On the Importance of Feature Decorrelation for Unsupervised Representation Learning in Reinforcement Learning

Hojoon Lee [1 2] **Koanho Lee** [1] **Dongyoon Hwang** [1] **Hyunho Lee** [2] **Byungkun Lee** [1] **Jaegul Choo** [1]

Abstract

Recently, unsupervised representation learning (URL) has improved the sample efficiency of Reinforcement Learning (RL) by pretraining a model from a large unlabeled dataset. The underlying principle of these methods is to learn temporally predictive representations by predicting future states in the latent space. However, an important challenge of this approach is the representational collapse, where the subspace of the latent representations collapses into a low-dimensional manifold. To address this issue, we propose a novel URL framework that causally predicts future states while increasing the dimension of the latent manifold by decorrelating the features in the latent space. Through extensive empirical studies, we demonstrate that our framework effectively learns predictive representations without collapse, which significantly improves the sample efficiency of state-of-the-art URL methods on the Atari 100k benchmark. The code is available at `https://github.com/dojeon-ai/SimTPR`.

1. Introduction

Deep Reinforcement Learning (RL) has made a significant advance in solving various sequential decision-making problems (Mnih et al., 2015; Silver et al., 2016; Gu et al., 2017; Raghu et al., 2017; Vinyals et al., 2019; Nie et al., 2021). The learning process of RL is inherently online, involving an iterative loop of data collection and policy optimization. However, in many real-world problems, the availability of online data collection is often limited as it can be expensive (e.g., robotics, and educational agents) or even dangerous (e.g., autonomous driving, and healthcare) (Levine et al., 2020; LeCun, 2022). To alleviate the burden of online data collection, researchers have explored the use of offline datasets by adopting the *pretrain then finetune* paradigm in RL (Oord et al., 2018; Stooke et al., 2021; Schwarzer et al., 2021; Zhang et al., 2022).

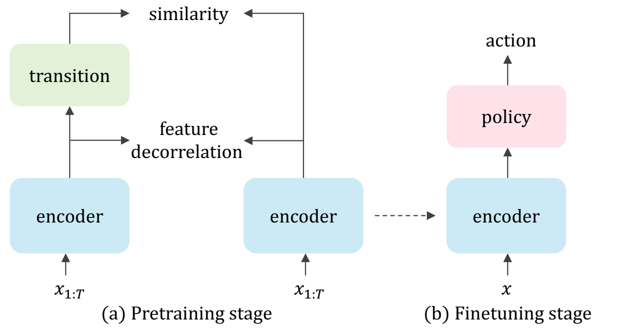


Figure 1. **SimTPR framework**. At pretraining, the encoder is trained to predict future states while decorrelating the encoded features in the latent space. Then, the pretrained encoder is used as a prior to optimize the policy at the finetuning stage.

In this paradigm, the agent trains an encoder from the offline dataset during the pretraining phase. Subsequently, during the finetuning phase, the agent leverages this encoder to efficiently optimize the policy for a downstream task with a limited number of online interactions. When the action or reward labels are present in the offline dataset, pretraining is typically performed in a supervised manner, with the goal of predicting the labels for each state (Christiano et al., 2016; Kumar et al., 2020; Fujimoto et al., 2019; Chen et al., 2021). Despite their simplicity, the acquisition of the labels is often limited, as it requires additional effort from human annotators. As a result, recent studies have focused on developing unsupervised representation learning methods from the datasets that only consist of states, which are easily accessible, large, and do not require any additional labeling process (Hafner et al., 2019c; Stooke et al., 2021; Seo et al., 2022; Baker et al., 2022).

RL has a rich history of research on unsupervised representation learning from states, with seminal studies dating back to the 1990s and early 2000s (Dayan, 1993; Tesauro et al., 1995; Littman & Sutton, 2001; Singh et al., 2003). A common principle across these studies is learning temporally predictive representations, which are obtained by training a model to predict future states using an autoregressive tran-

[1]KAIST [2]KaKao Enterprise. Correspondence to: Hojoon Lee <joonleesky@kaist.ac.kr>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

sition model. However, when states are high-dimensional data, such as images, predicting the raw, high-dimensional state may require an overly complex model to reconstruct fine-grained details of the states. Therefore, recent studies have focused on predicting future states in the latent space. They utilized a Siamese model to encode future states into latent representations and trained the model to maximize the similarity between the predicted future latents and ground truth future latents (Oord et al., 2018; Schwarzer et al., 2020; Stooke et al., 2021; Schwarzer et al., 2021).

When maximizing the latent representations' similarity with a Siamese model, there exists a potential pitfall known as *representational collapse*, wherein the subspace of the latent representations collapses into a low-dimensional manifold (i.e., low feature rank) (Chen & He, 2021; Hua et al., 2021). Having a sufficiently high feature rank is an important factor for the pretrained model to learn downstream tasks, as the feature rank decides the representation power of the model to adapt to new tasks (Lyle et al., 2022; Garrido et al., 2022). In the extreme case where the feature rank is 0, the encoder will output the same vector for all input states, making it infeasible to learn the policy for downstream tasks.

To prevent representational collapse, contrastive learning (Anand et al., 2019; Laskin et al., 2020; Stooke et al., 2021; Oord et al., 2018) or batch-normalization with stop-gradient operation (Schwarzer et al., 2020; 2021) are commonly employed in unsupervised representation learning. They learn temporally predictive representations without representational collapse by *repulsing* the representation of different states within the same mini-batch while maximizing the similarity between predictions and future states. However, this repulsion is known to negatively impact representation quality when there exist relevant states in the mini-batch (Grill et al., 2020; Chen & He, 2021; Khosla et al., 2020). This problem can be amplified in the unsupervised state representation for RL since each mini-batch generally consists of a batch of consecutive, relevant states.

In response, we introduce a simple temporally predictive representation learning framework for RL (SimTPR), which prevents representational collapse without repulsing the representations of different states in the mini-batch. Instead, motivated from (Cogswell et al., 2016; Hua et al., 2021; Zbontar et al., 2021), we directly maximize the feature rank of the latent manifold by standardizing the cross-correlation matrix between the predictions and the targets in the latent space. By enforcing the off-diagonal entries of the cross-correlation matrix to 0, this objective makes features to be independent and increases the feature rank of the latent representation's manifold. An overview of SimTPR is illustrated in Figure 1.

Through extensive experiments, we show that SimTPR achieved state-of-the-art performance in unsupervised representation learning on the Atari benchmark. Following the evaluation protocol from (Schwarzer et al., 2021; Zhang et al., 2022), we finetuned an MLP-based policy layer for 100k steps on top of the frozen encoder. When pretrained from a state dataset, SimTPR achieved a human-normalized IQM score of 0.451, representing a 10% improvement over the previous best, unsupervised state representation learning method, ATC (Stooke et al., 2021). Furthermore, through empirical studies, we discovered that feature rank is an important factor that affects the performance of downstream tasks. Also, we found that increasing the feature rank with the repulsive methods can harm the representation quality as it pushes away the relevant states in the mini-batch.

2. Related Work

2.1. Unsupervised Representation Learning for Reinforcement Learning.

Unsupervised state representation learning has a long history of research in RL, where a common underlying principle is to learn temporally predictive representations (Dayan, 1993; Tesauro et al., 1995; Littman & Sutton, 2001; Singh et al., 2003). To learn temporally predictive representations, models are generally trained to predict future states using an autoregressive transition model. When the state consists of high-dimensional data, such as images, high-capacity models are introduced to precisely predict future state information (Doerr et al., 2018; Buesing et al., 2018; Hafner et al., 2019b). However, predicting the full-state information may result in representations that contain information that is irrelevant to finding an optimal policy. For example, consider a task of maze navigation, where a TV is placed in the maze displaying random images. Although the TV content is irrelevant to the task, the model aims to learn representations to predict the screen of TV. To mitigate this issue, recent works have focused on predicting the latent information of future states, in which a Siamese model is adopted to encode the future states into latent representations. Then, the model is trained to maximize the similarity between the predictions and the future states in the latent space (Oord et al., 2018; Hafner et al., 2019a; Stooke et al., 2021; Schwarzer et al., 2020; 2021).

However, solely maximizing similarity can result in a representational collapse, in which the subspace of the latent representations collapses into a low-dimensional manifold or even to a constant. To prevent representational collapse, recent studies have employed the contrastive learning (Anand et al., 2019; Laskin et al., 2020; Oord et al., 2018; Stooke et al., 2021) or architectural designs including the use of batch-normalization and stop-gradient operation (Schwarzer et al., 2020; 2021). They prevent representational collapse by repulsing the representations within the same mini-batch. While effective in preventing representational collapse, this repulsion has been noted to have a risk of pushing away semantically relevant states (Grill et al., 2020; Chen & He, 2021; Khosla et al., 2020). This problem can be further amplified in state representation learning, where each mini-batch generally consists of a batch of consecutive states (Schwarzer et al., 2020).

In light of these limitations, we propose an alternative objective, referred to as the *feature decorrelation* loss. The feature decorrelation loss aims to mitigate representational collapse by directly maximizing the dimensions of the latent space's manifold, without pushing away representations within a given mini-batch.

2.2. Unsupervised Visual Representation Learning.

Unsupervised visual representation learning aims to learn rich representations from high-dimensional, unlabeled images. Among these approaches, invariance learning is the state-of-the-art representation learning method (Chen et al., 2020; He et al., 2020; Grill et al., 2020; Chen & He, 2021; Caron et al., 2020). Invariance learning involves maximizing the similarity between representations of two differently augmented views of the same image in the latent space. However, solely maximizing the similarity may result in a representational collapse (Hua et al., 2021; Wang & Liu, 2021; Jing et al., 2022).

To avoid representational collapse, contrastive learning (Chen et al., 2020; He et al., 2020) or architectural variants with batch normalization and stop-gradient operation (Grill et al., 2020; Richemond et al., 2020; Chen & He, 2021) have been proposed. They prevent the collapse by repulsing the representations of different images in the mini-batch. Recently, another line of research has explored the use of feature decorrelation objectives in unsupervised visual representation learning (Huang et al., 2020; Zbontar et al., 2021; Hua et al., 2021; Bardes et al., 2021). This approach aims to maximize the dimensionality of the latent manifold by standardizing the covariance matrix of the representations in the mini-batch (Huang et al., 2018; Kessy et al., 2018). This alternative approach has achieved competitive performance, compared to state-of-the-art methods.

