Pretraining Representations for Data-Efficient Reinforcement Learning

Max Schwarzer*[1,2], **Nitarshan Rajkumar***[1,2], **Michael Noukhovitch**[1,2], **Ankesh Anand**[1,2]
Laurent Charlin[1,3,5], **Devon Hjelm**[1,4], **Philip Bachman**[4], **Aaron Courville**[1,2,5]

[1]Mila, [2]Université de Montréal, [3]HEC Montréal, [4]Microsoft Research, [5]CIFAR

Abstract

Data efficiency is a key challenge for deep reinforcement learning. We address this problem by using unlabeled data to pretrain an encoder which is then finetuned on a small amount of task-specific data. To encourage learning representations which capture diverse aspects of the underlying MDP, we employ a combination of latent dynamics modelling and unsupervised goal-conditioned RL. When limited to 100k steps of interaction on Atari games (equivalent to two hours of human experience), our approach significantly surpasses prior work combining offline representation pretraining with task-specific finetuning, and compares favourably with other pretraining methods that require orders of magnitude more data. Our approach shows particular promise when combined with larger models as well as more diverse, task-aligned observational data – approaching human-level performance and data-efficiency on Atari in our best setting. We provide code associated with this work at https://github.com/mila-iqia/SGI.

1 Introduction

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

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

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

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

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

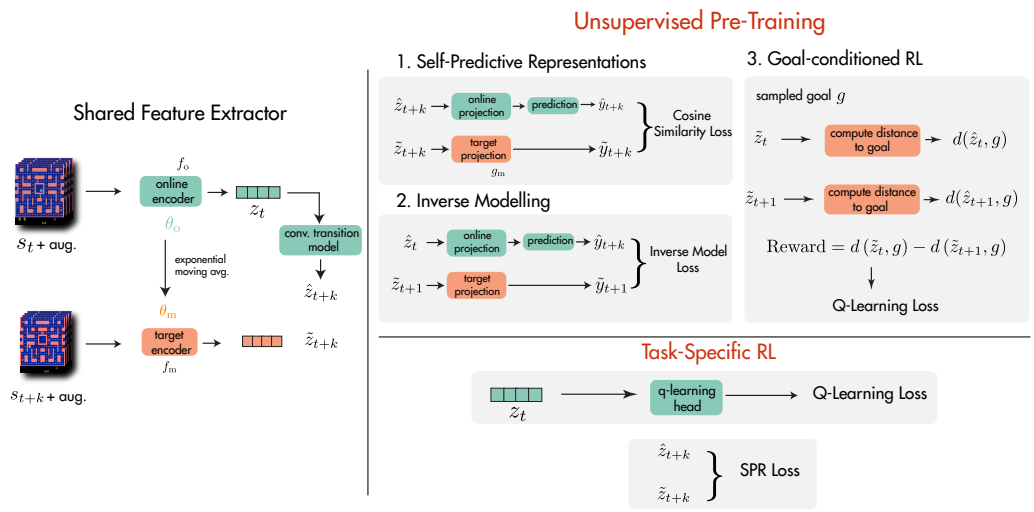


Figure 1: A schematic diagram showing our two stage pretrain-then-finetune method. All unsupervised training losses and task-specific RL use the shared torso on the left.

To this end, we propose to use a combination of self-supervised objectives for representation learning on offline data, requiring orders of magnitude less pretraining data than existing methods, while approaching human-level data-efficiency when finetuned on downstream tasks. We summarize our work below:

RL-aligned representation learning objectives: We propose to pretrain representations using a combination of latent dynamics modeling, unsupervised goal-conditioned reinforcement learning, and inverse dynamics modeling – with the intuition that a collection of such objectives can capture more information about the dynamical and temporal aspects of the environment of interest than any single objective. We refer to our method as **SGI** (**S**PR, **G**oal-conditioned RL and **I**nverse modeling).

Significant advances for data-efficiency on Atari: SGI's combination of objectives performs better than any in isolation and significantly improves performance over previous representation pretraining baselines such as ATC (Stooke et al., 2021). Our results are competitive with exploration-based approaches such as APT or VISR (Liu & Abbeel, 2021; Hansen et al., 2020), which require two to three orders of magnitude more pretraining data and the ability to interact with the environment during training, while SGI can learn with a small offline dataset of exploration data.

Scaling with data quality and model size: SGI's performance also scales with data quality and quantity, increasing as data comes from better-performing or more-exploratory policies. Additionally, we find that SGI's performance scales robustly with model size; while larger models are unstable or bring limited benefits in standard RL, SGI pretraining allows their finetuning performance to significantly exceed that of smaller networks.

We assume familiarity with RL in the following sections (with a brief overview in Appendix A).

