# S2P: State-conditioned Image Synthesis for Data Augmentation in Offline Reinforcement Learning

**Daesol Cho***
Department of Aerospace Engineering
Seoul National University

**Dongseok Shim***
Interdisciplinary Program in AI
Seoul National University

**H. Jin Kim**
Department of Aerospace Engineering
Seoul National University

{dscho1234, tlaehdtjr01, hjinkim}@snu.ac.kr

## Abstract

Offline reinforcement learning (Offline RL) suffers from the innate distributional shift as it cannot interact with the physical environment during training. To alleviate such limitation, state-based offline RL leverages a learned dynamics model from the logged experience and augments the predicted state transition to extend the data distribution. For exploiting such benefit also on the image-based RL, we firstly propose a generative model, S2P (State2Pixel), which synthesizes the raw pixel of the agent from its corresponding state. It enables bridging the gap between the state and the image domain in RL algorithms, and virtually exploring unseen image distribution via model-based transition in the state space. Through experiments, we confirm that our S2P-based image synthesis not only improves the image-based offline RL performance but also shows powerful generalization capability on unseen tasks.

## 1 Introduction

Deep learning algorithms have shown significant development thanks to the large pre-collected dataset, such as SQuAD [49] in natural language processing (NLP), and ImageNet [4] in computer vision. On the contrary, reinforcement learning (RL) requires an online trial-and-error in training process to collect the data by interacting with the environment, which hinders its utilization in many real-world applications. Due to this intrinsic property of current online RL algorithms, there exist some approaches that try to deploy large and diverse pre-recorded datasets without online interaction with the environment, which is called offline RL.

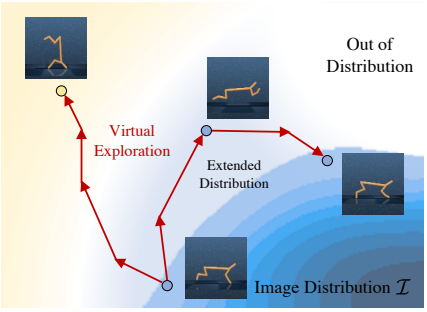



Figure 1: S2P generates the dynamics-consistent image transition data by virtually exploring in the state space to extend the distribution of the offline datasets.

However, recent studies have observed that the current online RL algorithms [10, 36] perform poorly in an offline setting. It is primarily attributed to the large extrapolation error when the Q-function is evaluated on out-of-

*Equal contribution

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

distribution actions, which is called the distributional shift [26, 22, 8]. That is, due to the offline setting that limits online data collection, the offline RL has struggled to generalize beyond the given offline dataset. Even though some offline RL methods [27, 25, 56] achieve reasonable performances in some settings, their training is still limited to behaviors within the given offline dataset distribution, and detours the evaluation on out-of-distribution data rather than directly addressing such an empty space of the offline dataset. Thus, there exists a growing need for the development of algorithms specialized to directly address such out-of-distribution data by extending the offline dataset distribution's support.

To alleviate the fixed dataset distribution problem, recent studies propose some data augmentation strategies. In state-based RL, model-based algorithms [2, 1, 28, 16] which learn a dynamics model from the pre-recorded dataset and augment the dataset with generated state transitions have emerged as a promising paradigm. As the model-based approach trains dynamics models in a supervised manner, it allows a stable training process and generates reliable state transition data for augmentation. Thus, it can be a plausible choice that it enables the generalization into the unseen state-action by performing dynamics-consistent planning on unseen state distribution.

When it comes to the image domain, however, there is still no augmentation strategy to mitigate the aforementioned distribution shift. Even though some model-based image RL methods [12, 11] that propose to learn latent dynamics using reconstruction error from ELBO objective [18, 53, 24] can be exploited for generating image transition data in a similar manner to the state-based methods, the quality of the output images from these approaches is not satisfactory because 1) their focus is on learning latent representation suitable for the RL network's inputs rather than generating high-quality and accurate images, and 2) ELBO-based objective cannot generate photo-realistic images compared to other generative models such as GAN [9] or diffusion [14, 43] and it usually produces blurry outputs. Above all, 3) model-based image RL only exploits image input and it makes the generative model fail to capture the accurate dynamics and the details of the agent's posture or objects in the image [42, 44, 3]. These undesired properties discourage offline RL algorithms from adding the reconstruction output of model-based image RL to their training data as an augmentation strategy.

Therefore, we propose S2P (State2Pixel) which utilizes multi-modal input (the state, and the previous image observation of the agent) to synthesize the realistic image from its corresponding state. The key element of S2P is a multi-modal affine transformation (MAT) which effectively exploits both state and image cross-modality information. Unlike previous learned affine transformation [19–21, 46, 35], which leverages a single domain input, MAT fuses the cross-modal representation from the state and the image to produce the scale and the bias modulation parameters. This multi-modality of S2P makes it possible to generate the dynamics-consistent images from the reliable state transition while preserving high-quality image generation capability.

