Local-guided Global: Paired Similarity Representation for Visual Reinforcement Learning

Hyesong Choi¹, Hunsang Lee², Wonil Song³, Sangryul Jeon⁴, Kwanghoon Sohn³, Dongbo Min¹†
¹Ewha W. University  ²Hyundai Motor Company  ³Yonsei University  ⁴University of Michigan

Abstract

Recent vision-based reinforcement learning (RL) methods have found extracting high-level features from raw pixels with self-supervised learning to be effective in learning policies. However, these methods focus on learning global representations of images, and disregard local spatial structures present in the consecutively stacked frames. In this paper, we propose a novel approach, termed self-supervised Paired Similarity Representation Learning (PSRL) for effectively encoding spatial structures in an unsupervised manner. Given the input frames, the latent volumes are first generated individually using an encoder, and they are used to capture the variance in terms of local spatial structures, i.e., correspondence maps among multiple frames. This enables for providing plenty of fine-grained samples for training the encoder of deep RL. We further attempt to learn the global semantic representations in the action aware transform module that predicts future state representations using action vectors as a medium. The proposed method imposes similarity constraints on the three latent volumes; transformed query representations by estimated pixel-wise correspondence, predicted query representations from the action aware transform model, and target representations of future state, guiding action aware transform with locality-inherent volume. Experimental results on complex tasks in Atari Games and DeepMind Control Suite demonstrate that the RL methods are significantly boosted by the proposed self-supervised learning of paired similarity representations.

1. Introduction

Deep reinforcement learning (RL) has been an appealing tool for training agents to solve various tasks including complex control and video games [12]. While most approaches have focused on training RL agent under the assumption that compact state representations are readily available, this assumption does not hold in the cases where raw visual observations (e.g. images) are used as inputs for training the deep RL agent. Learning visual features from raw pixels only using a reward function leads to limited performance and low sample efficiency.

To address this challenge, a number of deep RL approaches [1,10,38,40,43,44,46] leverage the recent advance of self-supervised learning which effectively extracts high-level features from raw pixels in an unsupervised fashion. In [38,46], they propose to train the convolutional encoder for pairs of images using a contrastive loss [24,50]. For training the RL agent, given a query and a set of keys consisting of positive and negative samples, they minimize the contrastive loss such that the query matches with the positive sample more than any of the negative samples [38,46]. While the parameters of the query encoder are updated through back-propagation using the contrastive loss [50], the parameters of the key encoder are computed with an exponential moving average (EMA) of the query encoder parameters. The output representations of the query encoder are passed to the RL algorithm for training the agent. These approaches have shown compelling performance and high sample efficiency on the complex control tasks when compared to existing image-based RL approaches [31,33,51].

While these approaches can effectively encode the global semantic representations of images with the self-supervised representation learning, there has been no attention on the local fine-grained structures present in the consecutively stacked images. Our key observation is that spatial deformation, i.e., the change in terms of the spatial structures across the consecutive frames, can provide plenty of local samples for training the RL agent. Establishing dense correspondence [19,34,39,42,55], which has been widely used for various tasks such as image registration and recognition in computer vision, can be an appropriate tool in modeling the local spatial deformation.

In this work, we propose a novel approach, termed self-supervised Paired Similarity Representation Learning (PSRL), that learns representations for deep RL by effectively encoding the spatial structures in a self-supervised...
fashion. The query representations generated from an encoder are used to predict the correspondence maps among the input frames. A correspondence aware transform is then applied to generate future representations. We further extend our framework by introducing the concept of future state prediction, originally used for action planning in RL [8, 11], into the proposed action aware transform in order to learn temporally-consistent global semantic representations. The proposed method is termed ‘Paired Similarity’ as it encodes both local and global information of agent observations. More structured details of the terms are provided in the supplementary material due to lack of space. To learn the proposed paired similarity representation, we impose similarity constraints on the three representations; transformed query representations by the estimated pixel-wise correspondence, predicted query representations from the action aware transform module, and target representations of future state. When applying the paired similarity constraint, the prediction and projection heads of global similarity constraint are shared with the local constraint head, inducing locality-inherent volume to guide the global prediction. Finally, the well-devised paired similarity representation is then used as input to the RL policy learner.

We evaluate the proposed method with two challenging benchmarks including Atari 2600 Games [31, 51] and DMControl Suite [48], which are the common benchmarks adopted to evaluate the performance of recent sample-efficient deep RL algorithms. The proposed method competes favorably compared to the state-of-the-arts in 13 out of 26 environments on Atari 2600 Games and in 4 out of 6 tasks on DMControl Suite, in terms of cumulative rewards per episode.

We highlight our contributions as follows.

- While prior approaches place emphasis only on encoding global representations, our method takes advantage of spatial deformation to learn local fine-grained structures together, providing sufficient supervision for training the encoder of deep RL.
- We propose to impose the paired similarity constraints for visual deep RL by guiding the global prediction heads with locality-inherent volume.
- We introduce the action aware transform module to self-supervised framework to learn temporally-consistent instance discriminability by using action as a medium.

