: Hyperparameters for SPR on Atari, with and without augmentation.

Parameter	Setting (for both variations)	
Gray-scaling	True	
Observation down-sampling	84x84	
Frames stacked	4	
Action repetitions	4	
Reward clipping	[-1, 1]	
Terminal on loss of life	True	
Max frames per episode	108K	
Update	Distributional Q	
Dueling	True	
Support of Q-distribution	51	
Discount factor	0.99	
Minibatch size	32	
Optimizer	Adam	
Optimizer: learning rate	0.0001	
Optimizer: β_1	0.9	
Optimizer: β_2	0.999	
Optimizer: ϵ	0.00015	
Max gradient norm	10	
Priority exponent	0.5	
Priority correction	0.4 \rightarrow 1	
Exploration	Noisy nets	
Noisy nets parameter	0.5	
Training steps	100K	
Evaluation trajectories	100	
Min replay size for sampling	2000	
Replay period every	1 step	
Updates per step	2	
Multi-step return length	10	
Q network: channels	32, 64, 64	
Q network: filter size	$8 \times 8, 4 \times 4, 3 \times 3$	
Q network: stride	4, 2, 1	
Q network: hidden units	256	
Non-linearity	ReLU	
Target network: update period	1	
λ (SPR loss coefficient)	2	
K (Prediction Depth)	5	
Parameter	With Augmentation	Without Augmentation
Data Augmentation	Random shifts (± 4 pixels) & Intensity(scale=0.05)	None
Dropout	0	0.5
τ (EMA coefficient)	0	0.99

Table 18: Mean episodic returns on the 26 Atari games considered by [Kaiser et al., 2019](#) after 100k environment steps. The results are recorded at the end of training and averaged over 10 random seeds. SPR outperforms prior methods on all aggregate metrics, and exceeds expert human performance on 7 out of 26 games while using a similar amount of experience.

Game	Random	Human	SimPLe	DER	OTRainbow	CURL	DrQ	SPR (no Aug)	SPR
Alien	227.8	7127.7	616.9	739.9	824.7	558.2	771.2	847.2	801.5
Amidar	5.8	1719.5	88.0	188.6	82.8	142.1	102.8	142.7	176.3
Assault	222.4	742.0	527.2	431.2	351.9	600.6	452.4	665.0	571.0
Asterix	210.0	8503.3	1128.3	470.8	628.5	734.5	603.5	820.2	977.8
Bank Heist	14.2	753.1	34.2	51.0	182.1	131.6	168.9	425.6	380.9
BattleZone	2360.0	37187.5	5184.4	10124.6	4060.6	14870.0	12954.0	10738.0	16651.0
Boxing	0.1	12.1	9.1	0.2	2.5	1.2	6.0	12.7	35.8
Breakout	1.7	30.5	16.4	1.9	9.8	4.9	16.1	12.9	17.1
ChopperCommand	811.0	7387.8	1246.9	861.8	1033.3	1058.5	780.3	667.3	974.8
Crazy Climber	10780.5	35829.4	62583.6	16185.3	21327.8	12146.5	20516.5	43391.0	42923.6
Demon Attack	152.1	1971.0	208.1	508.0	711.8	817.6	1113.4	370.1	545.2
Freeway	0.0	29.6	20.3	27.9	25.0	26.7	9.8	16.1	24.4
Frostbite	65.2	4334.7	254.7	866.8	231.6	1181.3	331.1	1657.4	1821.5
Gopher	257.6	2412.5	771.0	349.5	778.0	669.3	636.3	774.5	715.2
Hero	1027.0	30826.4	2656.6	6857.0	6458.8	6279.3	3736.3	5707.4	7019.2
Jamesbond	29.0	302.8	125.3	301.6	112.3	471.0	236.0	367.2	365.4
Kangaroo	52.0	3035.0	323.1	779.3	605.4	872.5	940.6	1359.5	3276.4
Krull	1598.0	2665.5	4539.9	2851.5	3277.9	4229.6	4018.1	3123.1	3688.9
Kung Fu Master	258.5	22736.3	17257.2	14346.1	5722.2	14307.8	9111.0	15469.7	13192.7
Ms Pacman	307.3	6951.6	1480.0	1204.1	941.9	1465.5	960.5	1247.7	1313.2
Pong	-20.7	14.6	12.8	-19.3	1.3	-16.5	-8.5	-16.0	-5.9
Private Eye	24.9	69571.3	58.3	97.8	100.0	218.4	-13.6	52.6	124.0
Qbert	163.9	13455.0	1288.8	1152.9	509.3	1042.4	854.4	606.6	669.1
Road Runner	11.5	7845.0	5640.6	9600.0	2696.7	5661.0	8895.1	10511.0	14220.5
Seaquest	68.4	42054.7	683.3	354.1	286.9	384.5	301.2	580.8	583.1
Up N Down	533.4	11693.2	3350.3	2877.4	2847.6	2955.2	3180.8	6604.6	28138.5
Mean Human-Norm'd	0.000	1.000	0.443	0.285	0.264	0.381	0.357	0.463	0.704
Median Human-Norm'd	0.000	1.000	0.144	0.161	0.204	0.175	0.268	0.307	0.415
Mean DQN@50M-Norm'd	0.000	23.382	0.232	0.239	0.197	0.325	0.171	0.336	0.510
Median DQN@50M-Norm'd	0.000	0.994	0.118	0.142	0.103	0.142	0.131	0.225	0.361
# Superhuman	0	N/A	2	2	1	2	2	5	7

Table 22: Wall-clock runtimes for various algorithms for a complete training and evaluation run on a single Atari game using a P100 GPU. Rainbow (controlled) is roughly comparable to DrQ, although its runtime will differ due different DQN hyperparameters. Runtime for SimPLe is taken from its v3 version on Arxiv, although the latest version doesn't mention runtime. All runtimes are approximate, as exact running times vary from game to game.

Model	Runtime in hours (100k env steps)
SPR	4.6
Rainbow (controlled)	2.1
SPR (No aug)	3.0
Rainbow (controlled, no aug)	1.4
SimPLe	500

APPENDIX TO THE THIRD ARTICLE

13.1 AGENT DETAILS

13.1.1 *MuZero*

Network Architecture: We follow the same network architectures follow the ones used in MuZero ReAnalyse (Schrittwieser et al., 2021). For pixel based inputs, the images are first sent through a convolutional stack that downsamples the 96×96 image down to an 8×8 tensor (for Procgen) or a 16×16 tensor (for Meta-World). This tensor then serves as input to the encoder. Both the encoder and the dynamics model were implemented by a ResNet with 10 blocks, each block containing 2 layers. For pixel-based inputs (Procgen and ML-45) each layer of the residual stack was convolutional with a kernel size of 3×3 and 256 planes. For state based inputs (ML-1) each layer was fully-connected with a hidden size of 512.

Hyperparameters We list major hyper-parameters used in this work in Table 23.

Table 23: Hyper-parameters

HYPER-PARAMETER	VALUE (PROCGEN)	VALUE (META-WORLD)
TRAINING		
Model Unroll Length	5	5
TD-Steps	5	5
ReAnalyze Fraction	0.945	0.95
Replay Size (in sequences)	50000	2000
MCTS		
Number of Simulations	50	50
UCB-constant	1.25	1.25
Number of Samples	n/a	20
SELF-SUPERVISION		
Reconstruction Loss Weight	1.0	1.0
Contrastive Loss Weight	1.0	0.1
SPR Loss Weight	10.0	1.0
OPTIMIZATION		
Optimizer	Adam	Adam
Initial Learning Rate	10^{-4}	10^{-4}
Batch Size	1024	1024

Additional Implementation Details: For Meta-World experiments, we provide the agent with past reward history as well. We found this to be particularly helpful when training on Meta-World since implicit task inference becomes easier. In both Procggen and Meta-World, the agent is given a history of the 15 last observations. The images are concatenated by channel and then input as one tensor to the encoder. For each game, we trained our model using 2 TPUv3-8 machines. A separate actor gathered environment trajectories with 1 TPUv3-8 machine.

13.2 CONTROLLED MODEL-FREE BASELINE

Our controlled Q-Learning baseline begins with the same setup as the MuZero agent (Section 13.1.1) and modifies it in a few key ways to make it model-free rather than model-based.

The Q-Learning baseline uses n -step targets for action value function. Given a trajectory $\{s_t, a_t, r_t\}_{t=0}^T$, the target action value is computed as follow

$$Q_{target}(s_t, a_t) = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n \max_{a \in \mathcal{A}} Q_{\zeta}(s_{t+n}, a) \quad (11)$$

Where Q_{ζ} is the target network whose parameter ζ is updated every 100 training steps.