2 Representation Learning Objectives

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

Structure in RL goes far beyond forward dynamics, however. We propose to combine multiple representation learning objectives, covering different agent-centric and temporal aspects of the MDP. Based on the intuition that knowledge of an environment is best represented in multiple ways (Hessel et al., 2021; Degris & Modayil, 2012), we expect this to outperform monolithic representation learning methods such as temporal contrastive learning (e.g., Stooke et al., 2021). In deciding which

tasks to use, we consider questions an adequate representation should be able to answer about its environment, including:

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

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

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

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

2.1 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 its base algorithm, Rainbow (Hessel et al., 2018), SPR learns a convolutional encoder, denoted as f_o, which produces representations of states as $z_t = f_o(s_t)$. SPR then uses a *dynamics model* h to recursively estimate the representations of future states, as $\hat{z}_{t+k+1} = h(\hat{z}_{t+k}, a_{t+k})$, beginning from $\hat{z}_t \triangleq z_t$. These representations are projected to a lower-dimensional space by a projection function p_o to produce $\hat{y}_{t+k} \triangleq p_o(\hat{z}_{t+k})$.

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

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

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: $\theta_m = \tau\theta_m + (1-\tau)\theta_o$.

2.2 Goal-Conditioned Reinforcement Learning

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

We select goals using a scheme inspired by hindsight experience replay (Andrychowicz et al., 2017), seeking to generate goal vectors that are both semantically meaningful and highly diverse. As in hindsight experience replay, we begin by sampling a state from another trajectory or the future of the current trajectory. However, we take the additional step of applying stochastic noise to encourage goals to lie somewhat off of the current representation manifold. We provide details in Appendix C.2.

2.3 Inverse Dynamics Modeling

We propose to use an inverse dynamics modeling task (Lesort et al., 2018), in which the model is trained to predict a_t from s_t and s_{t+1}. Because this is a classification task (in discrete control) or

regression task (continuous control), it is naturally not prone to representational collapse, which may complement and stabilize our other objectives. We directly integrate inverse modeling into the rollout structure of SPR, modeling $p(a_{t+k}|\hat{y}_{t+k}, \tilde{y}_{t+k+1})$ for each $k \in (0, \ldots, K-1)$, using a two-layer MLP trained by cross-entropy.

3 Related Work

Data-Efficiency In order to address data efficiency in RL, Kaiser et al. (2019) introduced the Atari 100k benchmark, in which agents are limited to 100,000 steps of environment interaction, and proposed SimPLe, a model-based algorithm that substantially outperformed previous model-free methods. However, 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) combined data augmentation with an auxiliary self-supervised learning objective. SGI employs SPR as one of its objectives in offline pretraining, leading to significant improvements in data-efficiency.

Exploratory pretraining A number of recent works have sought to improve reinforcement learning via the addition of an unsupervised *pretraining* stage prior to finetuning on the target task. One common approach has been to allow the agent a period of fully-unsupervised interaction with the environment, during which the agent is trained to maximize a surrogate exploration-based task such as the diversity of the states it encounters, as in APT (Liu & Abbeel, 2021) or ProtoRL (Yarats et al., 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) and Burda et al. (2018) use the loss of an inverse dynamics model like that used in SGI, while Sekar et al. (2020) uses the disagreement between an ensemble of latent-space dynamics models. Finally, Campos et al. (2021) report strong results based on massive-scale unsupervised pretraining.

Many of these methods are used to pretrain agents that are later adapted to specific reinforcement learning tasks. However, SGI differs in that it can be used offline and is agnostic to how data is collected. As such, if no pre-existing offline data is available, one of the methods above can be used to generate a dataset for SGI; we use Burda et al. (2018) for this in Section 4.2.

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

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

Two works similar to ours, Anand et al. (2019) and Stooke et al. (2021), propose reward-free temporal-contrastive methods to pretrain representations. Anand et al. (2019) show that representations from encoders trained with ST-DIM contain a great deal of information about environment states, but they do not examine whether or not representations learned via their method are, in fact, useful for reinforcement learning. However, Stooke et al. (2021) employ a similar algorithm and find only relatively minor improvements in performance compared to standard baselines in the large-data

regime; our controlled comparisons show that SGI's representations are far better for data-efficiency. Concurrent to our work, FERM (Zhan et al., 2020) propose contrastive pretraining from human demonstrations in a robotics setting. As FERM is quite similar to ATC, we are optimistic that our improvements over ATC in Atari 100k would translate to FERM's setting. Finally, Yang & Nachum (2021) propose a family of algorithms for learning representations for DRL from state observations; we believe that these might be promising in DRL from pixels, but they have not yet been adapted to that setting.

4 Experimental Details

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

4.1 Environment and Evaluation

To address the first challenge, we focus our experimentation on the Atari 100k benchmark introduced by Kaiser et al. (2019), in which agents are allowed only 100k steps of interaction with their environment.[2] This is roughly equivalent to the two hours human testers were given to learn these games by Mnih et al. (2015), providing a baseline of human sample-efficiency.