2. Related Work

Self-supervised Representation Learning: The self-supervised representation learning aims to learn general features from large-scale unlabeled images or videos without expensive data annotations. The contrastive methods have achieved state-of-the-art performance in the self-supervised representation learning [2, 4, 6, 7, 15, 24, 25, 27, 49, 50, 54]. The contrastive learning aims to bring positive samples closer while separating negative samples from each other [20]. Wu et al. [54] formulate the contrastive learning as a non-parametric classification problem at the instance level, and propose to learn visual features with the memory bank and noise contrastive estimation (NCE) [16, 41]. The method in [50] proposes a probabilistic contrastive loss, called InfoNCE, for inducing representations by leveraging positive and negative samples. The InfoNCE loss has widely been adopted in [6, 24, 25, 49]. Chen et al. [6] present a simple framework for contrastive self-supervised learning without specialized architecture [2, 25] or memory bank [54], but it requires a large batch size for using enough negative samples when computing the InfoNCE loss [50]. He et al. [24] propose to build a dynamic dictionary with a queue to avoid the use of large batches when collecting negative samples, and also uses the moving averaged (momentum) encoder for target data (positive and negative samples of query data). Grill et al. [15] use the momentum encoder to produce representations of the targets as a means of stabilizing the bootstrap step. This enables for learning the representations with only positive samples, which are generated by data augmentation, for a given query without the need to carefully set up negative samples. The method in [7] further extends this idea by using only stop-gradient operation without using the momentum update. Hjelm et al. [27] propose Deep InfoMax (DIM) that learns representations by maximizing mutual information between the input and learned features from deep networks. This was extended in [2] by maximizing mutual information between features extracted from multiple images of a shared context, e.g., augmented images. While these approaches focuses on learning global representations of a single image, our method proposes to learn paired similarity representations for effectively encoding the spatial structures in the consecutive images.

Self-supervised Representation Learning in Deep RL: Representation learning is crucial for RL algorithms to learn policies with high-dimensional visual observations. Contrastive learning has been used to extract desired latent representations of visual observations used in the RL algorithms. For training robot agents, Sermanet et al. [44] present the time-contrastive networks (TCN) that train viewpoint-invariant representations using a metric learning such that multiple viewpoints of the same scene are encouraged to be close, while negative images taken from a different timestep are separated. This work was extended in [10] by embedding multiple frames at each timestep for learning task-agnostic representations such as position and velocity attributes in continuous control tasks. In [40], a new objective based on DIM [27] was presented for adapting to RL algorithms. In [1], the representations for RL algorithms are learned by maximizing mutual information.
Overall framework of the PSRL method: Multiple representations generated by the query and target encoders are used to infer a set of pixel-wise correspondence maps. The transformed representation $Z_{q,tr}^{k+1}$ is produced using an inverse warping with the set of pixel-wise correspondence maps. The action aware transform module $G$ with an action $a_k$ predicts the future representation $Z_{q,pr}^{k+1}$. The proposed method imposes paired similarity constraints on the three latent volumes, $Z_{q,tr}^{k+1}$, $Z_{q,pr}^{k+1}$ and $Z_t^{k+1}$, guiding global prediction with local spatial structure. The target encoder and projection heads are updated using the stop-gradient operation. The encoder representation $Z_q^k$ is used as an input in the RL algorithm. In our work, Rainbow DQN [51] ($M = 3$) and SAC [17] ($M = 2$) are used as RL algorithms.

### 3. Method

We consider the Markov Decision Process (MDP) setting where an agent interacts with environments in a sequence of observations, actions, and rewards. We denote $o_k, a_k,$ and optical flow estimation [9, 28, 47] have been advanced largely thanks to the expressive power of deep networks. Though both approaches share a similar objective of finding corresponding pixels across views, the optical flow is known to be effective for encoding temporal motion trajectories, while the stereo matching is tailored to predicting 3D depth map in the scene. The commonly used architecture for two-frame correspondence estimation involves the feature map extraction of two frames, correlation volume computation, a series of convolutions for refinement, and regression. Some unsupervised learning approaches have attempted to infer correspondence maps with an image reconstruction loss for imposing the constraint that corresponding pixels should have similar intensities. Note that the image reconstruction loss has also been used for self-supervised monocular depth estimation [13, 14] and stereo matching [53]. In our work, we present the self-supervised correspondence estimation network that learns fine-grained dynamics information from the consecutive frames used in the RL algorithms.