To sum up, our work makes the following key contributions.

- We propose a state-to-pixel generative model (S2P) which generates dynamics-consistent image and multimodal affine transformation (MAT) module for aggregating cross-modal inputs.
- To the best of the author's knowledge, this work is the first to propose image augmentation for offline image RL and overcome innate fixed distribution problem by implicitly leveraging reliable state transition.
- We evaluate our S2P on the DMControl [54] benchmark environments with the offline setting, and it results in $1.0 - 3.0$x higher offline RL performance by augmenting the generated synthetic image transition data.
- Even with the state distributions of the unseen tasks, S2P can generalize to unseen image distribution, and we show that the agent can be trained by offline RL with these generated images only.

## 2 Related Work

### 2.1 Image Synthesis

Generative Adversarial Networks (GAN) [9] based deep generative models enjoy huge success in synthesizing high-resolution photo-realistic images via style mapping function and the learned affine

transformation [19–21, 46, 35]. StyleGAN [19] firstly utilizes a style vector $\mathbf{w}$ and Adaptive Instance Normalization (AdaIN) [5, 15] in the generative networks to disentangle the latent space and control the scale-specific synthesis. Following studies [20] pinpoint that the AdaIN operation which leads to information loss in the feature magnitude makes undesired droplet-like artifacts in the synthesized images and proposes weight demodulation by assuming the variance of input features. SPADE [46] proposes an architecture to synthesize the image using its corresponding semantic masks and spatially learned affine transformation. ManiGAN [35] suggests Text-Image Affine Combination Module (ACM) which enables the network to manipulate the images using text descriptions given by users. The difference between our proposed MAT and the previous studies is that we leverage cross-modal data, state and image, to estimate the modulation parameters for the learned affine transformation whereas other studies use a single data type such as text or image.

### 2.2 Offline Reinforcement Learning & Data Augmentation

Offline RL [6, 30, 34] is the task of learning policies from a given static dataset, which is different from online RL that learns useful behaviors through trial-and-error in the environment. Prior offline RL algorithms are designed to constrain the policy to the behavior policy used for offline data collection via direct state or action constraints [8, 38], maximum mean discrepancy [26], KL divergence [56, 61, 17], or learning conservative critics [27, 25]. However, most of these methods are limited to exploiting the state-action distribution of the given static dataset, rather than exploring and extending the distribution. As the offline setting prohibits online interaction with the environment, we suggest the synthetic data generation method to enable the offline RL agent to virtually explore and extend the distribution by leaving the support of the dataset.

Recent works in model-based state RL that involves learning a policy with a dynamics model [22, 2, 1, 29, 59, 60] suggest the need to augment the data with generated transitions from the model for extending the data distribution. Also, augmentation strategies in image domain [31, 51, 32, 57, 58] emphasize the importance of image augmentation for sample efficiency and robust representation learning. But, these works focus on purely image manipulation techniques on the given image such as cropping rather than generating image transition. Some image-based methods [47, 11, 12] that use the variational model to train the latent dynamics are studied in a similar concept to the state-based ones. However, the generated image from these methods is the byproduct of learning the effective image representation rather than the main purpose of these works, and as these methods only utilize the image input, it leads to missing objects or inaccurate dynamics of the agent in the reconstructed image. Therefore, to bridge the gap between the model-based state transition data augmentation techniques and image augmentation, we propose the method that generates dynamics-consistent image transition data along timesteps with multi-modal inputs.

## 3 Method

### 3.1 S2P Generator

**Architecture.** The goal of S2P (State2Pixel) generator is to synthesize the image $\hat{I}_t$ which perfectly represents all the information of its corresponding state $s_t$. Unfortunately, a state-sole condition cannot formulate a single deterministic rendered image because, in most cases, state does not provide the agent's position from the global coordinate, but rather from an egocentric coordinate. Also, image-based RL algorithms utilize sequential images to capture the agent's velocity using the change of the background, e.g. ground checkerboard, between input images. It means that the image of the current step $I_t$ is dependent not only on the current state $s_t$, but also on the image of the previous step $I_{t-1}$. We, therefore, build the generator $G$ to synthesize the image $\hat{I}_t$ from both the state $s_t$ and the previous image $I_{t-1}$ so that the generated image $\hat{I}_t$ can preserve the dynamics-consistency in the physical environment.

$$\hat{I}_t = G(s_t, I_{t-1}) \tag{1}$$

At the first layer of S2P, the input image $I_{t-1}$ and the state $s_t$ are projected to the feature space via convolution and MLP encoders respectively. Both features are then fed to the hierarchical generator block which consists of several residual connections [13] and the upsampling layer. After passing through each generator block, the spatial size of the feature map is doubled while the channel dimension is halved. The image features are converted to the RGB images at the last layer of the generator with a single convolution layer.

