Pretraining Reward-Free Representations for Data-Efficient Reinforcement Learning

Max Schwarzer^{*1,2}, **Nitarshan Rajkumar**^{*1,2}, **Michael Noukhovitch**^{1,2}, **Ankesh Anand**^{1,2}
Laurent Charlin^{1,3}, **Devon Hjelm**^{1,4}, **Philip Bachman**^{1,4}, **Aaron Courville**^{1,2}
^{1}Mila, ^{2}Université de Montréal, ^{3}HEC Montréal, ^{4}Microsoft Research

Abstract

Data efficiency poses a major challenge for deep reinforcement learning. We approach this issue from the perspective of self-supervised representation learning, leveraging reward-free exploratory data to pretrain encoder networks. We employ a novel combination of latent dynamics modelling and goal-reaching objectives, which exploit the inherent structure of data in reinforcement learning. We demonstrate that our method scales well with network capacity and pretraining data. When evaluated on the Atari 100k data-efficiency benchmark, our approach significantly outperforms previous methods combining unsupervised pretraining with task-specific finetuning, and approaches human-level performance.

1 Introduction

In deep reinforcement learning (RL), it is standard to train networks *tabula rasa* from random initializations, using only value learning based on task-specific rewards as supervision. Model-free RL algorithms which follow this approach typically suffer from severe overfitting (Zhang et al., 2018) and poor sample efficiency compared to humans (Tsividis et al., 2017).

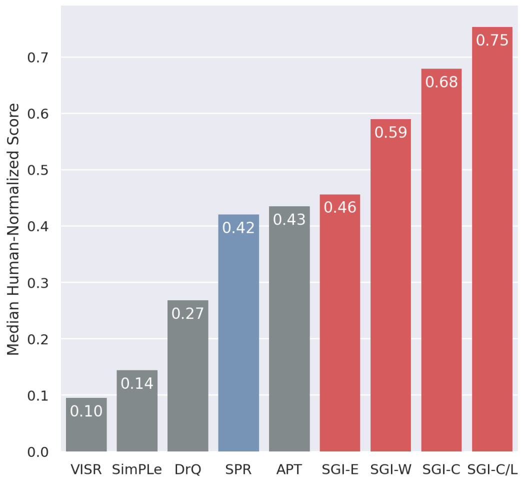

Figure 1: Median Human-Normalized Scores over 26 games in the Atari-100k data-efficiency benchmark. Our proposed SGI methods (red) use reward-free data for pretraining.

Unsupervised pretraining can be used to learn a strong prior (Erhan et al., 2010), and in RL can lead to better sample-efficiency when reward-based feedback becomes available. Particularly in real-world settings, unsupervised RL pretraining brings obvious benefits when the reward function is difficult to define or evaluate during training, such as when it depends on human feedback (Christiano et al., 2017).

One approach to pretraining is to collect massive amounts of exploratory data (Hansen et al., 2020; Liu & Abbeel, 2021), but this is impractical for real-world systems which impose physical limits on interaction: agents cannot be run faster than real-time, and only a limited number can be run in parallel. Furthermore, in safety-critical domains like autonomous driving where an agent is capable of causing serious harm, exploratory policies would require significant human oversight and intervention. It is thus important to develop algorithms that can accelerate learning but are pretrained with practical quantities of reward-free data.

Self-supervised learning has emerged as a promising approach for pretraining on unlabelled data and then fine-tuning on downstream tasks with limited labelled data, in settings such as computer vision (Grill et al., 2020; Chen et al., 2020a) and NLP (Devlin et al., 2019; Brown et al., 2020). We draw inspiration from this pretrain-then-finetune paradigm and apply it to the context of RL – our contributions are summarized below:

*{max.schwarzer, nitarshan}@mila.quebec

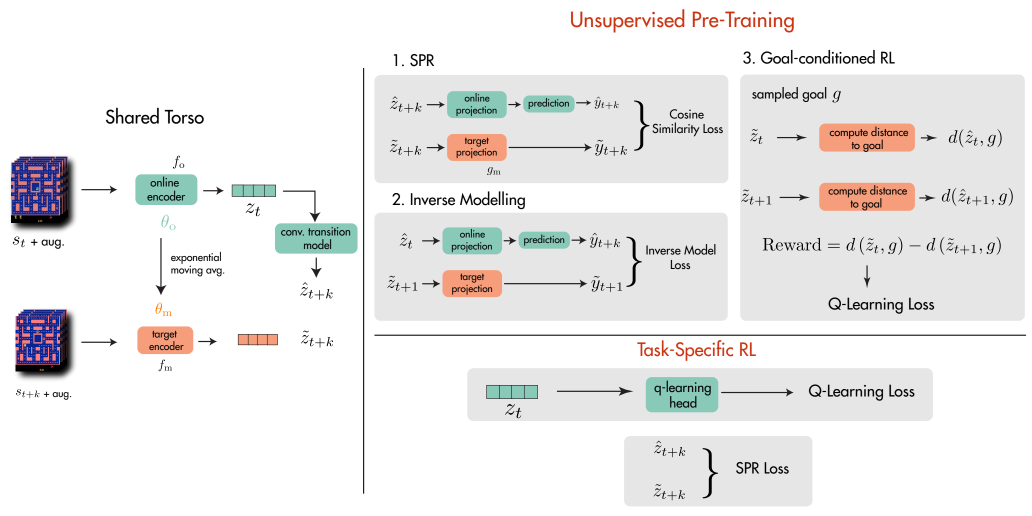

Figure 2: A schematic diagram showing our two stage pretraining-then-finetune method. All unsupervised training losses and task-specific RL use the shared torso on the left.