3. Method

As illustrated in Figure 2, SimTPR aims to learn temporally predictive representations by causally predicting the future states within the latent space. To achieve this, SimTPR utilizes an autoregressive transition model to predict future states and maximizes the similarity between the predictions and the future states in the latent space. Then, SimTPR integrates a feature decorrelation loss that prevents representational collapse by standardizing the cross-correlation matrix of the representations in the latent space.

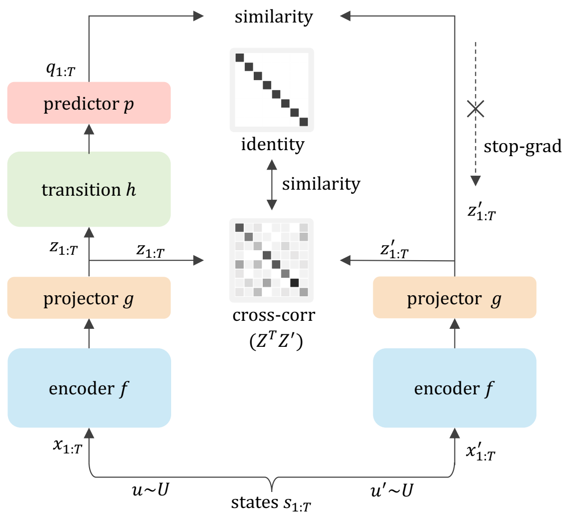


Figure 2. **SimTPR architecture.** Two augmented views of a sequence of states are processed with the same encoder f and the same projection MLP g. Then a transition model h and prediction MLP p are applied on one side to predict the future states in latent space. While the model maximizes the similarity, it simultaneously standardizes the cross-correlation matrix to prevent collapse.

Pretraining from State. Given a dataset of states, $\mathcal{S}$, we uniformly sample a sequence of T states $s_{1:T} \sim \mathcal{S}$. Then, we sample two different augmentation functions, u and u', from augmentation distribution $\mathcal{U}$. By applying augmentations, u and u', to each sequence of states, $s_{1:T}$, we generate two different views, $x_{1:T} \triangleq u(s_{1:T})$ and $x'_{1:T} \triangleq u'(s_{1:T})$.

Two augmented views, $x_{1:T}$ and $x'_{1:T}$, are then processed by four different model components as follows:

- A encoder, f, which encodes representation vectors from augmented states. Our framework allows various choices of the network without any constraints. Following previous work (Schwarzer et al., 2021), we use a 36-layer convolutional network as an encoder, which is the modified version of EfficientNet (Tan & Le, 2019).
- A projector, g, which maps encoded representations to the d-dimensional latent space. Following (Chen et al., 2020; Schwarzer et al., 2021), we use a simple one-hidden layer MLP with ReLU activation. For each augmented view, we obtain $z_{1:T} = g(f(x_{1:T}))$ and $z'_{1:T} = g(f(x'_{1:T}))$ where $z_{1:T}, z'_{1:T} \in \mathbb{R}^{T \times d}$.
- A transition model, h, which maps states' latent representations to future time step latent representations. We use a causal transformer (Radford et al., 2018) to model the transition dynamics.
- A predictor, p, that predicts the representations of the future states in the d-dimensional latent space. We use a one-hidden layer MLP with ReLU activation. By applying the transition model followed by the predictor, we obtain $q_{1:T} = p(h(z_{1:T}))$ where $q_{1:T} \in \mathbb{R}^{T \times d}$.

Now, we define our loss function, which composes the *similarity* and *feature decorrelation* loss.

In training, the inputs are processed in batches of N sequences. We denote the batch of projected representations as $Z = [z_{1,1:T}, ..., z_{N,1:T}]$, $Z' = [z'_{1,1:T}, ..., z'_{N,1:T}]$, the predicted representations of Z and Z' as $Q = [q_{1,1:T}, ..., q_{N,1:T}]$, and $Q' = [q'_{1,1:T}, ..., q'_{N,1:T}]$ respectively, where $Z, Z', Q, Q' \in \mathbb{R}^{N \times T \times d}$. We then ℓ_2-normalize each state representation from Z, Z', Q, Q' to $\bar{Z}, \bar{Z}', \bar{Q}, \bar{Q}'$, where $\bar{z}_{i,j} = z_{i,j}/\|z_{i,j}\|_2$, $\bar{z}'_{i,j} = z'_{i,j}/\|z'_{i,j}\|_2$, $\bar{q}_{i,j} = q_{i,j}/\|q_{i,j}\|_2$ and $\bar{q}'_{i,j} = q'_{i,j}/\|q'_{i,j}\|_2$.

First, following (Grill et al., 2020; Chen & He, 2021), we maximize the *similarity* between the predictions and the targets by minimizing their mean squared error as:

$$\mathcal{D}(\bar{Q}, \bar{Z}') = -\frac{1}{N(T-k)} \sum_{n=1}^{N} \sum_{t=1}^{T-k} \|\bar{q}_{n,t} - \bar{z}'_{n,t+k}\|_2^2, \quad (1)$$

where k denotes the number of future steps to associate the predictions. For the sake of simplicity, we set $k = 1$.

This similarity loss can be also interpreted as maximizing the cosine-similarity between the un-normalized representations: Q and Z (Chen & He, 2021; Grill et al., 2020).

Then, we symmetrized a loss function as (Grill et al., 2020):

$$\mathcal{L}_{\text{sim}} = \frac{1}{2}\mathcal{D}(\bar{Q}, \text{sg}(\bar{Z}')) + \frac{1}{2}\mathcal{D}(\bar{Q}', \text{sg}(\bar{Z})), \quad (2)$$

where $\text{sg}(\cdot)$ is stop-gradient operation.

Second, we define a feature decorrelation loss, $\mathcal{L}_{\text{decorr}}$, which maximizes the dimensions of the latent representation's manifold by standardizing the cross-correlation matrix.

The cross-correlation matrix, $\mathcal{C}$, between $\bar{Z}$ and $\bar{Z}'$ is:

$$\mathcal{C}(\bar{Z}, \bar{Z}')_{i,j} = \frac{\sum_{n=1}^{N} \sum_{t=1}^{T} \bar{z}_{n,t,i} \cdot \bar{z}'_{n,t,j}}{\sqrt{\sum_{n=1}^{N} \sum_{t=1}^{T} (\bar{z}_{n,t,i})^2} \sqrt{\sum_{n=1}^{N} \sum_{t=1}^{T} (\bar{z}'_{n,t,j})^2}} \quad (3)$$

where i and j indicate the dimension of the vectors.

Then, following (Zbontar et al., 2021), we standardize the cross-correlation matrix as:

$$\mathcal{L}_{\text{decorr}} = \sum_{i} (1 - \mathcal{C}_{ii})^2 + \lambda_o \sum_{i} \sum_{j \neq i} \mathcal{C}_{ij}^2 \quad (4)$$

where λ_o controls the importance between two terms.

Intuitively, the first term in equation 4 generates low-variance features by enforcing the on-diagonal terms to be 1. The second term in equation 4 generates features that are independent of each other by enforcing the off-diagonal terms to be 0. While the second term is a crucial component to prevent the representational collapse, the first term is also important to induce the encoded representations to be not dependent on a few, high-variance features.

The overall loss function, $\mathcal{L}_{\text{state}}$, is a weighted average of the $\mathcal{L}_{\text{sim}}$ and $\mathcal{L}_{\text{decorr}}$, as:

$$\mathcal{L} = \mathcal{L}_{\text{sim}} + \lambda_d \mathcal{L}_{\text{decorr}}, \quad (5)$$

where λ_d controls the importance between two terms. The pseudo-code is described in Algorithm 1.

Pretraining from Demonstration. In the case where our dataset contains action labels, we can simply extend our framework to further predict the action labels on top of the future state prediction task. Here, we slightly modify the transition model's input to incorporate the action information and initialize another predictor model to predict the action labels. We refer to this action predictor as r and use a one-hidden layer MLP with ReLU activation that outputs the number of actions, n_a. The rest of the components remain identical to the state pretraining setup.

Given a dataset of states, $\mathcal{S}$, and actions, $\mathcal{A}$, we sample a sequence of states $s_{1:T} \sim \mathcal{S}$, with their corresponding actions $a_{1:T} \sim \mathcal{A}$. The actions are then linearly embedded into d-dimensional vectors, represented as $y_{1:T} \in \mathbb{R}^{T \times d}$.

Then, we construct the trajectory representation as $\tau = [z_1, y_1, ..., z_T, y_T] \in \mathbb{R}^{2T \times d}$. By applying the transition model h, we obtain the context representations, $c = h(\tau)$. This context representation is then decomposed into state and action representations which are processed by the state predictor p, and the action predictor r respectively as, $q_{1:T} = p(\{c_i\}_{i=2,4,...,2T})$ and $l_{1:T} = r(\{c_i\}_{i=1,3,...,2T-1})$.

In training, the inputs are processed in batches of N sequences. We denote the batch of action prediction logits as $L = [l_{1,1:T}, ..., l_{N,1:T}]$ and the action labels as $A = [a_{1,1:T}, ..., a_{N,1:T}]$, where $L \in \mathbb{R}^{N \times T \times n_a}$ and $A \in \mathbb{R}^{N \times T}$.

Then, we define the action prediction loss as:

$$\mathcal{L}_{\text{act}} = -\frac{1}{NT} \sum_{n=1}^{N} \sum_{t=1}^{T} \log\left(\frac{\exp(l_{n,t,a_{n,t}})}{\sum_{i=1}^{n_a} \exp(l_{n,t,i})}\right) \quad (6)$$

which minimizes the negative log-likelihood of the action for each state.

The overall loss function, $\mathcal{L}$, is a weighted average of the $\mathcal{L}_{\text{sim}}$, $\mathcal{L}_{\text{decorr}}$, and $\mathcal{L}_{\text{act}}$, as:

$$\mathcal{L} = \mathcal{L}_{\text{sim}} + \lambda_d \mathcal{L}_{\text{decorr}} + \lambda_a \mathcal{L}_{\text{act}}, \quad (7)$$

where λ_a controls the importance of the action prediction loss. The pseudo-code is described in Algorithm 2.

4. Experiments

4.1. Experimental Setup

Here, we present the experimental setups. A detailed explanation for each setup is described in Appendix A.

In this study, we evaluate our method using the Atari100k benchmark (Bellemare et al., 2013). This benchmark is widely used to measure the sample efficiency of Reinforcement Learning (RL) algorithms (Yarats et al., 2020; Schwarzer et al., 2020; Laskin et al., 2020). Following (Schwarzer et al., 2021; Zhang et al., 2022), we pretrain the models on 26 different games in the Atari environment and finetune each model for 100k steps of online interactions.

