Does Self-supervised Learning Really Improve Reinforcement Learning from Pixels?

Anonymous Author(s)
Affiliation
Address
email

Abstract

We investigate whether self-supervised learning (SSL) can improve online reinforcement learning (RL) from pixels. We extend the contrastive reinforcement learning framework (e.g., CURL) that jointly optimizes SSL and RL losses and conduct an extensive amount of experiments with various self-supervised losses. Our observations suggest that the existing SSL framework for RL fails to bring meaningful improvement over the baselines only taking advantage of image augmentation when the same amount of data and augmentation is used. We further perform an evolutionary search to find the optimal combination of multiple self-supervised losses for RL, but find that even such a loss combination fails to meaningfully outperform the methods that only utilize carefully designed image augmentations. Often, the use of self-supervised losses under the existing framework lowered RL performances. We evaluate the approach in multiple different environments including a real-world robot environment, and confirm that no single self-supervised loss or image augmentation method can dominate all environments and that the current framework for joint optimization of SSL and RL is limited. Finally, we empirically investigate the pretraining framework for SSL + RL and the properties of representations learned with different approaches.

1 Introduction

Learning to act from image observations is crucial in many real-world applications. One popular approach is online reinforcement learning (RL), which requires no human demonstration or expert trajectories. Since all training samples are collected by the agent during policy learning in online RL, the collected data often has strong correlations and high variance, challenging the policy learning. Meanwhile, the cost of interacting with environments requires the RL algorithms to have higher sample efficiency. Compared to RL using state-based features, pixel-based RL continuously takes images as inputs, which usually come with a much higher dimensionality than numerical states. Such properties pose serious challenges to image representation learning in RL.

Recent work tackles such problems mainly from two different directions. (1) Inspired by the great success of self-supervised learning (SSL) with images and videos (e.g., [4, 5, 7, 9, 13, 14, 16, 20, 29, 30, 35, 38, 48, 50, 51, 54, 63]), some RL methods (e.g., CURL [40] and SAC+AE [72]) take advantage of self-supervised learning. This is typically done by applying both self-supervised loss and reinforcement learning loss in one batch. In this paper, we dub such joint optimization of the self-supervised loss and the RL loss as the *Joint Learning Framework*. (2) On the other hand, approaches such as RAD [41] and DrQ [73, 74] investigate how online RL can take advantage of image augmentations. They show significant improvements by applying relatively simple image augmentations to observations of RL agents.

Our objective is to study how well a single or combination of self-supervised losses and augmentations work under the joint learning framework, and to empirically identify their impact in RL systems.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

In this paper, we extend the existing joint (SSL + RL) learning framework, conduct experiments comparing multiple self-supervised losses with augmentations, and empirically test them in many environments from different benchmarks. We confirm that a single self-supervised loss under such joint learning framework typically fails to bring meaningful improvements to existing image augmentation-based methods. We also computationally search for a better combination of losses and image augmentations for RL with the joint learning framework. The experiments in different environments and tasks show inconsistency in self-supervised learning's capability to improve reinforcement learning.

With all our findings, we present this work as a thorough reference for investigating better frameworks and losses for SSL + RL and inspiring future research. Our contributions can be summarized as follows:

1. We conduct an extensive comparison of various self-supervised losses with the existing joint learning framework for pixel-based reinforcement learning in many environments from different benchmarks, including one real-world environment. We demonstrate the properties of representations learned by different SSL-based methods and image augmentation-based methods.
2. We perform evolutionary search for the optimal combination of multiple self-supervised losses and the magnitudes of image augmentation, and confirm its limits.
3. We empirically study a pretraining framework for combining SSL with RL as an alternative of the joint learning framework.

2 Preliminaries

2.1 Reinforcement Learning

In this paper, we extend the configurations of previous work [40, 72] and exploit SAC (**S**oft **A**ctor **C**ritic) [24, 25] and Rainbow DQN [31] for the environments with continuous action space and discrete action space respectively.

Soft Actor Critic is an off-policy actor-critic algorithm that takes advantage of the maximum entropy to encourage the agent to explore more states during the training. It maintains a policy network π_ψ and two critic networks Q_{ϕ_1} and Q_{ϕ_2}. The goal of π_ψ is to maximize the expected sum of rewards and a γ-discounted entropy simultaneously, where the entropy encourages the agent to explore during learning.
Rainbow DQN is a variant of DQN [47] with a bag of improvements such as double Q-learning [28, 64], prioritized sampling [55], noisy net [18], distributional RL [3], dueling networks [69] and multi step reward.