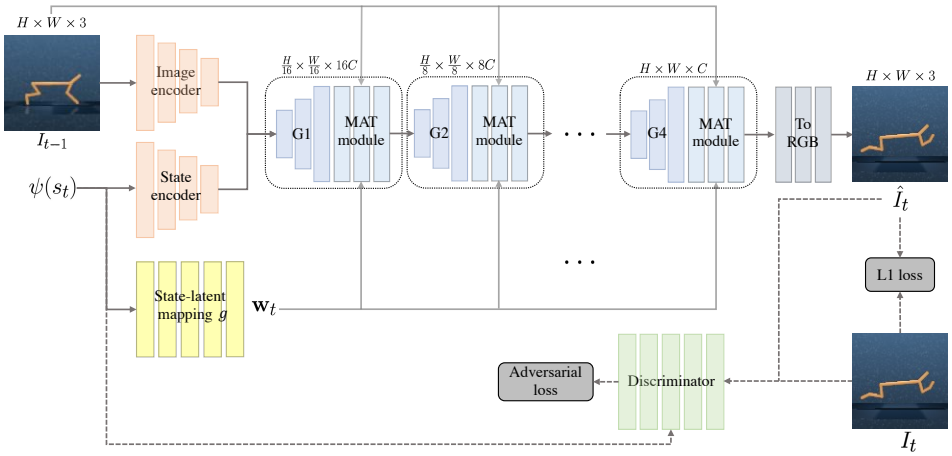



Figure 2: An overview of S2P architecture. State $s_t$ and the previous image $I_{t-1}$ are used as input to generate current step image $\hat{I}_t$. The spatial size of the features gets larger as it passes through multiple upsampling generators. G and MAT indicate the generator block and the Multimodal Affine Transformation respectively

.

We observe that the input signals, i.e. state $s_t$ and image $I_{t-1}$, become attenuated as they pass through deeper generating layers and the network produces images with poor quality. So, similar to recent style-based image synthesis algorithms [19–21, 46, 35], we adopt a learned affine transformation architecture to inject auxiliary signals to the generator. The difference between the previous style-based generative models and S2P is that we propose a multimodal affine transformation (MAT) to produce the learnable modulation parameters, $\gamma$ and $\beta$, with the cross-modality representation via a multimodal feature extractor and state-to-latent mapping function. The overall architecture of our proposed S2P is depicted in Figure 2.

A non-linear latent mapping function $g : \mathcal{S} \rightarrow \mathcal{W}$ which is implemented as an 8-layer MLP produces a latent code $\mathbf{w} \in \mathcal{W}$ from the given state $s$ in the state space $\mathcal{S}$. The latent code $\mathbf{w}$ is spatially expanded as the same size of the input feature of MAT module $\mathbf{x}$, and the conditioned image $I_{t-1}$ is also linearly interpolated to make its size same as $\mathbf{x}$. The spatially expanded $\mathbf{w}$ and the resized $I_{t-1}$ are channel-wisely concatenated and fed to the multimodal feature extractor to fuse the state and image cross-modality representation. Each estimator then produces the learnable scale $\gamma$ and bias $\beta$ for effective cross-modal affine transformation.

Finally, our proposed MAT operation is defined as:

$$\mathbf{y}^i = \gamma^i(\mathbf{w}, I_{t-1}) \odot \frac{\mathbf{x}^i - \mu_c(\mathbf{x}^i)}{\sigma_c(\mathbf{x}^i)} + \beta^i(\mathbf{w}, I_{t-1}), \tag{2}$$

where $\mu_c(\cdot)$ and $\sigma_c(\cdot)$ are the channel-wise mean and the standard deviation of the input feature of MAT $\mathbf{x}^i$ from the $i^{th}$ block of the generator, and $\odot$ denotes Hadamard element-wise product. The design of MAT is illustrated in Figure 3.

In addition, it is shown that the neural network is biased toward learning a low frequency mapping and has difficulty in representing a high frequency information [48, 40]. To mitigate such undesired tendency, we do not use naïve state vector $s$ as input, but employ a positional encoding with the high frequency function $\psi : \mathbb{R} \rightarrow \mathbb{R}^{2L}$ which is defined as:

$$\psi(x) = (\sin(2^0\pi x), \cos(2^0\pi x), \cdots, \sin(2^{L-1}\pi x), \cos(2^{L-1}\pi x)), \tag{3}$$

where $x$ indicates each component of state vector $s$.

We utilize multi-scale discriminators in [55] to increase the receptive field without deeper layers or larger convolution kernels for alleviating the overfitting. Two discriminators with identical architecture are adopted during training, and the synthesized and real images which are resized to several spatial sizes are fed to each discriminator.