In order to make the model architecture most similar to what is used in the MuZero agent, we decompose the action value function into two parts: a reward prediction \hat{r} and a value prediction V , and model these two parts separately. The total loss function is, therefore, $\mathcal{L}_{total} = \mathcal{L}_{reward} + \mathcal{L}_{value}$. The reward loss is exactly the same as that of MuZero. For the value loss, we can decompose [Equation 11](#) in the same way:

$$\begin{aligned} Q_{target}(s_t, a_t) &= \hat{r}_t + \gamma V_{target}(s) \\ &= \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n \max_{a \in \mathcal{A}} \left(\hat{r}_{t+n} + \gamma V_{\zeta}(s') \right) \\ \implies V_{target}(s) &= \sum_{i=1}^{n-1} \gamma^{i-1} r_{t+i} + \gamma^{n-1} \max_{a \in \mathcal{A}} \left(\hat{r}_{t+n} + \gamma V_{\zeta}(s') \right) \end{aligned} \quad (12)$$

Since the reward prediction should be taken care of by \mathcal{L}_{reward} and it usually converges fast, we assume $\hat{r}_t = r_t$ and the target is simplified to [Equation 12](#). We can then use this value target to compute the value loss $\mathcal{L}_{value} = \text{CE}(V_{target}(s), V(s))$.

In Meta-World, since it has a continuous action space, maximizing over the entire action space is infeasible. We follow the Sampled Muzero approach ([Hubert et al., 2021](#)) and maximize only over the sampled actions.

The Q-Learning baseline uses 5-step targets for computing action values. This is the same as in MuZero, but unlike MuZero, the Q-Learning baseline only trains the dynamics for a single step. However, we also provide results for a 5-step dynamics function which we call QL+Model and QL+Model+Recon which further adds an auxiliary for reconstruction.

13.2.1 MuZero + Reconstruction

The decoder architecture mirrors the ResNet encoder with 10 blocks, but with each convolutional layer replaced by a de-convolutional layer (Noh et al., 2015). In addition to the regular mean reconstruction loss, we add an additional max-loss term which computes the maximum reconstruction loss across all pixels. This incentivizes the reconstruction objective to not neglect the hardest to reconstruct pixels, and in turn makes the reconstructions sharper. The reconstruction loss can be used as follows:

$$\mathcal{L}(X, Y) = \left(\frac{1}{|X|} \sum_{i,j,f} \ell(X_{ijf}, Y_{ijf}) \right) + \max_{i,j,f} \ell(X_{ijf}, Y_{ijf}),$$

where $\ell(x, y) = (x - y)^2$ is the element-wise squared distance between each feature (i.e. RGB) value in the image array, where i and j are the pixel indices, where f is the feature index, and where $|X|$ is the size of the array.

13.2.2 MuZero + Contrastive

Our implementation of the contrastive loss does not rely on any image augmentations but focuses on the task of predicting future embeddings of the image encoder (Banino et al., 2021; Oord et al., 2018). However, in this case, the dynamics function produces a predicted vector x_n at each step n in the unroll, making the contrastive task action-conditional similar to CPC|Action (Guo et al., 2018). We utilize the same target network used in train temporal difference in MuZero to compute our target vectors. We then attempt to match x_n with the corresponding target vector y_n directly computed from the observation at that timestep. We compute the target vector y_n with the same target network weights used to compute the temporal difference loss in MuZero. Since the contrastive loss is ultimately a classification problem, we also need negative examples. For each pair x_n and y_n , the sets $\{x_i\}_{i=0, i \neq n}^N$ and $\{y_i\}_{i=0, i \neq n}^N$ serve as the negative examples with N the set of randomly sampled examples in the minibatch. We randomly sample 10% of the examples to constitute N . As in previous work (Banino et al., 2021; Chen et al., 2020a; Mitrovic et al., 2021) we use a cosine distance between x_i and y_i and feed this distance metric through a softmax function. We use the loss function in ReLIC (Mitrovic et al., 2021) and CoBERL (Banino et al., 2021) and use a temperature of 0.1.

The 256-dimensional vector x_n is derived from a two layer critic function which is first a convolutional layer with stride 1 and kernel size 3 on top of the state of the dynamics function. This is then fed into a fully connected layer to produce x_n . For each step n , the target vector y_n , which serves as a positive example, is derived from the final convolutional layer (8) that is fed into the residual stack of the encoder. The vector is computed by inputting the encoder, using the target network weights, with the corresponding image at that future step n . The encoding is then fed into the critic network to compute y_n .

The encoder of MuZero uses 15 past images as history. However, when we compute the target vectors, our treatment of the encoding history is different from that of the agent; instead of stacking all of the historical images ($n - 15 \dots n - 1$) up to the corresponding step n , we simply replace the history stack with 15 copies of the image at the current step n . We found that this technique enhanced the performance of contrastive learning.

13.2.3 *MuZero + SPR*

The implementation of SPR is similar to the contrastive objective except that it does not use negative examples. Thus it only uses a mean squared error instead of InfoNCE. The architecture for SPR is nearly identical to what is described in [Section 13.2.2](#). The same layers mentioned compute predicted and target vectors x_n and y_n respectively. However, SPR relies on an additional projection layer for learning. We thus add a projection layer p_n of 256 dimensions on top of the critic function mentioned in [Section 13.2.2](#). Instead of computing a loss between x_n and y_n , we follow the original formulation of SPR and compute mean squared error between L2-normalized p_n and L2-normalized y_n .

Game	Mode	MuZero				Q-Learning		
		MZ	MZ+Contr	MZ+Recon	MZ+SPR	QL	QL+Model	QL+Model+Recon
bigfish	train	0.77	0.73	0.80	0.79	0.60	0.53	0.57
	test	0.60	0.55	0.61	0.59	0.58	0.59	0.51
bossfight	train	0.91	0.93	0.94	0.94	0.51	0.64	0.71
	test	0.92	0.94	0.92	0.95	0.57	0.65	0.79
caveflyer	train	0.91	0.90	0.87	0.67	0.82	0.83	0.88
	test	0.77	0.81	0.77	0.58	0.65	0.64	0.73
chaser	train	0.28	0.89	0.26	0.89	0.16	0.53	0.54
	test	0.24	0.89	0.24	0.89	0.17	0.68	0.65
climber	train	0.72	0.96	0.92	0.94	0.42	0.58	0.81
	test	0.42	0.87	0.76	0.83	0.27	0.40	0.68
coinrun	train	0.98	0.98	0.98	0.97	0.85	0.87	0.87
	test	0.64	0.71	0.68	0.73	0.58	0.60	0.68
dodgeball	train	0.78	0.30	0.74	0.42	0.58	0.60	0.62
	test	0.77	0.29	0.73	0.41	0.54	0.64	0.59
fruitbot	train	0.33	0.44	0.55	0.52	0.56	0.61	0.65
	test	0.31	0.38	0.57	0.48	0.69	0.72	0.75
heist	train	0.13	0.17	0.18	0.13	0.14	0.14	0.25
	test	-0.17	-0.09	-0.18	-0.16	-0.17	-0.17	0.05
jumper	train	0.75	0.89	0.87	0.90	0.54	0.61	0.82
	test	0.32	0.78	0.77	0.73	0.06	0.09	0.66
leaper	train	0.98	0.98	0.97	0.97	0.70	0.75	0.74
	test	0.86	0.90	0.88	0.90	0.80	0.85	0.85
maze	train	0.58	0.38	0.54	0.44	0.26	0.30	0.39
	test	-0.25	-0.39	-0.19	-0.31	-0.41	-0.39	-0.39
miner	train	0.82	0.96	1.06	1.06	0.27	0.46	0.56
	test	0.59	0.78	0.85	0.94	0.23	0.15	0.11
ninja	train	0.98	0.98	0.98	0.98	0.70	0.77	0.83
	test	0.84	0.87	0.83	0.85	0.52	0.61	0.78
plunder	train	0.38	0.97	0.31	0.97	0.32	0.41	0.50
	test	0.17	0.94	0.13	0.91	0.04	0.05	0.40
starpilot	train	1.19	1.16	1.23	0.93	0.65	0.76	0.77
	test	0.98	0.95	0.91	0.84	0.66	0.77	0.80
Average	train	0.72 (0.70/0.75)	0.78 (0.78/0.79)	0.75 (0.74/0.88)	0.79 (0.71/0.79)	0.51 (0.49/0.51)	0.60 (0.59/0.60)	0.66 (0.58/0.66)
	test	0.50 (0.48/0.55)	0.63 (0.61/0.65)	0.57 (0.54/0.75)	0.64 (0.57/0.64)	0.36 (0.35/0.37)	0.45 (0.44/0.47)	0.54 (0.45/0.54)

Table 24: Final normalised training and test scores on Procgen when training on 500 levels for 30M environment frames, reporting the median across 3 seeds (for MZ, MZ+Recon across 5 seeds). Min/max over seeds are shown in parentheses. Final scores for each seed were computed as means over data points from the last 1M frames. A subset of the experiments on Maze have terminated before 30M frames therefore we computed final scores at 27M for all agents on this game.