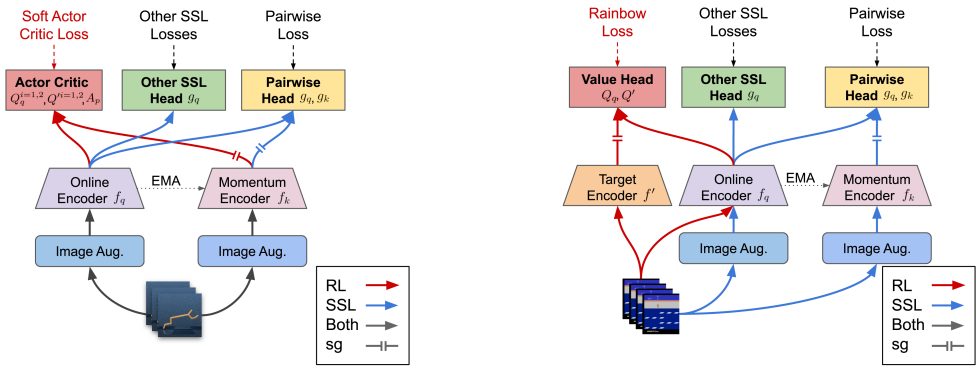
2.2 Pairwise Learning

We coin the term "Pairwise" Learning for the frameworks that learn visual representations based on semantic invariance between dual-stream encoder representations. A general pairwise learning method first generates multiple augmented views by applying a series of random image augmentations to the input sample, then clusters views with the same semantics in the representation space. Optionally in such frameworks, methods using contrastive losses repel samples with different semantics. In this paper, we focus on four representative pairwise learning methods, MoCo [10, 11, 29], BYOL [21], SimSiam [9] and DINO [5]. We have a detailed explanation and comparison on these methods at Sec. A.1.

2.3 Representation Learning for Pixel-based RL

Previous works explore the possibility of learning better visual representation which may finally benefit the policy learning. One direction is using image augmentation for policy learning, where RAD [41] and DrQ [73] achieve significant performance using simple image augmentation. Another direction is to combine SSL with Pixel-based RL, in which two representative methods are SAC+AE [72] and CURL [40].

RAD (**R**einforcement Learning with **A**ugmented **D**ata) [41] investigates the impact of different types of image augmentations for both image and state inputs. By applying random translation or random crop to the input image, RAD significantly improves data-efficiency solely through image



(a) SAC backbone

(b) Rainbow backbone

Figure 1: General joint learning framework for SSL + RL. The red solid arrow represents the RL flow; the blue one represents the SSL flow and the black one means a shared flow for both; sg stands for stop gradients.

augmentation without any auxiliary losses.

DrQ (**D**ata-**r**egularized **Q**) [73] further investigates the possibilities of utilizing image augmentation. DrQ applies image augmentation twice on the input images, and averages the Q value over two augmented images which is assigned as the Q value of the input images.

SAC+AE [72] takes advantage of a RAE (deterministic **R**egularized **A**uto**E**ncoder) [19], in replacement of $\beta-$VAE Higgins et al. [32] in SAC backbone to improve learning stability. The RAE is jointly trained with SAC by performing both SAC update and RAE update alternatingly in one batch.

CURL (**C**ontrastive **U**nsupervised Representations for **R**einforcement **L**earning) [40] combines contrastive learning with an online RL algorithm by introducing an additional contrastive learning head at the end of the image encoder. Similar to the aforementioned SAC+AE, here the contrastive loss and reinforcement learning loss are applied alternatively at training.

3 Self-supervision for Reinforcement Learning

To effectively evaluate different self-supervised losses, we extend the joint learning framework used in SAC+AE [72] and CURL [40] by replacing its contrastive learning head with a general self-supervised learning head. We keep the same backbone using SAC in tasks with continuous action space and using Rainbow DQN in tasks with discrete action space.

3.1 General Joint Learning Framework

With SAC Fig. 1a shows a general joint learning framework with SAC backbone. The unmodified SAC backbone contains two image encoders, online encoder f_q, target (or momentum) encoder f_k and an actor head A_p. Each encoder is followed by two critic heads. Besides that, we attach an additional self-supervised head g_q after the online encoder. For some pairwise learning losses, we concatenate a momentum SSL head g_k after the target encoder.

For every sampled batch of transitions, we first apply image augmentation to both the current state s and the next state s', and update the SAC model $(f_q, Q_q^{i=1,2}, A_p)$ using the augmented images. Note that for stability concerns, we do not update the parameters of the image encoder when updating the actor head A_p. Then, the target networks are updated by Exponential Moving Average (EMA). This is followed by also performing an EMA update of the SSL head if required. And finally the online encoder f_q and the self-supervised head g_q are updated by the self-supervised loss. By alternatingly performing RL and SSL in every batch, we jointly train all the components in the framework. The pseudo code of SAC update alternating RL and SSL is provided in Algorithm 1.

With Rainbow DQN Fig. 1b demonstrates how to jointly apply SSL to Rainbow DQN backbone. The unmodified Rainbow DQN backbone maintains an online encoder f_q and a target encoder f', followed by two state value heads Q_q and Q'. We introduce an additional momentum encoder f_k and self-supervised heads g_q and g_k as suggested in CURL. For each batch, the self-supervised losses are computed using augmented images, while the RL loss is computed using the original data. Finally, the online encoder f_q and the self-supervised head g_q are updated by the self-supervised loss. The pseudo code of Rainbow DQN update can be found at Algorithm 2.