**Visual Correspondence Learning:** Visual correspondence estimation [19, 34, 42, 55] is a long-standing research in the computer vision community. It aims to establish a pair of corresponding pixels between two (or more) views taken under different locations (stereo matching) or timestep (optical flow). Recent methods for stereo matching [5, 57, 58] and optical flow estimation [9, 28, 47] have been advanced largely thanks to the expressive power of deep networks. Though both approaches share a similar objective of finding corresponding pixels across views, the optical flow is known to be effective for encoding temporal motion trajectories, while the stereo matching is tailored to predicting 3D depth map in the scene. The commonly used architecture for two-frame correspondence estimation involves the feature map extraction of two frames, correlation volume computation, a series of convolutions for refinement, and regression. Some unsupervised learning approaches have attempted to infer correspondence maps with an image reconstruction loss for imposing the constraint that corresponding pixels should have similar intensities. Note that the image reconstruction loss has also been used for self-supervised monocular depth estimation [13, 14] and stereo matching [53]. In our work, we present the self-supervised correspondence estimation network that learns fine-grained dynamics information from the consecutive frames used in the RL algorithms.
and \( r_k \) as the observation, the action of the agent, and the reward received at timestep \( k \). Since our method is a general framework that leverages the representation learning for training the RL agent, it can be combined with any RL algorithm. Following the state-of-the-arts RL approaches [38, 43, 46] using the self-supervised learning, we adopt the Soft Actor Critic (SAC) method [17] for continuous control task in DeepMind Control Suite benchmark, and Rainbow DQN [51] for discrete control task in Atari Games. The proposed self-supervised paired similarity representation learning (PSRL) is used as an auxiliary task for training RL agents.

3.1. Self-supervised Correspondence Estimation

We start with how to generate the locality-inherent representations for capturing spatial deformations from the consecutively stacked frames in a self-supervised manner. An instance used by the model-free off-policy RL algorithms [17, 51] is a stack of images, not a single image. Given an input raw observation \( o_k = \{ I_k, ..., I_{k+M} \} \) where \( I_k \) is an image at timestep \( k \), the latent encoder features \( e_k = \{ z_k, ..., z_{k+M} \} \) are first generated by applying an encoder individually to each of the input observations \( o_k \). Note that \( z \in \mathbb{R}^{h \times w \times d} \) is a 3-D volume with a spatial resolution \( h \times w \) and a feature dimension \( d \). We apply query encoder and target encoder to \( o_k \) and \( o_{k+1} \), respectively, and denote the output of the query encoder \( E^q \) as \( z^q \), and the output of the target encoder \( E^t \) as \( z^t \). While the existing methods [1, 10, 38, 40, 43, 44, 46] feeds the stacked frames to the encoder at once, which can be viewed as an early fusion [32], our method generates the set of the latent representations individually with the encoder. Later, they are fused using \( 1 \times 1 \) convolutional layer in a manner similar to a late fusion [45].

The set of representations is used to predict the spatial deformations, i.e., correspondence maps between two consecutive frames. We compute a correlation volume \( V_{a,b} \in \mathbb{R}^{h \times w \times r^2} \) using a dot product between two latent representations \( z_a \) and \( z_b \) [9] as follows:

\[
V_{a,b}(u, v, \delta) = < z_a(u + \delta), z_b(v + \delta) >, \tag{1}
\]

where \( u \) and \( v \) represent 2D feature position in \( z_a \) and \( z_b \), \( \delta \in [-\bar{r}, \bar{r}] \), and \( \bar{r} \) indicates the kernel size for computing correlation, \( r = 2\bar{r} + 1 \). Computing the patch similarity in (1) for all combinations of \( u \) and \( v \) (totally, \( h^2 \cdot w^2 \) times) causes a huge amount of computation. Thus, the maximum displacement for computing the patch similarity is limited for \( e \in \mathcal{N}(u) \) where \( \mathcal{N}(u) \) represents neighboring pixels of \( u \) within pre-defined search range.

The correlation volume is fed into a series of convolutions followed by the refinement layers, producing a correspondence map \( c_{a-b} \in \mathbb{R}^{h \times w \times 2} \) from \( I_a \) to \( I_b \). As PSRL is a fully self-supervised framework, the correspondence estimation module \( C \) is trained by self-supervised loss \( L_r \) as follows:

\[
L_r(c_{a-b}) = \sum_p |I_a(p) - I_b(p + c_{a-b})| + L_{reg}, \tag{2}
\]

where \( I(p) \) indicates an intensity at the pixel corresponding to 2D feature position \( p \). For computing the loss \( L_r \), we resize \( I_a \) and \( I_b \) to the size of the latent representations, \( h \times w \). We additionally use the Charbonnier regularization loss \( L_{reg} \) [3] for producing spatially smooth correspondence maps. In Figure 1, we denote ‘correspondence matching’ block as the self-supervised correspondence estimation module \( C \) including the correlation volume computation, the series of convolutions, and the refinement layers as in Figure 2.