Training levels	Mode	MZ	MZ+Contr	MZ+Recon	MZ+SPR
10	train	0.73	0.76	0.76	0.68
	test	0.07	0.13	0.12	0.09
100	train	0.72	0.76	0.82	0.79
	test	0.29	0.46	0.47	0.40
500	train	0.72	0.78	0.75	0.79
	test	0.50	0.63	0.57	0.64
All	train	0.71	0.82	0.93	0.85
	test	0.68	0.80	0.93	0.83

Table 25: MuZero with self-supervised losses on Procgen. Final mean normalised training and test scores for different number of training levels. Reporting median values over 1 seed on 10, 100 levels; on 500 levels 3 seeds for MZ+Contr, MZ+SPR and 5 seeds for MZ, MZ+Recon; on infinite levels 1 seed for MZ+Contr, MZ+SPR and 2 seeds for MZ, MZ+Recon. Final scores for each seed were computed as means over data points from the last 1M frames.

	reach	push	pick-place
Agent	Test	Test	Test
RL ²	100%	96.5%	98.5%
MuZero	100%	100%	100%
Q-Learning	97.5%	99.8%	99.9%

Table 26: Zero-shot test performance on the ML-1 procedural generalization benchmark of Meta-World. Shown are baseline results on RL² (Duan et al., 2016) from the Meta-World paper (Yu et al., 2020a) as well as our results on MuZero. Note that RL² is a meta-learning method and was trained for 300M environment steps, while MuZero and Q-Learning do not require meta-training were only trained for 50M steps and fine-tune for 10M. We average the success rate of the last 1M environment steps for each seed and report medians across three seeds for MuZero and Q-Learning (RL² results from (Duan et al., 2016)).

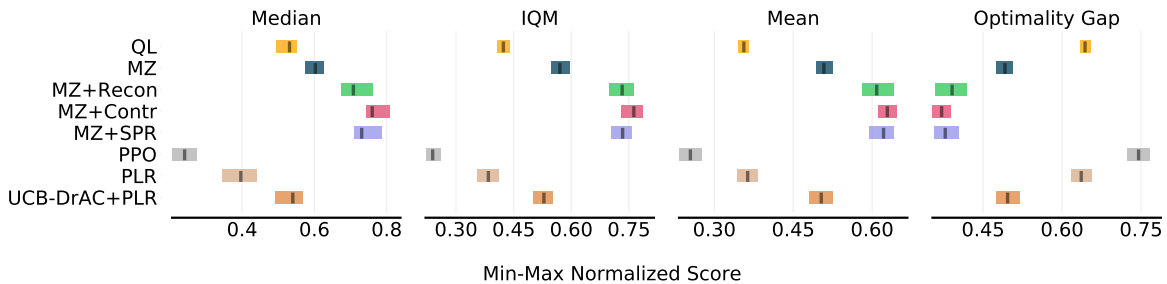


Figure 20: Additional metrics (proposed in Agarwal et al. (2021)) indicating zero-shot test performance of different methods on ProcGen. IQM corresponds to the Inter-Quartile Mean among all runs, and Optimality Gap refers to the amount by which the algorithm fails to meet a minimum score of 1.0.

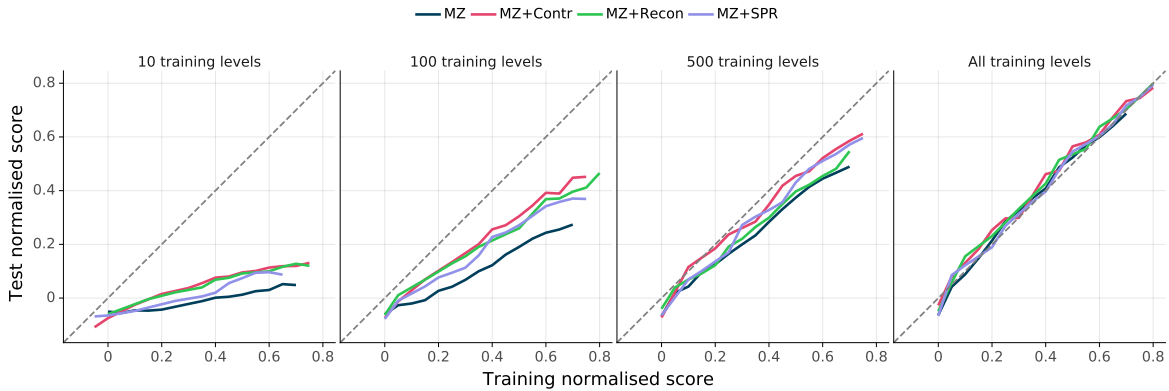
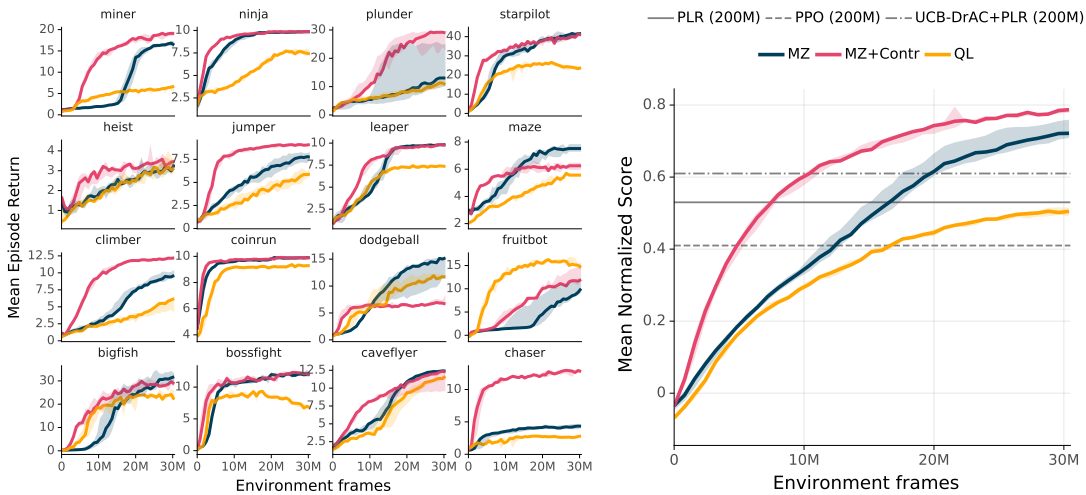


Figure 21: Isolating the interaction between self-supervision and data diversity. Each plot shows the generalization performance as a function of training performance for different numbers of training levels. Also shown is the unity line, where training and testing performance are equivalent (note that with infinite training levels, all lines collapse to the unity line, as in this case there is no difference between train and test). Generally speaking, self-supervision results in stronger generalization than MuZero. Reporting median values over 1 seed on 10, 100 levels; on 500 levels 3 seeds for MZ+Contr, MZ+SPR and 5 seeds for MZ, MZ+Recon; on infinite levels 1 seed for MZ+Contr, MZ+SPR and 2 seeds for MZ, MZ+Recon. For visibility, min/max seeds are not shown.



(a) Individual scores across all 16 Procgen games

(b) Mean normalized Score

Figure 22: Training performance on Procgen. See Figure 12 in the main text for results on the test set. Reporting median values over 3 seeds (5 seeds for MZ), with shaded regions indicating min/max seeds.

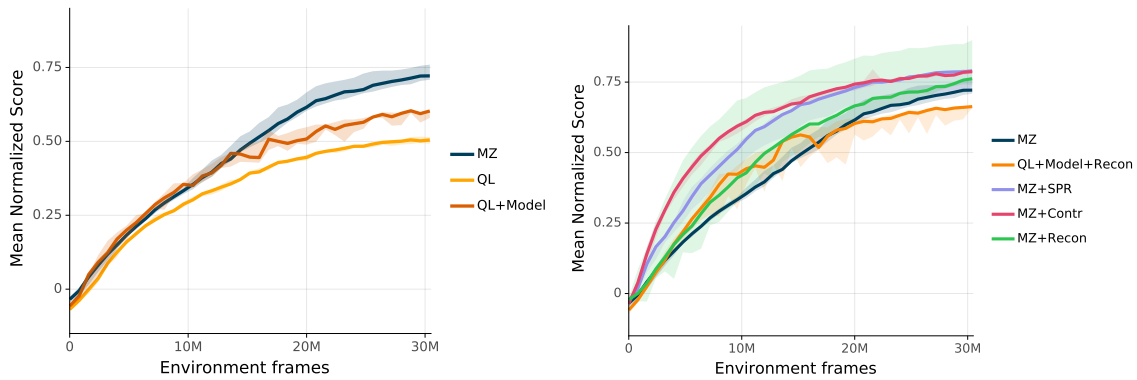


Figure 23: Training performance on Procgen. See Figure 13 in the main text for corresponding results on the test set. Left: the effect of planning. Right: the effect of self-supervision. Reporting median performance over 3 seeds (5 seeds for MZ and MZ+Recon), with shaded regions indicating min/max seeds.