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

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

4.2 Pretraining Data

The second challenge pertains to pretraining data. Although some prior work on offline representational pretraining has focused on expert-quality data (Stooke et al., 2021), we expect real-world pretraining data to be of greatly varying quality. We thus construct four different pretraining datasets to approximate different data quality scenarios.

- **(R)andom** To assess performance near the lower limit of data quality, we use a random policy to gather a dataset of 6M transitions for each game. To encourage the agent to venture further from the starting state, we execute each action for a random number of steps sampled from a Geometric($\frac{1}{3}$) distribution.
- **(E)xploratory** To emulate slightly better data that covers a larger range of the state space, we use an exploratory policy. Specifically, we employ the **IDF** (inverse dynamics) variant of the algorithm proposed by Burda et al. (2018). We log the first 6M steps from an agent trained in each game. This algorithm achieves better-than-random performance on only 70% of tasks, setting it far below the performance of more modern unsupervised exploration methods.

To create higher-quality datasets, we follow Stooke et al. (2021) and use experience gathered during the training of standard DQN agents (Mnih et al., 2015). We opt to use the publicly-available DQN Replay dataset (Agarwal et al., 2020), which contains data from training for 50M steps (across all 57 games, with five different random seeds). Although we might prefer to use data from recent

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

unsupervised exploration methods such as APT (Liu & Abbeel, 2021), VISR (Hansen et al., 2020), or CPT (Campos et al., 2021), none of these works provide code or datasets, making this impractical. We address using data collected from on-task agents with a behavioural cloning baseline in Section 5, with surprising findings relative to prior work.

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

We compare the agents used for our datasets to those for unsupervised exploration pretraining baselines in Table 1. We estimate the performance of the Weak and Mixed agents as the average of the corresponding logged evaluations in the Dopamine (Castro et al., 2018) baselines. Even our largest dataset is quite small compared to the amounts of data gathered by unsupervised exploration methods (see the "Data" column in Table 1); this is intentional, as we believe that unsupervised interactional data may be expensive in real world applications. We show the performance of the non-random data collection policies in Table 2 (note that a fully-random policy has a score of **0** by definition).

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

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

[1] Calculated from ICLR 2021 OpenReview submission; unreported in arXiv version.
[2] Upper-bound estimate from averaging evaluation performance of corresponding agents in Dopamine.

4.3 Training Details

We optimize our three representation learning objectives jointly during unsupervised pretraining, summing their losses. During finetuning, we optimize only the reinforcement learning and forward dynamics losses, following Schwarzer et al. (2021) (see Section 5.5), and lower the learning rates for the pretrained encoder and dynamics model by two orders of magnitude (see Section 5.4).

We consider the standard three-layer convolutional encoder introduced by Mnih et al. (2015), a ResNet inspired by Espeholt et al. (2018), as well as an enlarged ResNet of the same design. In other respects, our implementation matches that of SPR and is based on its publicly-released code. Full implementation and hyperparameter details are provided in Appendix C. We refer to agents by the model architecture and pretraining dataset type used: **SGI-R** is pretrained on Random, **SGI-E** on Exploratory, **SGI-W** on Weak, and **SGI-M** on Mixed. To investigate scaling, we vary the size of the encoder used in **SGI-M**: the standard Mnih et al. (2015) encoder is **SGI-M/S** (for small), our standard ResNet is simply **SGI-M** and using a larger ResNet is **SGI-M/L** (for large)[4]. For **SGI-M/L** we also use the 6M step dataset described earlier. All ablations are conducted in comparison to **SGI-M** unless

[3] Our data-collection agents are weaker than those used by ATC on seven of the eight games they consider.
[4] See Appendix C for details on these networks

otherwise noted. Finally, agents without pretraining are denoted **SGI-None**; SGI-None/S would be roughly equivalent to SPR (Schwarzer et al., 2021).

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

5 Results and Discussion

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

Pretraining data efficiency SGI achieves strong performance with only limited pretraining data; our largest dataset contains 6M transitions, or roughly 4.5 days of experience. This compares favourably to recent works on unsupervised exploration such as APT or CPT, which require far larger amounts of data and environment interaction (250M steps or 193 days for APT, 4B steps or 8.45 years for CPT). We expect SGI would perform even better if used in these large-data settingss, as we find that it scales robustly with data (see Section 5.2).

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

5.1 Data quality matters

In principle, SGI can be used with any offline dataset but we demonstrate that it scales with

Table 2: HNS on Atari100k for SGI and baselines.