Pre-training. We follow the pretraining protocol from (Schwarzer et al., 2021; Zhang et al., 2022), where we use the publicly-available DQN replay dataset (Agarwal et al., 2020). It contains the training logs of a DQN agent for 50 million steps. We select 1.5 million frames from the 3.5 to 5 million steps of the replay dataset as Zhang et al. (2022). This selection corresponds to the training logs of a weak, partially trained agent, which reflects the nature of publicly available datasets in internet platforms that generally contain sub-optimal trajectories (Baker et al., 2022).

As our default configuration, we use two simple image augmentations used in (Schwarzer et al., 2020; 2021; Zhang et al., 2022; Stooke et al., 2021) which are *random shift* followed by an *intensity jittering*. For the encoder, we use a 30-layer convolutional network from (Schwarzer et al., 2021) and a 2-layer MLP as a projector, which projects to a 512-dimensional latent space. For the transition model, we use two layers of causal transformer block (Radford et al., 2018) with 512 hidden dimensions. The predictor network is also a 2-layer MLP which maps to a 512-dimensional latent space. To balance the weights between the loss functions, we set $\lambda_o = 0.005, \lambda_d = 0.01$, and $\lambda_a = 1.0$. For each training step, a training batch consists of 640 samples with $N = 64$ and $T = 10$. We use an AdamW optimizer (Loshchilov & Hutter, 2017) with a learning rate of 3×10^{-4} and a weight decay of 10^{-6} and train for 100 epochs.

To identify representational collapse, we measure the feature rank (Feat.Rank) of the latent representations during the training process. In training, we randomly sample n states and feed them into the model to obtain the projection matrix, $Z \in \mathbb{R}^{n \times d}$. Then, we perform the singular value decomposition to the projection matrix, Z, and count the number of singular values of its diagonal matrix that are larger than a constant, ϵ. This estimation measures the dimension of the subspace spanned by the features in Z, after removing the highly correlated features (Golub et al., 1976; Lyle et al., 2022). We use $n = 1000$ and $\epsilon = 0.01$.

Linear Probing. To evaluate the learned representations, we follow the linear evaluation protocol from (Zhang et al., 2022). After pretraining, we train a linear classifier on top of the frozen encoder and predict the reward or action of the expert. For the expert dataset, we use the last 100k frames from the DQN replay dataset with a 4:1 train/eval split.

For reward prediction, we simplify the problem to a binary classification task and predict whether a reward has occurred or not in a given state. We report the mean F1 score across environments (Rew F1). For action prediction, we train a multiclass classifier and report the multiclass F1 score across environments (Act F1). These scores serve as a proxy to measure whether the agent can effectively learn the policy and the value function of the experts.

Finetuning. For each environment, we finetune the pretrained model for 100k steps. We follow the protocol from (Schwarzer et al., 2021; Zhang et al., 2022), where we train an MLP-based Q-learning policy on top of the frozen encoder using the Rainbow algorithm (Hessel et al., 2018). We do not use any other auxiliary loss during finetuning.

For each game, we compute the average score of 50 trajectories, evaluated at the end of the training. Then, this score is normalized as HNS=$\frac{\text{agent\_score - random\_score}}{\text{human\_score - random\_score}}$, which measures the relative performance to humans. To reduce the variance of each run, this score is averaged over 5 random seeds for our empirical studies and 10 random seeds for our main results, including all baselines. Following the guidelines of (Agarwal et al., 2021), we report a bootstrapped interval for the mean, median, interquartile mean (IQM), and optimality gap (OG) of HNS over 26 games. OG estimates the gap of the average probability to satisfy the HNS by 1.0.

4.2. Empirical Study

In this section, we conduct empirical studies to investigate the importance of the feature rank and the potential pitfalls of the repulsive methods in maximizing the feature rank. We focus on unsupervised representation learning where the encoder is pretrained from the state dataset.

Effect of Feature Decorrelation. First, we analyze the effect of the decorrelation loss by varying the decorrelation strength λ_d for SimTPR. We fix the remaining hyperparameters to its optimal configuration.

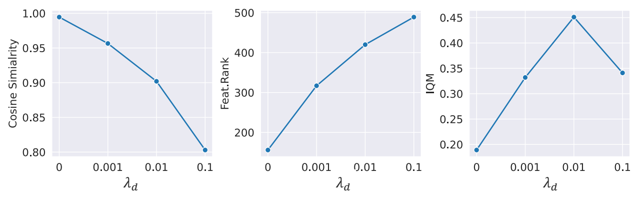


Figure 3. **Effect of decorrelation strength (λ_d).** There exists a trade-off between the similarity loss and the decorrelation loss where SimTPR achieved the highest IQM when $\lambda_d = 0.01$.

Figure 3 illustrates the effect of the decorrelation strength on the representation similarity (i.e., the cosine similarity between the predicted states and future states), feature rank, and finetuning performance of SimTPR. In Figure 3, we observed that stronger decorrelation leads to an increase in the feature rank while it leads to a decrease in the predictive similarity. Among the variants, the SimTPR with a moderate decorrelation strength ($\lambda_d = 0.01$) shows the highest finetuning performance, with an IQM score of 0.451. These empirical findings reveal that there exists a trade-off between learning temporally predictive representations and feature decorrelation. Thus, it is important to balance these two terms to achieve good finetuning performance.

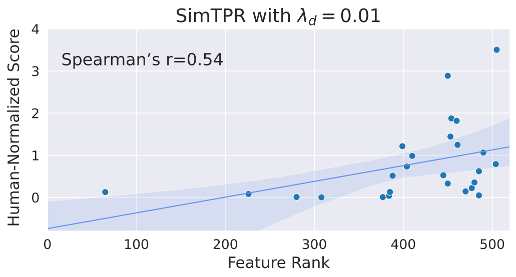


Figure 4. **Relationship between feature rank and finetuning performance.** The figure presents the human-normalized score of each game, plotted against the corresponding feature rank of the pretrained model. While the feature rank is not the sole determinant of agent performance, there exists a positive correlation between feature rank and human-normalized score.

In Figure 4, we illustrate the human-normalized score for each individual Atari game along with their respective feature rank of the pretrained encoders. Here, we use SimTPR with decorrelation strength of $\lambda_d = 0.01$, The scatterplot in the figure illustrates that there exists a clear correlation between feature rank and human-normalized scores, as quantified by Spearman's correlation coefficient of $r = 0.54$. While there exist several games with low fine-tuning performance with high feature rank, all the games with low feature rank (≤ 300) show low fine-tuning performance.

In summary, our findings indicate that although there are numerous factors (e.g., credit assignments, learning algorithm, optimizer) that affect the learning process of the RL agent, having a sufficiently high feature rank is an important factor to attain good finetuning performance.

Feature Decorrelation vs Repulsion. In this section, we compare the feature decorrelation loss to repulsion-based methods. Specifically, we consider two repulsion-based methods: contrastive learning and batch normalization (BN). These methods prevent collapse by repulsing the representations of different states in a mini-batch. For contrastive learning, we replace the similarity loss $\mathcal{L}_{\text{sim}}$ with the contrastive loss $\mathcal{L}_{\text{cont}}$ as in (Chen et al., 2020) and vary the repulsion temperature from $\{0.05, 0.1, 0.2\}$ to select the best trade-off between similarity maximization and repulsion. We use the temperature of 0.1 which yields the highest finetuning performance. For BN, we apply the BN layer to the output of the first layer of the projector.

Table 1. **Comparison of collapse prevention methods**. The latent dimension is 512. The grey row indicates our default setup.

	$\mathcal{L}_{\text{cont}}$	batch-norm	$\mathcal{L}_{\text{decorr}}$	Feat.Rank	IQM ↑
(a)	-	-	-	156	0.189
(b)	✓	-	-	443	0.331
(c)	-	✓	-	305	0.068
(d)	-	-	✓	421	**0.451**
(e)	✓	-	✓	472	0.342
(f)	-	✓	✓	500	0.264

Table 1 reports the finetuning performance for different configurations of collapse prevention methods. By comparing Table 1.(a) and (b), we observed that using the contrastive loss effectively alleviates representational collapse (Feat.Rank=443), resulting in an improved IQM score of 0.331. In Table 1.(c), we found that although batch-normalization alleviates the collapse (Feat.Rank=305), it results in a significant drop in finetuning performance, with 0.068 IQM. We speculate that applying batch-normalization brings extra difficulties in training the transition model, as it makes the input, $z_{1:T}$, and the target, $z'_{1:T}$, of the transition model to fluctuate depending on the distribution of the mini-batch. Among the variants, SimTPR with the decorrelation loss shows the best finetuning performance with 0.451 IQM (Table 1.(d)). From Table 1.(e),(f) we observed that further using contrastive loss or batch normalization leads to a higher feature rank but decreases the finetuning performance with an IQM of 0.342 and 0.260 respectively.

Although both contrastive and feature decorrelation loss has shown to effectively prevent representation collapse (i.e., high feature rank), we observed a large gap of IQM scores between these methods (Table 1.(b),(d)). We hypothesize that this discrepancy stems from the repulsive force of the contrastive loss which may even separate the relevant states in the mini-batch. To validate our hypothesis, we conduct an in-depth comparison of the pretrained representations of the contrastive loss to feature decorrelation loss.

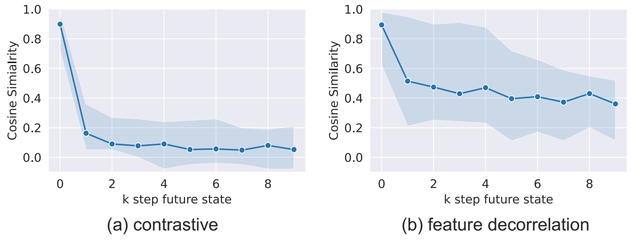
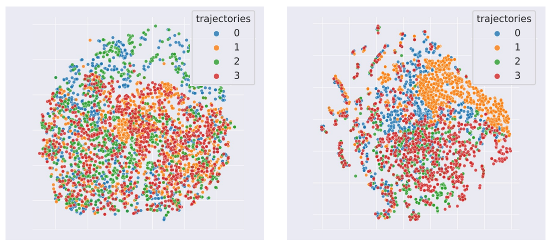


Figure 5. **Cosine Similarity between state representations**. The figure presents the average cosine similarity across 26 Atari games.