Algorithm 1 Update SAC with Self-supervised Losses
Green: additional operations for SSL; Orange: only for BYOL and DINO.

procedure UPDATESACWITHSSL(s: current state, s': next state, a: action, r: reward, d: done signal, step: model update step counter, f_q: online encoder, f_k: target/momentum encoder, A_p: actor head, Q_q^i: online critic head, Q'^i: target critic head, τ : target/momentum network update rate, g_q: online SSL head, g_k: momentum SSL head)
  $s_a, s'_a \leftarrow$ IMAGEAUGMENTATION(s), IMAGEAUGMENTATION(s')
  $f_q, Q_q^{i=1,2}, A_p \leftarrow$ UPDATESOFTACTORCRITIC(s_a, s'_a, a, r, d)
  $f_k, Q'^{i=1,2} \leftarrow \tau(f_q, Q_q^{i=1,2}) + (1-\tau)(f_k, Q'^{i=1,2})$ ▷ EMA update of SAC
  $g_k \leftarrow \tau g_q + (1-\tau) g_k$ ▷ EMA update of the momentum SSL head
  $f_q, g_q \leftarrow$ UPDATESSL(s_a, s'_a, a, r)
end procedure

Algorithm 2 Update Rainbow with Self-supervised Losses
Green: additional operations for SSL; Orange: only for BYOL and DINO.

procedure UPDATERAINBOWDQNWITHSSL(s: current state, s': next state, a: action, r: reward, d: done, step: model update step counter, f_q: online encoder, f': target encoder, Q_q: online value head, Q': target value head, f_k: momentum networks, τ : momentum network update rate, g_q: online SSL head, g_k: momentum SSL head, w_{SSL}: weights of self-supervised losses)
  $s_a, s'_a \leftarrow$ IMAGEAUGMENTATION(s), IMAGEAUGMENTATION(s')
  $\mathcal{L}_{SSL} \leftarrow$ CALCULATESSLOSS(s_a, s'_a, a, r)
  $\mathcal{L}_{\text{Rainbow}} \leftarrow$ CALCULATERAINBOWLOSS(s, s', a, r, d)
  $\mathcal{L} \leftarrow \mathcal{L}_{\text{Rainbow}} + w_{SSL}\mathcal{L}_{SSL}$
  $f_q, Q_q, g_q \leftarrow$ ONLINENETWORKSUPDATE$(\mathcal{L})$
  $f', Q' \leftarrow f_q, Q_q$ ▷ Copy parameters from online networks to target networks
  $f_k, g_k \leftarrow \tau(f_q, g_q) + (1-\tau)(f_k, g_k)$ ▷ EMA update of momentum networks and SSL head
end procedure

3.2 Losses for Self-supervised Learning

The self-supervised losses we investigated can be categorized into four classes: pairwise learning, transformation awareness, reconstruction, and reinforcement learning context prediction.

Pairwise Learning We investigate three representative pairwise learning methods: BYOL [21], DINO [5] and SimSiam [9], along with existing CURL whose framework is similar to MoCo [29]. BYOL, DINO and SimSiam only explicitly pull positive samples closer without the need of large number of negative samples. CURL uses a contrastive loss taking both positive and negative samples into consideration.
Given the general joint learning framework described in Sec. 3.1, by substituting the self-supervised head and loss, we can easily formulate different agents w.r.t. self-supervised losses. For BYOL, as shown in Fig. 9a, a projector and a predictor are appended to the online encoder sequentially. And an momentum projector is attached to the end of the target/momentum encoder. DINO (Fig. 9c) maintains only projector in both online and target branches. The momentum projector is also updated by EMA. The two encoders in BYOL and DINO operates on two augmented views of the data respectively whereas SimSiam (see Fig. 9b), uses only the online network and a projector for processing both the augmented views.

Transformation Awareness Recent works (e.g., [12, 20, 34, 37, 43, 48]) have shown that the awareness of transformations (like rotation, Jigsaw puzzle and temporal ordering) improves many downstream tasks in computer vision like image classification and action recognition. Typically such awareness can be acquired by explicitly asking a classifier to identify the applied transformation from the pixel representation. Therefore, we investigate two simple classification losses, rotation classification (*RotationCLS*) and shuffle classification (*ShuffleCLS*), and set a two-layer MLP classifier as the self-supervised head in the joint learning framework.
RotationCLS represents the methods that encourage spatial transformation awareness. Inspired by RotNet [20] and E-SSL [12], we rotate the input image after augmentation by 0°, 90°, 180°and 270°. The classifier predicts the rotation angle from the visual representation and it is trained by cross-entropy loss.
Shuffle Tuple [46] encourages the encoder to develop awareness of action causality by predicting if two frames appear in order. We adapt Shuffle Tuple by randomly shuffling the current state image

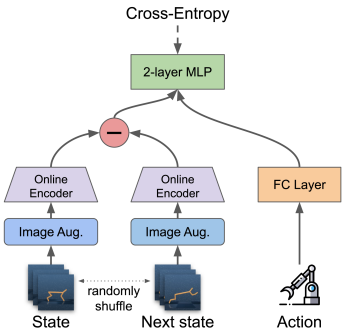


Figure 2: ShuffleCLS

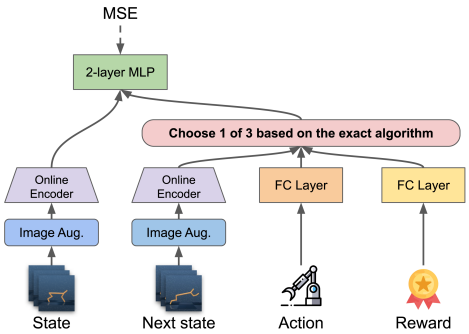


Figure 3: General RL context prediction

Table 1: I/O of RL context prediction losses

	Extract-A	Extract-R	Guess-A	Guess-F	Predict-F	Predict-R	Extract-AR	Guess-AF	Predict-FR
Rep. of s'	Input	Input	-	Output	Output	-	Input	Output	Output
Action a	Output	-	Output	-	Input	Input	Output	Output	Input
Reward r	-	Output	Input	Input	-	Output	Output	Input	Output

and next state image in a state transition tuple and predicting whether it is shuffled or not. The classifier also takes action into consideration because some of the transitions are reversible. The overall architecture of ShuffleCLS is shown as Fig. 2.

