AMSI SUMMER RESEARCH SCHOLARSHIPS 2023–24

SET YOUR SIGHTS ON RESEARCH THIS SUMMER

Data Efficient Reinforcement Learning

Joel Woodfield
Supervised by Dr. Nan Ye
University of Queensland

AMSI AUSTRALIAN MATHEMATICAL SCIENCES INSTITUTE

Abstract

Data efficient reinforcement learning is important for learning how to interact with an environment when this interaction is expensive. One approach is to focus on robustness, where with limited data we aim to learn latent representations of the environment's state that generalise well to unseen states. Self-predictive representations (SPR) is a reinforcement learning algorithm that succeeds in doing this, achieving state-of-the-art performance on the benchmark Atari-100k tasks. Our research has two objectives: (a) simplify SPR by assessing the necessity of each component and (b) apply masked-reconstruction extensions in the pixel level to SPR to increase the robustness of the model. We find that that the predictor model in SPR seems to not be necessary for it to achieve similar performance. We also find that our implementation of masked-reconstruction hurts the performance of SPR. Our analysis reveals some fundamental issues that make masked reconstruction difficult to apply successfully in reinforcement learning settings. These include masking hiding important small objects, and reconstruction reinforcing representations without the small objects.

1 Introduction

Reinforcement learning (RL) is the task of learning to interact in an unknown environment to maximise an objective (Sutton and Barto 2018). This has applications in many fields such as learning behaviour for robotics or learning dynamic treatment regimes in healthcare (Li 2017). Deep reinforcement learning makes useful of deep artificial neural networks, and have seen successes in complex tasks such as playing Atari games (Badia et al. 2020). However, they require a lot of environment interactions to learn, which can be a problem for tasks where this interaction is expensive (Schwarzer et al. 2020).

Data efficient RL aims to undertake this learning with a limited number of environment interactions to learn from. One way to approach this is to learn representations of the environment from the limited interactions that is robust to new, unseen interactions. Self-predictive representation (SPR) is a model that succeeds in learning more robust representations. It aims to make the representations predictive of the representations of future timesteps. It achieves state-of-the-art performance at the time of release (Schwarzer et al. 2020). Another method for making representations robust is masked reconstruction. It helps encourage the representations of images to be invariant when parts of the image is hidden. The Masked Autoencoder model (He et al. 2022) applies this technique and helped achieve state-of-the-art performance on image-classification tasks at the time of release.

Given these successes, our project has two goals:

- Goal 1: Simplify the SPR model by evaluating the necessity of each component. We do this by performing an ablation study.
- Goal 2: Apply extensions based on masked reconstruction to SPR.

For the scope of this research, we limit our evaluation of our models to 6 representative games out of 26 Atari 100k benchmark games (Kaiser et al. 2020). This is a common benchmark to evaluate data efficient RL

algorithms, which limits the number of interactions used to train the algorithm to 100,000. This is about 2 hours of playtime (Kaiser et al. 2020).

We find that one of the components of SPR, the predictor, is not essential for the model to perform just as well on the chosen games. We also find that our implementations of masked-reconstruction seems to hurt the model's performance on Atari games more than it helps. This is likely due to fundamental issues with masking hiding important objects that are small and reconstruction encouraging learning representations with these objects not incorporated.

The remainder of this report is structured as follows. Section 2 will cover related work on masked-reconstruction and other representational learning techniques in RL. Section 3 will go over how the RL problems and algorithms are formulated and the methods involved in SPR and masked-reconstruction. Section 4 will give considerations on evaluating model performance when simplifying or extending the SPR algorithm. Section 5 will show the ablation experiments to simplify SPR. Section 6 will cover the methods and results from applying masked reconstruction to SPR. Finally, Section 7 will discuss the implications of the results obtained.

Statement of Authorship

We used the implementation of the SPR algorithm by Schwarzer et al. (2020) as a foundation for our investigations. Joel Woodfield extended the code for the ablation experiments and the masked-reconstruction extensions, and performed analyses and diagnostic experiments on the results. Dr. Nan Ye contributed the idea for this research and gave guidance and feedback on all aspects of this research.

2 Related Work

There have been some previous work done on applying masked reconstruction for RL tasks. Yu et al. (2022) applies large amount of masking on space-time cubes, and performs reconstruction in the latent-space. A sequence of latent representations is used to reconstruct each latent representation. Seo et al. (2023) applies the masked autoencoder algorithm on the latent convolutional outputs of the inputs. By masking and reconstructing the convolutional outputs, it can avoid completely hiding small objects. Our research differs from these methods by focusing on smaller masking proportions and performing reconstruction in the pixel-space rather than the latent space.

There have also been other techniques for learning robust representations for RL. Contrastive unsupervised representations (Srinivas, Laskin, and Abbeel 2020) aims to make the representations invariant to augmentations while keeping representations of different inputs different. Data-regularized Q (Kostrikov, Yarats, and Fergus 2021) makes use of multiple image augmentations to generate new examples for the models to learn from, making the representations more robust.

3 Background

Markov Decision Processes (MDPs) are a general class of mathematical models for describing how an agent interacts with the environment, and existing RL works focus on this. Q-learning algorithms are commonly used to learn to make optimal decisions within MDPs. SPR, the algorithm of our focus, builds upon the Q-learning algorithm called Rainbow. In this section, we will introduce each of these ideas along with a description on SPR and the inspiration for masked-reconstruction.

3.1 Markov Decision Process

MDP is a framework that can model the dynamics of the environment interaction in RL (Sutton and Barto 2018). Formally, an MDP is specified by the tuple (s_t, a_t, r_t, s_{t+1}) for each timestep $t \geq 0$, where s_t is the complete state of the environment from the set of possible states $\mathbb{S}$, a_t is the action taken from the set of possible actions $\mathbb{A}$, $r_t \in \mathbb{R}$ is the reward obtained, and $s_{t+1} \in \mathbb{S}$ is the state at the next timestep.