Table 2. **Finetuning on Atari-100k benchmark**. In pre-training, all methods are trained for 100 epochs with DQN replay dataset. Then, we finetuned a 2-layer Q-learning policy on top of the frozen encoder for 10 random seeds. We report 95% Confidence Intervals (CI) of IQM, Median, Mean, and Optimality Gap scores where CIs are estimated using the percentile bootstrap with stratified sampling. All competitors except SGI (Schwarzer et al., 2021) and BarlowBalance (Zhang et al., 2022) based on our reproduction.

Method	Act F1 ↑	Rew F1 ↑	IQM ↑	Median ↑	Mean ↑	Optimality Gap ↓
No Pretraining						
DrQ (Yarats et al., 2020)	-	-	0.161 (0.149, 0.174)	0.157 (0.129, 0.184)	0.308 (0.285, 0.332)	0.724 (0.710, 0.737)
Pretrained from State						
VAE (Kingma & Welling, 2014)	22.1	56.8	0.266 (0.236, 0.301)	0.266 (0.198, 0.293)	0.556 (0.498, 0.620)	0.623 (0.600, 0.645)
BarlowTwins (Zbontar et al., 2021)	21.3	58.0	0.224 (0.203, 0.247)	0.249 (0.204, 0.289)	0.448 (0.410, 0.488)	0.662 (0.644, 0.680)
CURL (Laskin et al., 2020)	15.5	52.5	0.247 (0.222, 0.273)	0.208 (0.185, 0.264)	0.426 (0.400, 0.452)	0.654 (0.636, 0.673)
RSSM (Hafner et al., 2019b)	23.6	61.1	0.302 (0.255, 0.348)	0.235 (0.199, 0.288)	0.707 (0.611, 0.811)	0.595 (0.571, 0.620)
ATC (Stooke et al., 2021)	25.8	65.6	0.353 (0.325, 0.384)	0.376 (0.262, 0.409)	0.647 (0.585, 0.714)	0.570 (0.553, 0.586)
SimTPR (ours)	**25.9**	**67.7**	**0.451** (0.410, 0.494)	**0.434** (0.317, 0.507)	**0.773** (0.707, 0.837)	**0.522** (0.503, 0.541)
Pretrained from Demonstration						
BC	**27.1**	66.2	0.413 (0.376, 0.450)	0.344 (0.289, 0.402)	0.705 (0.660, 0.751)	0.536 (0.517, 0.555)
IDM	26.5	64.7	0.343 (0.308, 0.380)	0.279 (0.237, 0.337)	0.564 (0.520, 0.614)	0.579 (0.556, 0.601)
SGI (Schwarzer et al., 2021)	26.8	64.0	0.380 (0.329, 0.436)	0.490 (0.355, 0.573)	0.751 (0.678, 0.828)	0.557 (0.530, 0.587)
BarlowBalance (Zhang et al., 2022)	22.7	61.3	0.338 (0.296, 0.382)	0.201 (0.131, 0.316)	**1.089** (0.983, 1.203)	0.588 (0.569, 0.607)
SimTPR (ours)	26.9	**71.2**	**0.500** (0.464, 0.537)	**0.515** (0.381, 0.572)	0.757 (0.715, 0.800)	**0.493** (0.476, 0.511)

First, we investigate the cosine similarity between the latent representation, z_t, and the k-step distant latent representations, z_{t+k}, of the pretrained encoder. In Figure 5, we plot the average similarity of the latent representations w.r.t k. As the distance k increases, we observed a drastic decrement of similarity for contrastive loss whereas the feature decorrelation loss was able to maintain a relatively higher level of similarity.



(a) contrastive (b) feature decorrelation

Figure 6. **t-SNE visualization.** We visualize the t-SNE plot of the latent representations on the Pong environment pretrained with (a) contrastive loss and (b) feature decorrelation loss. We use 4 distinct trajectories, each with a length of $T = 1000$.

In Figure 6, we further visualize the t-SNE plot of the latent representations. We use the Pong environment and visualize the four distinct trajectories with 1000 steps for each trajectory. While the latent representations of the contrastive loss were relatively spread out regardless of the trajectories, the latent representations of the feature decorrelation loss were clustered based on the trajectories.

In summary, we found that although repulsive methods can effectively prevent representational collapse, they push away the representations of the consecutive, relevant states. We believe that this repulsion between the consecutive states is a key factor that brings the decrement of the performance for the contrastive loss to the decorrelation loss.

4.3. Main Result

In Table 4.2, we compare SimTPR with state-of-the-art unsupervised representation learning methods in RL. To ensure a fair comparison, we fix the encoder, latent dimension, augmentation, pretraining data, and the number of pretraining steps across all methods. The results for SGI and BarlowBalance were obtained by running their official code, whereas the results for the remaining baselines are based on our reproduced results. When reproducing baselines, we strictly followed the training recipes outlined in the original papers. Further details are described in Appendix B.

When pretrained from state datasets, we observed that pretraining consistently led to improved performance compared to learning from scratch. In addition, we discover that methods that learn temporally predictive representations (i.e., RSSM, ATC, SimTPR) achieved superior performance compared to methods that do not learn any temporal dynamics (i.e., VAE, BarlowTwins, CURL). Among these methods, SimTPR achieved the highest scores in all metrics, with an improvement of 10% in IQM (0.451) over the previous best method, ATC (0.353).

Training with demonstration datasets generally leads to enhanced finetuning performance compared to the methods with pretraining from states, since the action labels provide additional information to learn good representations. By leveraging action labels, the finetuning performance of

SimTPR increases to an IQM score of 0.500, which is the best result among the demonstration pretraining methods. Although BarlowBalance achieved the highest mean score, this is due to a few games with exceptionally high human normalized scores (HNS of 6.7 in the game Up'n Down).

5. Discussion

Here, we present ablations and discussions for SimTPR to give an intuition of its behavior and performance. In this section, we focus on pretraining SimTPR from states.

5.1. Robustness to Batch Size and Latent Dimension

First, we investigate the robustness of SimTPR w.r.t different batch sizes and latent dimensions. We maintained the other hyperparameters identical to their optimal configurations.

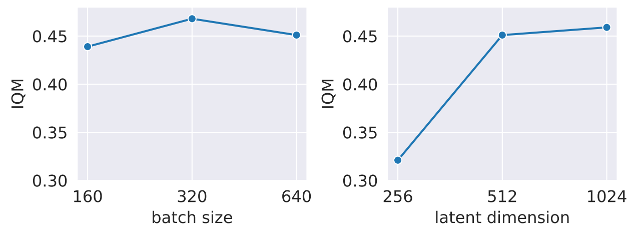


Figure 7. **Effect of batch size and latent dimension.** SimTPR was robust to the changes of batch size and showed a positive correlation between the latent dimension and IQM.

Figure 7 displays SimTPR's performance with varying batch sizes and latent dimensions. Our observations indicate that SimTPR's performance remains consistent across different batch sizes, underscoring its robustness. Moreover, we observed a positive correlation between the size of the latent dimension and the fine-tuning performance, as quantified by the IQM. This correlation implies that increasing the latent dimension could potentially improve the model's performance by facilitating more expressive representations.

Interestingly, these observations align with the results from (Zbontar et al., 2021), who utilized a feature decorrelation loss in vision tasks. They report that the feature decorrelation loss remains robust to changes in batch size while performance improves as the latent dimension increases.

5.2. Transition Model: Causal vs Non-Causal

Here, we discuss the training strategy and architecture of the transition model. For the training strategy, we compare the performance of causal (Radford et al., 2018) and non-causal (Devlin et al., 2019) transition models, using the transformer's encoder and decoder respectively. In training the non-causal model, we randomly masked input representations and reconstruct the masked representations. For the model architecture, we compare the performance of the transformer to the simple recurrent transition model which consists of the 2-layers of GRU block (Cho et al., 2014).

Table 3. **Comparison of the transition model**. A non-causal and causal transformer uses the encoder and decoder of transformer architecture, respectively. The grey row indicates the default setup.

	strategy	architecture	mask ratio	Feat.Rank	IQM ↑
(a)			0.3	435	0.270
(b)	non-causal	transformer	0.5	478	0.385
(c)			0.7	452	0.367
(d)	causal	transformer	-	421	**0.451**
(e)	causal	gru	-	438	0.387

In Table 3.(a), we observed that the non-causal variant with a 30% mask ratio shows the lowest IQM of 0.270. We speculate that with a low masking ratio, the model may easily reconstruct the masked representation by interpolating the representations of the past and future. Although the variants with higher mask ratios showing the improved performance (Table 3.(b) and (c)), they still performed worse than the causal model in Table 3.(d). Comparing Table 3.(e) to (d), we found that a simple recurrent model can learn temporally predictive representations with an IQM score of 0.384, but using a powerful transition model brings further benefits.

5.3. Finetuning matters

Throughout this paper, we have focused on evaluating models that were pretrained, followed by finetuning a policy on top of the frozen encoder. However, in this section, we broaden our scope to consider other finetuning strategies that can potentially enhance the finetuning performance.

One crucial finding from recent studies is that the loss of plasticity significantly contributes to the sample inefficiency of RL (Nikishin et al., 2022; Cetin et al., 2022; Lyle et al., 2023). The term *loss of plasticity* describes the phenomenon where an agent overfits to initial interactions and subsequently struggles to adapt to new data. We find this issue to be especially pronounced in the pretrain-then-finetune paradigm. This is because pretrained models, with their inherent capability to rapidly adapt, are more susceptible to overfitting to earlier interactions.

Table 4. **Comparison of finetuning strategy**. For the reset, we periodically reset the policy layer for every 40,000 update steps.

	pretrain	encoder	reset policy	IQM ↑
(a)	-	Frozen	-	0.161
(b)		Frozen	-	0.451
(c)	✓	Finetuned	-	0.366
(d)			✓	**0.601**

To mitigate this issue, we implement reset techniques as suggested in (Nikishin et al., 2022; Igl et al., 2021). These techniques involve periodically reinitializing the policy parameters to reintroduce plasticity into the model. Following (Nikishin et al., 2022), we reset the policy parameters for every 40,000 updates.

Table 4 presents the finetuning performance for different strategies employed in the case of SimTPR. As in (Schwarzer et al., 2021), simply finetuning the encoder resulted in a relatively low IQM score of 0.366 (Table 4.(c)). This score was improved upon when we froze the encoder, leading to an IQM score of 0.451 (Table 4.(b)). However, the most significant enhancement was achieved by using the reset strategy with pretrained models. Here, SimTPR achieved an IQM score of 0.601 (Table 4.(d)), outperforming the other strategies by a large margin.