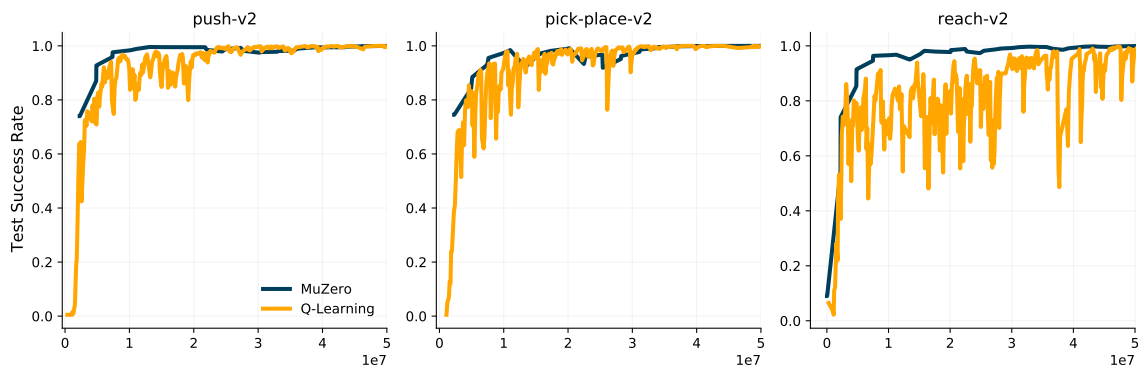


Figure 24: Zero-shot test success rates MuZero and Q-Learning on unseen goals on the ML-1 procedural generalization benchmark of Meta-World. While both MuZero and Q-Learning achieve near optimal performance, MuZero is more stable and a bit more data-efficient than Q-Learning. Shown are medians across three seeds (for visibility, min/max seeds are not shown).

Game	Mode	MuZero				Q-Learning
		MZ	MZ+Contr	MZ+Recon	MZ+SPR	QL
ML10	train	97.6%	78.1%	97.8%	82.5%	85.2%
	zero-shot test	26.5%	17.0%	25.0%	17.8%	6.8%
	fine-tune test	94.1%	60.0%	97.5%	51.3%	80.1%
ML45	train	77.2%	57.5%	74.9%	54.4%	55.9%
	zero-shot test	17.7%	12.5%	18.5%	16.0%	10.8%
	fine-tune test	76.7%	77.3%	81.7%	77.2%	78.1%

Table 27: Success rates for MuZero (with and without self-supervision) and Q-Learning on ML-10 and ML-45. Training results are shown at 50M frames and fine-tuning results are shown at 10M frames. Reporting average of the last 1M frames and then medians across 3 seeds (2 for MZ and MZ+Recon on ML10).

BIBLIOGRAPHY

- [1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. “Deep reinforcement learning at the edge of the statistical precipice.” In: *NeurIPS*. 2021 (cit. on pp. [17](#), [82](#), [125](#)).
- [2] Guillaume Alain and Yoshua Bengio. “Understanding intermediate layers using linear classifier probes.” In: *ICLR (Workshop Track)*. 2017 (cit. on p. [38](#)).
- [3] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. “Deep speech 2: End-to-end speech recognition in english and mandarin.” In: *ICML*. 2016 (cit. on pp. [30](#), [36](#)).
- [4] Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. “Unsupervised state representation learning in atari.” In: *NeurIPS*. 2019 (cit. on pp. [53](#), [65](#), [84](#), [93](#)).
- [5] Ankesh Anand, Jacob Walker, Yazhe Li, Eszter Vértés, Julian Schrittwieser, Sherjil Ozair, Théophane Weber, and Jessica B Hamrick. “Procedural generalization by planning with self-supervised world models.” In: *ICLR*. 2021 (cit. on pp. [54](#), [94](#)).
- [6] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Joze-fowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. “Learning dexterous in-hand manipulation.” In: *The International Journal of Robotics Research* (2020) (cit. on p. [81](#)).
- [7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. “Layer normaliza-tion.” In: *arXiv preprint arXiv:1607.06450* (2016) (cit. on p. [26](#)).
- [8] Philip Bachman, R Devon Hjelm, and William Buchwalter. “Learning rep-resentations by maximizing mutual information across views.” In: *NeurIPS*. 2019 (cit. on pp. [41](#), [56](#)).
- [9] Philip Bachman, Ouais al Sharif, and Doina Precup. “Learning with Pseudo-Ensembles.” In: *NeurIPS*. 2014 (cit. on p. [63](#)).

- [10] 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 (cit. on p. 56).
- [11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate.” In: *ICLR*. 2015 (cit. on p. 26).
- [12] Andrea Banino, Adrià Puidomenech Badia, Jacob Walker, Tim Scholtes, Jovana Mitrovic, and Charles Blundell. “CoBERL: Contrastive BERT for Reinforcement Learning.” In: *arXiv preprint arXiv:2107.05431* (2021) (cit. on pp. 80, 121).
- [13] Victor Bapst, Alvaro Sanchez-Gonzalez, Carl Doersch, Kimberly Stachenfeld, Pushmeet Kohli, Peter Battaglia, and Jessica Hamrick. “Structured agents for physical construction.” In: *ICML*. 2019 (cit. on p. 79).
- [14] David Barber and Felix Agakov. “The IM Algorithm: A Variational Approach to Information Maximization.” In: *NeurIPS*. 2003 (cit. on p. 38).
- [15] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. “Mutual Information Neural Estimation.” In: *ICML*. 2018. URL: <http://proceedings.mlr.press/v80/belghazi18a.html> (cit. on pp. 37, 38, 41).
- [16] Marc G Bellemare, Will Dabney, and Rémi Munos. “A distributional perspective on reinforcement learning.” In: *ICML*. 2017 (cit. on pp. 28, 58).
- [17] 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) (cit. on pp. 38, 56, 58).
- [18] Aurélien Bellet, Amaury Habrard, and Marc Sebban. “A survey on metric learning for feature vectors and structured data.” In: *arXiv preprint arXiv:1306.6709* (2013) (cit. on p. 32).
- [19] Richard Bellman. “A Markovian decision process.” In: *Journal of mathematics and mechanics* (1957), pp. 679–684 (cit. on p. 24).
- [20] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957 (cit. on p. 24).
- [21] Yoshua Bengio. “Learning deep architectures for AI.” In: *Foundations and Trends in Machine Learning* 2.1 (2009), pp. 1–127 (cit. on p. 44).

- [22] Yoshua Bengio, Aaron Courville, and Pascal Vincent. “Representation Learning: A Review and New Perspectives.” In: *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)* 35.8 (2013), pp. 1798–1828 (cit. on pp. 30, 44).
- [23] William Bialek and Naftali Tishby. “Predictive information.” In: *arXiv preprint cond-mat/9902341* (1999) (cit. on p. 40).
- [24] G Branwen. *The Scaling Hypothesis*. <https://www.gwern.net/Scaling-hypothesis>. 2021 (cit. on p. 25).
- [25] G Branwen. *Why Tool AIs Want to Be Agent AIs*. <https://www.gwern.net/Tool-AI>. 2021 (cit. on p. 96).
- [26] 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: *NeurIPS*. 2020 (cit. on pp. 25, 70).
- [27] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. “Large-scale study of curiosity-driven learning.” In: *ICLR* (2019) (cit. on p. 48).
- [28] Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. “MONet: Unsupervised Scene Decomposition and Representation.” In: *arXiv preprint arXiv:1901.11390* (2019) (cit. on p. 39).
- [29] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. “Deep clustering for unsupervised learning of visual features.” In: *ECCV*. 2018 (cit. on p. 44).
- [30] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. “A simple framework for contrastive learning of visual representations.” In: *ICML*. 2020 (cit. on pp. 56, 60, 61, 112, 121).
- [31] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. “Big Self-Supervised Models are Strong Semi-Supervised Learners.” In: *NeurIPS*. 2020 (cit. on pp. 56, 70).
- [32] Xi Chen, Yan Duan, Rein Houthoofd, John Schulman, Ilya Sutskever, and Pieter Abbeel. “Infogan: Interpretable representation learning by information maximizing generative adversarial nets.” In: *NeurIPS*. 2016 (cit. on p. 38).