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

[1] APT claims 0.6955, but we calculate 0.666 from their reported per-game scores.
[2] CPT@0 does not do any finetuning.
[3] Our implementation (see Appendix C.6)

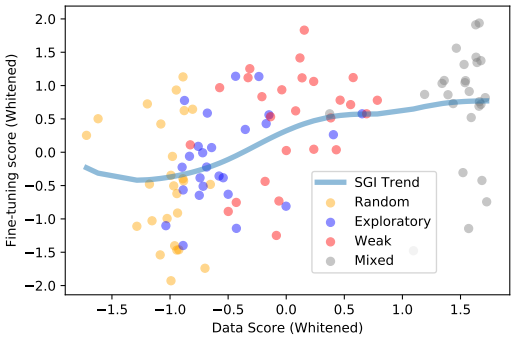


Figure 2: SGI finetuning performance vs. pretraining data score for all combinations of game and dataset. Data score is estimated as clipped return per episode, trend calculated via kernel regression. Values whitened per-game for clarity.

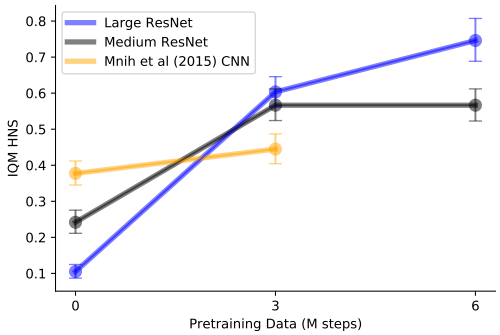


Figure 3: Finetuning performance of SGI for different CNN sizes and amounts of pretrained data from the **Mixed** dataset. We plot IQM HNS with confidence intervals (see Appendix B).

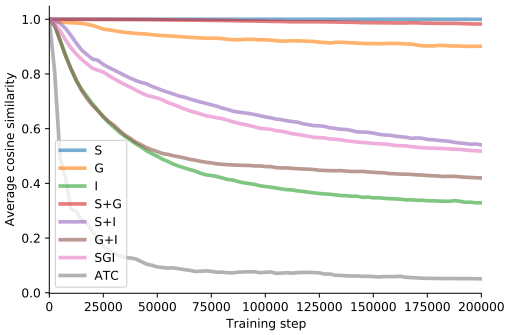


Figure 4: Average cosine similarity between representations over pretraining, averaged across the 26 Atari 100k games. 1 indicates representations are identical, 0 perfect dissimilarity.

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

Importantly, the pattern we observe is very different from what would be expected for imitation learning. In particular, SGI-W's strong performance shows that expert data is not required. To characterize this, we plot the average clipped reward [5] experienced per episode for each of our pretraining datasets in Figure 2. Normalizing across tasks, we find a strong positive correlation between task reward engagement ($p < 0.0001$) and finetuning performance. Moreover, we find diminishing returns to further task engagement.

5.2 Pretraining unlocks the value of larger networks

The three-layer network introduced by Mnih et al. (2015) has become a fixture of deep reinforcement learning, and has been employed by previous works examining pretraining in this field (e.g. Liu & Abbeel, 2021; Stooke et al., 2021). However, we find that representational pretraining with this network (**SGI-M/S**) provides only minor benefits compared to training from scratch. In contrast, larger networks struggle without pretraining but shine when pretrained as shown in Figure 3.

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

5.3 Combining SGI's objectives improves performance

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

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

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

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

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

5.4 Naively finetuning ruins pretrained representations

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

Table 4: HNS on Atari 100K for finetuning schemes for SGI.

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

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

5.5 Not all SSL objectives are beneficial during finetuning

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

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

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

6 Conclusion

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

7 Acknowledgements

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

References

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *ICML*, 2020. 5

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A., and Bellemare, M. G. Deep reinforcement learning at the edge of the statistical precipice. *NeurIPS*, 2021. 5, 15, 19

Anand, A., Racah, E., Ozair, S., Bengio, Y., Côté, M.-A., and Hjelm, R. D. Unsupervised state representation learning in atari. In *NeurIPS*, 2019. 4, 5

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., and Zaremba, W. Hindsight experience replay. In *Advances in neural information processing systems*, pp. 5048–5058, 2017. 3

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 18

Bachman, P., Hjelm, R. D., and Buchwalter, W. Learning representations by maximizing mutual information across views. In *NeurIPS*, 2019. 4

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. *ICML*, 2017. 15

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 1

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. Large-scale study of curiosity-driven learning, 2018. 4, 5

Campos, V., Sprechmann, P., Hansen, S. S., Barreto, A., Blundell, C., Vitvitskyi, A., Kapturowski, S., and Badia, A. P. Coverage as a principle for discovering transferable behavior in reinforcement learning, 2021. URL https://openreview.net/forum?id=INhwJdJtxn6. 1, 4, 5, 6, 7

Castro, P. S., Moitra, S., Gelada, C., Kumar, S., and Bellemare, M. G. Dopamine: A Research Framework for Deep Reinforcement Learning. 2018. URL http://arxiv.org/abs/1812.06110. 6

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. *ICML*, 2020a. 1, 4, 8

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. Big self-supervised models are strong semi-supervised learners. In *NeurIPS*, 2020b. 4