**Loss Function.** Our S2P generator is trained by linearly combined multiple objectives. First, we leverage a pixel-wise $\mathcal{L}_1$ loss between the output of the generator $\hat{I}_t$ and the real image $I_t$,

$$\mathcal{L}_1 = ||I_t - \hat{I}_t||_1. \tag{4}$$

In addition to the pixel-wise loss, we also utilize the ImageNet [50] pre-trained VGG19 [52] network to calculate the perceptual similarity loss $\mathcal{L}_{per}$ between two images,

$$\mathcal{L}_{per} = ||\phi^{(i)}(I_t) - \phi^{(i)}(\hat{I}_t)||_1, \tag{5}$$

where $\phi^{(i)}$ denotes the i-th layer of VGG19.

We implement the adversarial objective $\mathcal{L}_{adv}$ for both S2P generator and multi-scale discriminators same as pix2pixHD [55]. The difference is that we replace the least square loss with the hinge-based loss [37] and we condition state information to the discriminator $D$ so that the generator is induced to produce the dynamics-consistent outputs.

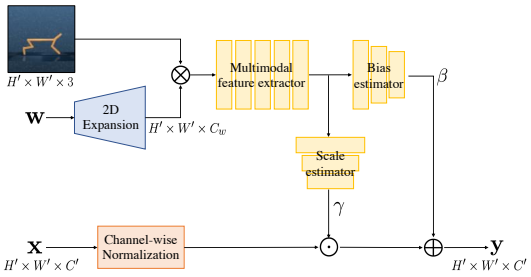



Figure 3: Multimodal Affine Transformation (MAT) module.

$$\begin{gathered}\mathcal{L}_D = -\mathbb{E}[\min(0, -1 + D(s_t, I_t))] - \mathbb{E}[\min(0, -1 - D(s_t, \hat{I}_t))],\ \mathcal{L}_G = -\mathbb{E}[-D(s_t, \hat{I}_t)] \\ \mathcal{L}_{adv} = \mathcal{L}_G + \mathcal{L}_D\end{gathered} \tag{6}$$

The total loss function to optimize the S2P generator can be defined as:

$$\mathcal{L} = \lambda_1 \mathcal{L}_1 + \lambda_2 \mathcal{L}_{per} + \lambda_3 \mathcal{L}_{adv}, \tag{7}$$

where $\lambda_1$, $\lambda_2$, and $\lambda_3$ are the hyperparameters to balance among the objectives.

### 3.2 Offline reinforcement learning with synthetic data

We consider the Markov decision process (MDP) $\mathcal{M} = (\mathcal{I}, \mathcal{S}, \mathcal{A}, T, r, \rho_0, \gamma)$, where $\mathcal{I}$ denotes the image space, $\mathcal{S}$ the state space corresponding to $\mathcal{I}$, $\mathcal{A}$ the action space, $T(s'|s,a)$ the transition dynamics, $r(s,a)$ the reward function, $\rho_0$ the initial distribution, and $\gamma$ the discount factor. We denote the discounted image visitation distribution of a policy $\pi$ using $d^{\pi}_{\mathcal{M}}(I) := (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \mathcal{P}(I_t = I|\pi)$, where $\mathcal{P}(I_t = I|\pi)$ is the probability of reaching image observation $I$ at time $t$ by using $\pi$ in $\mathcal{M}$. Similarly, we denote the image-action visitation distribution with $d^{\pi}_{\mathcal{M}}(I,a) := d^{\pi}_{\mathcal{M}}(I)\pi(a|I)$. The objective of RL is to optimize a policy $\pi(a|I)$ that maximizes the expected discounted return $J(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{(I,a)\sim d^{\pi}_{\mathcal{M}}(I,a)}[r(s,a)]$.

In the offline RL setting, the algorithm has access to a static dataset $D = \{(I_t, s_t, a_t, r_t, I_{t+1}, s_{t+1})\}_{t=0}^{N}$ collected by unknown behavior policy $\pi_\beta$. In other words, the dataset $D$ is obtained from $d^{\pi_\beta}_{\mathcal{M}}(I,a) := d^{\pi_\beta}_{\mathcal{M}}(I)\pi_\beta(a|I)$ and the goal is to find the best possible policy using the static dataset without online interaction with the environment.

To utilize the dynamics-consistent transition data for augmentation in offline RL, we take the model-based approach that trains an ensemble of dynamics and reward model $\hat{T}_\theta(s', r|s,a)$, which outputs the predicted next state, and reward $\hat{r}(s,a)$. Once a model has been learned, we can construct the learned MDP $\widehat{\mathcal{M}} = (\mathcal{I}, \mathcal{S}, \mathcal{A}, \hat{T}, \hat{r}, \rho_0, \gamma)$, which has the same spaces, but uses the learned dynamics and reward function. Naively optimizing the RL objective with the $\widehat{\mathcal{M}}$ is known to fail in the offline RL setting, both in theory and practice [22, 59], due to the distribution shift and model-bias. To overcome these, we take an uncertainty estimation algorithm like bootstrap ensembles [45, 39], and obtain $u(s,a)$, an estimate of uncertainty in dynamics. Then we could utilize the uncertainty penalized reward $\tilde{r}(s,a) = \hat{r}(s,a) - \lambda u(s,a)$, where $\lambda$ is a hyperparameter. We consider the following uncertainty quantification method that uses the maximum learned variance over the ensemble, $u(s,a) = \max_{i=1,\dots N} ||\Sigma^i_\theta(s,a)||_F$ [59, 22].