Reconstruction Reconstructing the input image with an hourglass architecture has been shown to be an effective way to learn image representation [19, 32, 38]. We simply extend SAC+AE by changing the input and reconstruction target to be augmented images. The reconstruction loss and regularization from RAE [19] are left untouched.
Recent study on **M**asked **A**uto**E**ncoder [30] (MAE) adapts the reconstruction task for patch-based Vision Transformers [17]. The objective in MAE includes reconstructing the entire image from input masked image patches. Inspired from this, we adapt SAC+AE into SAC+MAE by replacing the augmented input image with its masked version, and only penalizing the reconstruction error for the masked patches.

RL Context Prediction Besides the self-supervised learning methods that are specifically designed for pixels, we investigate the losses using attributes naturally collected during RL process. For any state transition that is not the end of a trajectory, it contains four components: current state s, next state s', action a and reward r, with trajectory termination signal omitted. Inspired by Shelhamer et al. [60], we concatenate the visual representation of the current state s and another representation h as the input. Without loss of generality, the second input representation h can be any of these three representations of s', a and r. Then, we predict the remaining components using a two-layer MLP. For continuous outputs, mean-squared error (MSE) loss is applied, while for the discrete target (e.g., action in discrete action space), we use cross-entropy loss. The architecture of this group of self-supervised losses is shown in Fig. 3. From the combination of inputs and outputs, we define nine losses whose I/O specifications are provided in Table 1. For those losses whose outputs include two components, two target prediction networks share the same SSL head except the last task-specific layer.

3.3 Evolving Multiple Self-supervised Losses

Besides a single self-supervised loss or handcrafted combination of two losses, we further investigate how multiple self-supervised losses affect the policy learning together with the joint learning framework. We formulate the combination of multiple losses as a weighted sum $\mathcal{L}_{\text{Combo}} = \sum_{i=1}^{N_l} w_i \cdot \mathcal{L}_i$ where w_i is the weight of a specific loss $\mathcal{L}_i$ and N_l is the total number of losses in the search space. In the joint learning framework, we apply both self-supervised $\mathcal{L}_{\text{Combo}}$ and RL losses jointly to the networks for every mini-batch. Considering that the policy learning is quite sensitive to hyper-parameters (see Sec. A.4), it is non-trivial to find each weight for every SSL loss.

ELo (**E**volving **Lo**sses) [53] shows promising results in unsupervised video representation learning [51, 63], by using evolutionary search to automatically find optimal combination of many self-supervised losses. In the spirit of ELo, we turn to evolutionary search for automatically find-

ing the optimal solution. Assume an unknown objective function whose inputs are weights of multiple losses w_i and the magnitudes of image augmentation $m_{j=1,2}$ for the online encoder and momentum encoder. The function output is the score achieved by the trained agent in its environment: $\mathcal{R}_{\text{env}}^{\text{seed}}(m_{j=1,2}, w_{i=1,2,\ldots N_l})$. The optimization algorithm approaches the maximum value of the objective function by continuously testing the value of the objective function, which in our case is the training and evaluation of an agent with the given parameters. We choose an off-the-shelf optimization algorithm PSO (**P**article **S**warm **O**ptimization) [36] for its simplicity, and perform the evolutionary search on **one** environment from DMControl [62] or Atari game benchmark [2]. For each set of inputs, we find it critical to run with multiple random seeds and report IQM (interquartile mean)[1] for a stable and robust search. The optimization process is presented as: $\text{argmax}_{m_{j=1,2}, w_{i=1,\ldots,N_l}} \text{IQM}(\mathcal{R}_{\text{env}}^{\text{seed}=1,\ldots,5}(m_{j=1,2}, w_{i=1,\ldots,N_l}))$

Note that we are also implicitly searching for the balance between the self-supervised loss and the RL loss by performing this search, as it has the capability to adjust absolute weights of the self-supervised losses overall. Please refer to Sec. A.2.3 for our detailed configurations and search results.

4 Experiments

We conduct experiments in three directions, in order to better understand how we should integrate SSL with RL. First, we demonstrate how different self-supervised losses affect the RL process, by trying them on multiple challenging tasks. Then, we benchmark a pretraining framework as an alternative of the joint learning framework (Sec. A.7). Finally, we perform further empirical analysis on the visual representations learned with the joint learning framework (Sec. A.8).

Evaluation Scheme Thorough evaluation of reinforcement learning algorithms is challenging due to the high variances between each run and the extensive requirement of computation. Consequently, we run all experiments with multiple different random seeds and report the interquartile mean and the standard deviation of the scores as suggested by Agarwal et al. [1]. For a quantitative comparison of the different methods mentioned in Section 3.2, in addition to the absolute scores, we assign a *Relative Score* to each method. We denote the interquartile mean of scores achieved by agent A in environment $e \in E$ as $\text{IQM}^{A,e}$ and denote the collection of all interquartile mean scores achieved in environment e by different agents as IQM^{e}. The Relative Score of agent A is computed as $S_{\text{Relative}}^{A} = \sum_{e \in E} (\text{IQM}^{A,e} - \text{mean}(\text{IQM}^{e}))/\text{std}(\text{IQM}^{e})$.