Chu, B., Madhavan, V., Beijbom, O., Hoffman, J., and Darrell, T. *Best Practices for Fine-Tuning Visual Classifiers to New Domains*, pp. 435–442. 11 2016. ISBN 978-3-319-49408-1. doi: 10.1007/978-3-319-49409-8_34. 9

Dabney, W., Barreto, A., Rowland, M., Dadashi, R., Quan, J., Bellemare, M. G., and Silver, D. The value-improvement path: Towards better representations for reinforcement learning. 2021. 3, 9

Degris, T. and Modayil, J. Scaling-up knowledge for a cognizant robot. In *AAAI Spring Symposium on Designing Intelligent Robots: Reintegrating AI.*, 2012. 2

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *ACL*, 2019. 1

Dubey, R., Agrawal, P., Pathak, D., Griffiths, T., and Efros, A. Investigating human priors for playing video games. In *International Conference on Machine Learning*, 2018. 1

Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., and Hester, T. An empirical investigation of the challenges of real-world reinforcement learning. 2020. 5

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. *CoRR*, abs/1802.01561, 2018. URL http://arxiv.org/abs/1802.01561. 6, 8, 18

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2018. 4

Gelada, C., Kumar, S., Buckman, J., Nachum, O., and Bellemare, M. G. Deepmdp: Learning continuous latent space models for representation learning. *ICML*, 2019. 2, 4

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., et al. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*, 2020. 1, 4

Guo, D., Pires, B. A., Piot, B., Grill, J.-b., Altché, F., Munos, R., and Azar, M. G. Bootstrap latent-predictive representations for multitask reinforcement learning. In *ICML*, 2020. 4

Guo, Z. D., Azar, M. G., Piot, B., Pires, B. A., and Munos, R. Neural predictive belief representations. *ICML*, 2018. 2

Hadsell, R., Rao, D., Rusu, A. A., and Pascanu, R. 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. 9

Hansen, S., Dabney, W., Barreto, A., Warde-Farley, D., de Wiele, T. V., and Mnih, V. Fast task inference with variational intrinsic successor features. In *ICLR*, 2020. 1, 2, 4, 5, 6, 7

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del R'ıo, J. F., Wiebe, M., Peterson, P., G'erard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. 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. 15

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016. 18

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 4

Hénaff, O. J., Srinivas, A., De Fauw, J., Razavi, A., Doersch, C., Eslami, S., and Oord, A. v. d. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019. 1, 4

Hernandez, D., Kaplan, J., Henighan, T., and McCandlish, S. Scaling laws for transfer, 2021. 8

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M. G., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, 2018. 3, 15

Hessel, M., Danihelka, I., Viola, F., Guez, A., Schmitt, S., Sifre, L., Weber, T., Silver, D., and van Hasselt, H. Muesli: Combining improvements in policy optimization. *arXiv preprint arXiv:2104.06159*, 2021. 2

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. Learning deep representations by mutual information estimation and maximization. *ICLR*, 2019. 4

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 18

Kaiser, Ł., Babaeizadeh, M., Miłos, P., Osiński, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al. Model based reinforcement learning for atari. In *ICLR*, 2019. 4, 5, 6, 7, 21, 22, 23, 24

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 8

Kielak, K. P. Do recent advancements in model-based deep reinforcement learning really improve data efficiency?, 2020. URL https://openreview.net/forum?id=Bke9u1HFwB. 4

Kostrikov, I., Yarats, D., and Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *ICLR*, 2021. 4, 5, 7, 9, 18

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017. 1

Lesort, T., Díaz-Rodríguez, N., Goudou, J.-F., and Filliat, D. State representation learning for control: An overview. *Neural Networks*, 108, 2018. 3, 4

Li, H., Chaudhari, P., Yang, H., Lam, M., Ravichandran, A., Bhotika, R., and Soatto, S. Rethinking the hyperparameters for fine-tuning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1g8VkHFPH. 9

Liu, H. and Abbeel, P. Unsupervised active pre-training for reinforcement learning, 2021. URL https://openreview.net/forum?id=cvNYovr16SB. 1, 2, 4, 5, 6, 7, 8, 21

Mazoure, B., Combes, R. T. d., Doan, T., Bachman, P., and Hjelm, R. D. Deep reinforcement and infomax learning. In *NeurIPS*, 2020. 4

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540), 2015. 5, 6, 8, 15, 18

Ng, A. Y., Harada, D., and Russell, S. J. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, 1999. 18

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 4

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 15

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *ICML*. PMLR, 2017. 4

Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A. C. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018. 18

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018. 18

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 2021. 8

Schwarzer, M., Anand, A., Goel, R., Hjelm, R. D., Courville, A., and Bachman, P. Data-efficient reinforcement learning with self-predictive representations. In *ICLR*, 2021. 2, 3, 4, 5, 6, 7, 15, 18, 21, 22

Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., and Pathak, D. Planning to explore via self-supervised world models. In *ICML*, 2020. 4

Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2019. 4

Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N., Cubuk, E. D., Kurakin, A., Zhang, H., and Raffel, C. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020. 4

Stooke, A. and Abbeel, P. rlpyt: A research code base for deep reinforcement learning in pytorch. *arXiv preprint arXiv:1909.01500*, 2019. 15

Stooke, A., Lee, K., Abbeel, P., and Laskin, M. Decoupling representation learning from reinforcement learning, 2021. 1, 2, 4, 5, 6, 7, 8, 19, 24

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html. 15

Tan, M. and Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114, 2019. 18

Tarvainen, A. and Valpola, H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017. 4

Tsividis, P., Pouncy, T., Xu, J. L., Tenenbaum, J. B., and Gershman, S. J. Human learning in atari. In *AAAI Spring Symposia*, 2017. 1

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *AAAI*, 2016. 15

van Hasselt, H. P., Hessel, M., and Aslanides, J. When to use parametric models in reinforcement learning? In *NeurIPS*, 2019. 4, 7

Wang, T. and Isola, P. Understanding contrastive representation learning through alignment and uniformity on the hypersphere, 2020. 9

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *ICML*, 2016. 15

Yang, M. and Nachum, O. Representation matters: Offline pretraining for sequential decision making. *arXiv preprint arXiv:2102.05815*, 2021. 5

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Reinforcement learning with prototypical representations. *arXiv preprint arXiv:2102.11271*, 2021. 4

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *ICML*, 2017. 9

Zhan, A., Zhao, P., Pinto, L., Abbeel, P., and Laskin, M. A framework for efficient robotic manipulation, 2020. 5

Zhang, C., Vinyals, O., Munos, R., and Bengio, S. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018. 1

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

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

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 Uncertainty-aware comparisons

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

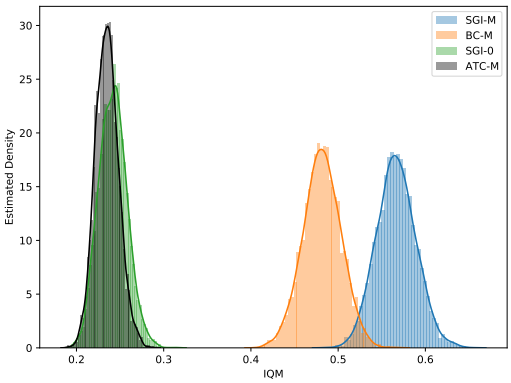
As raw per-run data is required for this, which was not reported for prior work, we do so only for experiments conducted ourselves. In the interests of improving practices in the community moving forward, we also commit to making this data for our experiments available to other researchers in the future.

In Figure 5a through Figure 5e we show estimated uncertainty via bootstrapping for the various comparisons drawn throughout Section 5, while Table 6 gives IQM human-normalized scores and 95% bootstrap confidence intervals for the same results. All comparisons in Figure 5a through Figure 5e are statistically significant ($p < 0.05$) except for:

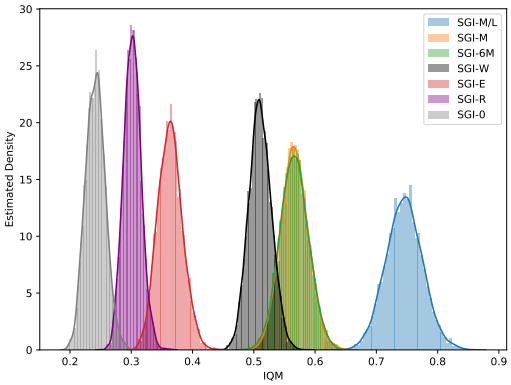
- ATC-M vs SGI-None in Figure 5a ($p \gg 0.05$)
- SGI-M vs SGI-W in Figure 5b ($p \approx 0.05$)
- SGI-M vs SGI-M w/ SGI FT in Figure 5d ($p \approx 0.4$)
- SGI-M vs G+I and S+I in Figure 5e ($p \approx 0.1$)

C Implementation Details

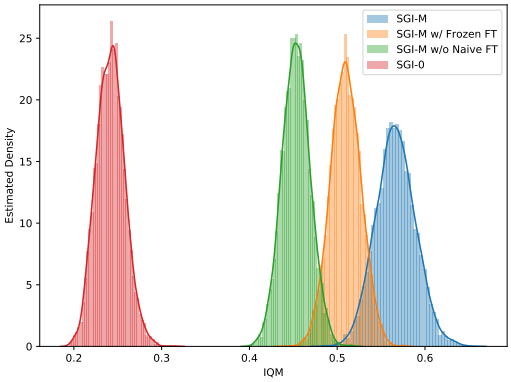
We base our work on the code released for SPR (Schwarzer et al., 2021), which in turn is based on rlpyt (Stooke & Abbeel, 2019), and makes use of NumPy (Harris et al., 2020) and PyTorch (Paszke et al., 2019).