In summary, despite the propensity of pretrained representations to overfit to earlier interactions, our results illustrate that pretraining consistently outperforms training from scratch. Importantly, the introduction of a periodic reset was able to significantly enhance the fine-tuning performance of these pretrained models.

6. Conclusion, Limitations, and Future Work

In this paper, we introduce SimTPR, a novel unsupervised representation learning framework for reinforcement learning. By using a feature decorrelation loss to prevent representational collapse, SimTPR efficiently learns temporally predictive representations. Our experimental findings reveal that SimTPR enhances sample efficiency notably on the Atari100k benchmark and attains superior performance in both state and demonstration representation learning. Furthermore, through a systematic analysis of our framework, we find that the feature rank of the pretrained representations is an important factor affecting the fine-tuning performance.

Despite promising results, our study has a few limitations. First, our evaluations are confined to Atari environments. Future research could explore its applicability in diverse environments such as Deepmind Control Suite (Tassa et al., 2018) or Procedural Generation (Cobbe et al., 2020).

Second, we explored a simple variant for fine-tuning, the reset mechanism (Nikishin et al., 2022). Despite its simplicity, the reset significantly improves the performance of a naive fine-tuning strategy. Future work should broaden this exploration to include other fine-tuning techniques (e.g., self-supervised objectives, knowledge distillation), to effectively leverage the pretrained representations at fine-tuning.

Last, our observation on the importance of feature rank lacks theoretical validation. Recent research emphasizes the need for maintaining model plasticity for sample-efficient RL, which refers to its adaptability to new targets (Lyle et al., 2023). They discovered that the factors such as the number of active units (Abbas et al., 2023), the smoothness of the loss landscape (Lyle et al., 2023), and the feature rank (Lyle et al., 2022) are potential contributors to model plasticity. Further investigating these factors can guide the design of better pretraining objectives that generate generalizable and plastic representations.

7. Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2021-0-02068, Artificial Intelligence Innovation Hub), No. 2020-0-00368, A Neural-Symbolic Model for Knowledge Acquisition and Inference Techniques), and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2022R1A2B5B02001913). We also thank Kakao Enterprise for its computational support with GPU Cloud.

References

Abbas, Z., Zhao, R., Modayil, J., White, A., and Machado, M. C. Loss of plasticity in continual deep reinforcement learning. *arXiv preprint arXiv:2303.07507*, 2023.

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *Proc. the International Conference on Machine Learning (ICML)*, pp. 104–114. PMLR, 2020.

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. Deep reinforcement learning at the edge of the statistical precipice. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 34: 29304–29320, 2021.

Anand, A., Racah, E., Ozair, S., Bengio, Y., Côté, M.-A., and Hjelm, R. D. Unsupervised state representation learning in atari. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Baker, B., Akkaya, I., Zhokhov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., and Clune, J. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Bardes, A., Ponce, J., and LeCun, Y. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *Proc. the International Conference on Learning Representations (ICLR)*, 2021.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Buesing, L., Weber, T., Racaniere, S., Eslami, S., Rezende, D., Reichert, D. P., Viola, F., Besse, F., Gregor, K., Hassabis, D., et al. Learning and querying fast generative models for reinforcement learning. *arXiv preprint arXiv:1802.03006*, 2018.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 33: 9912–9924, 2020.

Cetin, E., Ball, P. J., Roberts, S., and Celiktutan, O. Stabilizing off-policy deep reinforcement learning from pixels. *Proc. the International Conference on Machine Learning (ICML)*, 2022.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *Proc. the International Conference on Machine Learning (ICML)*, pp. 1597–1607. PMLR, 2020.

Chen, X. and He, K. Exploring simple siamese representation learning. In *Proc. of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 15750–15758, 2021.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Christiano, P., Shah, Z., Mordatch, I., Schneider, J., Blackwell, T., Tobin, J., Abbeel, P., and Zaremba, W. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. Leveraging procedural generation to benchmark reinforcement learning. In *Proc. the International Conference on Machine Learning (ICML)*, pp. 2048–2056. PMLR, 2020.

Cogswell, M., Ahmed, F., Girshick, R., Zitnick, L., and Batra, D. Reducing overfitting in deep networks by decorrelating representations. *Proc. the International Conference on Learning Representations (ICLR)*, 2016.

Dayan, P. Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624, 1993.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 4171–4186, 2019.

Doerr, A., Daniel, C., Schiegg, M., Duy, N.-T., Schaal, S., Toussaint, M., and Sebastian, T. Probabilistic recurrent state-space models. In *Proc. the International Conference on Machine Learning (ICML)*, pp. 1280–1289. PMLR, 2018.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *Proc. the International Conference on Machine Learning (ICML)*, 2019.

Garrido, Q., Balestriero, R., Najman, L., and Lecun, Y. Rankme: Assessing the downstream performance of pretrained self-supervised representations by their rank. *arXiv preprint arXiv:2210.02885*, 2022.

Golub, G., Klema, V., and Stewart, G. W. Rank degeneracy and least squares problems. Technical report, STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1976.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. Bootstrap your own latent-a new approach to self-supervised learning. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Gu, S., Holly, E., Lillicrap, T., and Levine, S. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. 2017.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In *Proc. the International Conference on Machine Learning (ICML)*, pp. 2555–2565. PMLR, 2019b.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In *Proc. the International Conference on Machine Learning (ICML)*, 2019c.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *Proc. the AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Hua, T., Wang, W., Xue, Z., Ren, S., Wang, Y., and Zhao, H. On feature decorrelation in self-supervised learning. In *Proc. of the IEEE international conference on computer vision (ICCV)*, pp. 9598–9608, 2021.

Huang, J., Gong, S., and Zhu, X. Deep semantic clustering by partition confidence maximisation. In *Proc. of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 8849–8858, 2020.

Huang, L., Yang, D., Lang, B., and Deng, J. Decorrelated batch normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 791–800, 2018.

Igl, M., Farquhar, G., Luketina, J., Boehmer, W., and Whiteson, S. Transient non-stationarity and generalisation in deep reinforcement learning. *Proc. the International Conference on Learning Representations (ICLR)*, 2021.

Jing, L., Vincent, P., LeCun, Y., and Tian, Y. Understanding dimensional collapse in contrastive self-supervised learning. *Proc. the International Conference on Learning Representations (ICLR)*, 2022.

Kaiser, Ł., Babaeizadeh, M., Miłos, P., Osiński, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al. Model based reinforcement learning for atari. In *Proc. the International Conference on Learning Representations (ICLR)*, 2019.

Kessy, A., Lewin, A., and Strimmer, K. Optimal whitening and decorrelation. *The American Statistician*, 72(4):309–314, 2018.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. Supervised contrastive learning. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 33:18661–18673, 2020.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *stat*, 1050:1, 2014.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Laskin, M., Srinivas, A., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. In *Proc. the International Conference on Machine Learning (ICML)*, 2020.

LeCun, Y. A path towards autonomous machine intelligence. *preprint posted on openreview*, 2022.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Littman, M. and Sutton, R. S. Predictive representations of state. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 14, 2001.

Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *Proc. the International Conference on Learning Representations (ICLR)*, 2017.

Lyle, C., Rowland, M., and Dabney, W. Understanding and preventing capacity loss in reinforcement learning. *Proc. the International Conference on Learning Representations (ICLR)*, 2022.

Lyle, C., Zheng, Z., Nikishin, E., Pires, B. A., Pascanu, R., and Dabney, W. Understanding plasticity in neural networks. *Proc. the International Conference on Machine Learning (ICML)*, 2023.

Mairal, J. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

Nie, X., Brunskill, E., and Wager, S. Learning when-to-treat policies. *Journal of the American Statistical Association*, 116(533):392–409, 2021.

Nikishin, E., Schwarzer, M., D'Oro, P., Bacon, P.-L., and Courville, A. The primacy bias in deep reinforcement learning. In *Proc. the International Conference on Machine Learning (ICML)*, pp. 16828–16847. PMLR, 2022.

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

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018.

Raghu, A., Komorowski, M., Ahmed, I., Celi, L., Szolovits, P., and Ghassemi, M. Deep reinforcement learning for sepsis treatment. *arXiv preprint arXiv:1711.09602*, 2017.

Richemond, P. H., Grill, J.-B., Altché, F., Tallec, C., Strub, F., Brock, A., Smith, S., De, S., Pascanu, R., Piot, B., et al. Byol works even without batch statistics. *arXiv preprint arXiv:2010.10241*, 2020.

Schwarzer, M., Anand, A., Goel, R., Hjelm, R. D., Courville, A., and Bachman, P. Data-efficient reinforcement learning with self-predictive representations. In *Proc. the International Conference on Learning Representations (ICLR)*, 2020.

Schwarzer, M., Rajkumar, N., Noukhovitch, M., Anand, A., Charlin, L., Hjelm, R. D., Bachman, P., and Courville, A. C. Pretraining representations for data-efficient reinforcement learning. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Seo, Y., Lee, K., James, S. L., and Abbeel, P. Reinforcement learning with action-free pre-training from videos. In *Proc. the International Conference on Machine Learning (ICML)*, pp. 19561–19579. PMLR, 2022.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

Singh, S. P., Littman, M. L., Jong, N. K., Pardoe, D., and Stone, P. Learning predictive state representations. In *Proc. the International Conference on Machine Learning (ICML)*, pp. 712–719, 2003.

Stooke, A., Lee, K., Abbeel, P., and Laskin, M. Decoupling representation learning from reinforcement learning. In *Proc. the International Conference on Machine Learning (ICML)*, 2021.

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

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Tesauro, G. et al. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 2019.

Wang, F. and Liu, H. Understanding the behaviour of contrastive loss. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2495–2504, 2021.

Yarats, D., Kostrikov, I., and Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *Proc. the International Conference on Machine Learning (ICML)*, 2020.

Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. Barlow twins: Self-supervised learning via redundancy reduction. In *Proc. the International Conference on Machine Learning (ICML)*, pp. 12310–12320. PMLR, 2021.

Zhang, W., GX-Chen, A., Sobal, V., LeCun, Y., and Carion, N. Light-weight probing of unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2208.12345*, 2022.

A. Implementation Details

A.1. Pretraining

In this section, we describe the pretraining details of SimTPR. First, we explain the architectural details of four major components of our model: convolutional encoder, projector, transition model, and predictor.