DMControl Experiments DMControl (**D**eep**M**ind **C**ontrol suite) contains many challenging visual continuous control tasks, which are widely utilized by recent papers. We evaluate all the methods introduced in Sec. 3, along with two important baselines, SAC-NoAug and SAC-Aug(100), in six environments of DMControl that are commonly used in previous papers [40, 41, 72, 73]. Other methods that only take advantage of image augmentation, like RAD [41] and DrQ [73] are also benchmarked for comparison.

The difference between these methods can be summarized as follows: SAC-NoAug is the original pixel-based SAC [24, 25]. SAC-Aug(88) and SAC-Aug(100) use random crop as the only image augmentation, where (88) means the original image has a size of 88×88 before randomly cropping to 84×84 and (100) means the original image has a size of 100×100. These two methods should be regarded as variants of RAD with different augmentation choices. The Random crop from 100×100 to 84×84 is the default image augmentation method for all the methods introduced in Sec. 3.2. Essentially, if we remove their self-supervised loss, they will fall back to SAC-Aug(100). Similarly, we test DrQ variants by replacing its default random shift augmentation with random crop, reported as DrQ(88) and DrQ(100). Meanwhile, RAD uses random translate by default except on walker walk; ELo-SAC renders images at 100 × 100 and first perform center crop with sizes of 87 × 87 and 86 × 86. Then two central patches are randomly cropped to 84 × 84 as the inputs for the online networks and the target networks respectively. For the policy learning part, all the methods share the same model. However, DrQ, DrQ(88) and DrQ(100) apply an additional $\tanh$ activation after the convolutional layers in the encoder. Therefore, we also report DrQ-w/o-Tanh removing $\tanh$ activation from DrQ for reference.

We mainly follow the hyper-parameters reported in CURL, except that we use the same learning rate 10^{-3} in all environments for simplicity. All the methods are benchmarked at 100k environment steps, with training batch size 512 under 10 random seeds. The relative score of each tested algorithm on

[1] Mean using only the data between the first and third quartiles [70]

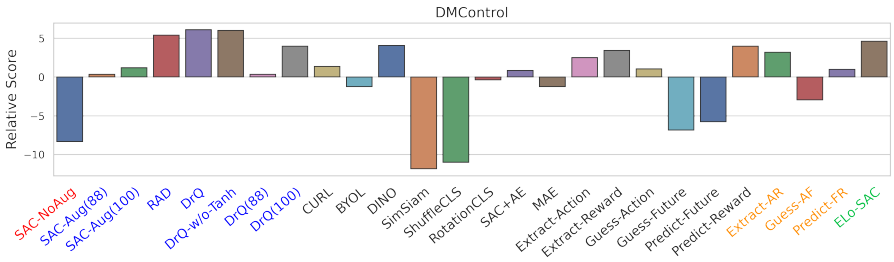


Figure 4: Relative Scores on six DMControl tasks, environment step=100k, batch size=512, Number of seeds=10. SAC-NoAug uses no image augmentation; The methods in blue (like DrQ) only take advantage of image augmentation without any SSL; the methods in black (like CURL) apply only one self-supervised loss; the methods in orange (like Extract-AR) manually combines two self-supervised losses and the two losses share one self-supervised head except the last layer; ELo-SAC combines multiple self-supervised losses with specific weights from an evolutionary search, and each loss has its individual SSL head. From this figure, **No** existing SSL-based method with the joint learning framework achieves better performance than DrQ and RAD which only use well-designed image augmentation. ELo-SAC achieves higher Relative Score than all the self-supervised methods, but it still performs worse than DrQ.

DMControl is reported as Fig. 4. We strongly encourage readers to check full results at Table 7 for a full picture.

From the first glace at Fig. 4, no tested SSL-based method within the joint learning framework achieves better performance than DrQ and RAD which are carefully designed to take the best advantage of specific image augmentations. We also find that compared to the baseline SAC-Aug(100), approaches with a single self-supervised loss frequently (11 out of 18) fail to improve reinforcement learning. Some SSL methods (like SimSiam, ShuffleCLS) ruin the policy learning resulting in performance even worse than SAC-NoAug, which suggests that an improper use of self-supervised loss can damage the benefits brought by image augmentation. Note that there are several self-supervised losses that outperform CURL in its joint learning framework, which shows some potential on studying novel self-supervised losses for RL. Table 7 further shows that except in rare cases which the performance is constantly lower than others like SimSiam and ShuffleCLS, there is no strong evidence showing that any self-supervised loss is consistently better than others across all the environments. Furthermore, how different tasks "respond" to a certain self-supervised loss can be regarded as a property, which can help us understand the similarities of different tasks and benefit the loss section.

Then, regarding combining losses, Guess-AF and Predict-FR, which are manually designed to combine two individual losses, are not better than the single self-supervised loss in their combinations (see Guess-Action and Predict-Reward). Considering that Extract-AR, Guess-AF and Predict-FR concatenate both the outputs and apply supervision by averaging loss per element of the output, the target with a higher dimension will naturally get more penalty due to the larger number of elements in the output. We further test '-Balanced' configuration, where we only modify how the supervision is applied, and this trick bring overall improvements (See Sec. A.5).

ELo-SAC finds the desired combination by searching in one task. This combination generalizes to other environments on DMControl with better overall performance than any approach in the search space. In the 'cheetah run' where the search was performed, it obtained the best result among the approaches with SSL. Further, ELo-SAC achieves the best score in 'walker, walk' than any other methods although only searched in 'cheetah run'. This demonstrates the feasibility of ELo-SAC and implies that the obtained combination through evolutionary search has a potential to generalize to other environments in DMControl. However, weaker performance in 'finger, spin' and 'reacher, easy' made ELo-SAC relatively worse than DrQ (which does not use any self-supervision) on average.