3.2. Paired Similarity Representation Learning

Figure 1 illustrates the overall architecture of the proposed PSRL approach. Following the prior work on the self-supervised learning [7, 15, 24], we use the query encoder \( E^q \) with the parameters \( \theta^q \) and the target encoder \( E^t \) with the parameters \( \theta^t \) for encoding the query observation \( o_k \) and the target observation \( o_{k+1} \), respectively. While the parameters \( \theta^q \) of the query encoder are updated through back-propagation, the parameters \( \theta^t \) of the target encoder are updated with the query encoder parameters \( \theta^q \) using a stop-gradient operation [7] as \( \theta^t \leftarrow \theta^q \).

Pixel-wise Correspondence Learning and Correspondence Aware Transform (CAT): By minimizing (2), we first compute a set of \( M + 1 \) external correspondence maps \( \{ c_{k+i+1 \rightarrow k+i}^e \}_{i=0, \ldots, M} \) with the self-supervised correspondence estimation module \( C \) such that

\[
c_{k+i+1 \rightarrow k+i}^e(z_{k+i}^q) = C(z_{k+i+1}^q, z_{k+i}^q) \quad \text{for} \quad i = 0, \ldots, M. \tag{3}
\]

Note that the external correspondence map is predicted from the target feature \( z_{k+i+1}^q \) to the query feature \( z_{k+i}^q \). Then, we transform the query features \( c_{k+i}^e = \{ z_{k+i}^q, ..., z_{k+i+M}^q \} \) into the future state via the inverse warping [30] using \( M + 1 \) external correspondence maps. The transformed query features \( \{ z_{k+i+1}^q, ..., z_{k+i+M}^q \} \) are then fused using \( 1 \times 1 \) convolution, producing the transformed query representation \( Z_{k+i+1}^q \) at the timestep \( k + 1 \). As an additional exploitation of predicted volumes, we can also predict internal correspondence maps within the...
Various combinations of a projection and prediction heads and the target projection heads. When applying the paired similarity constraint, the heads of global similarity constraint are shared with the local constraint head, inducing locality-inherent volume to guide the global prediction.

query features $e^q_k = \{z^q_k, \ldots, z^q_{k+M}\}$ as $e^{\text{int}}_{a \rightarrow b} = C(z^q_a, z^q_b)$. Various combinations of $a$ and $b$ are possible for computing the internal correspondence maps, and we choose to compute a single correspondence map $e^{\text{int}}_{a \rightarrow b}$. We found that this is an appropriate choice in terms of computational efficiency and accuracy as the external correspondence maps are already used to impose the structural similarity constraint between multiple frames, and is effective in dealing with the case where the external spatial difference between two consecutive frames is relatively small. More details are presented in the supplementary material. The loss function $L_c$ for computing the internal and external correspondence maps is given as

$$L_c = L_r(e^{\text{int}}_{k \rightarrow k+M}) + \sum_{i=0}^{M} L_r(e^{\text{ext}}_{k+1 \rightarrow k+i}). \quad (4)$$

To measure the similarity between the transformed query representation $Z^q_{k+1}$ and the target representation $Z^t_{k+1}$ which is the fusion of target encoder features $e^{\text{int}}_{k+1} = \{z^q_{k+1}, \ldots, z^q_{k+M+1}\}$, we use two projection heads and one predictor. We project the two representations $Z^q_{k+1}$ and $Z^t_{k+1}$ into a smaller latent space by passing them through the query projection head $\psi^q$ with parameters $\xi^q$ and the target projection head $\psi^t$ with parameters $\xi^t$, and also apply an additional query prediction head $\phi^q$ to the query projection. The target projection head parameters $\xi^t$ are updated with the stop-gradient operation as in the target encoder, i.e., $\xi^t \leftarrow \xi^q$. The prediction loss $L_s$ is computed using the cosine similarity between the transformed query representation $y^q_{k+1}$ and the observed target representation $y^t_{k+1} = \phi^t(\psi^t(Z^t_{k+1}))$ such that

$$L_s(y^q_{k+1}, y^t_{k+1}) = -\frac{<y^q_{k+1}, y^t_{k+1}>}{\|y^q_{k+1}\| \|y^t_{k+1}\|}. \quad (5)$$

In Figure 3, we depict the module consisting of the query projection and prediction heads and the target projection heads.

**Action Aware Transform (AAT):** We further extend our method by leveraging an action aware transform module conditioned on an action. We generate the query representation $Z^q_{k+1}$ by applying $1 \times 1$ convolution to the query features $\{z^q_k, \ldots, z^q_{k+M}\}$ and then feed it into the convolutional prediction model $G$. Then, we use a single next prediction $Z^p_{k+1} = G(z^q_{k}, a_k)$ from the query representation $Z^q_{k}$. The predicted global query representation $Z^p_{k+1}$ is fed into the query projection head $\psi^q$ and the query prediction head $\phi^q$ such that $y^q_{k+1} = \phi^q(\psi^q(Z^q_{k+1}))$. Note that, $Z^p_{k+1}$ is a 3-dimensional representation and it becomes 1-dimensional vector, $y^q_{k+1}$ after passing the heads. The prediction loss is also computed using the cosine similarity loss $L_s(y^q_{k+1}, y^t_{k+1})$.