These tuples are generated sequentially using a few dynamics functions. We start with an initial state s_0 sampled from an initial state distribution p_0. Given the current state and action s_t and a_t, every succeeding state s_{t+1} is sampled from the transition distribution $T(s|s_t, a_t)$. From each state s_t, we choose action a_t by sampling from the agent's policy distribution $\pi(a|s_t)$. The reward r_t is associated with each state and the action taken from that state by the reward function $R(s_t, a_t)$.

As explained by Mnih et al. (2015), the 6 Atari games that we use for performance evaluation can each be formulated as an MDP. 4 consecutive frames of each game in grayscale contains enough information to know the positions of all the objects and how they are moving. Thus, we consider every 4 frames of the game as the state. The actions, which the agent takes every 4 frames, are the game's buttons. The reward is the points obtained or lost during the game.

3.2 Reinforcement Learning by Q-Learning

How the game is played is determined by the policy distribution π, and this can be automatically learned by the RL algorithm called Q-learning. As explained by Sutton and Barto (2018), from each state $s \in \mathbb{S}$, Q-learning predicts the expected sum of future discounted rewards for each action $a \in \mathbb{A}$ you could take from s. This is done by building a model for the function $Q^\pi(s, a) = E\left(\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right)$. The goal for our policy is to maximise Q^π over all $\pi : \mathbb{S} \to \mathbb{A}$, which we can define as Q^*

$$Q^*(s, a) = \max_\pi E\left(\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right)$$

Our policy would then be to choose actions that maximise Q^*.

Bellman's optimality equation can be used to build the model for Q^*. This equation relates the value of Q^* at neighbouring timesteps.

$$Q^*(s_t, a_t) = E\left(R(s_t, a_t) + \gamma \max_a Q^*(s_{t+1}, a)\right)$$

If we define an operator $H(Q) = E\left(R(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a)\right)$, and have a randomly intialised function $Q : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$, it has been shown that Q converges to Q^* if we iteratively update Q for all $i > 0$ by

$$Q_{i+1} = H(Q_i) \tag{1}$$

We can iteratively collect the states, actions, and rewards from the environment and solve Equation 1 for all $i > 0$. We collect these objects from the MDP's dynamics functions and store it in a buffer B (see Algorithm 1). We try to solve Equation 1 by sampling a tuple (s_k, a_k, r_k, s_{k+1}) randomly from B, and then updating our model to minimise the Q-loss function

$$L_Q = (R(s_k, a_k) + \gamma \max_a Q(s_{k+1}, a) - Q(s_k, a_k))^2$$

We switch between collecting experiences from the environment and updating Q in a way shown in Algorithm 2 (Mnih et al. 2015).

Algorithm 1: CollectExperience

Input: Starting timestep t, state s_t, policy π, transition model T, reward function R, experience buffer B, number of timesteps to collect N_c

Output: Last collected timestep t, experience buffer B

for i *in* $1 : N_c$ **do**
    $a_t \sim \pi(a|s_t)$
    $r_t = R(s_t, a_t)$
    $s_{t+1} \sim T(s|s_t, a_t)$
    Insert(B, (s_t, a_t, r_t, s_{t+1}))
    $t = t + 1$
end

3.3 Deep Reinforcement Learning

Deep reinforcement learning parameterises Q using a neural network Q_θ (Mnih et al. 2015). Q_θ can be learned to approximate Q^* by minimizing L_Q iteratively using gradient descent based methods. A simple form updates the parameters in the opposite direction to a semi-gradient of L_Q at each iteration

$$\theta_{i+1} = \theta_i - \alpha(R(s_t, a_t) + \gamma \max_a Q_{\theta_i}(s_{t+1}, a) - Q_{\theta_i}(s_t, a_t))\nabla_\theta Q_{\theta_i}, \quad \alpha \in \mathbb{R}$$

For our model, we use the Adam method (Kingma and Ba 2014), which is based on this gradient descent.

For image-based problems, one way to structure the neural network as done by Schwarzer et al. (2020) is to split Q_θ into a combination of two functions h and f.

$$Q_\theta(s_t, a_t) = h(f(s_t), a_t)$$

Algorithm 2: Q-Learning

Input: Initial state s_0, Q-function Q_0, policy π_0 (dependent on Q_0), transition model T, reward function R, experience buffer B, number of training steps N_t, number of timesteps to collect per training step N_c, number of times to update per training step N_u

Output: trained policy π_i

$t = 0$
$i = 0$
for j *in* $1 : N_t$ **do**
    $t, B = \text{CollectExperience}(t, s_t, \pi_i, T, R, B, N_c)$
    for k *in* $1 : N_u$ **do**
        /* Sample tuple randomly from B */
        $(s_k, a_k, r_k, s_{k+1}) \sim B$
        $L_{Q_i} = (R(s_k, a_k) + \gamma \max_a Q_i(s_{k+1}, a) - Q_i(s_k, a_k))^2$
        $Q_{i+1} = \text{Update}(Q_i, L_{Q_i})$
        $i = i + 1$
    end
end

where s_t and a_t are the state and action at timestep t. We consider $f : \mathbb{S} \to \mathbb{Z}$ to be the encoder, which maps the state to a latent representation from the set of latent representations $\mathbb{Z}$. We consider $h : \mathbb{Z} \to \mathbb{R}$ to be the head, which maps the latent representation and action taken to the expected sum of future discounted rewards. By splitting Q_θ as so, we separate the tasks of getting useful features from the state and predicting the future rewards. We can also visualize the function composition in a flowchart diagram (see Figure 1).

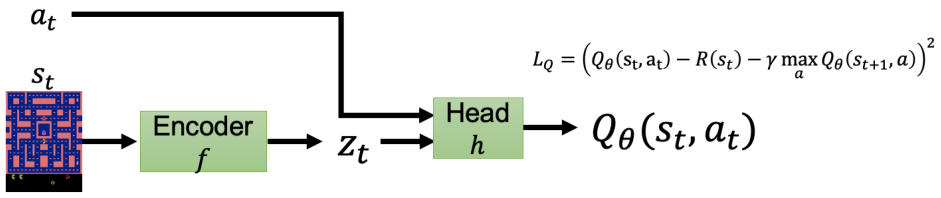


Figure 1: Flow chart diagram for Q_θ. For each time step t, the game's frame s_t gets input into the encoder and outputs the latent representation of the game's state z_t. Both the action a_t and representation z_t is input into the head to output the Q function value.