- [33] François Chollet. “On the measure of intelligence.” In: *arXiv preprint arXiv:1911.01547* (2019) (cit. on p. 76).
- [34] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. “Empirical evaluation of gated recurrent neural networks on sequence modeling.” In: *arXiv preprint arXiv:1412.3555* (2014) (cit. on p. 26).
- [35] Adam Coates, Andrew Ng, and Honglak Lee. “An analysis of single-layer networks in unsupervised feature learning.” In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011 (cit. on p. 39).
- [36] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. “Leveraging procedural generation to benchmark reinforcement learning.” In: *ICML*. 2020 (cit. on pp. 75–78, 82, 83).
- [37] Karl Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. “Phasic policy gradient.” In: *ICML*. 2021 (cit. on p. 82).
- [38] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. “Quantifying generalization in reinforcement learning.” In: *ICML*. 2019 (cit. on p. 76).
- [39] Remi Tachet des Combes, Mohammad Pezeshki, Samira Shabianian, Aaron Courville, and Yoshua Bengio. “On the learning dynamics of deep neural networks.” In: *arXiv preprint arXiv:1809.06848* (2018) (cit. on pp. 50, 64).
- [40] Alexis Conneau and Douwe Kiela. “SentEval: An Evaluation Toolkit for Universal Sentence Representations.” In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*. 2018 (cit. on p. 39).
- [41] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. “Two-stage learning kernel algorithms.” In: *ICML*. 2010 (cit. on p. 32).
- [42] Rémi Coulom. “Efficient selectivity and backup operators in Monte-Carlo tree search.” In: *International conference on computers and games*. 2006 (cit. on pp. 29, 81).
- [43] Kenneth James Williams Craik. *The nature of explanation*. Vol. 445. CUP Archive, 1952 (cit. on p. 75).
- [44] Giuseppe Cuccu, Julian Togelius, and Philippe Cudré-Mauroux. “Playing atari with six neurons.” In: *International Conference on Autonomous Agents and Multiagent Systems* (2019) (cit. on p. 39).

- [45] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. “Implicit quantile networks for distributional reinforcement learning.” In: *ICML*. 2018 (cit. on p. [81](#)).
- [46] Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. “The helmholtz machine.” In: *Neural computation* (1995) (cit. on p. [76](#)).
- [47] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. “Density estimation using Real NVP.” In: *ICLR*. 2017 (cit. on p. [37](#)).
- [48] Andrea Dittadi, Frederik Träuble, Manuel Wüthrich, Felix Widmaier, Peter Gehler, Ole Winther, Francesco Locatello, Olivier Bachem, Bernhard Schölkopf, and Stefan Bauer. “Representation Learning for Out-Of-Distribution Generalization in Reinforcement Learning.” In: *arXiv preprint arXiv:2107.05686* (2021) (cit. on p. [76](#)).
- [49] 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: *CVPR*. 2021 (cit. on p. [80](#)).
- [50] Carl Doersch and Andrew Zisserman. “Multi-task self-supervised visual learning.” In: *ICCV*. 2017 (cit. on pp. [37](#), [44](#)).
- [51] Monroe D Donsker and SR Srinivasa Varadhan. “Asymptotic evaluation of certain Markov process expectations for large time. IV.” In: *Communications on Pure and Applied Mathematics* 36.2 (1983), pp. 183–212 (cit. on p. [38](#)).
- [52] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. “An image is worth 16x16 words: Transformers for image recognition at scale.” In: 2021 (cit. on p. [26](#)).
- [53] Wuyang Duan. “Learning state representations for robotic control: Information disentangling and multi-modal learning.” MA thesis. Delft University of Technology, 2017 (cit. on p. [37](#)).
- [54] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. “RL²: Fast reinforcement learning via slow reinforcement learning.” In: *arXiv preprint arXiv:1611.02779* (2016) (cit. on pp. [20](#), [22](#), [89](#), [91](#), [125](#)).

- [55] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. “Challenges of real-world reinforcement learning.” In: *ICML*. 2019 (cit. on pp. 25, 56).
- [56] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. “Adversarially learned inference.” In: *ICLR*. 2017 (cit. on p. 37).
- [57] Cian Eastwood and Christopher KI Williams. “A framework for the quantitative evaluation of disentangled representations.” In: *ICLR*. 2018 (cit. on p. 44).
- [58] Steve Engelhardt. *BJARS.com Atari Archives*. <http://bjars.com>. [Online; accessed 1-March-2019]. 2019 (cit. on p. 43).
- [59] Robert Epstein. “The empty brain.” In: *Aeon*, May 18 (2016) (cit. on p. 31).
- [60] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-agnostic meta-learning for fast adaptation of deep networks.” In: *ICML*. 2017 (cit. on pp. 20, 79, 89).
- [61] Chelsea Finn, Ian Goodfellow, and Sergey Levine. “Unsupervised learning for physical interaction through video prediction.” In: *NeurIPS*. 2016, pp. 64–72 (cit. on pp. 30, 37).
- [62] Chelsea Finn and Sergey Levine. “Deep visual foresight for planning robot motion.” In: *ICRA*. 2017 (cit. on pp. 76, 79, 80).
- [63] Meire Fortunato et al. “Noisy Networks For Exploration.” In: *ICLR*. 2018 (cit. on p. 64).
- [64] 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) (cit. on p. 56).
- [65] Karl Friston. “A theory of cortical responses.” In: *Philosophical transactions of the Royal Society B: Biological sciences* 360.1456 (2005), pp. 815–836 (cit. on p. 40).
- [66] Kunihiko Fukushima and Sei Miyake. “Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition.” In: *Competition and cooperation in neural nets*. Springer, 1982 (cit. on p. 26).
- [67] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. “Deepmdp: Learning continuous latent space models for representation learning.” In: *ICML*. 2019 (cit. on pp. 65, 68, 70).

- [68] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016 (cit. on p. 26).
- [69] Robert D Gordon and David E Irwin. “What’s in an object file? Evidence from priming studies.” In: *Perception & Psychophysics* 58.8 (1996), pp. 1260–1277 (cit. on pp. 30, 36).
- [70] Karol Gregor, Danilo Jimenez Rezende, Frederic Besse, Yan Wu, Hamza Merzic, and Aaron van den Oord. “Shaping belief states with generative environment models for rl.” In: *NeurIPS*. 2019 (cit. on p. 79).
- [71] Jean-Bastien Grill, Florent Altché, Yunhao Tang, Thomas Hubert, Michal Valko, Ioannis Antonoglou, and Rémi Munos. “Monte-carlo tree search as regularized policy optimization.” In: *ICML*. 2020 (cit. on p. 80).
- [72] 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 (cit. on pp. 32, 56, 60, 61, 69, 71, 113, 114).
- [73] Christopher Grimm, André Barreto, Satinder Singh, and David Silver. “The value equivalence principle for model-based reinforcement learning.” In: *NeurIPS*. 2020 (cit. on pp. 80, 81, 88, 91).
- [74] Charles G Gross. *Learning, perception, and the brain*. 1968 (cit. on pp. 30, 37).
- [75] Arthur Guez, Mehdi Mirza, Karol Gregor, Rishabh Kabra, Sébastien Racanière, Théophane Weber, David Raposo, Adam Santoro, Laurent Orseau, Tom Eccles, et al. “An investigation of model-free planning.” In: *ICML*. 2019 (cit. on pp. 79, 81, 84).
- [76] 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 (cit. on pp. 65, 80).
- [77] Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Bernardo A Pires, and Rémi Munos. “Neural predictive belief representations.” In: *ICML* (2018) (cit. on pp. 65, 84, 111, 121).

- [78] Zhaohan Daniel Guo, Shantanu Thakoor, Miruna Pislar, Bernardo Avila Pires, Florent Alché, Corentin Tallec, Alaa Saade, Daniele Calandriello, Jean-Bastien Grill, Yunhao Tang, et al. “Byol-explore: Exploration by bootstrapped prediction.” In: *NeurIPS*. 2022 (cit. on p. 54).
- [79] 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 (cit. on pp. 32, 39, 41).
- [80] David Ha and Jürgen Schmidhuber. “Recurrent world models facilitate policy evolution.” In: *NeurIPS*. 2018, pp. 2450–2462 (cit. on pp. 30, 37, 76, 79).
- [81] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. “Dream to control: Learning behaviors by latent imagination.” In: *ICLR*. 2020 (cit. on pp. 30, 64).
- [82] 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 (cit. on pp. 57, 64).
- [83] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. “Mastering atari with discrete world models.” In: *ICLR*. 2021 (cit. on pp. 76, 79, 80).
- [84] Jessica B Hamrick. “Analogues of mental simulation and imagination in deep learning.” In: *Current Opinion in Behavioral Sciences* 29 (2019), pp. 8–16 (cit. on pp. 76, 79).
- [85] Jessica B Hamrick, Abram L Friesen, Feryal Behbahani, Arthur Guez, Fabio Viola, Sims Witherspoon, Thomas Anthony, Lars Buesing, Petar Veličković, and Théophane Weber. “On the role of planning in model-based deep reinforcement learning.” In: *ICLR*. 2021 (cit. on pp. 79–81, 91).
- [86] Hado P van Hasselt, Matteo Hessel, and John Aslanides. “When to use parametric models in reinforcement learning?” In: *NeurIPS*. 2019 (cit. on pp. 57, 58, 63–65, 112).
- [87] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. “Masked autoencoders are scalable vision learners.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022 (cit. on p. 25).