RL-aligned representation learning objectives We formulate a novel representation learning task by combining latent dynamics modeling using SPR (Schwarzer et al., 2021), unsupervised goal-conditioned reinforcement learning, and inverse dynamics modeling. We refer to our method as **SGI** (**S**PR, **G**oal-conditioned RL and **I**nverse modeling), and provide a visual overview of it in Figure 2.

Significant advances for pre-training data-efficiency in visual domains SGI learns representations that, even without further fine-tuning, significantly advance performance on the Atari 100k data-efficiency benchmark (Kaiser et al., 2019), as shown in Figure 1. These results surpass those from exploration-based approaches which require orders of magnitude more unsupervised data.

Scalability with data quality and model size SGI surpasses the performance of prior work even when trained with limited data from weak or even entirely random behavioral policies. When increasing the amount of data and the quality of the behavioural policy, SGI's performance scales straightforwardly. Additionally, where larger models fail to provide any benefit in online RL, we find that using them for unsupervised pretraining with SGI substantially improves performance.

2 Related Work

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

SGI draws from this tradition, but incorporates objectives specifically designed to exploit additional structure inherent to the interactive data used in reinforcement learning.

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

SGI extends SPR to offline pretraining, and demonstrates that this leads to further improvements to performance in data-efficient RL.

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

Many of these methods are used to pretrain agents that are later adapted to specific reinforcement learning tasks. However, SGI differs in that it can be used offline and is agnostic to the offline data-collection strategy. As such, if operating in a setting where offline data is not available, one could use one of the methods above either to generate a dataset for SGI or in conjunction with SGI in an online exploratory setting.

Representation Learning for RL Recent advances in representation learning in computer vision have spurred similar growth in methods aimed specifically at improving performance in RL. We refer the reader to Lesort et al. (2018) for a review of earlier methods, including inverse dynamics modeling which is used in SGI. More recently, research has focused on leveraging latent-space dynamics modeling as an auxiliary task. Gelada et al. (2019) propose a simple next-step prediction task, coupled with reward prediction, but found it to be prone to latent space collapse and needed to use an auxiliary reconstruction loss in experiments on Atari. Guo et al. (2020) use a pair of networks for both forward and backward prediction, and show improved performance in extremely large-data multi-task settings. Mazoure et al. (2020) use a temporal contrastive objective for representation learning, and show improvement on the continual RL setting of Procgen (Cobbe et al., 2020). Concurrently, SPR (Schwarzer et al., 2021) proposed a multi-step latent prediction task with similarities to BYOL (Grill et al., 2020).

The works most similar to ours, Anand et al. (2019) and Stooke et al. (2020), both propose to use reward-free temporal-contrastive methods to pretrain representations. Anand et al. (2019) show that representations from encoders trained with ST-DIM contain a great deal of information about environment states, but they do not examine whether or not representations learned via their method are, in fact, useful for reinforcement learning. Meanwhile, Stooke et al. (2020) focus solely on the large-data regime and find only minor improvements in performance compared to standard baselines. Additionally, they only evaluate their method on 8 Atari games, limiting the degree to which their performance can be compared to other algorithms.

Offline RL SGI's use of a goal-conditioned reinforcement learning objective on offline data connects it to work in offline (or *batch*) RL (see for example Agarwal et al., 2020; Kumar et al., 2020), in which the agent is trained on an entirely fixed dataset without any environment interactions. However, unlike these methods, SGI does not assume that its offline data includes rewards, or that the data collection agent was solving the same task.

3 Representation Learning Objectives

Self-supervised methods require careful design of pretext tasks to exploit the inherent structure present in unlabelled data. We propose to combine three representation learning tasks suitable for the RL setting, designed to leverage the particular agent-centric and temporal nature of MDPs. We provide an overview of our objectives below; detailed pseudocode is available in Appendix C.

As the core of our algorithm, we employ SPR (Schwarzer et al., 2021), which currently achieves state-of-the-art performance on the data-efficient Atari 100k benchmark when combined with a Rainbow-based DQN. Based on an intuition that the success of SPR is partially due to the interplay between the SPR and RL objectives, we propose to replace the standard RL task with a self-supervised goal-conditioned RL objective, optimized entirely off-policy. Finally, as a shield against possible representational collapse, we also employ an *inverse modeling* task (Lesort et al., 2018), in which we learn a model of the distribution $p(a_t|s_t, s_{t+1})$.

3.1 Self-Predictive Representations

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

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

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

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

However, like BYOL (Grill et al., 2020), SPR is nominally vulnerable to collapse, and in Schwarzer et al. (2021) this risk was mitigated by the presence of a second objective. Moreover, while objectives such as BYOL and SimCLR are capable all-purpose representation learning algorithms, they depend on heavy data augmentation (see ablations in Grill et al., 2020; Chen et al., 2020a), which has thus far not been found promising in DRL (Kostrikov et al., 2021).

3.2 Goal-Conditioned Reinforcement Learning

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

We select goals using a scheme based on hindsight experience replay (Andrychowicz et al., 2017) combined with noise, seeking to generate goal vectors that are both semantically meaningful and highly diverse; using purely random goal vectors would likely violate the former goal, while pure hindsight experience replay might violate the latter. We generate goals in a three-stage process: a goal g for state s_t is initially chosen to be the target representation of a state sampled uniformly from the near future, $g \leftarrow \tilde{z}_{t+i}, i \sim \text{Uniform}(50)$, before being combined with a normalized vector of isotropic Gaussian noise n as $g \leftarrow \alpha n + (1-\alpha)g$, where $\alpha \sim \text{Uniform}(0, 0.5)$. Finally, we exchange goal vectors between states in the minibatch with probability 0.2, to ensure that some goals correspond to states reached in entirely different trajectories.