We measure the paired similarity loss $L_{sim}$ between the three representations $y^q_{k+1}, y^q_{k+1}$, and $y^t_{k+1}$ as

$$L_{sim} = L_s(y^q_{k+1}, y^t_{k+1}) + L_s(y^q_{k+1}, y^q_{k+1}) + \sum_{i=1}^{M} L_s(y^q_{k+1}, y^t_{k+i}) \quad (6)$$

We also include pixel-level $L_1$ loss on the original spatial latent space to guide the semantic loss with additional pixel-level similarity. Note that when applying $L_{sim}$, the projection and the prediction heads of global similarity constraint are shared with the local constraint head, inducing the locality-inherent volume generated from the correspondence to guide the global prediction process. Finally, the query representation $Z^q_{k+1}$ is fed into the deep RL algorithm.

**Final Loss:** The final loss function is summarized as

$$L_{total} = L_c + \alpha L_{sim} + L_{RL}(Z^q_{k}), \quad (7)$$

where $L_{RL}(Z^q_{k})$ indicates the loss of the RL algorithm which uses $Z^q_{k}$ as an input. $\alpha$ is a hyper-parameter that balances the loss function. We summarize the overall method in Algorithm 1.

### 3.3. Implementation Details

**Self-supervised Correspondence Estimation Module:** The input image $I_i$ is of $84 \times 84$ for Atari Games and DeepMind Control (DMControl) Suites. The query and target encoders generates $z^q_i, z^t_{i+1} \in \mathbb{R}^{7 \times 7 \times 64}$ ($i = k, \ldots, k + 3$) for Atari Games and $z^q_i, z^t_{i+1} \in \mathbb{R}^{32 \times 32 \times 32}$ ($i = k, \ldots, k + 2$) for DMControl Suites, respectively. The search window for computing the correlation volume $V$ is $6 \times 6$ for Atari games and DMControl Suites. The correlation volume goes through $3 \times 3$ convolution layers 3 times. The decoder is then applied to provide a dense correspondence map. The decoder includes three un-convolutional layers, consisting of un-pooling and convolution, and the coarser correspondence maps and encoder feature maps are concatenated into each.
Algorithm 1: Self-Supervised Paired Similarity Representation Learning (PSRL)

\[ E^q, E^t: \text{Query encoder, Target encoder} \]
\[ \psi^q, \psi^t: \text{Query projection head, Target projection head} \]
\[ \phi^q: \text{Query prediction head} \]
Initialize replay buffer and network parameters.

while Training do
    (1) PSRL
    Generate \( z^q_{k+1} \rightarrow z^t_{k+1} \) with \( E^q, E^t \) for \( i = 0, \ldots, M \).
    Generate query representation \( Z^q_k \) by fusing a set of query features \( z^q_{i+k} \) for \( i = 0, \ldots, M \).
    Generate target representation \( Z^t_{k+1} \) by fusing a set of target features \( z^t_{i+k+1} \) for \( i = 0, \ldots, M \).

    (1-1) Correspondence Learning
    Learn external and internal correspondences with (4).

    (1-2) Correspondence Aware Transform
    Generate transformed query representation \( Z^q_{k+1} \) with external correspondence \( z^q_{k+1} \rightarrow z^t_{k+1} \) for \( i = 0, \ldots, M \).

    (1-3) Action Aware Transform
    Generate predicted query representation \( Z^q_{k+1} \) from \( Z^q_k \) using action aware transform model \( G \).

    (2) Training
    \( Z^q_k \) goes into RL MLP head.
    Compute global and local similarities of (6) as in Figure 3.
    Optimize the networks by minimizing (7).
    Update parameters of \( E^q, \psi^q, \psi^t \) with \( E^t, \psi^q, \psi^t \).
end

un-convolutional layer.

Action aware transform Model: The action aware transform model includes two convolutional layers interweaved with ReLU and batch normalization [29], with the current representations \( Z^q_k \) and the action \( q_k \) of one-hot vector taken to each location being fed to the first convolutional layer.

Other Details: The query and target projection heads, \( \psi^q \) and \( \psi^t \), are implemented as the multi-layer perceptron (MLP). For the query prediction head \( \phi^q \), we reuse the first linear layer of the RL head. We used \( \alpha = 5 \) in (7) to balance the weight of the losses. More details are presented in the supplementary material.