- [88] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. "Momentum contrast for unsupervised visual representation learning." In: *CVPR*. 2020 (cit. on pp. [56](#), [61](#), [71](#)).
- [89] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016 (cit. on p. [26](#)).
- [90] Nicolas Heess, Greg Wayne, David Silver, Timothy Lillicrap, Yuval Tassa, and Tom Erez. "Learning continuous control policies by stochastic value gradients." In: *NeurIPS*. 2015 (cit. on p. [80](#)).
- [91] 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) (cit. on p. [56](#)).
- [92] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. "Deep reinforcement learning that matters." In: *AAAI*. 2018 (cit. on p. [68](#)).
- [93] 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: *ICML*. 2021 (cit. on p. [80](#)).
- [94] 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 (cit. on pp. [28](#), [58](#), [64](#)).
- [95] Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. "Towards a Definition of Disentangled Representations." In: *arXiv preprint arXiv:1812.02230* (2018) (cit. on pp. [39](#), [44](#)).
- [96] Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. "Darla: Improving zero-shot transfer in reinforcement learning." In: *ICML*. 2017 (cit. on p. [37](#)).

- [97] 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 (cit. on pp. [34](#), [56](#)).
- [98] 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) (cit. on pp. [37](#), [38](#), [40](#), [41](#), [44](#)).
- [99] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory.” In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cit. on p. [26](#)).
- [100] Herke van Hoof, Nutan Chen, Maximilian Karl, Patrick van der Smagt, and Jan Peters. “Stable reinforcement learning with autoencoders for tactile and visual data.” In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 3928–3934 (cit. on p. [39](#)).
- [101] Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Mohammadamin Barekatin, Simon Schmitt, and David Silver. “Learning and Planning in Complex Action Spaces.” In: *ICML*. 2021 (cit. on pp. [29](#), [82](#), [84](#), [120](#)).
- [102] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent component analysis*. Vol. 46. John Wiley & Sons, 2004 (cit. on p. [43](#)).
- [103] Aapo Hyvärinen and Petteri Pajunen. “Nonlinear independent component analysis: Existence and uniqueness results.” In: *Neural Networks* 12.3 (1999), pp. 429–439 (cit. on p. [48](#)).
- [104] Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. “Generalization in reinforcement learning with selective noise injection and information bottleneck.” In: *NeurIPS*. 2019 (cit. on p. [78](#)).
- [105] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.” In: *ICML*. 2015 (cit. on p. [63](#)).
- [106] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift.” In: *ICML*. 2015 (cit. on p. [26](#)).

- [107] Eric Jang. *Just Ask for Generalization*. <https://evjang.com/2021/10/23/generalization.html>. 2021 (cit. on p. 96).
- [108] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. “When to Trust Your Model: Model-Based Policy Optimization.” In: *NeurIPS*. 2019 (cit. on p. 79).
- [109] Thomas Jentzsch and CPUWIZ. *AtariAge Atari 2600 Forums*. 2019. URL: <http://atariage.com/forums/forum/16-atari-2600/> (cit. on p. 43).
- [110] Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. “Prioritized level replay.” In: *ICML*. 2021 (cit. on pp. 15, 79, 82, 83, 86).
- [111] Rico Jonschkowski and Oliver Brock. “Learning state representations with robotic priors.” In: *Autonomous Robots* 39.3 (2015), pp. 407–428 (cit. on p. 39).
- [112] Rico Jonschkowski, Roland Hafner, Jonathan Scholz, and Martin Riedmiller. “Pves: Position-velocity encoders for unsupervised learning of structured state representations.” In: *arXiv preprint arXiv:1705.09805* (2017) (cit. on p. 39).
- [113] Niels Justesen, Ruben Rodriguez Torrado, Philip Bontrager, Ahmed Khalifa, Julian Togelius, and Sebastian Risi. “Illuminating generalization in deep reinforcement learning through procedural level generation.” In: *AAAI*. 2019 (cit. on p. 76).
- [114] Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłoś, Błażej Osiański, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Koza-kowski, Sergey Levine, et al. “Model Based Reinforcement Learning for Atari.” In: *ICLR*. 2019 (cit. on pp. 13, 56, 57, 64–66, 114, 116).
- [115] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Koza-kowski, Sergey Levine, et al. “Model-Based Reinforcement Learning for Atari.” In: *arXiv preprint arXiv:1903.00374* (2020) (cit. on pp. 53, 79, 80).
- [116] Ken Kansky, Tom Silver, David A Mély, Mohamed Eldawy, Miguel Lázaro-Gredilla, Xinghua Lou, Nimrod Dorfman, Szymon Sidor, Scott Phoenix, and Dileep George. “Schema networks: Zero-shot transfer with a generative causal model of intuitive physics.” In: *ICML*. 2017 (cit. on p. 101).

- [117] 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) (cit. on pp. 25, 26).
- [118] Mina Khan, P Srivatsa, Advait Rane, Shriram Chenniappa, Rishabh Anand, Sherjil Ozair, and Pattie Maes. “Pretrained encoders are all you need.” In: *arXiv preprint arXiv:2106.05139* (2021) (cit. on p. 35).
- [119] Kacper Piotr Kielak. *Do recent advancements in model-based deep reinforcement learning really improve data efficiency?* 2020. URL: <https://openreview.net/forum?id=Bke9u1HFwB> (cit. on pp. 64, 110).
- [120] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes.” In: *ICLR*. 2014 (cit. on pp. 37, 45).
- [121] Thomas Kipf, Elise van der Pol, and Max Welling. “Contrastive learning of structured world models.” In: *ICLR*. 2019 (cit. on pp. 65, 80).
- [122] Louis Kirsch, Sjoerd van Steenkiste, and Jürgen Schmidhuber. “Improving generalization in meta reinforcement learning using learned objectives.” In: *ICLR*. 2020 (cit. on p. 91).
- [123] Levente Kocsis and Csaba Szepesvári. “Bandit based Monte-Carlo planning.” In: *European conference on machine learning*. 2006 (cit. on p. 81).
- [124] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. “Revisiting Self-Supervised Visual Representation Learning.” In: *CVPR*. 2019 (cit. on pp. 37, 44).
- [125] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks.” In: *NeurIPS*. 2012 (cit. on pp. 30, 36).
- [126] Samuli Laine and Timo Aila. “Temporal ensembling for semi-supervised learning.” In: *ICLR*. 2017 (cit. on p. 63).
- [127] 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) (cit. on pp. 36, 75).
- [128] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. “Reinforcement Learning with Augmented Data.” In: *NeurIPS*. 2020 (cit. on pp. 64, 65, 78).

- [129] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. “Backpropagation applied to handwritten zip code recognition.” In: *Neural computation* (1989) (cit. on p. 26).
- [130] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. “Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model.” In: *NeurIPS*. 2020 (cit. on p. 64).
- [131] Shane Legg and Marcus Hutter. “Universal intelligence: A definition of machine intelligence.” In: *Minds and machines* 17.4 (2007), pp. 391–444 (cit. on p. 24).
- [132] Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Francois Goudou, and David Filliat. “State representation learning for control: An overview.” In: *Neural Networks* 108 (2018) (cit. on pp. 37, 65).
- [133] Long-Ji Lin. “Self-improving reactive agents based on reinforcement learning, planning and teaching.” In: *Machine learning* (1992) (cit. on p. 28).
- [134] Seppo Linnainmaa. “The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors.” PhD thesis. Master’s Thesis (in Finnish), Univ. Helsinki, 1970 (cit. on p. 26).
- [135] Francesco Locatello, Stefan Bauer, Mario Lucic, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. “Challenging common assumptions in the unsupervised learning of disentangled representations.” In: *ICML* (2019) (cit. on p. 43).
- [136] Xiuyuan Lu, Benjamin Van Roy, Vikranth Dwaracherla, Morteza Ibrahimi, Ian Osband, and Zheng Wen. “Reinforcement learning, bit by bit.” In: *arXiv preprint arXiv:2103.04047* (2021) (cit. on p. 96).
- [137] Zhuang Ma and Michael Collins. “Noise Contrastive Estimation and Negative Sampling for Conditional Models: Consistency and Statistical Efficiency.” In: *EMNLP* (2018) (cit. on pp. 39, 41).
- [138] 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: *Journal of Artificial Intelligence Research* 61 (2018), pp. 523–562 (cit. on p. 64).