Convolutional encoder: f**.** A convolutional encoder is used to encode the augmented states into latent representations. The encoder is composed of a 30-layer convolutional network, based on the design from (Schwarzer et al., 2021). The encoder is largely divided into three blocks, where each block comprises of one downsample residual block and two residual blocks. Each residual block follows the design from EfficientNet (Tan & Le, 2019), which is three convolutional layers with an inverted bottleneck structure. The three layers are composed of a convolutional layer with 1x1 filters, a group convolutional layer with 3x3 filters that increases the number of channels by double the input channel, and a convolutional layer with 1x1 filters with output channels. Batch normalization and ReLU activation are applied between these convolutional layers. The spatial resolution of the encoder's output is $(7, 7, 3136)$.

Projector: g**.** Projector maps the encoded representations to the d-dimensional latent space ($d = 512$). We compose the projector as an MLP layer with 512 hidden dimensions where ReLU activation is in-between.

Transition model: h**.** Transition model maps the output of the projector into latent representations of future states. We utilized a casual transformer, based on the design from (Radford et al., 2018). The casual transformer is composed of 2 casual transformer blocks, each with 512 input dimensions, 8 heads, and 2048 hidden dimensions.

Predictor: q**.** Predictor maps the output of the transition model to the d-dimensional latent space ($d = 512$). The predictor is composed of an MLP layer with 512 hidden dimensions where ReLU activation and batch normalization are in-between.

In pretraining, we use the same hyperparameter of the state and demonstration pretraining where the only difference is the use of action prediction weight in the demonstration dataset. For simplicity, we use the action prediction weight as $\lambda_a = 1.0$. Detailed hyperparameters are described in Table 5. On a single A100 GPU, the pretraining takes 1 to 2 days for each run.

Table 5. Hyperparameters of SimTPR for pretraining on DQN replay dataset.

Hyperparameter	Value
State size	(84, 84, 4)
Grey scaling	True
Data Augmentation	Random Shifts and Intensity Jittering
Random Shifts	$\pm$ 4 pixels
Intensity Jiterring scale	0.05
Channels for each block	(16, 64, 64)
Strides for each block	(3, 2, 2)
Number of future steps to associate (k)	1
Off-diagonal weight (λ_o)	0.005
Decorrelation weight (λ_d)	0.01
Action prediction weight (λ_a)	1
Epochs	100
Batch size	640
Number of Sequences (N)	64
Sequence length (T)	10
Optimizer (β_1, β_2, ϵ)	AdamW (0.9, 0.999, 0.000015)
Learning rate	0.0003
Weight decay	0.00001
Max gradient norm	0.5

A.2. Linear Probing

Following the evaluation protocol from (Zhang et al., 2022), a linear classifier was trained on top of the frozen encoder and predicted the reward and action for each state. To evaluate the effectiveness of the pretrained representations in capturing the policy and value function of experts, we utilized the last 100,000 frames from the 50 million frame DQN replay dataset (Agarwal et al., 2020) with a 4:1 train/eval split.

To simplify the reward probing task, we formulated it as a binary classification problem where the objective is to predict the occurrence of a reward for a given state. We trained a logistic regression model using the majorization-minimization optimization (MISO) algorithm (Mairal, 2015) for a total of 300 iterations.

For the action probing task, we train a multi-class classifier with a softmax focal loss (Lin et al., 2017) to address the class imbalance of the action labels. The model was trained for 50 epochs using the SGD optimizer with a learning rate of 0.2, batch size of 256, and a weight decay of 10^{-6}. For each epoch, the learning rate was decayed by a step-wise scheduler with a step size of 10 and gamma of 0.1.

Both tasks were performed on an A100 GPU and completed within 5 minutes.

A.3. Finetuning

Following the evaluation protocol from (Zhang et al., 2022; Schwarzer et al., 2021), we focus on the Atari 100k benchmark (Kaiser et al., 2019), where only 100k steps of interactions are allowed at the finetuning phase. We train a Q-learning head on top of the frozen encoder using the Rainbow algorithm (Hessel et al., 2018). Here we do not apply any auxiliary self-supervised loss and rely on noisy layers for exploration. A detailed hyperparameter for the finetuning setup is described in Table 6. On an A100 GPU, each run takes from 2 to 3 hours.

Table 6. Hyperparameters for finetuning on the Atari100k benchmark.

Hyperparameter	Value
State downsample size	(84, 84)
Grey scaling	True
Data augmentation	Random Shifts and Intensity Jittering
Random Shifts	$\pm$ 4 pixels
Intensity Jiterring scale	0.05
Frame skip	4
Stacked frames	4
Action repeat	4
Training steps	100k
Update	Distributional Q
Dueling	True
Support of Q-distribution	51
Discount factor γ	0.99
Batch size	32
Optimizer (β_1, β_2, ϵ)	Adam (0.9, 0.999, 0.000015)
Learning rate	0.00003
Max gradient norm	10
Priority exponent	0.5
Priority correction	0.4 $\rightarrow$ 1
Exploration	Noisy nets
Noisy nets parameter	0.5
Replay buffer size	100k
Min buffer size for sampling	2000
Replay per training step	1
Updates per replay step	2
Multi-step return length	10
Q-head hidden units	512
Q-head non-linearity	ReLU
Evaluation trajectories	50

B. Baseline Implementations

Our comparison in Table 4.2 is based on our reproduction of the related methods with the exception of SGI and BarlowBalance. We have implemented the competitors as faithfully as possible following each individual paper. In order to ensure fairness in comparison, we have employed the same convolutional encoder, latent dimension, data augmentation, pretraining data, number of pretraining epochs, and the optimizer (AdamW) across all methods. For each method, we have varied the learning rate from $\{0.001, 0.0003, 0.0001\}$ during pretraining and selected the model with the highest linear probing score for each method. Then, we finetuned the selected models for 10 random seeds and report the finetuning performance. A detailed description of the training setup for each method is provided in the following section.

B.1. VAE

For the variational autoencoder (VAE), data augmentation was applied to each state, followed by the convolutional encoder to obtain the latent representations. Subsequently, the mean (μ) and log variance ($\log \sigma^2$) of the variational posterior of VAE were obtained by the MLP layer with 256 hidden dimensions, respectively. Then, the random variable z was sampled from the variational posterior $N(z; \mu, \sigma^2 I)$ and used to reconstruct the original states through the convolutional decoder. The architecture of the convolutional decoder is symmetric to the convolutional encoder as (Long et al., 2015). We trained VAE by jointly minimizing the KL divergence between the variational posterior and the prior $N(0, I)$, and the Mean Squared Error (MSE) between the reconstructed states and the original states. To balance the loss between two terms, we weighted the KL divergence loss by a factor of $\lambda_{KL} = 0.02$. To ensure compatibility with a batch size of 640, we set a number of sequences of $N = 640$, with a sequence length of $T = 1$.

B.2. BarlowTwins

BarlowTwins learns the representations by decorrelating the features of the augmented states. BarlowTwins uses siamese architecture where two differently augmented states are processed by an encoder followed by a projector. Then, the projected representations, Z, and Z' are obtained through this siamese architecture. Note that the convolutional encoder and projector are identical to our configuration. The model is trained to standardize the cross-correlation matrix between projected representations Z and Z' to the identity matrix. To balance the on-diagonal and off-diagonal terms, $\lambda_{Barlow} = 0.05$ is applied to weight the off-diagonal terms of the cross-correlation matrix. To ensure compatibility with a batch size of 640, we utilize a number of sequences, $N = 640$, with a sequence length of $T = 1$.

B.3. CURL

Contrastive Unsupervised Representations for Reinforcement Learning (CURL) employs a siamese model, where the convolutional encoder and a projection layer are identical to our setup. The siamese model is coupled with the Momentum Encoder, with a momentum coefficient of $\tau = 0.99$. Two different data augmentations are applied to each state, followed by the siamese model to obtain Z and Z'. A prediction layer, consisting of a residual MLP layer with 512 hidden dimensions, is applied to Z to obtain the prediction Q. Then, CURL is trained to minimize the contrastive loss between Q and Z', with a temperature of 0.1. For a fair comparison, the contrastive loss is also computed symmetrically between Q' and Z. To ensure compatibility with a batch size of 640, we utilized a number of sequences as $N = 640$, with a sequence length of $T = 1$.

B.4. RSSM

Recurrent State-Space Model (RSSM) learns temporally predictive representations by predicting future states in a latent space and reconstructing them in the original state space. To begin with, data augmentation was applied to each state, followed by the convolutional encoder to obtain the latent representations. Let s_t be the latent representation of the state at time step t, We explain the architectural details of four major components of RSSM: the recurrent model, the transition model, the representation model, and the image decoder, as follows:

$$\begin{aligned} \text{Recurrent model: } & h_{t+1} = f_\theta(h_t, z_t), \\ \text{Transition model: } & \hat{z}_{t+1} = p_\phi(\hat{z}_{t+1} \mid h_{t+1}), \\ \text{Representation model: } & z_{t+1} = q_\phi(z_{t+1} \mid h_{t+1}, s_{t+1}), \\ \text{Image Decoder: } & \hat{s}_{t+1} = f_\phi(\hat{s}_{t+1} \mid h_{t+1}, z_{t+1}). \end{aligned} \tag{8}$$

Recurrent model. Recurrent model predicts the latent of the next states and consists of an MLP layer with 512 hidden dimensions, ReLU activation, and a gated recurrent unit (GRU) cell with 512 hidden dimensions.

Transition model. Transition model serves as a prior and consists of a Linear layer with hidden dimensions of 512, ReLU activation, an output Linear layer with 256 output dimensions, and a Softplus activation to obtain mean μ and standard deviation σ with 128 hidden dimensions, respectively.

Representation model. Representation model serves as a posterior and is identical to the structure of the transition model.

Image decoder. Image decoder reconstructs the target state through the convolutional decoder. The architecture of the convolutional decoder is symmetric to the convolutional encoder (Long et al., 2015), which is identical to the architecture used in VAE.

RSSM was trained to minimize the KL divergence between the representation model and the transition model, as well as the Mean Squared Error (MSE) between the reconstructed states and the target states. In order to balance the loss between the KL divergence and MSE error, we weigh the KL divergence loss by a factor of $\lambda_{KL} = 0.025$. To ensure compatibility with a batch size of 640, we used a total of $N = 64$ sequences, each with a sequence length of $T = 10$.