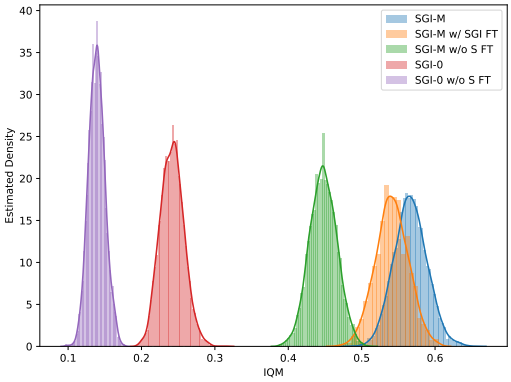
(a) Comparisons to behavioral cloning (BC) and ATC.



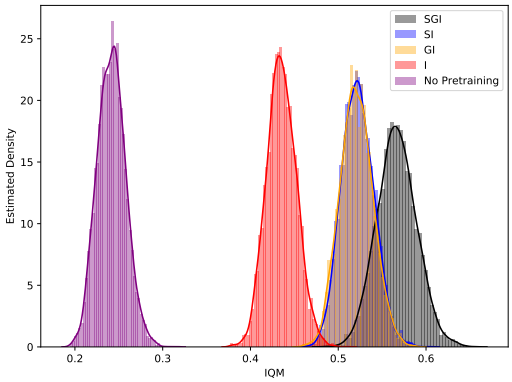
(b) Ablations over different pretraining datasets.



(c) Ablations over various fine-tuning.



(d) Ablations over SSL objectives during fine-tuning.



(e) Ablations over pretraining SSL objectives.

Figure 5: Bootstrapping distributions for uncertainty in IQM measurements.

Table 6: Interquartile mean, median and mean human-normalized scores for variants of SGI and controls, evaluated after finetuning over all 10 runs for each of the 26 Atari 100k games. Confidence intervals computed by percentile bootstrap with 5000 resamples.

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

C.1 Training

We set $\lambda^{\text{SPR}} = 2$ and $\lambda^{\text{IM}} = 1$ during pre-training. Unless otherwise noted, all settings match SPR during fine-tuning, including batch size, replay ratio, target network update period, and λ^{SPR}. We use a batch size of 256 during pre-training to maximize throughput, and update both the SPR and goal-conditioned RL target network target networks with an exponential moving average with $\tau = 0.99$. We pre-train for a number of gradient steps equivalent to 10 epochs over 6M samples, no matter the amount of data used. Due to the cost of pretraining, we pre-train a single encoder per game for each configuration tested. However, we use 10 random seeds at fine-tuning time, allowing us to average over variance due to exploration and data order. Finally, we reduce fine-tuning learning rates for pretrained encoders and dynamics models by a factor of 100, and by a factor of 3 for other pretrained weights. We find this crucial to SGI's performance, and discuss it in detail in Section 5.4.

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

C.2 Goal-Conditioned Reinforcement Learning

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

with probability 0.2, to ensure that some goals correspond to states reached in entirely different trajectories.

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

$$R(\tilde{z}_t, \tilde{z}_{t+1}, g) = d(\tilde{z}_t, g) - d(\tilde{z}_{t+1}, g) \tag{3}$$

$$d(\tilde{z}_t, g) = \exp\left(2\frac{\tilde{z}_t \cdot g}{||\tilde{z}||_2 \cdot ||g||_2} - 2\right). \tag{4}$$

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:

$$\text{FiLM}(F_c|\gamma_c, \beta_c) = \gamma_c F_c + \beta_c \tag{5}$$

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

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

C.3 Model Architectures

In addition to the standard three-layer CNN encoder introduced by Mnih et al. (2015), we experiment with larger residual networks (He et al., 2016). We use the design proposed by Espeholt et al. (2018) as a starting point, while still adopting innovations used in more modern architectures such as EfficientNets (Tan & 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.

C.4 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.

C.5 Temporal Augmentation

Initially due to a mismatch between the semantics of the DQN Replay dataset and the buffers used by rlpyt, SGI sampled actions in the offline dataset with a temporal offset of one (i.e., sampled a_{t+1} instead of a_t). Surprisingly, we found that fixing this did not improve performance as would be expected. Upon further inspection, the actions sampled were identical to the true actions a large fraction of the time, but sampling the true actions made SGI's pretraining tasks far easier. In other words, this was effectively serving as an additional form of data augmentation.

Table 7: Performance of SGI with various action offsets. We report results with offset 1 in the main paper.