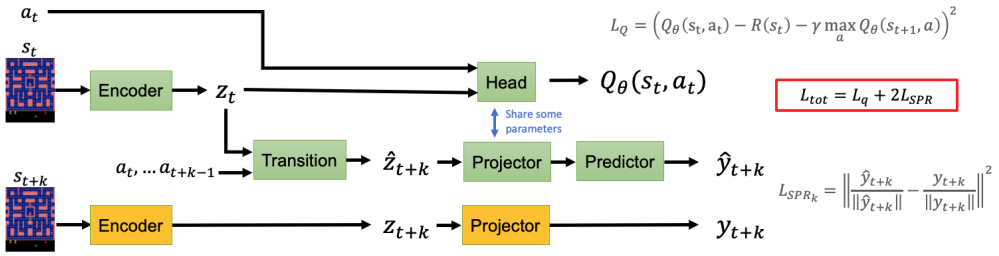


Figure 2: Flow chart diagram for the SPR model. It builds on top of the model in Figure 1. The latent representation z_t gets also input into the Transition model along with actions $a_t, \ldots, a_{t+k-1}$ to produce the prediction $\hat{z}_{t+k}$. We also obtain the true state s_{t+k} in the future and get z_{t+k}. Both z_{t+k} and $\hat{z}_{t+k}$ are projected into a lower dimensional space, with the prediction going through an additional predictor function. The normalised mean square error between the two projections $\hat{y}_{t+k}$ and y_{t+k} are minimised.

3.4 Rainbow Deep Q Network

Rainbow Deep Q Network developed by Hessel et al. (2017) is one of the most effective Q-learning algorithms. It combines many techniques that has been found to help learn a more optimal Q function. One important component is the use of convolutional neural networks. Because Atari games are image-based, Q_θ has to process images. Convolutional neural networks are designed to help learn spatially-invariant patterns in the data, which is helpful for analysing images (LeCun, Bengio, et al. 1995).

3.5 Self-Predictive Representations

Self predictive representations (SPR) was developed by Schwarzer et al. (2020). It aims to make the representations of the input states predictive by adding an additional objective for predicting the representations at future timesteps. We do this by using a transition model to predict the latent representation $\hat{z}_{t+k} \in \mathbb{Z}$ k steps into the future using the current representation $z_t \in \mathbb{Z}$. We then try to minimize the difference between the predicted $\hat{z}_{t+k}$ with the actual representation z_{t+k} obtained in the future.

The way SPR minimizes this difference is more complicated and is laid out in Figure 2 with components summarised in Table 1. The latent representations are projected into a lower dimensional space using the projector function. The projected prediction is then put through a predictor function. The difference between the prediction and true outputs are minimised by minimising the normalised mean square error L_{SPR_k}. The SPR model does this prediction for K successive future steps and minimises a combined objective L_{SPR}.

$$L_{SPR} = \frac{1}{K} \sum_{k=1}^{k} \left\| \frac{\hat{y}_{t+k}}{||\hat{y}_{t+k}||} - \frac{y_{t+k}}{||y_{t+k}||} \right\|^2 \tag{2}$$

This is then minimised simultaneously with L_Q by minimising the weighted sum

$$L_{total} = L_Q + 5L_{SPR}$$

The motivation for some of these decisions such as using the projector seems to be to prevent representational collapse, which is where the latent representations for all states to become too similar to each other. This would

Number	Name	Description
1	Encoder	Outputs latent representation of the input state.
2	Head	Outputs expected predicted future sum of rewards.
3	Transition	Predicts future latent representation from the current latent representations and the actions leading up to it.
4	Projector	Projects both true and predicted latent representations into a lower dimensional space
5	Predictor	Applies a linear transformation to only the projected predicted latent representation.

Table 1: Summaries for the components of the SPR model.

help minimise L_{SPR}, but will be unhelpful for using the latent representations to predict the future sum of rewards.

3.6 Masked Autoencoder

The masked autoencoder model by He et al. (2022) is a neural network model designed to learn better latent representations of images. Masking hides patches of the input image and aims to learn latent representations of them that can be used to fill in the masks and reconstruct the original image.

The model is comprised of the encoder and the decoder. The encoder outputs the latent representation from the masked image, and the decoder tries to reconstruct the original image from the latent representation. The parameters of both functions are updated to minimise the pixel-based reconstruction loss for the unmasked pixels

$$L_r = ||m'(s_t) - m'(\hat{s}_t)||^2$$

where $s_t, \hat{s}_t \in [0,1]^{3D}$ are the input image and reconstructed image respectively, each composed of D pixels each with 3 values for red green and blue. $m'(s) \in [0,1]^{3U}$ outputs the U unmasked pixels of $s \in [0,1]^{3D}$. A diagram for this algorithm is shown in Figure 8 in Appendix C.

The authors use vision transformer neural networks for the encoder and decoder. These are used in many state-of-the art computer vision models such as OmniVec (Srivastava and Sharma 2023). The authors mask a large proportion of patches so that masked patches cannot easily be filled in by looking at neighbouring patches (He et al. 2022).

An intuition on why masked-reconstruction can help produce robust representations is to consider masking as an image augmentation (Kong and Zhang 2023), and apply intuition on augmentations as explained by Tarvainen and Valpola (2017). Masking slightly alters the image, but the high-level content of the image, which can be encoded by the representation, does not change. By applying multiple sets of masking to the same image, we get multiple new data points with a different image but the same latent representaiton to train our

model on. This helps produce a more robust representation model when there is limited data to learn from.

4 Performance Evaluation

To evaluate SPR's agent's performance on the Atari games, Schwarzer et al. (2020) uses the Human Normalised Score and aggregate it across the games tested using the mean or the median. For each game g and our agent A, the human normalized score HN_g is defined as

$$HN_g = \frac{A_g - R_g}{H_g - R_g}$$

where A_g is our agent's score on the game, R_g is the score obtained by an agent taking random actions, and H_g is the score obtained by a human player as collected by Mnih et al. (2015).