3.2.1 Goal-Conditioned Rewards

In defining our synthetic goal-conditioned rewards, we take inspiration from potential-based reward shaping (Ng et al., 1999). Using the target representations $\tilde{z}_t \triangleq f_m(s_t)$ and $\tilde{z}_{t+1} \triangleq f_m(s_{t+1})$, we define the reward as follows:

$$R(\tilde{z}_t, \tilde{z}_{t+1}, g) = d(\tilde{z}_t, g) - d(\tilde{z}_{t+1}, g) \tag{2}$$

$$d(\tilde{z}_t, g) = \exp\left(2\frac{\tilde{z}_t \cdot g}{||\tilde{z}||_2 \cdot ||g||_2} - 2\right). \tag{3}$$

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

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

$$\text{FiLM}(F_c|\gamma_c, \beta_c) = \gamma_c F_c + \beta_c \tag{4}$$

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

3.3 Inverse Dynamics Modeling

We propose to use an inverse dynamics modeling task, in which the model is trained to predict a_t from s_t and s_{t+1}. This has a long history in representation learning for RL (Lesort et al., 2018; Hansen et al., 2021). Because this is a classification task (in discrete control) or regression task (continuous control), it is naturally not prone to representational collapse, which may complement and stabilize our other objectives. We directly integrate inverse modeling into the rollout structure of SPR, modeling $p(a_{t+k}|\hat{y}_{t+k}, \tilde{y}_{t+k+1})$ for each $k \in (0, \dots, K{-}1)$, using a two-layer MLP trained by cross-entropy.

4 Experimental Details

We base our work on the code released for SPR (Schwarzer et al., 2021), which in turn is based on rlpyt (Stooke & Abbeel, 2019), and makes use of NumPy (Harris et al., 2020) and PyTorch (Paszke et al., 2019). Implementation details are provided in Appendix F

4.1 Environment & Evaluation

We focus our experimentation on the Atari 100k benchmark introduced by Kaiser et al. (2019), in which agents are allowed only 100k interactions with their environment[1]. This is roughly equivalent to the two hours human testers were given to learn these games by Mnih et al. (2015), providing a baseline of human sample-efficiency. In this setting, agents are typically evaluated by *human-normalized score*, defined as $\frac{agent\_score - random\_score}{human\_score - random\_score}$. We report both mean and median human-normalized scores across games, as well as on how many games the agent achieves super-human performance. We evaluate agents by averaging scores over 100 trajectories at the end of training.

4.2 Pretraining Dataset

SGI can be applied to any offline dataset. To assess how SGI's performance depends on its pretraining data, we create several pretraining datasets varying in both quantity and quality. While it would be ideal to use data gathered by an usupervised method such as APT (Liu & Abbeel, 2021) or VISR (Hansen et al., 2020), neither these methods nor their results datasets have been made publicly available at this time. Instead, we propose to use the DQN Replay dataset (Agarwal et al., 2020), which contains the experience gathered during the training of standard DQN agents (Mnih et al., 2015) for 50M steps (across all 57 games and five different random seeds). This dataset, besides being publicly available, has the advantage of allowing us to easily vary both the *quality* and *quantity* of the data used to train SGI. We thus propose the follow datasets:

- **Random** Finally, to assess SGI's performance near the lower limit of data quality, we use a random policy to gather a dataset of 6M transitions for each game. To encourage the agent to venture further from the starting state, we execute each action for a random number of steps sampled from a Geometric($\frac{1}{3}$) distribution.

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

- **Exploratory Data** To investigate performance with exploratory data, we employ the algorithm proposed by Burda et al. (2018), specifically the **IDF** (inverse dynamics) variant. We log the first 6M steps from an agent trained in each game.
- **Weak Agent** By selecting the first 1M steps for each agent (a total of 5M steps are available, across all random seeds), we are able to create a dataset gathered by an agent substantially worse than that learned by APT or VISR during pretraining, with a median human-normalized score of roughly 0.031 (Castro et al., 2018). Although this is comparable to that of the **IDF** exploration policy above, this data has above-random performance on more games (22 vs 18), suggesting that it is encountering more task-relevant transitions.
- **Competent Agent** To assess SGI when using data gathered from more competent policies, we concatenate multiple checkpoints evenly spread throughout training, creating a small dataset using 3M steps of data and a large dataset using 6M steps.

Note that even our largest dataset is quite small compared to the amounts of data gathered by unsupervised exploration methods (see the "Data" column in Table 4.4). We show the performance of the non-random data collection policies in Table 1 (note that a fully-random policy has a score of **0** by definition).

4.3 SGI Variants

We refer to agents by the model architecture and dataset type used: Agents pretrained on the Random, Exploratory and Weak datasets are denoted by **SGI-W**, **SGI-E** and **SGI-R** respectively. For the Competent dataset, we additionally vary the architecture used - we denote variants of SGI that employ the standard Mnih et al. (2015) encoder as **SGI-C/S** (for small), variants that employ our standard ResNet as simply **SGI-C** and variants that employ the enlarged ResNet as **SGI-C/L** (for large)[2]. For **SGI-C/L** we also double the amount of pretraining data used. We refer to agents without any pretraining as **SGI-None**.