As a final process, we train offline RL with the following hybrid objective : $J(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{(I,a)\sim d_f(I,a)}[r(s,a)]$, where $d_f(I,a) = f d^{\pi_\beta}_{\mathcal{M}}(I,a) + (1-f) d^{\eta}_{\widehat{\mathcal{M}}}(I,a)$, $f \in [0,1]$ is the

Table 1: Offline RL results for DMControl. The numbers are the averaged normalized scores proposed in [7], where 100 corresponds to expert and 0 corresponds to the random policy. The results with standard deviation are in Appendix B.

| Environment | Dataset | IQL | IQL +S2P | CQL | CQL +S2P | SLAC-off | SLAC-off +S2P |
|---|---|---|---|---|---|---|---|
| cheetah, run | random | 10.28 | **12.64**(16.21) | 4.89 | **11.77**(7.52) | 16.37 | **18.14**(35.38) |
| walker, walk | random | -0.28 | **4.03**(0.83) | -0.43 | **10.44**(1.99) | 18.23 | 17.38(20.15) |
| ball in cup, catch | random | 74.77 | **82.28**(80.39) | 84.87 | **92.81**(91.61) | 70.04 | **85.57**(52.81) |
| reacher, easy | random | 33.75 | **70.45**(55.33) | 52.32 | **75.01**(81.48) | 77.43 | **85.76**(87.84) |
| finger, spin | random | -0.17 | **0.46**(-0.11) | -0.01 | -0.11(0.07) | 30.24 | 27.62(32.65) |
| cartpole, swingup | random | 24.52 | **38.59**(29.1) | 27.67 | **32.93**(42.12) | 35.03 | 31.01(52.22) |
| cheetah, run | mixed | 41.68 | **88.53**(70.44) | 92.63 | **93.16**(93.48) | 16.63 | **26.39**(24.42) |
| walker, walk | mixed | 96.07 | 95.49(97.80) | 97.18 | **97.84**(98.70) | 29.02 | **92.60**(67.09) |
| ball in cup, catch | mixed | 41.94 | 37.79(40.65) | 30.82 | **51.28**(37.21) | 28.54 | **40.41**(32.88) |
| reacher, easy | mixed | 66.88 | **75.61**(75.01) | 70.37 | **75.53**(77.54) | 62.49 | **63.59**(77.58) |
| finger, spin | mixed | 98.18 | 94.78(98.65) | 98.54 | 87.17(80.07) | 64.41 | **83.31**(83.29) |
| cartpole, swingup | mixed | 14.49 | 14.04(51.25) | 14.76 | -4.66(36.94) | 14.51 | **16.36**(25.41) |
| cheetah, run | expert | 79.89 | **87.18**(88.89) | 94.20 | **96.28**(95.54) | 8.92 | **14.41**(8.42) |
| walker, walk | expert | 94.34 | **94.97**(94.35) | 95.43 | **97.97**(98.47) | 11.71 | **70.95**(19.66) |
| ball in cup, catch | expert | 28.57 | **28.60**(28.48) | 28.42 | **28.62**(28.68) | 28.56 | **38.87**(28.69) |
| reacher, easy | expert | 52.13 | **58.19**(57.51) | 57.68 | 32.54(48.46) | 26.61 | **42.85**(34.49) |
| finger, spin | expert | 98.42 | 94.42(99.19) | 73.07 | **97.25**(99.51) | 24.75 | **81.05**(52.21) |
| cartpole, swingup | expert | 20.43 | 18.37(18.03) | 19.35 | 18.54(30.22) | 14.11 | 11.18(-3.80) |

ratio of the datapoints drawn from the offline dataset $D$, and $\eta(\cdot|s)$ is the state rollout distribution used with the trained dynamics ensemble model $\hat{T}_\theta$, and $\widetilde{\mathcal{M}}$ is same as $\widehat{\mathcal{M}}$ except that the reward is $\tilde{r}(s,a)$ instead of $\hat{r}(s,a)$. Samples from $d^{\eta}_{\widetilde{\mathcal{M}}}(I,a)$ can be obtained by rollout $\eta$ in $\widetilde{\mathcal{M}}$ and convert the obtained state transitions $\{(s_t,a_t,r_t,s_{t+1})\}_{t=0}^{N}$ into $\{(I_t,a_t,r_t,I_{t+1})\}_{t=0}^{N}$ by using the trained S2P generator in Section 3.1. For implementation, we collect synthetic image transition data in separate replay buffer $D_{model}$ and train the agent by any offline RL algorithms with the sampled mini-batches from $D$ and $D_{model}$ by the ratio of $f$ and $1-f$. The overall algorithm and more training details are summarized in Appendix B.