B.5. ATC

Augmented Temporal Contrast (ATC) learns temporally predictive representations by maximizing the similarity between the representations of current states and states with a short time difference. ATC employs a siamese model, where the convolutional encoder and a projection layer are identical to our setup. The siamese model is coupled with the Momentum Encoder, utilizing a momentum coefficient of $\tau = 0.99$. In training, two different data augmentations are applied to each state, followed by the siamese model to obtain Z and Z'. A prediction layer, which is also identical to our configuration is applied to Z to obtain the prediction Q. Then, a contrastive loss is applied, which maximizes the similarity between predictions from future states with k-steps distances (i.e., the similarity between Q_t and Z'_{t+k} at time step t). Following (Stooke et al., 2021; Schwarzer et al., 2021), we set $k = 3$. For a fair comparison, we symmetrize the loss function and set a number of sequences of $N = 64$, with a sequence length of $T = 10$.

B.6. BC

For behavioral cloning, we applied data augmentation to each state and trained a prediction model to predict the action for each state. The action prediction layer is applied to the output of the convolutional encoder which is implemented as a two-layer MLP with 512 hidden dimensions, where a ReLU activation is applied in between. To ensure compatibility with the batch size of 640, we utilized a number of sequences of $N = 640$ and a sequence length of $T = 1$. Through empirical analysis, we found that behavioral cloning serves as a strong baseline among the demonstration pretraining methods.

B.7. IDM

Inverse Dynamics Modeling (IDM) is trained to predict the action between two consecutive states. Similar to behavioral cloning, the action prediction layer is implemented as a two-layer multi-layer perceptron (MLP) with 512 hidden dimensions and ReLU activation is applied in between. In the training process, we first applied data augmentation to two consecutive states and used a convolutional encoder to obtain the latent representations for each state. The resulting representations were concatenated and processed by the action predictor to predict the action between the states. In order to ensure compatibility with the batch size of 640, we utilized a number of sequences of $N = 320$ and a sequence length of $T = 2$.

C. PseudoCode

C.1. State Pretraining

Algorithm 1 PyTorch-like Pseudocode of SimTPR for State Pretraining

```
# f:  encoder
# g:  projector
# h:  transition model
# p:  predictor

# n:  batch size
# t:  sequence length
# d:  dimension of the latent representations
# k:  future steps to associate with

for s in loader:  # load a batch of (n,t) samples
   # compute embeddings
   x1, x2 = aug(s), aug(s)
   z1, z2 = g(f(x1)), g(f(x2)) # projections, (n,t,d)
   q1, q2 = p(h(z1), p(h(z2)) # predictions, (n,t,d)

   # l2-normalize
   z1, z2 = normalize(z1, dim=2), normalize(z2, dim=2)
   q1, q2 = normalize(q1, dim=2), normalize(q2, dim=2)

   # compute loss
   L_sim = distance(q1, z2.detach())/2 + distance(q2, z1.detach())/2)
   L_decorr = standardize(z1, z2)
   L = L_sim + lambda_d.mul(L_decorr)

   # optimize
   L.backward()
   optimizer.step()

def distance(q, z):
   z1, z2 = z1[:, k:], z2[:, k:]  # (n,t-k,d)
   q1, q2 = q1[:, :-k], q2[:, :-k] # (n,t-k,d)

   return (q1 - z2).pow(2).sum(dim=2).mean()

def standardize(z1, z2):
   # reshape
   z1, z2 = z1.reshape(n*t,d), z2.reshape(n*t,d)
   q1, q2 = q1.reshape(n*t,d), q2.reshape(n*t,d)

   # compute cross-correlation matrix (d,d)
   z1_norm = (z1 - z1.mean(0)) / z_1.std(0)
   z2_norm = (z2 - z2.mean(0)) / z_2.std(0)
   c = mm(z1_norm.T, z2_norm)/n

   # loss
   c_on =on_diagonal(c) # (d)
   c_off =off_diagonal(c) # (d(d-1))
   return (c_on - ones(d)).sum() + lambda_o.mul(c_off.sum())
```

C.2. Demonstration Pretraining

Algorithm 2 PyTorch-like Pseudocode of SimTPR for Demonstration Pretraining

```
# f:  encoder
# g:  projector
# h:  transition model
# p:  predictor
# r:  action predictor

# n:  batch size
# t:  sequence length
# d:  dimension of the latent representations
# k:  future steps to associate with
# n_a:  a number of action in the environment

# distance(q, z):  a distance function defined in algorithm 1
# standardize(z1, z2):  a standardization function defined in algorithm 1
# cross_entropy(y, a):  a standard cross-entropy loss where y is the un-normalized logits

for s, a in loader:  # load a batch of (n,t) samples
   # compute embeddings
   x1, x2 = aug(s), aug(s)
   z1, z2 = g(f(x1)), g(f(x2)) # projections, (n,t,d)
   y = embed(a) # linearly embed actions, (n,t,d)
   tau1 = alternate_concat(z1, y) # tau1 = (z1_1,a_1,...,z1_t,a_t), (n,2*t,d)
   tau2 = alternate_concat(z2, y) # tau2 = (z2_1,a_1,...,z2_t,a_t), (n,2*t,d)
   c1, c2 = h(tau1), h(tau2)) # transition, (n,2*t,d)
   q1, q2 = p(c1[:,1,3,...,2t-1]), p(c2[:,1,3,...,2t-1])) # latent prediction, (n,t,d)
   l1, l2 = r(c1[:,0,2,...,2t-2]), r(c2[:,0,2,...,2t-2])) # action prediction, (n,t,n_a)

   # l2-normalize
   z1, z2 = normalize(z1, dim=2), normalize(z2, dim=2)
   q1, q2 = normalize(q1, dim=2), normalize(q2, dim=2)

   # compute loss
   L_sim = distance(q1, z2.detach())/2 + distance(q2, z1.detach())/2)
   L_decorr = standardize(z1, z2)
   L_act = cross_entropy(l1, a)/2 + cross_entropy(l2, a)/2
   L = L_sim + lambda_d.mul(L_decorr) + lambda_a.mul(L_act)

   # optimize
   L.backward()
   optimizer.step()
```

D. Uncertainty-Aware Comparison

D.1. State Representation Learning

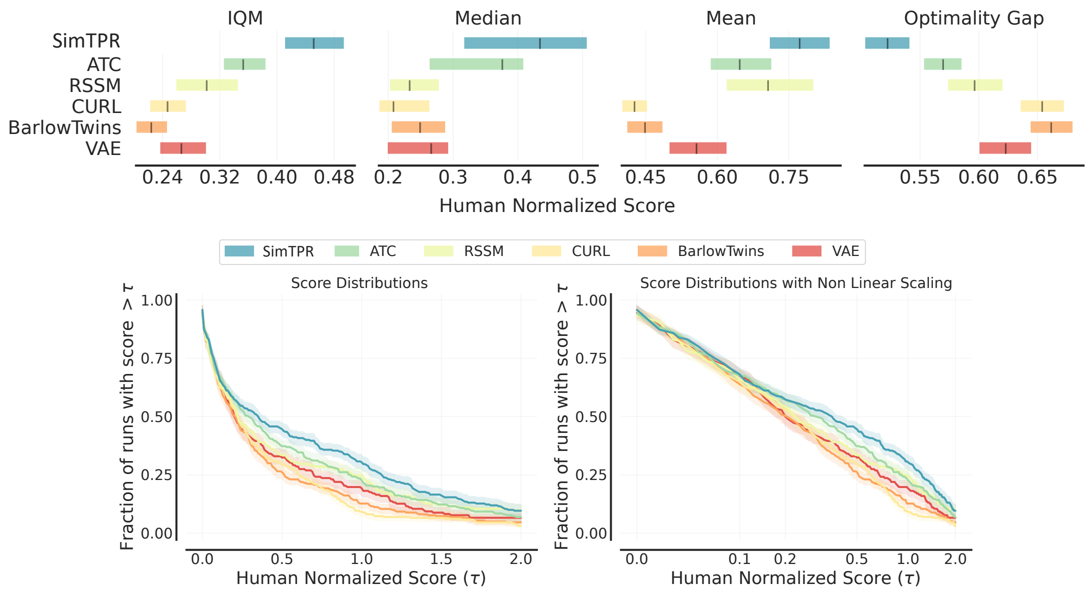


Figure 8. **Uncertainty-aware comparison for State Representation Learning. Top.** 95% Confidence Intervals for IQM, Median, Mean and Optimality Gap scores for each state representation learning method in Table 4.2. **Bottom.** Average score distributions for each method where shaded regions represent a pointwise 95% confidence interval. The confidence intervals are estimated using the percentile bootstrap with stratified sampling as (Agarwal et al., 2021).

D.2. Demonstration Representation Learning

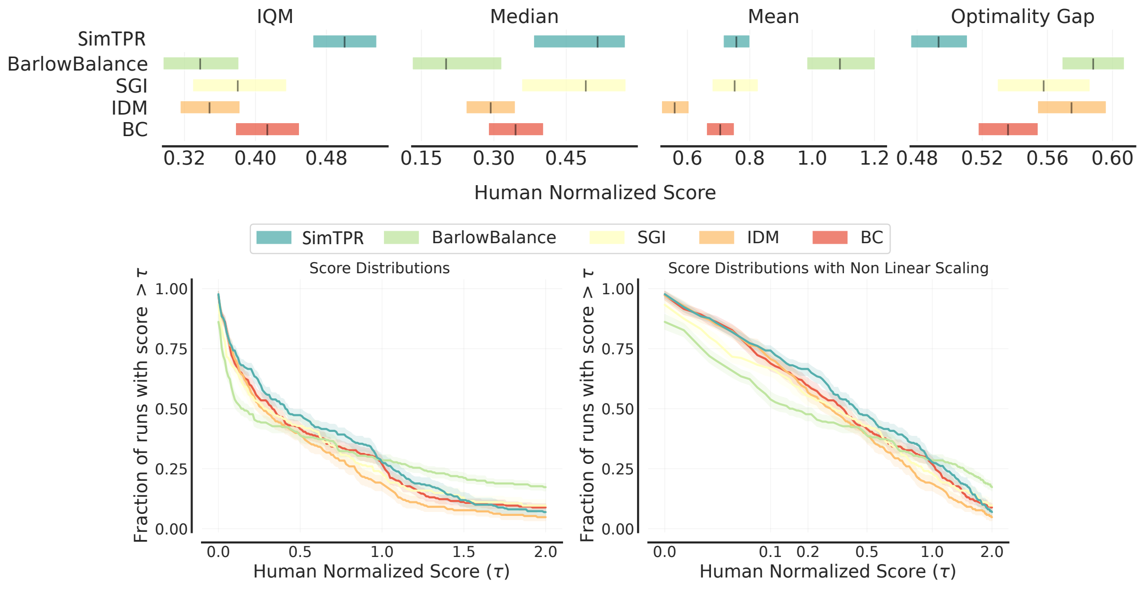


Figure 9. **Uncertainty-aware comparison for Demonstration Representation Learning. Top.** 95% Confidence Intervals for IQM, Median, Mean and Optimality Gap scores for each demonstration representation learning method in Table 4.2. **Bottom.** Average score distributions for each method where shaded regions represent a pointwise 95% confidence interval. The confidence intervals are estimated using the percentile bootstrap with stratified sampling as (Agarwal et al., 2021).