4.4 Training

We optimize our three representation learning objectives jointly during pretraining, while during fine-tuning we optimize only the reinforcement learning and SPR losses:

$$\mathcal{L}_\theta^{\text{Pretrain}} = \mathcal{L}_\theta^{\text{GCRL}} + \lambda^{\text{SPR}}\mathcal{L}_\theta^{\text{SPR}} + \lambda^{\text{IM}}\mathcal{L}_\theta^{\text{IM}} \quad (5)$$

$$\mathcal{L}_\theta^{\text{Finetune}} = \mathcal{L}_\theta^{\text{RL}} + \lambda^{\text{SPR}}\mathcal{L}_\theta^{\text{SPR}} \quad (6)$$

We set $\lambda^{\text{SPR}} = 2$ and $\lambda^{\text{IM}} = 1$ during pretraining. Unless otherwise noted, all settings match SPR during fine-tuning, including batch size, replay ratio, target network update period, and λ^{SPR}.

We use a batch size of 256 during pre-training to maximize throughput, and update both the SPR and goal-conditioned RL target network target networks with an exponential moving average with $\tau = 0.99$. We pre-train for a number of gradient steps equivalent to 10 epochs over 6M samples, no matter the amount of data used. Due to the cost of pretraining, we pre-train a single encoder per game for each configuration tested. However, we use 10 random seeds at fine-tuning time, allowing us to average over variance due to exploration and data order. Finally, we reduce fine-tuning learning rates for

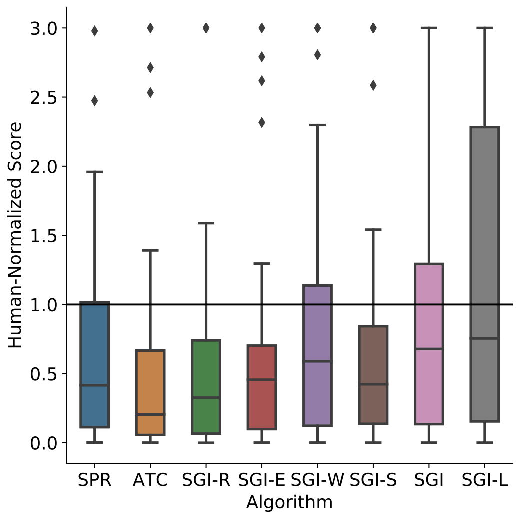

Figure 3: A boxplot of human-normalized scores across the 26 Atari100k games after fine-tuning. Each datapoint is the human-normalized score on a game, averaged across 10 random seeds. Values are clipped to a maximum of 3 for visual clarity.

[2]See Appendix F for details on these networks

pretrained encoders and dynamics models by a factor of 100, and by a factor of 3 for other pretrained weights. We find this crucial to SGI's performance, and discuss it in detail in Section 5.

ATC To make a controlled comparison, we adapt code for ATC (Stooke et al., 2020) to work with SGI's larger network architectures. Details are available in Appendix H

5 Results and Discussion

We find that pre-training with SGI robustly improves performance. When using a dataset of 6M pretraining steps gathered by a competent agent, **SGI-C/L** achieves a median human-normalized score of 0.753, an improvement of 81% over SPR. With a dataset of 3M steps, **SGI-C** achieves a median human-normalized score of 0.679, an improvement of 64% over SPR. SGI-C also greatly outperforms ATC, a comparable method for offline pre-training, in a controlled comparison. Meanwhile, SGI-E, which is directly comparable with works in the benchmark proposed by Hansen et al. (2020), achieves median performance robustly higher than competing methods in this setting, APT (Liu & Abbeel, 2021) and VISR (Hansen et al., 2020). We present aggregate measures in Table 1, and full per-game results in Appendix D. These results show that the strongest SGI setting outperforms all comparable methods except the recently proposed CPT (Campos et al., 2021), which uses several orders of magnitude more pretraining data than SGI.

Table 1: Human-normalized score across the 26 Atari100k games (Kaiser et al., 2019). Methods use varying amounts of pre-training data (right column) but all use 100k environment steps of online interaction. SGI variants are defined in Section 4.3. Full per-game results are presented in Appendix D.

Method	Median	Mean	>H	>0	Data
SimPLe	0.144	0.443	2	**26**	0
DER	0.161	0.285	2	**26**	0
DrQ	0.268	0.357	2	24	0
SPR	0.415	0.704	7	**26**	0
VISR	0.095	1.281	7	21	250M
APT	0.435	0.621	7	**26**	250M
Data-collection					
IDF	0.039	0.042	0	18	6M
DQN	0.031	0.026	0	22	1M
DQN	1.008	2.675	13	26	50M
Offline Pre-Training					
ATC (ours)	0.204	0.780	5	**26**	3M
SGI-R	0.326	0.888	5	**26**	6M
SGI-E	0.456	0.838	6	**26**	6M
SGI-W	0.589	1.144	8	**26**	5M
SGI-C/S	0.423	0.914	8	**26**	3M
SGI-C	0.679	1.149	**9**	**26**	3M
SGI-C/L	**0.753**	**1.598**	**9**	**26**	6M

Data quality matters Although SGI can in principle be used with any offline dataset, not all datasets are made equal. Even near the lower bound of data quality where all actions are selected randomly, pretraining with SGI (**SGI-R**) still provides some benefit, improving over an otherwise-identical randomly-initialized agent (**SGI-None**) on 16 out of 26 games, and increasing its mean score by 57%. In settings with more realistic-quality data, we find that data from an exploratory policy (**SGI-E**) substantially improves median performance, exceeding that of APT (Liu & Abbeel, 2021), which used 40 times more pre-training data. Using data from a poorly-performing non-exploratory policy (**SGI-W**) has an even larger effect, allowing a 72% improvement over median human-normalized score compared to training the same model from scratch, while data from a good policy (**SGI-C**) allows a 98% improvement.