Since DrQ outperforms the other methods without SSL, we further investigate the reason behind by comparing variants of DrQ with different augmentation designs. We confirm that the success of DrQ comes not only from the averaging mechanism taking advantage of image augmentation, but also from the specific well-designed image augmentation method. Another example is the different performance of RAD and SAC-Aug(100), given the only difference between these two methods is the augmentation. We also confirm the importance of image augmentation magnitude from the comparison of DrQ(88) and DrQ(100), as well as from the comparison of SAC-Aug(88) and SAC-

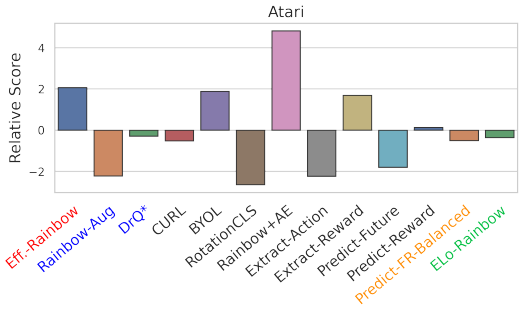


Figure 5: Relative Scores on seven Atari games, environment step=400k, batch size=32, Number of seeds=20. The color of a method reflects its category same as Fig. 4. The overall results show that image augmentation for RL does not benefit policy learning on Atari which is quite different from DMControl. Most of the self-supervised losses fail to bring improvements even given more computation and extra model capacity from the SSL head. Only Rainbow+AE outperforms Efficient Rainbow, which is inconsistent with SAC+AE. ELo-Rainbow achieves worse results even than some of the SSL-based methods in the search space like BYOL and Rainbow+AE. The high variance and the image domain gap between different games makes it extremely challenging for ELo-Rainbow to find the combined loss that generalizes to all environments.

Aug(100). Further ablations can be found at Sec. A.4. Overall, our observation suggests that it is critical to engineer image augmentation carefully when designing a RL system with or without SSL. Meanwhile, the better image augmentation for SSL + RL may need further investigations.

Atari Game Experiments Atari 2600 Games are also challenging benchmarks but with discrete action space. We choose seven games in this benchmark for selected methods. All the methods use Efficient Rainbow [65] as the backbone, which is a Rainbow [31] variant with modifications for better data-efficiency. Note that Efficient Rainbow, as a baseline, does not take advantage of image augmentation. Therefore, we also benchmark Rainbow-Aug which is essentially Efficient Rainbow taking the augmented images for policy learning instead. We use the same image augmentation and hyper-parameters reported by CURL for all applicable methods. For a fair comparison, the augmentation for DrQ* is also adopted from CURL, which is different from what the original DrQ paper suggested. We denote our setting as DrQ* to distinguish it from the original DrQ. For each game, we run 20 random seeds and benchmark the agent at 400K environment steps (100K model steps with frame skip of 4). We report interquartile mean, standard deviation and relative score same as DMControl (See Table 8).

Figure 5 shows a summary among the seven different tasks in Relative Score. Firstly, compared to vanilla baseline Efficient Rainbow which does not have any image augmentation or self-supervised learning, Rainbow-Aug performs worse overall with additional image augmentation for RL. This suggests that the image augmentation used for self-supervised learning in CURL does not easily transfer. Similarly, DrQ* achieves compromised performance than Efficient Rainbow, showing that using image augmentation for Rainbow on Atari does not benefit policy learning unlike SAC on DMControl. Based on the inconsistent impacts of image augmentation, further investigation is required when applying image augmentation to RL on Atari.

As for the self-supervised losses, BYOL, Rainbow+AE, Extract-Reward and Predict-Reward gain better performance than CURL. However, only Rainbow+AE shows significant improvement on Efficient Rainbow and outperforms all the other tested methods, which interestingly is inconsistent with SAC+AE on DMControl. Predict-FR-Balanced, which shows considerable improvements on DMControl by manually combining two self-supervised losses, even fails to surpass Predict-Reward on Atari. ELo-Rainbow, which searched in Frostbite, improves the baseline only in demon attack and frostbite. The high variance on this benchmark made the evolutionary search extremely difficult. Further, there is huge image domain gaps between games, which makes it even harder for ELo-Rainbow to work across multiple games on Atari.

Real Robot Experiments We further conduct experiments in a real-world robot environment, uArm reacher. The autonomous training environment set up and results are shown in Figs. 6 and 7. The robot fails to learn effective policy without image augmentation, while SAC-Aug(100) works better than CURL and DrQ(100). Please check Sec. A.6 for details.

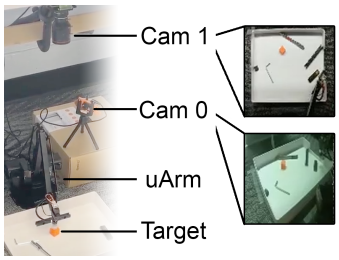


Figure 6: Real robot environment setup

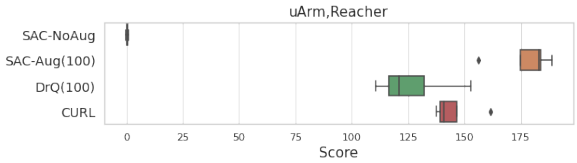


Figure 7: Scores on real robot uArm, reacher, environment step=200k, batch size=512. The agent fails to learn effective policy without image augmentation. SAC-Aug(100), using the augmented images for policy learning without any SSL, outperforms the other methods.