4. Experimental Results

4.1. Evaluation on Atari Games

To compare the performance of the proposed method with state-of-the-arts, we chose Atari 2600 Games introduced in [31, 51] where only 100K environment steps, corresponding to two hours of gameplay experiences, are available for training data. This sample-efficient setup, which uses much less environment steps than the standard setup of 50,000K environment steps, has been adopted for evaluating the performance of recent sample-efficient deep RL algorithms [31, 33, 38, 43, 51]. We compared our results with various RL algorithms including SimPLe [31] which learns to infer its own latent representations for Atari, Data-Efficient Rainbow (DER) [51] which modifies the Rainbow hyperparameters for improving the sample efficiency, OTRainbow [33] which is an over-trained version of the Rainbow for the sample efficiency, CURL [38] which proposes the use of image augmentation with the contrastive loss [50] for self-supervised representation learning, DrQ [36] which uses the modest image augmentation to improve the sample efficiency, and SPR [43] which trains an agent to predict its own latent state representations into the future. Following the experimental setup on the above-mentioned approaches, we evaluated on 26 environments of Atari 2600 games by measuring the average return after 100K interaction steps. We trained our method with 10 random seeds, similar to other methods.

As shown in Table 1, the proposed method (PSRL) achieved the best performance on 13 of 26 environments. CURL [38] recorded the highest mean in 7 games out of 26, and SPR [43] recorded the highest mean in 11 games out of 26. PSRL has the highest mean in 13 games out of 26. It can be interpreted that the performance increase of the PSRL is not small by considering the quantitative aspects of these games. Also, among the 13 games in which PSRL has an edge, in particular, in 8 games (Alien, Assault, Gopher, Jamesbond, Krull, Kung Fu Master, Ms Pacman, and Seaquest), PSRL records a remarkably higher performance compared to other methods. Even the performance of certain games is high enough to match that of humans. This is because the proposed method of capturing the local-global spatial structure is able to derive an effective representation from the images of the specific Atari Games with various movements.

However, PSRL may not be effective for some games. In particular, PSRL did not perform well in the task ‘Pong’ in Atari Games [31]. The biggest reason for this is that there are too few discriminative spatial structures available in the game images. Therefore, we can be sure that our representation learning method, which effectively captures the spatial structure, will work particularly well for data with much more complex structural features. In other words, while most of the simple methods suffer from training with data with complex structural features, PSRL can be a good substitute for addressing this.

4.2. Evaluation on DMControl Suite

Various approaches including ours have been benchmarked on the DMControl Suite where the agent operates from pixels to evaluate challenging visual continuous control tasks [48]. We compared our results with State-SAC which supposes that the agent has access to low-level state based features, Pixel-SAC [18] which directly operates from pixels, SAC+AE [36] which uses a joint learning of SAC with
Table 1. Quantitative evaluation of state-of-the-arts on the 26 Atari games [31] after 100K time steps using 10 random seeds: Numbers in bold represent 1st ranking, PSRL achieves the best performance on 13 out of 26 environments. We compared results with SimPlE [31], Data-Efficient Rainbow (DER) [51], OverTrained Rainbow (OTrRainbow) [33], CURL [38], DrQ [36], and SPR [43].

Table 2. Quantitative evaluation of mean and standard deviation with state-of-the-arts on the DMControl suite [48] after 100K time steps and 500K time steps using 10 random seeds. Numbers in bold represent 1st ranking, and PSRL achieves the best performance on 4 out of 6 environments for 500K time steps. We compared results with state-based SAC and pixel-based SAC [18], SAC+AE [56], Dreamer [21], PlaNet [22], CURL [38], RAD [37], and DrQ [36].

β-VAE [26], VAE [35], and regularized autoencoder [52], Dreamer [21] and PlaNet [22] which learn a latent space world model, CURL [38] which uses image augmentation with the contrastive loss [50], RAD [37] and DrQ [36] which demonstrate that data augmentation can greatly improve the performance of model-free RL algorithms and achieve state-of-the-art performance on DMControl Suite. We trained our method with 10 random seeds, and the results with 5 random seeds are provided in the supplementary material.

Table 2 demonstrates that the self-supervised paired similarity representations of PSRL achieved best performance on 4 out of 6 environments for 500K time steps including Cartpole Swingup, Reacher Easy, Walker Walk and Ball in Cup Catch. In general, the performance at 500K steps after most methods converge is widely adopted for the evaluation.

When compared to the performance improvement rate of other methods, the performance increase of PSRL is significant. The performance at 100K steps is usually based on ranking, and PSRL achieves the best performance on 4 out of 6 environments for 500K time steps. We compared results with state-based SAC and pixel-based SAC [18], SAC+AE [56], Dreamer [21], PlaNet [22], CURL [38], RAD [37], and DrQ [36].

β-VAE [26], VAE [35], and regularized autoencoder [52], Dreamer [21] and PlaNet [22] which learn a latent space world model, CURL [38] which uses image augmentation with the contrastive loss [50], RAD [37] and DrQ [36] which demonstrate that data augmentation can greatly improve the performance of model-free RL algorithms and achieve state-of-the-art performance on DMControl Suite. We trained our method with 10 random seeds, and the results with 5 random seeds are provided in the supplementary material.