- [139] David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc., 1982. ISBN: 0716715678 (cit. on pp. 30, 36).
- [140] Bogdan Mazoure, Ahmed M Ahmed, Patrick MacAlpine, R Devon Hjelm, and Andrey Kolobov. “Cross-Trajectory Representation Learning for Zero-Shot Generalization in RL.” In: *ICLR*. 2022 (cit. on pp. 76, 78, 80).
- [141] Bogdan Mazoure, Remi Tachet des Combes, Thang Doan, Philip Bachman, and R Devon Hjelm. “Deep Reinforcement and InfoMax Learning.” In: *NeurIPS*. 2020 (cit. on pp. 35, 65, 69, 78).
- [142] David McAllester and Karl Statos. “Formal Limitations on the Measurement of Mutual Information.” In: *arXiv preprint arXiv:1811.04251* (2018) (cit. on pp. 41, 50).
- [143] Trevor McInroe, Lukas Schäfer, and Stefano V Albrecht. “Learning Temporally-Consistent Representations for Data-Efficient Reinforcement Learning.” In: *arXiv preprint arXiv:2110.04935* (2021) (cit. on p. 54).
- [144] Marvin Minsky. “Steps toward artificial intelligence.” In: *Proceedings of the IRE* (1961) (cit. on p. 24).
- [145] Jovana Mitrovic, Brian McWilliams, Jacob Walker, Lars Buesing, and Charles Blundell. “Representation learning via invariant causal mechanisms.” In: *ICLR*. 2021 (cit. on pp. 84, 121).
- [146] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. “Asynchronous methods for deep reinforcement learning.” In: *ICML*. 2016 (cit. on p. 29).
- [147] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. “Playing Atari With Deep Reinforcement Learning.” In: *NeurIPS Deep Learning Workshop*. MIT Press, 2013. URL: <https://aeon.co/essays/your-brain-does-not-process-information-and-it-is-not-a-computer> (cit. on pp. 45, 98, 100).
- [148] 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* (2015) (cit. on pp. 28, 30, 36, 57, 58, 63, 64, 67).

- [149] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. “Model-based reinforcement learning: A survey.” In: *arXiv preprint arXiv:2006.16712* (2020) (cit. on p. 79).
- [150] Sharada Mohanty, Jyotish Poonganam, Adrien Gaidon, Andrey Kolobov, Blake Wulfe, Dipam Chakraborty, Gražvydas Šemetulskis, João Schapke, Jonas Kubilius, Jurgis Pašukonis, et al. “Measuring Sample Efficiency and Generalization in Reinforcement Learning Benchmarks: NeurIPS 2020 Progen Benchmark.” In: *arXiv preprint arXiv:2103.15332* (2021) (cit. on pp. 76, 79).
- [151] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. “Near-Optimal Representation Learning for Hierarchical Reinforcement Learning.” In: *ICLR*. 2019 (cit. on p. 39).
- [152] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. “Learning to adapt in dynamic, real-world environments through meta-reinforcement learning.” In: *ICLR*. 2019 (cit. on pp. 76, 79).
- [153] Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. “Gotta learn fast: A new benchmark for generalization in RL.” In: *arXiv preprint arXiv:1804.03720* (2018) (cit. on p. 76).
- [154] Evgenii Nikishin, Max Schwarzer, Pierluca D’Oro, Pierre-Luc Bacon, and Aaron Courville. “The primacy bias in deep reinforcement learning.” In: *ICML*. 2022 (cit. on p. 94).
- [155] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. “Learning deconvolution network for semantic segmentation.” In: *ICCV*. 2015 (cit. on p. 121).
- [156] Mehdi Noroozi and Paolo Favaro. “Unsupervised learning of visual representations by solving jigsaw puzzles.” In: *European conference on computer vision (ECCV)*. 2016 (cit. on p. 56).
- [157] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. “Action-conditional video prediction using deep networks in atari games.” In: *NeurIPS*. 2015, pp. 2863–2871 (cit. on pp. 37, 45, 98).
- [158] Junhyuk Oh, Satinder Singh, and Honglak Lee. “Value prediction network.” In: *NeurIPS*. 2017 (cit. on p. 81).

- [159] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. “Representation learning with contrastive predictive coding.” In: *arXiv preprint arXiv:1807.03748* (2018) (cit. on pp. [32](#), [34](#), [37](#), [38](#), [41](#), [44](#), [45](#), [56](#), [65](#), [80](#), [84](#), [98](#), [111](#), [112](#), [121](#)).
- [160] OpenAI et al. “Dota 2 with Large Scale Deep Reinforcement Learning.” In: *arXiv preprint arXiv:1912.06680* (2019) (cit. on pp. [25](#), [56](#)).
- [161] Sherjil Ozair, Yazhe Li, Ali Razavi, Ioannis Antonoglou, Aäron van den Oord, and Oriol Vinyals. “Vector Quantized Models for Planning.” In: *ICML. 2021* (cit. on p. [88](#)).
- [162] Sherjil Ozair, Corey Lynch, Yoshua Bengio, Aaron Van den Oord, Sergey Levine, and Pierre Sermanet. “Wasserstein Dependency Measure for Representation Learning.” In: *arXiv preprint arXiv:1903.11780* (2019) (cit. on pp. [41](#), [50](#)).
- [163] Stephanie E Palmer, Olivier Marre, Michael J Berry, and William Bialek. “Predictive information in a sensory population.” In: *Proceedings of the National Academy of Sciences* 112.22 (2015), pp. 6908–6913 (cit. on p. [40](#)).
- [164] 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* (cit. on p. [64](#)).
- [165] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. “Context Encoders: Feature Learning by Inpainting.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016* (cit. on p. [37](#)).
- [166] Ben Poole, Sherjil Ozair, Aäron Van den Oord, Alexander A Alemi, and George Tucker. “On Variational Bounds of Mutual Information.” In: *ICML. 2019* (cit. on pp. [32](#), [39](#), [41](#)).
- [167] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. “Learning transferable visual models from natural language supervision.” In: *ICML. 2021* (cit. on p. [25](#)).

- [168] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. "Learning transferable visual models from natural language supervision." In: *ICML*. 2021 (cit. on pp. 80, 81).
- [169] Roberta Raileanu and Rob Fergus. "Decoupling value and policy for generalization in reinforcement learning." In: *ICML*. 2021 (cit. on p. 82).
- [170] Roberta Raileanu, Max Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. "Automatic data augmentation for generalization in deep reinforcement learning." In: *NeurIPS*. 2021 (cit. on pp. 15, 83).
- [171] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. "Efficient off-policy meta-reinforcement learning via probabilistic context variables." In: *IMCL*. 2019 (cit. on p. 79).
- [172] Rajesh PN Rao and Dana H Ballard. "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects." In: *Nature neuroscience* 2.1 (1999), p. 79 (cit. on p. 40).
- [173] Danilo J Rezende, Ivo Danihelka, George Papamakarios, Nan Rosemary Ke, Ray Jiang, Theophane Weber, Karol Gregor, Hamza Merzic, Fabio Viola, Jane Wang, et al. "Causally correct partial models for reinforcement learning." In: *arXiv preprint arXiv:2002.02836* (2020) (cit. on p. 88).
- [174] Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. "Kornia: an open source differentiable computer vision library for pytorch." In: *The IEEE Winter Conference on Applications of Computer Vision*. 2020 (cit. on p. 63).
- [175] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*. Vol. 133. Springer, 2004 (cit. on p. 29).
- [176] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors." In: *nature* (1986) (cit. on p. 26).
- [177] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. "Prioritized experience replay." In: *ICLR*. 2016 (cit. on p. 28).
- [178] Jürgen Schmidhuber. "Curious model-building control systems." In: *Proc. international joint conference on neural networks*. 1991, pp. 1458–1463 (cit. on p. 76).

- [179] 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* (2020) (cit. on pp. [29](#), [56](#), [63](#), [71](#), [73](#), [75](#), [76](#), [80–82](#), [94](#)).
- [180] Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatin, Ioannis Antonoglou, and David Silver. “Online and offline reinforcement learning by planning with a learned model.” In: *NeurIPS* (2021) (cit. on pp. [76](#), [79](#), [81](#), [82](#), [118](#)).
- [181] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. “Proximal policy optimization algorithms.” In: *arXiv preprint arXiv:1707.06347* (2017) (cit. on pp. [15](#), [29](#), [45](#), [83](#), [101](#)).
- [182] Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. “Data-efficient reinforcement learning with self-predictive representations.” In: *ICLR*. 2021 (cit. on pp. [35](#), [80](#), [84](#), [85](#), [93](#)).
- [183] Max Schwarzer, Johan Samir Obando Ceron, Aaron Courville, Marc G Belle-mare, Rishabh Agarwal, and Pablo Samuel Castro. “Bigger, Better, Faster: Human-level Atari with human-level efficiency.” In: *International Conference on Machine Learning*. 2023 (cit. on p. [94](#)).
- [184] Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron C Courville. “Pretraining representations for data-efficient reinforcement learning.” In: *NeurIPS* 34 (2021) (cit. on pp. [54](#), [94](#)).
- [185] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. “Planning to explore via self-supervised world models.” In: *ICML*. 2020 (cit. on pp. [79](#), [91](#)).
- [186] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. “Time-contrastive networks: Self-supervised learning from video.” In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1134–1141 (cit. on p. [41](#)).
- [187] 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), p. 484 (cit. on pp. 25, 30, 36).
- [188] 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 (cit. on p. 29).
- [189] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play." In: *Science* 362.6419 (2018), pp. 1140–1144 (cit. on pp. 25, 29, 81).
- [190] Kihyuk Sohn. "Improved deep metric learning with multi-class n-pair loss objective." In: *NeurIPS*. 2016, pp. 1857–1865 (cit. on p. 41).
- [191] Aravind Srinivas, Michael Laskin, and Pieter Abbeel. "Curl: Contrastive unsupervised representations for reinforcement learning." In: *ICML*. 2020 (cit. on pp. 35, 61, 65, 69, 111).
- [192] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." In: *The Journal of Machine Learning Research* 15.1 (2014) (cit. on p. 63).
- [193] Adam Stooke and Pieter Abbeel. "rlpyt: A research code base for deep reinforcement learning in pytorch." In: *arXiv preprint arXiv:1909.01500* (2019) (cit. on pp. 54, 64).
- [194] Richard S Sutton. "Learning to predict by the methods of temporal differences." In: *Machine learning* (1988) (cit. on p. 28).
- [195] Richard S Sutton. "Dyna, an integrated architecture for learning, planning, and reacting." In: *ACM Sigart Bulletin* (1991) (cit. on pp. 76, 80).
- [196] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018 (cit. on p. 24).
- [197] Richard Sutton. "The bitter lesson." In: *Incomplete Ideas (blog)* (2019). <http://www.incompleteideas.net/IncIdeas/BitterLesson.html> (cit. on pp. 25, 56).