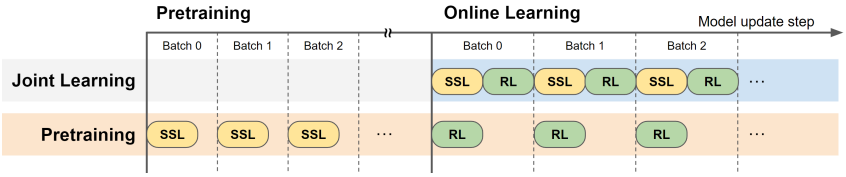


Figure 8: Comparison of two learning frameworks for SSL + RL. The rounded rectangle means updating the model with the labeled loss for one step.

Expriments with Pretraining Framework Besides the joint learning framework, we investigate the pretraining framework (shown as Fig. 8) which is widely used in previous work [58, 59, 61, 66, 77]. With the help of extra data, the pretraining framework usually works better than the joint learning framework, given the same amount of model *optimization* steps. However, the pretraining framework often achieves lower scores than the joint learning framework given the same amount of *environment* steps. Please see Sec. A.7 for more information.

5 Related Works

Self-supervised learning can fit in robot policy learning in multiple fashions and at different stages. Some works [23, 57–59, 61, 66, 71, 77] use SSL for representation learning in a pre-training stage before policy learning. Others [22, 26, 33, 40, 42, 44, 45, 50, 56, 72, 75, 76, 78, 79] jointly optimize the self-supervised loss with RL. Specifically, Transporter [39] and VAI [68] train an unsupervised keypoint detector to discovery critical objects in image for control. RRL [58] and VRL3 [66] also benefit from pre-training a deeper visual encoder on large datasets like ImageNet [15]. TCN [57] and CURL [40] take advantage of contrastive learning. After the agent is deployed, SSL can be used to continuously improve the policy [27]. Shelhamer et al. [60] study several self-supervised losses within both pretraining framework and the joint learning framework, while their selection of losses, number of runs and test environments are limited from a current point of view. Chen et al. [8] focus on imitation learning and test multiple SSL objectives for representation learning on various environments. They confirmed the critical role of image augmentation in imitation learning and showed inconsistencies of performance across environments. Our investigation supports some of their observations, beyond that, our evolving loss, real robot environment and representation analysis provide unique perspectives for online reinforcement learning.

6 Dicussion

From DMControl and the real robot experiments, we empirically show that compared to the image augmentation, the role of existing self-supervised losses with the joint learning framework is usually limited, even with the help of evolutionary search. While results on Atari show a different trend from DMControl, once again we confirm that there is no golden self-supervised loss or image augmentation that generalizes across environments. At the same time, it is usually challenging to conclude a consistent trend that one method is meaningfully better than others across multiple tasks. Though the pretraining framework can take advantage of extra data, it is also limited. One should cautiously decide the design choice of image augmentation or self-supervised loss for a specific RL task. We are excited to see future works that introduce more self-supervised losses designed specifically for RL, as well as novel training framework that can benefit policy learning.

References

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

[2] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

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

[4] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020.

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660, 2021.

[6] Lili Chen, Kimin Lee, Aravind Srinivas, and Pieter Abbeel. Improving computational efficiency in visual reinforcement learning via stored embeddings. *Advances in Neural Information Processing Systems*, 34, 2021.

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[8] Xin Chen, Sam Toyer, Cody Wild, Scott Emmons, Ian Fischer, Kuang-Huei Lee, Neel Alex, Steven H Wang, Ping Luo, Stuart Russell, et al. An empirical investigation of representation learning for imitation. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[9] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

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

[11] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised visual transformers. *arXiv e-prints*, pages arXiv–2104, 2021.

[12] Rumen Dangovski, Li Jing, Charlotte Loh, Seungwook Han, Akash Srivastava, Brian Cheung, Pulkit Agrawal, and Marin Soljacic. Equivariant self-supervised learning: Encouraging equivariance in representations. In *International Conference on Learning Representations*, 2021.

[13] Srijan Das and Michael S Ryoo. Stc-mix: Space, time, channel mixing for self-supervised video representation. *arXiv preprint arXiv:2112.03906*, 2021.

[14] Srijan Das and Michael S Ryoo. Viewclr: Learning self-supervised video representation for unseen viewpoints. *arXiv preprint arXiv:2112.03905*, 2021.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[16] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[18] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

[19] Partha Ghosh, Mehdi SM Sajjadi, Antonio Vergari, Michael Black, and Bernhard Schölkopf. From variational to deterministic autoencoders. *arXiv preprint arXiv:1903.12436*, 2019.

[20] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[21] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020.

[22] Zhaohan Daniel Guo, Bernardo Avila Pires, Bilal Piot, Jean-Bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap latent-predictive representations for multitask reinforcement learning. In *International Conference on Machine Learning*, pages 3875–3886. PMLR, 2020.

[23] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[24] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[25] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[26] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

[27] Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. *arXiv preprint arXiv:2007.04309*, 2020.

[28] Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

[29] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[30] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

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

[32] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

[33] Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. In *International Conference on Machine Learning*, pages 2117–2126. PMLR, 2018.