# 4 Experiments

## 4.1 Environments & Data collection

We evaluate our method on a large subset of the dataset from the DeepMind Control (DMControl) suite [54]. It includes 6 environments, which were typically used for online image-based RL benchmarks. However, to the best of our knowledge, all of these environments have never been properly evaluated in an offline image-based RL setting. The datasets in these benchmarks are generated as follows: **random** : rollout by a random policy that samples actions from a uniform distribution. **mixed** : train a policy using state-based SAC [10] until 500k steps for finger, cheetah and 100k steps for the others, then randomly samples trajectories from the replay buffer. 500k, 100k steps are minimum steps required to reach the expert level performance for each task. **expert** : train a policy using state-based SAC. After convergence, we collect trajectories from the converged policy.

## 4.2 Offline Reinforcement Learning

To validate whether the S2P can help improve the offline RL performance, we evaluate our algorithm with recent offline RL algorithms like **CQL** [27] which utilizes conservative training of the critic, and **IQL** [25] which trains critic by implicitly querying actions near the distribution of the dataset, and **SLAC-off** [33] which is state-of-the-art online image-based RL algorithm. We use this **SLAC-off** with the offline setting. We also compare the policy constraint-based offline RL algorithms like **BEAR** [26], and behavior cloning, **BC**. The results for these two algorithms are in Appendix B. To extend the offline RL into the image-based setting, we follow the image encoder architecture from [33] and train a variational model using the offline data. Then, we train in the latent space of this model.

We include the offline RL results on different types of environments and data in Table 1. The results on the originally given offline dataset (50k) are shown in the left column of each algorithm, and the results on the S2P-based augmented dataset are shown in the right column of each algorithm. The S2P augments the same amount of the original offline dataset (50k). As it physically has more data (100k) than the original dataset (50k), for a reference, we also include the results on the 100k offline dataset in the parenthesis in Table 1. Overall, the S2P-based method achieves better performance than the 50k dataset, even exceeding the 100k dataset's score in some tasks.

As the S2P answers how to generate the image transition data, we also have to consider where to generate the image transition data through S2P. Specifically, we use a random policy as $\eta(\cdot|s)$ in the mixed, expert dataset, and a policy trained by the state-based offline RL as $\eta(\cdot|s)$ in the random dataset. These strategies are considered due to the following two assumptions. Firstly, as the random dataset only has random behavior, the dataset may not have any meaningfully rewarded states even with the augmentation by the random policy, especially in the locomotion environment (e.g. cheetah, walker cannot leave the initial states by the random policy). As the S2P's objective is to extend the distribution of the given dataset, the policy trained in an offline manner can help leave the support of the dataset compared to the naive random policy. Secondly, as the non-random datasets may have relatively biased state-action distributions that receive meaningful rewards compared to the random dataset, it is difficult to get out of the support of these datasets by the trained policy as most of the state-action induced by the trained policy are included in the similar distribution of these datasets. However, the random policy can be effective as it can bring some exploration effects like noise injection or increasing entropy in [36, 10].

To verify such assumptions, we analyze the effect of different types of rollout distribution $\eta(\cdot|s)$ in offline RL performance (Table 2). We denote **50k dataset** as the results of the given original offline dataset, and **+S2P(random $\eta$)** as the results of the data augmented by rollout with the random actions, and **+S2P(offRL $\eta$)** as the results of the data augmented by rollout with the state-based offline RL policy. As expected, the random policy is more effective in the expert dataset. Also, we could find the opposite phenomenon in the random dataset, and trade-offs between these two strategies in the mixed dataset.

| DATASET | METHOD | 50K DATASET | +S2P (RANDOM $\eta$) | +S2P (OFFRL $\eta$) |
|---|---|---|---|---|
| CHEETAH RUN RANDOM | IQL | 10.28 | -0.107 | **12.64** |
| | CQL | 4.89 | -0.69 | **11.77** |
| | SLAC-OFF | 16.37 | 11.94 | **18.14** |
| CHEETAH RUN MIXED | IQL | 41.68 | **88.53** | 58.67 |
| | CQL | 92.63 | **93.16** | 89.6 |
| | SLAC-OFF | 16.63 | 26.39 | **26.53** |
| CHEETAH RUN EXPERT | IQL | 79.89 | **87.18** | 79.20 |
| | CQL | 94.20 | **96.28** | 93.69 |
| | SLAC-OFF | 8.92 | **14.41** | 7.79 |

Table 2: Experiments on each different rollout distribution $\eta(\cdot|s)$ in cheetah-run environment.