The large gap between SGI-W and SGI-E is notable. The similar mean and median scores of their data-collection policies suggests that it is the *degeneracy* of the pre-training data is critical. The IDF agents used to generate data for SGI-E have *sub-random* performance on twice as many games as the 1M-step DQN agents, suggesting a broader failure to engage with task-relevant behaviors.

Combining SSL objectives improves performance We test all possible combinations of our three SSL objectives, denoted by combinations of the letters S, G and I to indicate which objectives they employ. We present results in Table 2, where we see that performance monotonically increases as more objectives are used, with inverse dynamics modeling combined with either of the other objectives performing respectably well. We note that including inverse modeling appears to be critical to avoiding representational collapse, with **S**, **G** and **S+G** all exhibiting poor performance and at least partial collapse; more details are shown in Appendix B.

Representation learning requires larger networks The three-layer network introduced by Mnih et al. has become a fixture of deep reinforcement learning, and has been employed by previous works examining pretraining in this field (e.g., Liu & Abbeel, 2021; Stooke et al., 2020). However, we find that representational pretraining with this network (**SGI-C/S**) provides only minor benefits, with a median score identical to SPR's and a mean only 30% higher than SPR. With larger networks (**SGI-C** and **SGI-C/L**), representational pretraining via SGI performs qualitatively differently, offering large improvements in performance – despite that these larger networks harm performance on the Atari100k task without pretraining (**SGI-None** in Table 2, has a median score 17% below regular SPR).

This finding is consistent with recent work in unsupervised representation learning for classification, which has observed that unsupervised pretraining benefits disproportionately from larger networks (Chen et al., 2020a). Although large networks can perform well in deep reinforcement learning without pretraining in the very large data regime (Schrittwieser et al., 2021; Espeholt et al., 2018), SGI provides a simple way to extend these benefits to data-efficient RL.

Table 2: Human-normalized score on Atari for various controlled comparisons to SGI-C.

Method	Mdn	Mean	>H	> SPR
Ablated Pre-training				
None	0.343	0.565	3	10
S	0.009	-0.054	0	1
G	0.060	0.181	1	1
I	0.411	0.943	7	18
S+G	0.029	0.098	0	1
G+I	0.512	1.004	**9**	**20**
S+I	0.629	0.978	8	19
Ablated Fine-Tuning				
Naive	0.429	0.845	8	14
Frozen	0.499	0.971	8	15
No SPR	0.452	1.114	8	14
All Losses	0.397	1.011	8	17
SGI-C	**0.679**	**1.149**	**9**	**20**

Naively fine-tuning ruins pretrained representations We find that even high-quality pretrained representations provide little benefit if they are not used properly in fine-tuning. Using pretrained weights as an initialization while allowing them to change freely during fine-tuning leads to performance only somewhat better than initializing from scratch (denoted as **Naive SGI** in Table 2). Surprisingly, entirely freezing the SGI pre-trained encoder during fine-tuning (denoted as **Frozen SGI**) leads to better performance than both the naive baseline and SPR. SGI's approach, using significantly reduced learning rates for pre-trained parameters during fine-tuning, leads to superior performance (compare **SGI**, **Frozen SGI** and **Naive SGI** in Table 2). Interestingly, we also find that SGI's use of SPR at pretraining time substantially improves performance, even with these tweaks (compare **SGI** and **No SPR** in Table 2) and we hypothesize that using an auxiliary self-supervised learning loss may stabilize representational adaptation. However, we found no benefit to including the other self-supervised losses used by SGI during pre-training, suggesting that this effect may have sharp diminishing returns (**All Losses** in Table 2).

To explain these results, we hypothesize that representations learned by SGI are being disrupted by gradients early in fine-tuning, in a phenomenon analogous to catastrophic forgetting (Zenke et al., 2017; Hadsell et al., 2020) seen in continual learning. As representations do not automatically generalize between value functions at different stages of training (Dabney et al., 2021), allowing the encoder to strongly adapt to early value functions may make it *worse* at modeling later value functions, compared to the neutral initialization provided by SGI.

6 Conclusion

We presented SGI, a fully self-supervised (reward-free) approach to representation learning for reinforcement learning, which uses a combination of pretraining objectives to encourage the agent to learn multiple aspects of environment dynamics. We demonstrated that SGI enables significant improvements on the Atari 100k data-efficiency benchmark, especially in comparison to unsupervised exploration approaches which require orders of magnitude more pretraining data. We additionally investigated the various components of SGI, finding that it scales robustly with higher-quality behavioural data and larger models, that all three of the self-supervised objectives contribute to the success of this approach, and that careful reduction of fine-tuning learning rates is critical to optimal performance.

We hope that these findings demonstrate the value of explicit representation learning as an approach to moving away from tabula-rasa learning and towards imbuing agents with useful priors about the environments they will inhabit.

REFERENCES

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *ICML*, 2020.

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

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

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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

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

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

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

Víctor Campos, Pablo Sprechmann, Steven Stenberg Hansen, Andre Barreto, Charles Blundell, Alex Vitvitskyi, Steven Kapturowski, and Adria Puigdomenech Badia. Coverage as a principle for discovering transferable behavior in reinforcement learning, 2021. URL `https://openreview.net/forum?id=INhwJdJtxn6`.

Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G. Bellemare. Dopamine: A Research Framework for Deep Reinforcement Learning. 2018. URL `http://arxiv.org/abs/1812.06110`.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *ICML*, 2020a.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. In *NeurIPS*, 2020b.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30, pp. 4299–4307, 2017.

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

Will Dabney, André Barreto, Mark Rowland, Robert Dadashi, John Quan, Marc G Bellemare, and David Silver. The value-improvement path: Towards better representations for reinforcement learning. 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *ACL*, 2019.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? 11(19):625–660, 2010.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. *CoRR*, abs/1802.01561, 2018. URL `http://arxiv.org/abs/1802.01561`.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2018.