[34] Longlong Jing, Xiaodong Yang, Jingen Liu, and Yingli Tian. Self-supervised spatiotemporal feature learning via video rotation prediction. *arXiv preprint arXiv:1811.11387*, 2018.

[35] Kumara Kahatapitiya, Zhou Ren, Haoxiang Li, Zhenyu Wu, and Michael S Ryoo. Self-supervised pretraining with classification labels for temporal activity detection. *arXiv preprint arXiv:2111.13675*, 2021.

[36] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

[37] Dahun Kim, Donghyeon Cho, and In So Kweon. Self-supervised video representation learning with space-time cubic puzzles. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 8545–8552, 2019.

[38] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[39] Tejas D Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. *Advances in neural information processing systems*, 32, 2019.

[40] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.

[41] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in Neural Information Processing Systems*, 33:19884–19895, 2020.

[42] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33:741–752, 2020.

[43] Hankook Lee, Kibok Lee, Kimin Lee, Honglak Lee, and Jinwoo Shin. Improving transferability of representations via augmentation-aware self-supervision. *Advances in Neural Information Processing Systems*, 34, 2021.

[44] Kuang-Huei Lee, Ian Fischer, Anthony Liu, Yijie Guo, Honglak Lee, John Canny, and Sergio Guadarrama. Predictive information accelerates learning in rl. *Advances in Neural Information Processing Systems*, 33:11890–11901, 2020.

[45] Bogdan Mazoure, Remi Tachet des Combes, Thang Long Doan, Philip Bachman, and R Devon Hjelm. Deep reinforcement and infomax learning. *Advances in Neural Information Processing Systems*, 33:3686–3698, 2020.

[46] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer, 2016.

[47] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[48] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016.

[49] Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407. IEEE, May 2011.

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

[51] Tian Pan, Yibing Song, Tianyu Yang, Wenhao Jiang, and Wei Liu. Videomoco: Contrastive video representation learning with temporally adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11205–11214, 2021.

[52] Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Gupta. The unsurprising effectiveness of pre-trained vision models for control. *arXiv preprint arXiv:2203.03580*, 2022.

[53] AJ Piergiovanni, Anelia Angelova, and Michael S Ryoo. Evolving losses for unlabeled video representation learning. *arXiv preprint arXiv:1906.03248*, 2019.

[54] Kanchana Ranasinghe, Muzammal Naseer, Salman Khan, Fahad Shahbaz Khan, and Michael Ryoo. Self-supervised video transformer. *arXiv preprint arXiv:2112.01514*, 2021.

[55] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[56] Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with momentum predictive representations. *arXiv preprint arXiv:2007.05929*, 2(3), 2020.

[57] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[58] Rutav Shah and Vikash Kumar. Rrl: Resnet as representation for reinforcement learning. *arXiv preprint arXiv:2107.03380*, 2021.

[59] Jinghuan Shang and Michael S Ryoo. Self-supervised disentangled representation learning for third-person imitation learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 214–221. IEEE, 2021.

[60] Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. Loss is its own reward: Self-supervision for reinforcement learning. *arXiv preprint arXiv:1612.07307*, 2016.

[61] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pages 9870–9879. PMLR, 2021.

[62] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[63] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *arXiv preprint arXiv:2203.12602*, 2022.

[64] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[65] Hado P van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in reinforcement learning? *Advances in Neural Information Processing Systems*, 32, 2019.

[66] Che Wang, Xufang Luo, Keith Ross, and Dongsheng Li. Vrl3: A data-driven framework for visual deep reinforcement learning. *arXiv preprint arXiv:2202.10324*, 2022.

[67] Han Wang, Erfan Miahi, Martha White, Marlos C Machado, Zaheer Abbas, Raksha Kumaraswamy, Vincent Liu, and Adam White. Investigating the properties of neural network representations in reinforcement learning. *arXiv preprint arXiv:2203.15955*, 2022.

[68] Xudong Wang, Long Lian, and Stella X Yu. Unsupervised visual attention and invariance for reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6677–6687, 2021.

[69] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[70] Wikipedia contributors. Interquartile mean, 2022. URL `https://en.wikipedia.org/wiki/Interquartile_mean`. [Online; accessed 13-May-2022].

[71] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.

[72] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. *arXiv preprint arXiv:1910.01741*, 2019.

[73] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2020.

[74] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

[75] Pei Yingjun and Hou Xinwen. Learning representations in reinforcement learning: An information bottleneck approach. *arXiv preprint arXiv:1911.05695*, 2019.

[76] Tao Yu, Cuiling Lan, Wenjun Zeng, Mingxiao Feng, Zhizheng Zhang, and Zhibo Chen. Playvirtual: Augmenting cycle-consistent virtual trajectories for reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[77] Albert Zhan, Philip Zhao, Lerrel Pinto, Pieter Abbeel, and Michael Laskin. A framework for efficient robotic manipulation. *arXiv preprint arXiv:2012.07975*, 2020.

[78] Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

[79] Jinhua Zhu, Yingce Xia, Lijun Wu, Jiajun Deng, Wengang Zhou, Tao Qin, and Houqiang Li. Masked contrastive representation learning for reinforcement learning. *arXiv preprint arXiv:2010.07470*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Sec. A.9
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Please check supplemental material
   (b) Did you specify all the training details (e.g., data splits, hyper-parameters, how they were chosen)? [Yes] See Sec. A.2

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Sec. A.3

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [Yes]

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]