However, we find that using either the mean or median of these scores can lead to results that are biased towards a subset of the games. This is because our models achieve different magnitudes of values for HN_g for different games. The mean would be biased towards scores from games with consistently higher HN_g. The median would be biased towards games with HN_g that are consistently middle of the pack. Empirically, we find that in our ablation experiments in section 5, the median HN_g score for all our modifications were the average of the same two games (the median of an even number of scores).

To mitigate this problem, we instead use a simple modification of the human normalized score where we we replace the human score with the score that the SPR model achieves. We will call this the SPR normalized score

$$SPRN_g = \frac{A_g - R_g}{SPR_g - R_g}$$

where SPR_g is the game score obtained by the SPR model.

This alleviates the problem of having different magnitudes of values for HN_g for different games, as games that are harder to beat humans for our agent will also be hard for SPR agent. An additional side effect is that the score directly measures how much our agent improves over the original SPR. A score of greater than 1 means it is performing better, and a score less than 1 means it is performing worse.

5 Self-Predictive Representations Ablation Study

To determine the necessity of each component, we will analyse the performance of SPR for each component removed. This has been done already by Schwarzer et al. (2020) for some components, but not all.

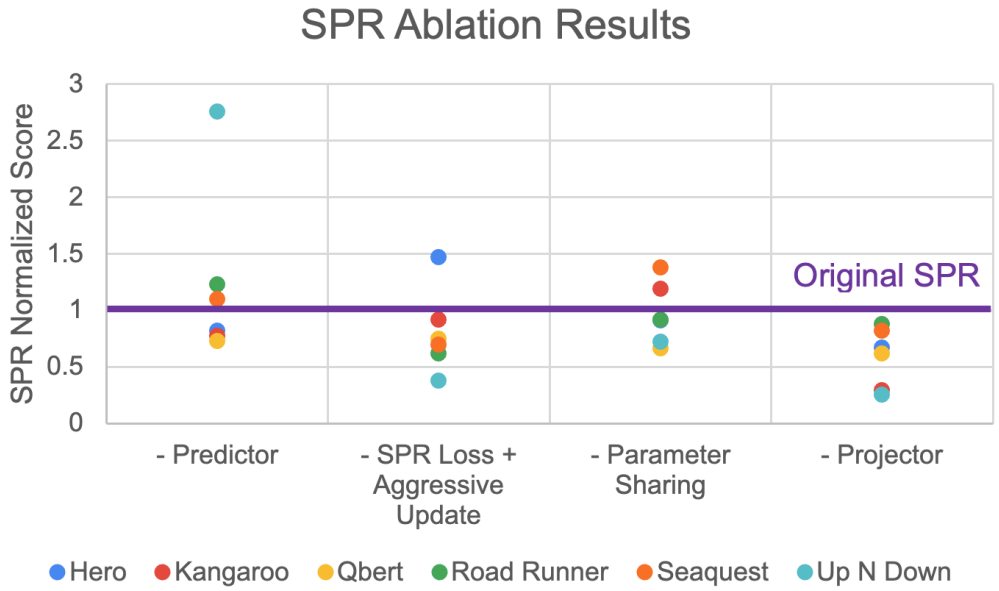
The predictor is the first component we will ablate. Although the predictor (number 5 in Table 1) has shown to be crucial to prevent representational collapse in self-supervised computer vision models (Balestriero et al. 2023), the SPR model is different with the addition of the transition model and L_Q, so it may not be needed.

Another component is the projector (number 4 in Table 1). Although Schwarzer et al. (2020) has tried removing the projector for both the prediction and the true representations, removing the projector for only

the true representation has not been tested. To make the dimensions match, we make the projector for the predicted representation project onto the same number of dimensions as the original latent representation.

The projector also shares some parameters with the head model (see Figure 2). The usefulness of this has yet to be tested, so we will also try having completely separate parameters between these two models.

Finally, there is the fact that for each sampled state s_t, K more states are used in computing L_{SPR}. This begs the question of whether the benefit of SPR comes from using each state on average $K+1$ times more to train the model, rather than through the use of the prediction model. Thus, we will also try training the RL model without the SPR components (numbers 3-5 in Table 1) and update the parameters more aggressively by updating the model through $K+1$ times more every training step. By an unfortunate error, we instead tested by using each state K times more rather than $K+1$, but it is still testing the same idea.



	Mean	Median
- Predictor	1.23	0.96
- SPR Loss + Aggressive Update	0.80	0.72
- Parameter Sharing	0.96	0.91
- Projector	0.58	0.64

Figure 3 & Table 2: The SPR normalized scores for the 4 modifications on 6 games. The most successful modification is removing the predictor. It gets a higher score than SPR on 3 out of 6 games, with a much higher score on Up N Down. It also gets a higher mean and similar median SPR normalized score. Other modifications do worse, especially removing thhe projector and removing SPR loss and adding aggressive updates.

Figure 3 shows the results of the experiments on the 6 games. Removing the predictor achieves a similar performance to SPR, having a median SPR normalized score close to 1 and a mean higher than 1. The higher mean can be attributed to the significantly higher performance on Up N Down. This suggests that the

predictor is not needed for SPR to achieve similar performance, and may help in certain tasks such as Up N Down. Removing the parameter sharing also achieves SPR Normalized scores close to 1, but is a bit lower. The other two modifications show clear signs of performing worse compared to the original SPR, having SPR normalized scores much less than 1. These features seem to be necessary for SPR to perform well.

6 Masked Reconstruction

To implement masked-reconstruction into SPR, we need to first decide the architecture for the autoencoder to produce the latent representations and the reconstructions. We also need to determine the amount of masking to apply. We also consider three different variants in incorporating masking and reconstruction into the SPR algorithm. We then analyse the results on the 6 Atari games and then perform diagnostic experiments to understand the results better.

6.1 Autoencoder Architecture

The Masked Autoencoder model uses transformer neural networks, which considers patches of pixels together. The masking is also applied to these patches. However, we find that we need to mask individual pixels rather than patches (see subsection 6.2). Using the transformer model for pixel inputs is unfortunately computationally expensive.