E. Does Higher Linear Probing Score indicates a better finetuning Performance?

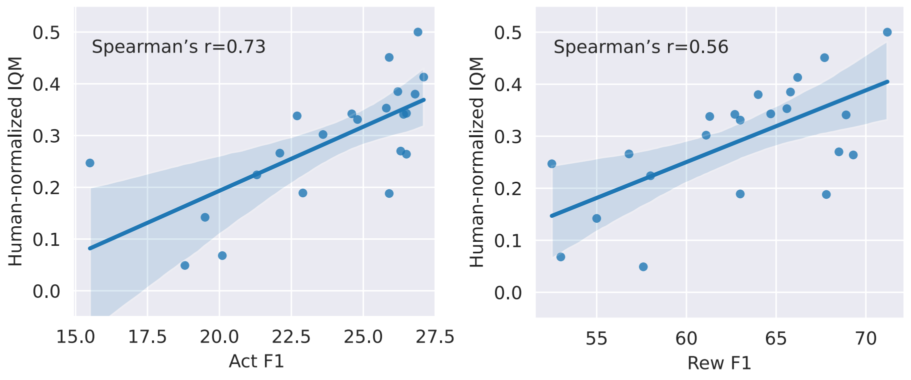


Figure 10. **Correlation between the Linear Probing Score and the finetuning Performance. Left.** Correlation between the Act F1 score and IQM of the HNS. **Right.** Correlation between the Rew F1 score and IQM of the HNS.

In order to assess the representation quality of a pretrained model, a commonly employed protocol is to train reinforcement learning (RL) algorithms on top of the pretrained model. However, this approach can be computationally intensive and may result in high-variance outcomes. To address this problem, a recent study by (Zhang et al., 2022) has proposed the use of two linear probing tasks as an alternative method for evaluating representation quality. These tasks, action prediction, and reward prediction are designed to predict the actions and rewards of experts and are computationally efficient with low variance. Empirically, a study by (Zhang et al., 2022) has shown that these probing scores have a high correlation to the RL agent's finetuning score.

In this study, we aim to investigate the relationship between linear probing scores and finetuning performance for reinforcement learning (RL) agents. We present an analysis of the correlation between the linear probing scores (Act F1 and Rew F1) and the finetuning performance (IQM of human-normalized score) for the RL agent in Figure 10. Each data point in the figure represents a different pretraining method, including ablation studies of SimTPR and various baselines. To evaluate the correlation, we applied Spearman's rank correlation test to the data. The results of this analysis indicate a moderate correlation between Action F1 and IQM, with a Spearman's correlation coefficient of 0.73, and a correlation coefficient between the Rew F1 and IQM of 0.56. Thus, based on the moderate correlation, it may not be sufficient to conclude that a higher linear probing score guarantees a better finetuning performance for the RL agent.

F. Full Results on Atari100k

F.1. State Representation Learning

Table 7. **Mean episodic scores of state representation learning.** We report the mean episodic scores on the 26 Atari games for state representation learning methods. The results are recorded at the end of training and averaged over 10 random seeds. SimTPR outperforms the other methods on all aggregate metrics.

Game	Random	Human	DrQ	VAE	BarlowTwins	CURL	RSSM	ATC	SimTPR
Alien	227.8	7127.7	798.52	1013.74	853.5	823.56	1013.23	845.84	784.12
Amidar	5.8	1719.5	123.91	250.49	157.44	128.49	191.57	124.87	158.35
Assault	222.4	742.0	534.88	783.52	635.76	616.87	860.19	716.34	853.58
Asterix	210.0	8503.3	605.5	560.86	528.6	538.6	537.5	574.7	521.4
BankHeist	14.2	753.1	124.72	56.57	57.18	91.96	35.37	395.7	402.14
BattleZone	2360.0	37187.5	10714.0	3274.29	5032.0	6718.0	6746.67	9168.0	6796.0
Boxing	0.1	12.1	3.61	32.71	25.23	10.01	41.66	32.76	42.13
Breakout	1.7	30.5	11.17	4.07	14.53	10.83	3.29	21.45	55.67
ChopperCommand	811.0	7387.8	914.4	736.57	873.8	1012.8	829.67	836.4	851.6
CrazyClimber	10780.5	35829.4	11203.6	32358.86	20787.6	26213.0	49406.67	69789.0	56273.2
DemonAttack	152.1	1971.0	384.7	471.57	704.84	785.67	801.92	925.24	1484.47
Freeway	0.0	29.6	26.94	15.22	22.29	16.49	25.79	0.0	6.68
Frostbite	65.2	4334.7	607.3	2587.0	1264.02	874.96	2222.07	252.0	268.48
Gopher	257.6	2412.5	344.88	800.06	453.84	628.44	657.67	761.56	1022.44
Hero	1027.0	30826.4	6967.47	7102.73	8206.1	7123.86	5870.07	10773.08	10804.96
Jamesbond	29.0	302.8	225.7	395.0	348.8	312.5	465.17	417.9	298.5
Kangaroo	52.0	3035.0	2907.0	1214.0	819.4	854.6	618.0	1520.0	3227.8
Krull	1598.0	2665.5	3178.16	4532.11	3911.5	4277.88	5332.93	3722.96	4678.62
KungFuMaster	258.5	22736.3	3085.4	17444.29	7982.6	13353.2	10321.67	16078.6	14197.6
MsPacman	307.3	6951.6	843.0	1636.57	1079.96	1180.2	1633.0	1268.54	1249.58
Pong	−20.7	14.6	−14.88	−10.04	−13.33	−4.59	−4.03	0.51	−2.57
PrivateEye	24.9	69571.3	80.0	85.71	90.0	122.33	−43.53	124.84	90.0
Qbert	163.9	13455.0	2337.6	3893.64	2709.65	2971.9	3747.08	3299.7	3091.7
RoadRunner	11.5	7845.0	6439.6	4411.43	8172.2	9915.2	8997.0	9837.2	9780.6
Seaquest	68.4	42054.7	407.16	703.66	352.24	465.88	753.8	517.12	424.96
UpNDown	533.4	11693.2	3198.66	11897.83	4900.44	2404.42	20043.8	15108.64	16623.82
IQM	0.0	1.0	0.161	0.266	0.224	0.247	0.302	0.353	**0.451**
Median	0.0	1.0	0.157	0.266	0.249	0.208	0.235	0.376	**0.434**
Mean	0.0	1.0	0.308	0.556	0.448	0.426	0.707	0.647	**0.773**
OG	0.0	1.0	0.724	0.623	0.662	0.654	0.595	0.570	**0.522**

F.2. Demonstration Representation Learning

Table 8. **Mean episodic scores of demonstration representation learning.** We report the mean episodic scores on the 26 Atari games for demonstration representation learning methods. The results are recorded at the end of training and averaged over 10 random seeds. SimTPR outperforms the other methods on all aggregate metrics except Mean.

Game	Random	Human	BC	IDM	SGI	BarlowBalance	SimTPR
Alien	227.8	7127.7	837.24	1007.2	1091.38	981.44	1006.1
Amidar	5.8	1719.5	143.47	134.97	233.47	170.37	127.74
Assault	222.4	742.0	735.42	535.04	569.69	903.22	1053.48
Asterix	210.0	8503.3	709.0	587.57	398.75	436.0	643.8
BankHeist	14.2	753.1	363.18	557.83	397.33	205.38	449.04
BattleZone	2360.0	37187.5	7838.0	7937.14	2920.0	3812.0	9014.0
Boxing	0.1	12.1	41.93	22.52	44.41	62.13	13.01
Breakout	1.7	30.5	34.61	24.47	13.74	1.11	40.56
ChopperCommand	811.0	7387.8	1124.6	931.14	815.0	658.4	829.6
CrazyClimber	10780.5	35829.4	62124.0	45292.86	43783.5	94975.4	47558.8
DemonAttack	152.1	1971.0	1587.94	748.46	182.7	343.5	2571.99
Freeway	0.0	29.6	21.4	16.75	18.53	19.27	25.65
Frostbite	65.2	4334.7	385.94	540.91	1207.2	638.76	245.38
Gopher	257.6	2412.5	994.8	751.89	1253.85	1149.36	1241.48
Hero	1027.0	30826.4	8232.99	7398.11	4336.45	5302.16	9986.87
Jamesbond	29.0	302.8	241.4	318.29	307.38	414.0	419.5
Kangaroo	52.0	3035.0	832.2	630.86	2195.0	2213.0	664.8
Krull	1598.0	2665.5	4809.3	4825.6	6313.55	5551.18	6307.36
KungFuMaster	258.5	22736.3	12934.0	12198.57	14343.25	14359.6	13142.0
MsPacman	307.3	6951.6	1313.14	1356.29	1600.2	415.06	1849.7
Pong	-20.7	14.6	11.35	7.41	3.26	3.51	9.29
PrivateEye	24.9	69571.3	90.0	71.67	61.5	60.2	100.2
Qbert	163.9	13455.0	2754.0	2998.21	714.06	185.5	4576.75
RoadRunner	11.5	7845.0	10749.4	7755.14	7995.25	21362.6	10694.2
Seaquest	68.4	42054.7	487.88	559.77	561.75	579.92	584.2
UpNDown	533.4	11693.2	4414.98	5744.23	18800.9	74548.54	9086.26
IQM	0.0	1.0	0.413	0.343	0.380	0.338	**0.500**
Median	0.0	1.0	0.345	0.279	0.490	0.201	**0.515**
Mean	0.0	1.0	0.705	0.564	0.751	**1.089**	0.757
OG	0.0	1.0	0.536	0.579	0.558	0.588	**0.493**