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

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.

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.

Raia Hadsell, Dushyant Rao, Andrei A. Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 2020. doi: https://doi.org/10.1016/j.tics.2020.09.004. URL `http://www.sciencedirect.com/science/article/pii/S1364661320302199`.

Nicklas Hansen, Yu Sun, Pieter Abbeel, Alexei A. Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. In *ICLR*, 2021.

Steven Hansen, Will Dabney, Andre Barreto, David Warde-Farley, Tom Van de Wiele, and Volodymyr Mnih. Fast task inference with variational intrinsic successor features. In *ICLR*, 2020.

Charles R. Harris, K. Jarrod Millman, St'efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fern'andez del R'ıo, Mark Wiebe, Pearu Peterson, Pierre G'erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

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

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

Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019.

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

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

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

Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model based reinforcement learning for atari. In *ICLR*, 2019.

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

Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *ICLR*, 2021.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv e-prints*, pp. arXiv–2006, 2020.

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

Hao Liu and Pieter Abbeel. Unsupervised active pre-training for reinforcement learning, 2021. URL `https://openreview.net/forum?id=cvNYovr16SB`.

Bogdan Mazoure, Remi Tachet des Combes, Thang Doan, Philip Bachman, and R Devon Hjelm. Deep reinforcement and infomax learning. In *NeurIPS*, 2020.

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

Andrew Y Ng, Daishi Harada, and Stuart J Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, 1999.

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

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.

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

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

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

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. *Nature*, 2021.

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.

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

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2019.

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

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

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning, 2020.

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

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114, 2019.

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

Pedro Tsividis, Thomas Pouncy, Jaqueline L. Xu, Joshua B. Tenenbaum, and Samuel J. Gershman. Human learning in atari. In *AAAI Spring Symposia*, 2017.

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

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

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere, 2020.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *ICML*, 2016.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017.

Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.

A Reinforcement Learning Background

We focus on reinforcement learning in a Markov decision process (MDP). An agent interacts with an environment, taking actions a and observing states s and rewards r. Interactions are broken up into *episodes*, series of states, actions and rewards that ultimately terminate. We denote the t-th state, action and reward as s_t, a_t and r_t respectively, following the convention that r_t is the reward received by the agent after taking action a_t in state s_t. In the control problem examined here, the agent seeks to maximize a discounted sum of rewards $G_t = \sum_{i \geq t} \gamma^i r_i$, where $\gamma \in [0, 1]$ is a discount factor balancing current and future rewards (Sutton & Barto, 2018).

Deep Q-Learning One common approach to this problem is to estimate $Q(s_t, a) = \mathbf{E}_{\pi^*}[G_t | a_t = a]$. Assuming that Q is known exactly, the problem of acting optimally is reduced to finding $\max_a Q(s_t, a)$ in each state s_t the agent encounters, which is trivial in environments that possess only a small number of possible actions. In practice, the true Q can be iteratively approximated by a parameterized Q_θ with a semi-gradient method, minimizing

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

where Q_ξ denotes an older version of Q_θ. This method has proven extremely successful when used with deep learning, a setting referred to as Deep Q-Networks (DQN) (Mnih et al., 2015), and more broadly is a common class of deep reinforcement learning (DRL) algorithms.

A number of variants of DQN have been proposed, including those that predict full distributions of future rewards (Bellemare et al., 2017), modifications to the $\max$ operation to reduce value overestimation (Van Hasselt et al., 2016), and architectural modifications to how Q_θ is predicted (Wang et al., 2016). We employ a somewhat modified (Schwarzer et al., 2021) version of Rainbow (Hessel et al., 2018), an algorithm that combines many of these innovations.

B Representational Collapse

We plot the average cosine similarity between representations y_t of different states for several pretraining methods in Fig. 4, using our ResNet encoder. We observe that pretraining with SPR (**S**) alone leads to almost-total representational collapse, with average cosine similarity nearly 1, indicating that representations of different states are nearly identical. All configurations that include inverse dynamics modeling avoid representational collapse, as does ATC, whose contrastive loss implicitly optimizes for representational diversity (Wang & Isola, 2020).

Intriguingly, we observe that increased representational diversity does not automatically improve performance during fine-tuning. For example, SGI's representations are less diverse than those from several other methods, including **ATC**, **G+I** and **I**, although SGI outperforms all of these methods by various margins in fine-tuning. Finally, we observe that adding SPR to a configuration appears to consistently pull representations towards collapse (see S+I and I, S+G and G, and SGI and G+I); how SPR does this while improving performance is an interesting question for future work.