To account for this, we will use the same convolutional neural network from the original SPR model for the encoder. For the decoder, we use transposed convolutions to project the latent representation back to the higher-dimensional state space. To train the encoder, we minimimise the reconstruction loss over all D pixels.

$$L_r = ||s_t - \hat{s}_t||^2$$

where $s_t \in [0,1]^D$ is the true state and $\hat{s}_t \in [0,1]^D$ is the reconstruction from the latent representation of s_t.

6.2 Masking Size

We decide to mask individual pixels instead of patches so that the masking does not alter the high level features (representation) of the game state. If too much masking is applied, it can hide important details of the image, such as the location of ghosts in Ms Pacman (see Figure 4). We found that by masking out patches of pixels at a time, it was easy for the mask to completely hide important objects in Ms Pacman. Hiding these objects will change the way the game should be played, which would mean a different representation.

We also control the proportion of pixels masked to optimise the benefit of masking. We have to balance maximising this proportion to increase the robustness of our model and not masking too much to hide important objects. To do this, we train our autoencoder with masking proportions 0, 0.1, 0.2, 0.3, and 0.4 on images from the 6 games obtained by taking random actions. We evaluate the effectiveness of each proportion by measuring its reconstruction loss. We find that a proportion of 0.3 has the lowest average reconstruction loss, so we used this proportion for our masking experiments.

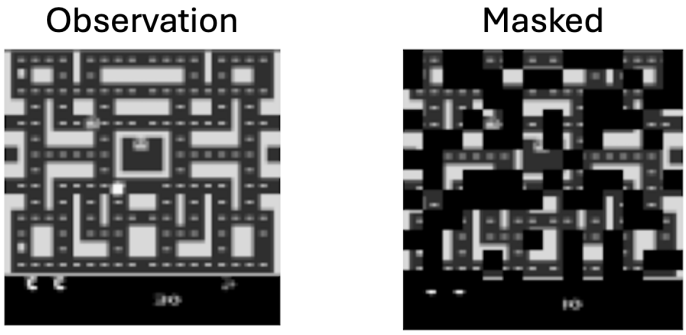


Figure 4: Ms Pacman with too much masking. The player in the lower left position is completely hidden.

The decision to use a smaller proportion than the masked autoencoder model and to mask pixels rather than patches goes against the idea He et al. (2022) brought up that reconstruction can be done too easily by looking at neighbouring pixels. However, the fact that we achieved a lower reconstruction loss by adding the low masking suggests that our masking at least helps with learning better representations for reconstruction.

6.3 Training Strategies

There are multiple ways to include masked reconstruction into the SPR model. We consider three variants. Detailed algorithms for each are shown in Appendix B.

6.3.1 Variant 1

For the first variant, we simply add masking to the state before it is input into the model and add the decoder that takes in the latent representation. The reconstruction loss is computed using the reconstruction, and it is added to the total loss so that it can be minimized concurrently with L_Q and L_{SPR}

$$L_{total} = L_Q + 5L_{SPR} + 10L_r$$

The decision to multiply L_r by 10 was to make the magnitudes of the reconstruction loss comparable to the magnitudes of L_{SPR} observed in the initial stages of training on Ms Pacman.

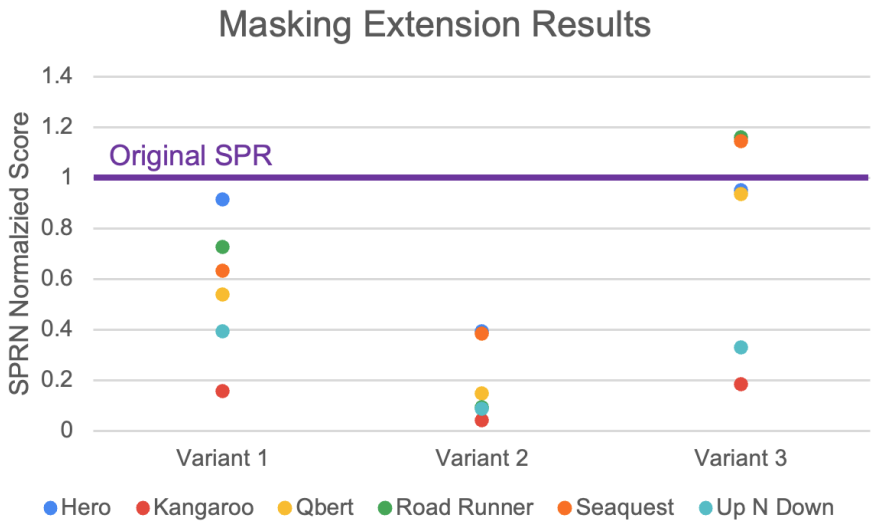
6.3.2 Variant 2

For the second variant, we first pretrain just the encoder and decoder on game images obtained by taking random actions. We then copy the parameters of the encoder to the encoder of SPR, and train SPR as normal without masking or reconstructing. Although this uses extra game interactions when pretraining, this method is similar to how the masked autoencoder is applied to computer vision tasks so we decide to test this.

6.3.3 Variant 3

For the third variant, we apply masking to the images but we do not add the decoder and the reconstruction. Unlike the masked autoencoder, our SPR algorithm has two other objectives to minimise, and the representation learned by minimising these objectives may already be improved just by applying the mask.

6.4 Results



	Mean	Median
Variant 1	0.56	0.59
Variant 2	0.19	0.12
Variant 3	0.78	0.94

Figure 5 & Table 3: SPR Normalized Scores from the 6 games for the 3 masking variants. All variants have a lower mean and median scores than the original SPR. Variant 3, which does not have a reconstruction objective, performs better than the others with higher mean and median scores. It also achieves better results on two out of the six games. Variant 2 performs much worse for all games.

The results are shown in Figure 5. It shows that for all variants, the performance is worse on the majority of the 6 games, especially Variant 2. This suggests that the method of masking we implemented, which is used in all 3 variants, hurts the performance of the SPR model. It also shows that the model performs better without doing the reconstruction, as seen with higher scores in Variant 3 which does not make use of the reconstruction.