### 4.3 Comparison with model-based image RL

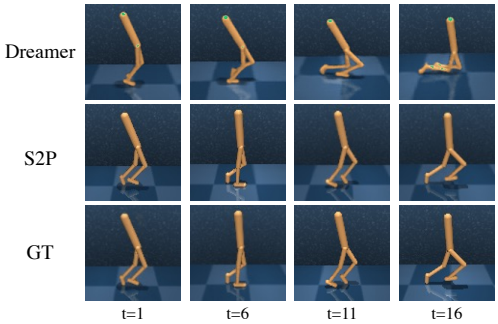



Figure 4: Qualitative comparison of S2P and Dreamer.

| dataset | method | 50k dataset | +S2P | +Dreamer |
|---|---|---|---|---|
| cheetah run mixed | IQL | 41.68 | **88.53** | 2.09 |
| | CQL | 92.63 | **93.16** | 58.93 |
| | SLAC-off | 16.63 | **26.39** | 4.68 |
| walker walk mixed | IQL | 96.07 | **95.49** | 1.28 |
| | CQL | 97.18 | **97.84** | 95.88 |
| | SLAC-off | 29.02 | **92.60** | 52.65 |
| cheetah run expert | IQL | 79.89 | **87.18** | 73.23 |
| | CQL | 94.20 | **96.28** | 53.69 |
| | SLAC-off | 8.92 | **14.41** | 3.65 |
| walker walk expert | IQL | 94.34 | **94.97** | 34.95 |
| | CQL | 95.43 | **97.97** | 96.14 |
| | SLAC-off | 11.71 | **70.95** | 52.03 |

Table 3: Quantitative comparison of S2P and Dreamer.

To show why the multi-modal inputs are necessary for augmenting the image transition data in offline image RL, we compare our S2P with the previous model-based image RL algorithm, Dreamer [12], as it can also reconstruct the images of the agent by training the reconstruction error from ELBO objective only using uni-modal inputs (previous images of the agent). For comparison, we trained Dreamer in an offline manner with the same dataset used for training S2P, and predicted future images with the episode context obtained from 5 consecutive ground truth images. Even though the Dreamer saw more previous steps' images compared to S2P (only a single image of the previous

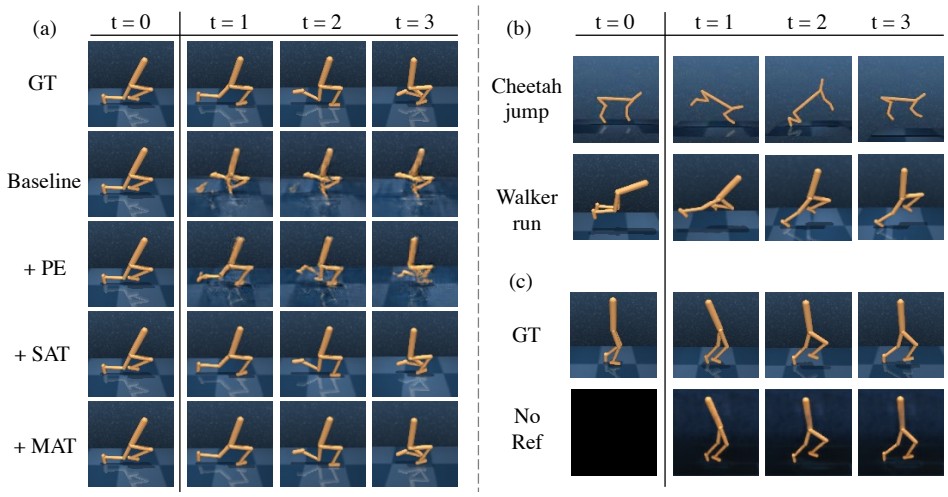



Figure 5: Qualitative results. We report the generator performance by synthesizing multiple steps with a single trajectory. (a) demonstrates **the effectiveness of each component** in the S2P generator where PE, SAT, and MAT indicate positional encoding, state affine transformation and multimodal affine transformation, respectively. SAT cannot perfectly estimate the correct location of the agent as there is **misalignment at the ground checkerboard** compared to MAT. (b) represents **unseen task adaptation** using the trajectories given from the state-level transition model. (c) shows that our model can recover the posture of the agent **without any reference image**. Additional qualitative results on several environments are provided in Appendix A.

step), it still has difficulty in generating accurate posture, which supports the S2P's advantages (Fig 4). It is because the priority of the model-based image RL algorithms is learning the effective latent representation for RL tasks, and they do not utilize a broader source of supervision from the state inputs.

To prove that state-inconsistent images from the model-based image RL method cannot improve offline RL performance at the S2P level, we perform the same experiment in Table 1, but replace the augmented images from S2P with images from Dreamer. We observe that the image augmentation from Dreamer even degrades the original RL performance in several tasks, and the performance improvements with S2P excels the augmentation from Dreamer by a large margin in all baselines (Table 3). Thus, we could say that the inaccurate posture and quality of the images generated by the model-based method trained with the uni-modal inputs (images) are not sufficient for augmenting image transition data in the offline setting. More experiments and details of the augmentation are in the Appendix.