Method	IQM@0	IQM@1	Median@0	Median@1	Mean@0	Mean@1
SGI-M	0.530	0.567	0.490	0.679	1.120	1.149
SGI-W	0.553	0.510	0.463	0.589	1.229	1.144

As we find that this in fact improves performance on the median game, the traditional focus of the community, we continue to report results with the original offset. However, under the evaluation method proposed by (Agarwal et al., 2021) these methods are roughly equivalent (see Table 7), and we suspect that methods seeking to build off of SGI for offline RL or imitation learning will wish to sample actions without an offset. We have therefore made this configurable in our released code.

C.6 Experiments with ATC

As ATC (Stooke et al., 2021) was not tested on the Atari100k setting, and as its hyperparameters (including network size and fine-tuning scheme) are very different from those used by SGI, we modify its code[6] 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.

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

D Pseudocode

Algorithm 1: Pre-Training with SGI

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

E Full Results on Atari100k

We report full scores for SGI agents across all 26 games in Table 8. 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.[7]

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

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

Table 9: 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-M
Alien	227.8	7127.7	835.9	1049.3	**1242.8**	1060.7	1117.6	1101.7
Amidar	5.8	1719.5	107.6	133.6	147.7	154.2	**206.0**	168.2
Assault	222.4	742.0	657.7	752.1	869.2	756.3	**1145.2**	905.1
Asterix	210.0	8503.3	832.9	**1029.3**	433.1	575.5	603.1	835.6
Bank Heist	14.2	753.1	613.2	**726.5**	273.6	365.8	323.4	608.4
Battle Zone	2360.0	37187.5	13490.0	**15708.0**	11754.0	13692.0	11689.8	13170.0
Boxing	0.1	12.1	6.6	24.0	**61.5**	34.7	42.7	36.9
Breakout	1.7	30.5	12.1	29.3	34.0	43.0	**62.6**	42.8
Chopper Command	811.0	7387.8	1085.2	1081.2	916.5	925.5	965.8	**1404.0**
Crazy Climber	10780.5	35829.4	19707.6	55002.4	65220.0	69505.6	69052.0	**88561.2**
Demon Attack	152.1	1971.0	778.8	850.0	1329.4	981.7	**1783.8**	968.1
Freeway	0.0	29.6	17.2	28.1	24.4	13.2	10.9	**30.0**
Frostbite	65.2	4334.7	1475.8	662.1	1045.4	482.1	**1664.9**	741.3
Gopher	257.6	2412.5	438.2	626.1	**2214.1**	1561.7	1998.7	1660.4
Hero	1027.0	30826.4	6472.0	5538.3	6353.3	5249.6	**8715.4**	7474.0
Jamesbond	29.0	302.8	157.4	324.2	358.2	346.8	**407.6**	366.4
Kangaroo	52.0	3035.0	**3802.8**	3091.6	800.0	685.6	999.5	2172.8
Krull	1598.0	2665.5	3954.0	5202.7	**6073.7**	5722.8	5323.9	5734.0
Kung Fu Master	258.5	22736.3	7929.4	11952.2	**19374.6**	15039.8	18123.2	16137.8
Ms Pacman	307.3	6951.6	990.2	1276.4	1663.3	1753.3	**1779.3**	1520.0
Pong	-20.7	14.6	-4.4	-4.2	3.8	3.9	-0.1	**7.6**
Private Eye	24.9	69571.3	62.8	**385.9**	96.7	90.5	90.0	90.0
Qbert	163.9	13455.0	720.0	664.8	587.6	681.3	**3015.8**	709.8
Road Runner	11.5	7845.0	5428.4	14629.7	14311.9	17036.5	13998.2	**18370.2**
Seaquest	68.4	42054.7	577.8	509.0	1054.4	**1397.8**	989.4	728.4
Up N Down	533.4	11693.2	46042.6	48856.6	29938.4	**105466.9**	45023.5	79228.8
Median HNS	0.000	1.000	0.343	0.425	0.499	0.452	0.397	**0.679**
Mean HNS	0.000	1.000	0.565	0.849	0.971	1.114	1.011	**1.149**
#Games > Human	0	0	3	8	8	8	8	**9**
#Games > SPR	0	19	10	14	15	14	17	**20**

Table 10: 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**

F 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 11. We pretrain on a dataset of 750k frames from each game in a clique (i.e. 3M frames for a clique of 4) and finetune on a single game. To show whether pretraining on other games is beneficial, we compare to a baseline of pretraining on just the 750k frames from the single Atari 100k game we use for finetuning.

Our results in Table 12 show that pretraining with the extra frames from the clique games is mostly unhelpful to finetune performance. Only Kangaroo shows a modest improvement, a few games show no difference in performance, and most games show a decrease in performance when pretraining with other games. We believe that Atari may not be as suitable to transferring representations as other domains, and previous work using Atari to learn transferable representations has also had negative results (Stooke et al., 2021). Though game semantics can be similar, we note that even small differences in rule sets and visual cues can make transfer difficult.

Table 11: Cliques of semantically similar games

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

Table 12: 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