6.5 Diagnostic Tests

To understand why masking does not seems to help, we can look at the features that the encoder detects by analysing the reconstruction images. To investigate further into why Variant 3 with no reconstruction objective performed better, we analyse the performance of SPR with different combinations of losses. We also analyse

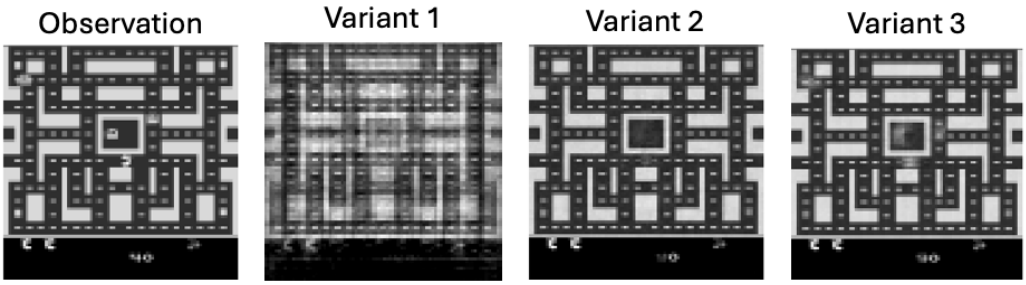


Figure 6: The reconstruction outputs for the three masking variants trained on Ms Pacman. The observation is the image input into the autoencoders for the three variants. For variant 1, only the background and the pellets are reconstructed. The same items are reconstructed in variant 2, but it is more clear. For variant 3, the ghosts and player are reconstructed along with the background.

the variance of the latent representations output by the encoder to check if the reconstruction objective caused representational collapse.

6.5.1 Reconstruction Images

We will analyse what features of the input images gets incorporated in the latent representation by analysing the reconstruction images produced by our masked-reconstruction variants trained on the game Ms Pacman. For variants 1 and 2, we can simply use the trained encoder and decoder. For variant 3, we create a new autoencoder with the trained encoder, and a randomly initialized decoder. We then train just the decoder to minimise the reconstruction loss. We use 100,000 sets of 4 frames - the same number used to train the other variants.

Figure 6 shows the reconstructions. A key insight is that the small, moving objects are not reconstructed in variants 1 and 2. This suggests, although we can't be sure, that these objects are hard for variants 1 and 2 to detect. On the other hand, Variant 3 reconstructs the small, moving objects blurrily. This suggests that Variant 3 does detect these objects.

6.5.2 Loss Compatibility

Loss Combination	Mean SPR Normalized	Median SPR Normalized
Q + SPR + R	0.56	0.59
Q + SPR	0.78	0.94
Q + R	0.46	0.36
Q	0.64	0.73

Table 4: SPR normalized scores from training SPR on the 6 games with different combinations of Q-loss (Q), SPR loss (SPR), and reconstruction loss (R).

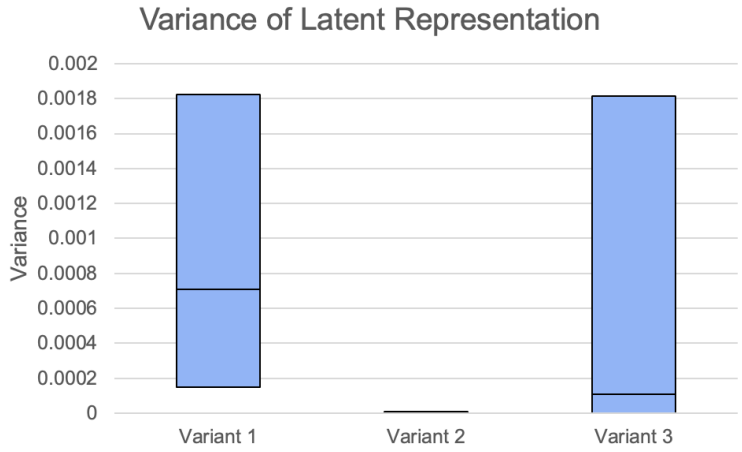


Figure 7: Box plot showing the quartiles for the sample variances for each of the 3136 components of the latent representation of Ms Pacman. Minimums and maximums are not shown. The sample variance is computed over 2000 sets of 4 frames obtained by taking random actions in the game. The middle 50% of the variances are similar for Variant 1 and Variant 3, but significantly smaller for Variant 2.

Table 4 shows the SPR normalized scores from training SPR with masked images with different combinations of losses. It shows that adding the reconstruction loss hurts the performance both with SPR loss and without. This suggests that the reconstruction loss is incompatible with minimising the Q-loss when masking is applied.

However, it should be noted that it is possible to get relatively good scores with just Q and SPR losses (Variant 3), and then separately train the decoder to produce good reconstructions as we have found in subsubsection 6.5.1. This suggests that it is rather the dynamics of minimising reconstruction loss at the same time as the Q-loss that makes it incompatible.

6.5.3 Variance of Latent Representation

Figure 7 shows the variances of the 64x7x7 dimensional latent representations produced by the encoders of the three variants trained on the game Ms Pacman. It shows that Variant 2 has much lower variance, which means that the latent representations of different states are more similar to each other. Variant 1 and 3 have similar variances.

7 Discussion and Conclusion

7.1 Necessity of the Predictor

The results of SPR without the predictor performed just as well as with the predictor. This suggests that the predictor is not needed in the SPR model to obtain the same performance. Although the predictor seems to be necessary in preventing representational collapse in computer vision pretraining settings, we hypothesise that adding the additional Q-loss objective and sharing the parameters in the projector and the head can prevent

representational collapse without the predictor. Testing on more problems is needed to confirm this result.

7.2 Hiding Small Objects

Another important insight is that masking can hide small objects that are crucial to playing the game. Although the amount of masking can be reduced so that the small objects remain partially visible, it is possible that it can increase the difficulty to detect these objects too much that it outweighs the benefits of learning more robust representations of the input. In some games such as Kangaroo, some important objects are so small such that any amount of masking could hide it in some frames. This may be the reason for the consistently low performance on Kangaroo. Further research on detecting small, moving objects may be helpful.