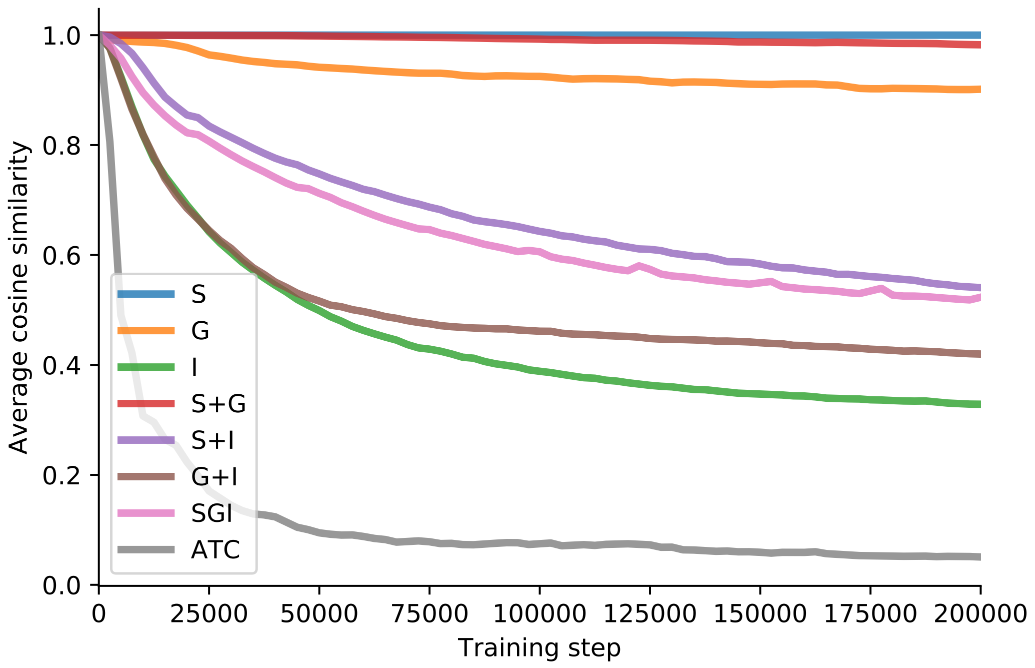

Figure 4: A plot of the average cosine similarity between representations of different states over the course of pre-training, using our ResNet encoder and several different pretraining algorithms, averaged across the 26 Atari games under consideration. An average cosine similarity of 1 indicates that representations are entirely identical, while 0 indicates perfect dissimilarity.

C Pseudocode

Algorithm 1: Pre-Training with SGI

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

D Full Results on Atari100k

We report full scores for SGI agents across all 26 games in Table 3. We do not reproduce the per-game scores for APT and VISR provided by Liu & Abbeel (2021), as we believe that the scores in the currently-available version of their paper may contain errors.[3]

Table 3: Mean return per episode for the 26 Atari100k games (Kaiser et al., 2019) after 100k steps. Agents are evaluated at the end of training, and scores for all methods are averaged over 10 random seeds. We reproduce scores for SPR from Schwarzer et al. (2021), whereas ATC scores are from our implementation.

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

E Transferring Representations between Games

One advantage of pretraining representations is the possibility of representations being useful across games. Intuitively, we expect better transfer between similar games so we chose five "cliques" of games with similar semantics and visual elements. The cliques are shown in Table 6. We pretrain on a dataset of 750k frames from each game in a clique (i.e. 3M frames for a clique of 4) and fine-tune on a single game. To show whether pretraining on other games is beneficial, we compare to a baseline of pretraining on just the 750k frames from the single Atari 100k game we use for fine-tuning.

Our results in Table 7 show that pretraining with the extra frames from the clique games is mostly unhelpful to fine-tune performance. Only Kangaroo shows a modest improvement, a few games show no difference in performance, and most games show a decrease in performance when pretraining with other games. We believe that Atari may not be as suitable to transferring representations as other domains, and previous work using Atari to learn transferable representations has also had negative results (Stooke et al., 2020). Though game semantics can be similar, we note that even small differences in rule sets and visual cues can make transfer difficult.

[3] In particular, we observed that multiple methods are claimed to have scores below -21 on Pong, which is impossible with standard settings.

Table 4: Mean return per episode for the 26 Atari100k games (Kaiser et al., 2019) after 100k steps for versions of SGI with modified fine-tuning, as discussed in Section 5. Agents are evaluated at the end of training, and scores for all methods are averaged over 10 random seeds. We reproduce scores for SPR from Schwarzer et al. (2021).

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

F Implementation

In addition to the standard three-layer CNN encoder introduced by Mnih et al. (2015), we experiment with larger residual networks (He et al., 2016). We use the design proposed by Espeholt et al. (2018) as a starting point, while still adopting innovations used in more modern architectures such as EfficientNets (Tan & Le, 2019) and MobileNetv2 (Sandler et al., 2018). In particular, we use inverted residual blocks with an expansion ratio of 2, and batch normalization (Ioffe & Szegedy, 2015) after each convolutional layer. We use three groups of three residual blocks with 32, 64 and 64 channels each, downscaling by a factor of three in the first group and two in each successive group. This yields a final representation of shape $64 \times 7 \times 7$ when applied to 84×84-dimensional Atari frames, identical to that of the standard CNN encoder. In our scaling experiment with a larger network, we increase to five blocks per group, with 48, 96 and 96 channels in each group, as well as using a larger expansion ratio of 4, producing a representation of shape $96 \times 7 \times 7$. This enlargement increases the number of parameters by roughly a factor of 5. Finally, our DQN head has 512 hidden units, as opposed to 256 in SPR.

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

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

Table 5: Mean return per episode for the 26 Atari100k games (Kaiser et al., 2019) after 100k steps for various combinations of SGI's pretraining objectives, as discussed in Section 5. Agents are evaluated at the end of training, and scores for all methods are averaged over 10 random seeds.