- [198] Antti Tarvainen and Harri Valpola. “Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results.” In: *NeurIPS*. 2017 (cit. on pp. [56](#), [60](#), [61](#), [63](#), [114](#)).
- [199] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. “Deepmind control suite.” In: *arXiv preprint arXiv:1801.00690* (2018) (cit. on p. [56](#)).
- [200] Gerald Tesauro et al. “Temporal difference learning and TD-Gammon.” In: *Communications of the ACM* (1995) (cit. on p. [24](#)).
- [201] Valentin Thomas, Jules Pondard, Emmanuel Bengio, Marc Sarfati, Philippe Beaudoin, Marie-Jean Meurs, Joelle Pineau, Doina Precup, and Yoshua Bengio. “Independently controllable factors.” In: *arXiv preprint arXiv:1708.01289* (2017) (cit. on p. [39](#)).
- [202] Yonglong Tian, Dilip Krishnan, and Phillip Isola. “Contrastive multiview coding.” In: *arXiv preprint arXiv:1906.05849* (2019) (cit. on p. [56](#)).
- [203] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. “Domain randomization for transferring deep neural networks from simulation to the real world.” In: *IROS*. 2017 (cit. on pp. [76](#), [81](#)).
- [204] Eleni Triantafillou, Richard Zemel, and Raquel Urtasun. “Few-Shot Learning Through an Information Retrieval Lens.” In: *NeurIPS*. 2017 (cit. on p. [39](#)).
- [205] Adam Tupper and Kouros Neshatian. “Evaluating Learned State Representations for Atari.” In: *2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ)*. 2020 (cit. on p. [35](#)).
- [206] Alan Turing. “Intelligent machinery (1948).” In: *B. Jack Copeland* (2004) (cit. on p. [24](#)).
- [207] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. “Deep image prior.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9446–9454 (cit. on p. [48](#)).
- [208] Hado Van Hasselt, Arthur Guez, and David Silver. “Deep reinforcement learning with double q-learning.” In: *AAAI*. 2016 (cit. on pp. [57](#), [58](#)).
- [209] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention is all you need.” In: *NeurIPS* (2017) (cit. on p. [26](#)).

- [210] Petar Veličković, Matko Bošnjak, Thomas Kipf, Alexander Lerchner, Raia Hadsell, Razvan Pascanu, and Charles Blundell. “Reasoning-Modulated Representations.” In: *arXiv preprint arXiv:2107.08881* (2021) (cit. on p. 35).
- [211] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. “Deep graph infomax.” In: 2019 (cit. on p. 37).
- [212] 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) (cit. on p. 56).
- [213] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. “Starcraft ii: A new challenge for reinforcement learning.” In: *arXiv preprint arXiv:1708.04782* (2017) (cit. on p. 25).
- [214] Jacob Walker, Eszter Vértés, Yazhe Li, Gabriel Dulac-Arnold, Ankesh Anand, Théophane Weber, and Jessica B Hamrick. “Investigating the role of model-based learning in exploration and transfer.” In: *arXiv preprint arXiv:2302.04009* (2023) (cit. on pp. 74, 94).
- [215] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding.” In: *International Conference on Learning Representations*. 2019. URL: <https://openreview.net/forum?id=rJ4km2R5t7> (cit. on p. 39).
- [216] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. “Learning to reinforcement learn.” In: *arXiv preprint arXiv:1611.05763* (2016) (cit. on p. 91).
- [217] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. “Benchmarking model-based reinforcement learning.” In: *arXiv preprint arXiv:1907.02057* (2019) (cit. on p. 76).
- [218] Tongzhou Wang and Phillip Isola. “Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere.” In: *ICML*. 2020 (cit. on pp. 111, 112).

- [219] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. "Dueling Network Architectures for Deep Reinforcement Learning." In: *ICML*. 2016 (cit. on pp. [58](#), [64](#), [66](#)).
- [220] David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. "Unsupervised control through non-parametric discriminative rewards." In: 2019 (cit. on p. [39](#)).
- [221] Christopher John Cornish Hellaby Watkins. "Learning from delayed rewards." In: (1989) (cit. on p. [28](#)).
- [222] Manuel Watter, Jost Springenberg, Joshka Boedecker, and Martin Riedmiller. "Embed to control: A locally linear latent dynamics model for control from raw images." In: *NeurIPS*. 2015 (cit. on pp. [37](#), [39](#)).
- [223] Théophane Weber, Sébastien Racanière, David P Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomenech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. "Imagination-augmented agents for deep reinforcement learning." In: *NeurIPS*. 2017 (cit. on pp. [76](#), [79](#), [80](#)).
- [224] Zach Whalen and Laurie N Taylor. "Playing the Past." In: *History and Nostalgia in Video Games*. Nashville, TN: Vanderbilt University Press (2008) (cit. on p. [43](#)).
- [225] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." In: *Machine learning* 8.3-4 (1992), pp. 229–256 (cit. on p. [29](#)).
- [226] Sam Witty, Jun Ki Lee, Emma Tosch, Akanksha Atrey, Michael Littman, and David Jensen. "Measuring and characterizing generalization in deep reinforcement learning." In: *arXiv preprint arXiv:1812.02868* (2018) (cit. on p. [76](#)).
- [227] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. "Google's neural machine translation system: Bridging the gap between human and machine translation." In: *arXiv preprint arXiv:1609.08144* (2016) (cit. on pp. [30](#), [36](#)).
- [228] Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. "Zero-Shot Learning - A Comprehensive Evaluation of the Good, the Bad and the Ugly." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018), 1–1 (cit. on p. [39](#)).

- [229] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. “Unsupervised data augmentation for consistency training.” In: *NeurIPS*. 2020 (cit. on p. 56).
- [230] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. “Reinforcement learning with prototypical representations.” In: *ICML*. 2021 (cit. on pp. 79, 80).
- [231] Denis Yarats, Ilya Kostrikov, and Rob Fergus. “Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels.” In: *ICLR*. 2021. URL: <https://openreview.net/forum?id=GY6-6sTvGaf> (cit. on pp. 63, 64, 66).
- [232] Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. “Mastering atari games with limited data.” In: *NeurIPS* 34 (2021) (cit. on p. 54).
- [233] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. “Gradient surgery for multi-task learning.” In: *NeurIPS*. 2020 (cit. on pp. 20, 22, 76, 89, 125).
- [234] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. “Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning.” In: *CoRL*. 2020 (cit. on pp. 75–77, 83, 90).
- [235] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. “Relational deep reinforcement learning.” In: *arXiv preprint arXiv:1806.01830* (2018) (cit. on p. 79).
- [236] Amy Zhang, Sainbayar Sukhbaatar, Adam Lerer, Arthur Szlam, and Rob Fergus. “Composable planning with attributes.” In: *ICML*. 2018 (cit. on p. 76).
- [237] Amy Zhang, Yuxin Wu, and Joelle Pineau. “Natural Environment Benchmarks for Reinforcement Learning.” In: *arXiv preprint arXiv:1811.06032* (2018) (cit. on pp. 76, 101).
- [238] Daniel Zhang et al. “The AI Index 2022 Annual Report.” In: *arXiv preprint arXiv:2205.03468* (2022) (cit. on p. 74).

- [239] Jin Zhang, Jianhao Wang, Hao Hu, Tong Chen, Yingfeng Chen, Changjie Fan, and Chongjie Zhang. “Metacure: Meta reinforcement learning with empowerment-driven exploration.” In: *ICML*. 2021 (cit. on p. 76).