7.3 Decoder Reinforcing Bad Representations

Another finding is that adding the reconstruction objective hurts the performance regardless of whether SPR is used. Because the reconstructions often do not include the small, moving objects, it is possible that this objective encourages the learning of the representations of bigger, non-moving objects. Because the small, moving objects are often more important to play the games, this would mean the reconstruction loss is encouraging the detection of distractions, which may explain the worse performance. This is perhaps further compounded by masking hiding smaller objects.

Variant 2, where its encoder was only trained using the reconstruction objective, has low variance in the latent representations. It also had the worst performance on the Atari games out of the three variants. This supports the idea that the reconstruction objective encourages learning the representations of unmoving objects rather than the more helpful moving objects.

7.4 Conclusion

Overall, we have found a possible way to simplify the SPR model by removing the predictor, and gained some insight into how masked-reconstruction strategies can prevent the SPR model from learning good representations to play Atari games. Some reasons for the unsuccessful results such as the masking hiding objects may not be applicable to tasks outside of Atari games. It may be worth exploring masking for those tasks. Additionally, further research into models that could better detect small objects may help learn better representations for tasks similar to Atari games.

References

Badia, Adrià Puigdomènech et al. (2020). "Agent57: Outperforming the atari human benchmark". In: *International conference on machine learning*. PMLR, pp. 507–517.

Balestriero, Randall et al. (2023). "A cookbook of self-supervised learning". In: *arXiv preprint arXiv:2304.12210*.

He, Kaiming et al. (2022). "Masked autoencoders are scalable vision learners". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009.

Hessel, Matteo et al. (2017). *Rainbow: Combining Improvements in Deep Reinforcement Learning*. arXiv: 1710.02298 [cs.AI].

Kaiser, Lukasz et al. (2020). *Model-Based Reinforcement Learning for Atari*. arXiv: 1903.00374 [cs.LG].

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Kong, Xiangwen and Xiangyu Zhang (2023). "Understanding masked image modeling via learning occlusion invariant feature". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6241–6251.

Kostrikov, Ilya, Denis Yarats, and Rob Fergus (2021). *Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels*. arXiv: 2004.13649 [cs.LG].

LeCun, Yann, Yoshua Bengio, et al. (1995). "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.10, p. 1995.

Li, Yuxi (2017). "Deep Reinforcement Learning: An Overview". In: *CoRR* abs/1701.07274. arXiv: 1701.07274. URL: http://arxiv.org/abs/1701.07274.

Mnih, Volodymyr et al. (2015). "Human-level control through deep reinforcement learning". In: *nature* 518.7540, pp. 529–533.

Schwarzer, Max et al. (2020). "Data-efficient reinforcement learning with self-predictive representations". In: *arXiv preprint arXiv:2007.05929*.

Seo, Younggyo et al. (2023). *Masked World Models for Visual Control*. arXiv: 2206.14244 [cs.RO].

Srinivas, Aravind, Michael Laskin, and Pieter Abbeel (2020). *CURL: Contrastive Unsupervised Representations for Reinforcement Learning*. arXiv: 2004.04136 [cs.LG].

Srivastava, Siddharth and Gaurav Sharma (2023). *OmniVec: Learning robust representations with cross modal sharing*. arXiv: 2311.05709 [cs.CV].

Sutton, Richard S and Andrew G Barto (2018). *Reinforcement learning: An introduction*. MIT press.

Tarvainen, Antti and Harri Valpola (2017). "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results". In: *Advances in neural information processing systems* 30.

Yu, Tao et al. (2022). "Mask-based latent reconstruction for reinforcement learning". In: *Advances in Neural Information Processing Systems* 35, pp. 25117–25131.

A Hyperparameters

For the SPR model, the same hyperparameters from Schwarzer et al. (2020) are used, except for the SPR loss coefficient, where we use 5 instead of 2. Table 5 shows the SPR hyperparameters. The mask-reconstruction-specific hyperparameters are shown in Table 6.

Parameter	Setting
Gray-scaling	True
Observation down-sampling	84x84
Frames stacked	4
Action repetitions	4
Reward clipping	[-1, 1]
Terminal on loss of life	True
Max frames per episode	108K
Update	Distributional Q
Dueling	True
Support of Q-distribution	51
Discount factor	0.99
Minibatch size	32
Optimizer	Adam
Optimizer: learning rate	0.0001
Optimizer: β_1	0.9
Optimizer: β_2	0.999
Optimizer: ϵ	0.00015
Max gradient norm	10
Priority exponent	0.5
Priority correction	$0.4 \rightarrow 1$
Exploration	Noisy nets
Training steps	100K
Evaluation trajectories	100
Min replay size for sampling	2000
Replay period every	1 step
Updates per step	2
Multi-step return length	10
Q network: channels	32, 64, 64
Q network: filter size	8x8, 4x4, 3x3

Continued on next page

Parameter	Setting
Q network: stride	4, 2, 1
Q network: hidden units	256
Non-linearity	ReLU
Target network: update period	1
SPR loss coefficient	5
K	5
Data Augmentation	Random shifts (± 4 pixels) & Intensity(scale=0.05)

Table 5: Hyperparameters used for the SPR model.

Parameter	Setting
Masking proportion	0.3
Masking patch size	1x1x1
Decoder: model	Transposed Convolutional Neural Network
Decoder: channels	64, 64, 32
Decoder: filter size	3x3, 4x4, 8x8
Decoder: stride	1, 2, 4
Decoder loss coefficient	10
Variant 2: Number of steps used in pretraining	100,000

Table 6: Hyperparameters used for the masking extensions.