	Random	Human	None	S	G	I	G+I	S+G	S+I	SGI
Alien	227.8	7127.7	835.9	278.7	964.3	1161.6	571.2	1172.3	**1203.0**	1101.7
Amidar	5.8	1719.5	107.6	37.8	54.8	198.1	58.0	**210.5**	175.4	168.2
Assault	222.4	742.0	657.7	517.9	512.3	868.1	567.2	813.5	820.3	**905.1**
Asterix	210.0	8503.3	832.9	292.6	416.1	475.6	431.8	506.3	648.5	**835.6**
Bank Heist	14.2	753.1	**613.2**	3.1	115.2	357.6	57.2	423.3	547.5	608.4
Battle Zone	2360.0	37187.5	13490.0	4665.0	3336.0	14807.0	3249.0	12528.0	**15491.0**	13170.0
Boxing	0.1	12.1	6.6	-21.8	12.5	40.1	-0.4	**42.9**	38.3	36.9
Breakout	1.7	30.5	12.1	0.9	2.1	24.1	3.2	41.0	41.6	**42.8**
Chopper Command	811.0	7387.8	1085.2	799.7	813.1	973.1	923.7	1097.2	978.3	**1404.0**
Crazy Climber	10780.5	35829.4	19707.6	243.3	17760.3	51203.9	581.0	66228.5	83995.4	**88561.2**
Demon Attack	152.1	1971.0	778.8	668.9	316.9	**1524.6**	756.4	1008.4	1286.6	968.1
Freeway	0.0	29.6	17.2	15.2	17.7	2.6	19.3	**30.5**	29.1	30.0
Frostbite	65.2	4334.7	**1475.8**	427.2	523.3	395.0	215.4	530.5	463.3	741.3
Gopher	257.6	2412.5	438.2	60.7	129.0	**1966.1**	99.0	1747.4	1778.7	1660.4
Hero	1027.0	30826.4	6472.0	2381.2	3590.2	7177.6	3998.7	**8251.2**	7366.2	7474.0
Jamesbond	29.0	302.8	157.4	41.8	236.0	373.1	183.6	365.6	**378.4**	366.4
Kangaroo	52.0	3035.0	**3802.8**	129.8	401.6	1041.4	222.6	830.8	760.2	2172.8
Krull	1598.0	2665.5	3954.0	720.1	1241.4	**5859.8**	1582.4	5778.8	5808.6	5734.0
Kung Fu Master	258.5	22736.3	7929.4	79.7	453.7	16914.7	686.2	**17825.1**	14681.9	16137.8
Ms Pacman	307.3	6951.6	990.2	418.7	528.5	1620.1	293.3	**1847.1**	1715.9	1520.0
Pong	-20.7	14.6	-4.4	-20.9	-20.4	-3.0	-21.0	0.9	1.7	**7.6**
Private Eye	24.9	69571.3	62.8	-20.7	89.4	**100.0**	12.7	98.2	**100.0**	90.0
Qbert	163.9	13455.0	**720.0**	201.0	277.4	706.5	215.2	650.5	601.9	709.8
Road Runner	11.5	7845.0	5428.4	780.3	5592.9	17698.4	2617.8	18229.4	17443.5	**18370.2**
Seaquest	68.4	42054.7	577.8	105.7	193.2	965.3	118.8	**1115.0**	792.1	728.4
Up N Down	533.4	11693.2	46042.6	892.2	4399.7	58142.0	1313.4	52772.9	39771.3	**79228.8**
Median HNS	0.000	1.000	0.343	0.009	0.060	0.411	0.029	0.512	0.629	**0.679**
Mean HNS	0.000	1.000	0.565	-0.054	0.181	0.943	0.098	1.004	0.978	**1.149**
#Games > Human	0	0	3	0	1	7	0	**9**	8	**9**
#Games > SPR	0	19	10	1	1	18	1	**20**	19	**20**

Table 6: Cliques of semantically similar games

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

G Self-Supervised Learning During Fine-Tuning

In addition to SGI's use of SPR during fine-tuning, we experiment with a variant that optimizes only the standard DQN objective during fine-tuning. As we still employ augmentation, this method is roughly equivalent to using DrQ (Kostrikov et al., 2021) with DQN hyperparameters set to match SGI. In this case, we find that pre-training with SGI dramatically improves performance, as training without pre-training is greatly harmed by the omission of SPR (compare **None** and **SPR Only** entries in Table 8.) Notably, SGI without SPR manages to achieve roughly the same mean human-normalized score as SGI with SPR, although removing SPR harms performance on performance on 19 out of 26 games and reduces median normalized score by roughly 33%. We also consider a variant that uses all of SGI's constituent objectives during fine-tuning (**All Losses** in Table 8), but find no benefit to doing so compared to using SPR alone. However, we do not modify the weights of these losses from their values during pre-training, and it is possible that results might be improved by tuning these hyperparameters.

H Experiments with ATC

As ATC (Stooke et al., 2020) was not tested on the Atari100k setting, and as its hyperparameters (including network size and fine-tuning scheme) are very different from those used by SGI, we

Table 7: Mean return per episode for clique games in Atari100k (Kaiser et al., 2019) after 100k steps. Agents are evaluated at the end of training, and scores for all methods are averaged over 10 random seeds. Games in the same clique are placed together.

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

Table 8: Human-normalized score across the 26 Atari100k games (Kaiser et al., 2019) for algorithms using various combinations of self-supervised losses during fine-tuning, with and without SGI pre-training.

Fine-Tuning SSL	Mdn	Mean	>H	> SPR
Without SGI Pre-Training				
None	0.161	0.315	2	1
SPR Only	0.343	0.565	3	10
With SGI Pre-Training				
No SPR	0.452	1.114	8	14
All Losses	0.397	1.011	8	17
SPR Only	**0.679**	**1.149**	**9**	**20**

modify its code[4] to allow it to be fairly compared to SGI. We replace the convolutional encoder with that used by SGI, and use the same optimizer settings, image augmentation, pre-training data, and number of pre-training epochs as in SGI. However, we retain ATC's mini-batch structure (i.e., sampling 32 subsequences of eight consecutive time steps, for a total batch size of 512), as this structure defines the negative samples used by ATC's InfoNCE loss. During fine-tuning, we transfer the ATC projection head to the first layer of the DQN MLP head, as in SPR; we otherwise fine-tune identically to SGI, including using SPR.

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