Table 2 demonstrates that the self-supervised paired similarity representations of PSRL achieved best performance on 4 out of 6 environments for 500K time steps including Cartpole Swingup, Reacher Easy, Walker Walk and Ball in Cup Catch. In general, the performance at 500K steps after most methods converge is widely adopted for the evaluation.

When compared to the performance improvement rate of other methods, the performance increase of PSRL is significant. The performance at 100K steps is usually based on ranking, and PSRL achieves the best performance on 4 out of 6 environments for 500K time steps. We compared results with state-based SAC and pixel-based SAC [18], SAC+AE [56], Dreamer [21], PlaNet [22], CURL [38], RAD [37], and DrQ [36].

β-VAE [26], VAE [35], and regularized autoencoder [52], Dreamer [21] and PlaNet [22] which learn a latent space world model, CURL [38] which uses image augmentation with the contrastive loss [50], RAD [37] and DrQ [36] which demonstrate that data augmentation can greatly improve the performance of model-free RL algorithms and achieve state-of-the-art performance on DMControl Suite. We trained our method with 10 random seeds, and the results with 5 random seeds are provided in the supplementary material.

Table 2 demonstrates that the self-supervised paired similarity representations of PSRL achieved best performance on 4 out of 6 environments for 500K time steps including Cartpole Swingup, Reacher Easy, Walker Walk and Ball in Cup Catch. In general, the performance at 500K steps after most methods converge is widely adopted for the evaluation.

When compared to the performance improvement rate of other methods, the performance increase of PSRL is significant. The performance at 100K steps is usually based on ranking, and PSRL achieves the best performance on 4 out of 6 environments for 500K time steps. We compared results with state-based SAC and pixel-based SAC [18], SAC+AE [56], Dreamer [21], PlaNet [22], CURL [38], RAD [37], and DrQ [36].

β-VAE [26], VAE [35], and regularized autoencoder [52], Dreamer [21] and PlaNet [22] which learn a latent space world model, CURL [38] which uses image augmentation with the contrastive loss [50], RAD [37] and DrQ [36] which demonstrate that data augmentation can greatly improve the performance of model-free RL algorithms and achieve state-of-the-art performance on DMControl Suite. We trained our method with 10 random seeds, and the results with 5 random seeds are provided in the supplementary material.

Table 2 demonstrates that the self-supervised paired similarity representations of PSRL achieved best performance on 4 out of 6 environments for 500K time steps including Cartpole Swingup, Reacher Easy, Walker Walk and Ball in Cup Catch. In general, the performance at 500K steps after most methods converge is widely adopted for the evaluation.

When compared to the performance improvement rate of other methods, the performance increase of PSRL is significant. The performance at 100K steps is usually based on ranking, and PSRL achieves the best performance on 4 out of 6 environments for 500K time steps. We compared results with state-based SAC and pixel-based SAC [18], SAC+AE [56], Dreamer [21], PlaNet [22], CURL [38], RAD [37], and DrQ [36].

β-VAE [26], VAE [35], and regularized autoencoder [52], Dreamer [21] and PlaNet [22] which learn a latent space world model, CURL [38] which uses image augmentation with the contrastive loss [50], RAD [37] and DrQ [36] which demonstrate that data augmentation can greatly improve the performance of model-free RL algorithms and achieve state-of-the-art performance on DMControl Suite. We trained our method with 10 random seeds, and the results with 5 random seeds are provided in the supplementary material.

Table 2 demonstrates that the self-supervised paired similarity representations of PSRL achieved best performance on 4 out of 6 environments for 500K time steps including Cartpole Swingup, Reacher Easy, Walker Walk and Ball in Cup Catch. In general, the performance at 500K steps after most methods converge is widely adopted for the evaluation.

When compared to the performance improvement rate of other methods, the performance increase of PSRL is significant. The performance at 100K steps is usually based on ranking, and PSRL achieves the best performance on 4 out of 6 environments for 500K time steps. We compared results with state-based SAC and pixel-based SAC [18], SAC+AE [56], Dreamer [21], PlaNet [22], CURL [38], RAD [37], and DrQ [36].

β-VAE [26], VAE [35], and regularized autoencoder [52], Dreamer [21] and PlaNet [22] which learn a latent space world model, CURL [38] which uses image augmentation with the contrastive loss [50], RAD [37] and DrQ [36] which demonstrate that data augmentation can greatly improve the performance of model-free RL algorithms and achieve state-of-the-art performance on DMControl Suite. We trained our method with 10 random seeds, and the results with 5 random seeds are provided in the supplementary material.

Table 2 demonstrates that the self-supervised paired similarity representations of PSRL achieved best performance on 4 out of 6 environments for 500K time steps including Cartpole Swingup, Reacher Easy, Walker Walk and Ball in Cup Catch. In general, the performance at 500K steps after most methods converge is widely adopted for the evaluation.