B Masking Strategies

Algorithm 3: Variant 1 Training

Input: Initial state s_0, policy π, transition T, reward R, encoder f, decoder d, network parameters θ, experience buffer B, random mask function m, SPR augmentation function A, number of training steps N_t, number of timesteps to collect per training step N_c, number of times to update per training step N_u

Output: Trained policy π

$t = 0$
for j *in* $1 : N_t$ **do**
  $t, B = \text{CollectExperience}(t, s_t, \pi, T, R, B, N_c)$
  for k *in* $1 : N_u$ **do**
    Radomly sample sequence of K experiences $\{s_{t'}, a_{t'}, r_{t'}, s_{t'+1}\}_{t'=i}^{i+K} \sim B$
    for t' *in* $(i, \ldots, i+K)$ **do**
      $s_{t'} = A(s_{t'})$
    end
    $s_i^{(m)} = m(s_i)$
    $z_i = f(s_i^{(m)})$
    $s_i' = d(z_i)$
    $L_r = ||s_i - s_i'||^2$
    $L_Q = |R(s_i^{(m)}, a_i) + \gamma \max_a' Q_i(s_{i+1}, a') - Q_i(s_i^{(m)}, a_i)|^2$
    /* Get SPR loss using Figure 2 and Equation 2 */
    $L_{SPR} = SPRLoss(z_i, \{s_{t'}, a_{t'}, r_{t'}, s_{t'+1}\}_{t'=i}^{i+K})$
    $L_{tot} = L_Q + 5L_{SPR} + 10L_r$
    Update(θ, L_{tot})
  end
end

Algorithm 4: Variant 2 Training

Input: Initial state s_0, policy π, transition T, reward R, encoder f with parameters θ_f, decoder d with parameters θ_d, other parameters in the network θ', experience buffer B, random mask function m, SPR augmentation function A, number of training steps for autoencoder N_{a_t}, number of training steps for SPR N_{s_t}, number of timesteps to collect to train autoencoder N_{a_c}, number of times to update SPR per training step N_{s_u}

Output: Trained policy π

$t = 0$

/* Collect experiences by choosing actions randomly */

$t, B = \text{CollectExperience}(t, s_t, \pi^{random}, T, R, B, N_c)$

/* Train autoencoder */

for i *in* $1 : N_{a_t}$ **do**

  $(s, a, r, s_{next}) \sim B$

  $s^{(m)} = m(s)$

  $z = f(s^{(m)})$

  $s' = d(z)$

  $L_r = ||s - s'||^2$

  $\text{Update}((\theta_e, \theta_d), L_r)$

end

Clear all experiences in B

/* Train SPR with pretrained encoder */

for j *in* $1 : N_{s_t}$ **do**

  for k *in* $1 : N_u$ **do**

    Randomly sample sequence of K experiences $(s_{t'}, a_{t'}, r_{t'}, s_{t'+1})_{t'=i:i+K} \sim B$

    for t' *in* $(i, \ldots, i + K)$ **do**

      $s_{t'} = A(s_{t'})$

    end

    $L_Q = |R(s_i, a_i) + \gamma \max'_a Q_i(s_{i+1}, a') - Q_i(s_i, a_i)|^2$

    /* Get SPR loss using Figure 2 and Equation 2 */

    $L_{SPR} = SPRLoss[z_i, (s_{t'}, a_{t'}, r_{t'}, s_{t'+1})_{t'=i+1:i+K}]$

    $L_{tot} = L_Q + 5L_{SPR}$

    $\text{Update}(\theta', L_{tot})$

  end

end

Algorithm 5: Variant 3 Training

Input: Initial state s_0, SPR network parameters θ, policy π_θ, transition T, reward R, encoder f, decoder d, experience buffer B, random mask function m, SPR augmentation function A, number of training steps N_t, number of timesteps to collect per training step N_c, number of times to update per training step N_u

Output: Trained policy π_θ

$t = 0$
for i *in* $1 : N_t$ **do**
  $t, B = \text{CollectExperience}(t, s_t, \pi_\theta, T, R, B, N_c)$
  for i *in* $1 : N_u$ **do**
    Randomly sample sequence of K experiences $\{s_{t'}, a_{t'}, r_{t'}, s_{t'+1}\}_{t'=i}^{i+K} \sim B$
    for t' *in* $(i, \ldots, i+K)$ **do**
      $s_{t'} = A(s_{t'})$
    end
    $s_i^{(m)} = m(s_i)$
    $z_i = f(s_i^{(m)})$
    $L_Q = (R(s_i^{(m)}, a_i) + \gamma \max'_a Q_i(s_{i+1}, a') - Q_i(s_i^{(m)}, a_i))^2$
    /* Get SPR loss using Figure 2 and Equation 2 */
    $L_{SPR} = SPRLoss(z_i, \{s_{t'}, a_{t'}, r_{t'}, s_{t'+1}\}_{t'=i}^{i+K})$
    $L_{tot} = L_Q + 5L_{SPR}$
    $\text{Update}(\theta, L_{tot})$
  end
end

C Masked Autoencoder Diagram

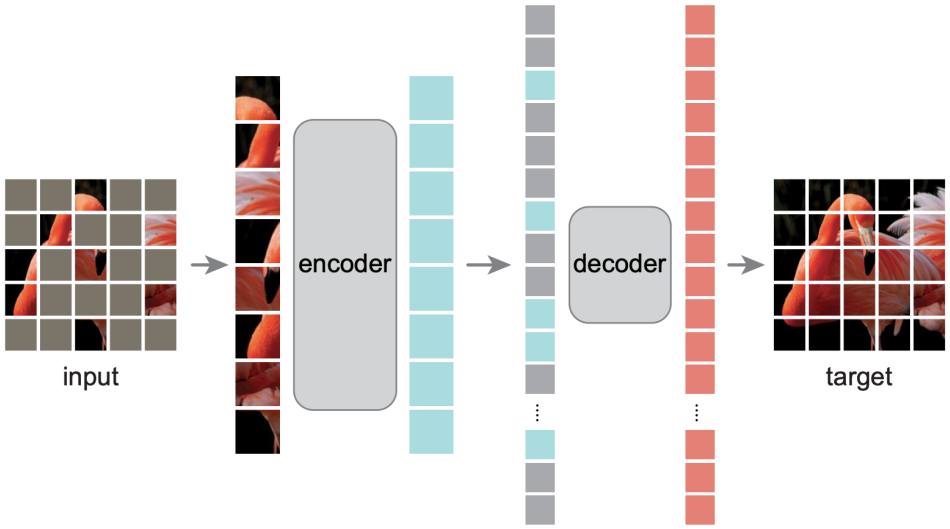


Figure 8: The masked autoencoder model. The input image is split into patches, and a subset is randomly masked. The unmasked patches are input into the vision transformer encoder to output the latent representation of the input. The latent representation is then input into the decoder that reconstructs the original input. Figure taken from (He et al. 2022).