When compared to the performance improvement rate of other methods, the performance increase of PSRL is significant. The performance at 100K steps is usually based on ranking, and PSRL achieves the best performance on 4 out of 6 environments for 500K time steps. We compared results with state-based SAC and pixel-based SAC [18], SAC+AE [56], Dreamer [21], PlaNet [22], CURL [38], RAD [37], and DrQ [36].
Table 3. To study the impact of several losses, we measured the average performance over 10 random seeds according to the combinations of losses on DMControl Suite [48] with 500K time steps. Refer to section 4.3 for ‘C’, ‘C+T’, ‘P’, and ‘C+P’.

<table>
<thead>
<tr>
<th>500K step scores</th>
<th>C</th>
<th>C+T</th>
<th>P</th>
<th>C+P</th>
<th>C+T+P (PSRL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finger, Spin</td>
<td>729±110</td>
<td>757±100</td>
<td>711±64</td>
<td>768±112</td>
<td>961±121</td>
</tr>
<tr>
<td>Cartpole, Swingup</td>
<td>819±38</td>
<td>876±19</td>
<td>793±15</td>
<td>868±27</td>
<td>895±39</td>
</tr>
<tr>
<td>Reacher, Easy</td>
<td>857±45</td>
<td>901±29</td>
<td>904±58</td>
<td>922±31</td>
<td>932±41</td>
</tr>
<tr>
<td>Cheetah, Run</td>
<td>501±73</td>
<td>586±40</td>
<td>691±104</td>
<td>690±73</td>
<td>685±80</td>
</tr>
<tr>
<td>Walker, Walk</td>
<td>770±87</td>
<td>879±67</td>
<td>822±53</td>
<td>876±41</td>
<td>930±75</td>
</tr>
<tr>
<td>Ball in Cup, Catch</td>
<td>849±42</td>
<td>953±26</td>
<td>848±107</td>
<td>951±20</td>
<td>988±54</td>
</tr>
</tbody>
</table>

Table 4. To study the impact of various data augmentation, we measured the average performance over 10 random seeds according to the data augmentation on DMControl Suite [48] with 500K time steps.

<table>
<thead>
<tr>
<th>500K step scores</th>
<th>PSRL + no aug</th>
<th>PSRL + crop</th>
<th>PSRL + translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finger, Spin</td>
<td>932±115</td>
<td>915±91</td>
<td>961±121</td>
</tr>
<tr>
<td>Cartpole, Swingup</td>
<td>895±39</td>
<td>837±16</td>
<td>872±51</td>
</tr>
<tr>
<td>Reacher, Easy</td>
<td>932±114</td>
<td>833±87</td>
<td>930±83</td>
</tr>
<tr>
<td>Cheetah, Run</td>
<td>635±74</td>
<td>611±59</td>
<td>686±80</td>
</tr>
<tr>
<td>Walker, Walk</td>
<td>914±30</td>
<td>930±75</td>
<td>886±51</td>
</tr>
<tr>
<td>Ball in cup, Catch</td>
<td>962±14</td>
<td>988±54</td>
<td>946±42</td>
</tr>
</tbody>
</table>

Slightly different from the result presented in RAD [37], Cartpole Swingup and Reacher Easy achieved the best performance when no augmentation was used, Finger Spin and Cheetah Run obtained the best performance for translation, and Walker Walk and Ball in cup Catch showed the best performance for crop. Since PSRL learns correspondence in an end-to-end manner with RL algorithm, it is analyzed that the results are different from those of RAD [37].

5. Discussion and Conclusion

We have presented the self-supervised paired similarity representation learning, termed PSRL, to encode global and local spatial structures in an unsupervised manner. The correspondence maps inferred by the proposed method offer plenty of supervision for learning the fine-grained latent representations, and also compute transformed predictions at future frame. PSRL achieves state-of-the-art performance on Atari benchmark with 100K steps and DMControl Suites with 100K/500K steps. We have shown the importance of learning the paired similarity representations in improving the performance and sample-efficiency of image-based RL algorithms. We hope this can facilitate future works at various aspects for RL based on self-supervised learning. Code will be available soon.

Limitations The increase in the computational cost during training is unavoidable because PSRL additionally leverage the correspondence estimation module, but we found that the additional computational cost for training is not so significant. For training on DMControl Suite [48] up to 500K on the same GPU environment, the proposed method takes about 16 hours, whereas the state-of-the-art methods CURL [38] and SPR [43] take about 10 hours and 13 hours, respectively. Note that the original SPR paper did not provide the code implemented for DM Control Suite, so we conducted the experiments by modifying the original SPR code. Additionally, the correspondence estimation module are used only during training, and the inference process is implemented in the same manner as other methods. Therefore, the inference time of our method is exactly the same as that of the state-of-the-arts methods (CURL [38], SPR [43], DrQ [36]) as long as the same encoder for query images is used.
References