### 4.4 Ablation

To observe how each component of the S2P contributes to the quality of the synthesized images, we perform an ablation study on the model architecture of the S2P and show its qualitative results on Figure 5(a). A baseline architecture without any contribution of our proposed method, i.e. positional encoding (PE) and multimodal affine transformation (MAT), shows the worst image quality. We report that a simple application of the high frequency function $\psi$ to the input state $s_t$ (PE) results in a noticeable increase in image quality. We also address the necessity of the input image $I_{t-1}$ to estimate the modulation parameters, $\gamma$, $\beta$, for the learned affine transformation. We ablate the input $I_{t-1}$ in MAT and utilize only the spatially expanded input state which is called State Affine Transformation (SAT). The generator which replaces the MAT module with SAT has difficulty in exploiting dynamics information from the previous image $I_{t-1}$ and the translation error is accumulated as the generator recurrently synthesizes the long-horizon trajectory. We can observe such dynamics inconsistency especially in the locomotion tasks, e.g. cheetah, and walker, which expresses the velocity in images by the change of the checkerboard in the ground. Compared to SAT, our proposed MAT which leverages both the state $s_t$ and the previous image $I_{t-1}$ shows better performance in reconstructing not only the posture of the agent but also its dynamics-consistent background.

### 4.5 Zero-shot Task Adaptation

To validate whether the S2P can help the offline RL process in settings that require generalization to tasks that are different from the given dataset, we construct two environments **cheetah-jump** and **walker-run**. In **cheetah-jump**, which is referred from [59], the agent should solve a task that is different from the original purpose of the behavior policy. Specifically, we relabel the rewards in the **cheetah-run-mixed** dataset to reward the cheetah to jump as high as possible. Then, we generate the image transition data from the states whose z positions are bigger than a threshold to validate the S2P-based augmentation's advantage in tasks that require generalization. By training with the relabeled reward, the agent with the S2P achieves a higher return and learns to bounce back and forth to take a leap higher (Figure 5(b)), even though the batch data contain little jumping motion.

To investigate whether the S2P can generate unseen image distributions from unseen state distribution, we collect **walker-run** state dataset by the same way of other mixed datasets in Section 4.1. Then, we generate images from these unseen states by recurrently using the S2P generator and we apply offline RL on these generated image transition data. Even though the state distribution is totally different from **walker-walk** dataset as the agent should run instead of walk, not only the S2P successfully generalizes to the unseen image distributions (Figure 5(b)), but also the agent can be trained to run by offline RL only with these synthesized images beyond the expertise of the dataset (Table 4). More details are in Appendix B.

We attribute such satisfactory task generalization to the model architecture of the S2P which leverages both $I_{t-1}$ and $s_t$ for synthesizing the $I_t$. The posture of the agent is deterministic with the sole-state condition and the background of the image such as the ground checkerboard is dependent both on the state and the previous image. Therefore, our S2P exploits the state input to generate the posture of the agent and exploits the image input to generate the background. It is well shown in Figure 5(c) where we intentionally replace all the image input with the zero matrix $O_{H\times W\times 3}$ during the inference phase. We observe that S2P still perfectly reconstructs the posture of the agent with its corresponding state only while it fails to recover the background and the ground checkerboard as we expected.

| METHOD | WALKER-RUN | METHOD | CHEETAH-JUMP |
|---|---|---|---|
| BATCH MEAN | 572.94 | BATCH MEAN | 2.05 |
| IQL | N/A | IQL | 36.6 |
| IQL+S2P | **659.25** | IQL+S2P | **54.4** |
| CQL | N/A | CQL | 40.8 |
| CQL+S2P | **652.19** | CQL+S2P | **47.2** |

Table 4: Average returns of the walker-run and cheetah-jump tasks. We include mean undiscounted return of the episodes in the batch data for comparison.

## 5 Conclusion

We firstly present the state-to-pixel (S2P) algorithm that synthesizes the raw pixel from its corresponding state and the previous image. As the augmentation paradigm of the S2P is generating dynamics-consistent image transition data, we demonstrate that S2P not only improves the image-based offline RL performance but also shows powerful generalization capability on unseen tasks. Even though S2P shows promising results in offline RL by data augmentation techniques, the assumption that datasets consist of pairs of images and states is still a strong assumption. Thus, for future work, we plan to further extend the idea with more relaxed assumptions such as unpaired datasets or extend other state-based applications to the image-based RL algorithms.

## 6 Acknowledgement

This research was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) [NO.2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University)]

## References

[1] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in Neural Information Processing Systems*, 31, 2018.

[2] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472. Citeseer, 2011.

[3] Fei Deng, Ingook Jang, and Sungjin Ahn. Dreamerpro: Reconstruction-free model-based reinforcement learning with prototypical representations. *arXiv preprint arXiv:2110.14565*, 2021.

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

[5] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *arXiv preprint arXiv:1610.07629*, 2016.

[6] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

[7] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[8] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf`.

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

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

[12] Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=S1lOTC4tDS`.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[15] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

[16] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems*, 32:12519–12530, 2019.

[17] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

[18] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[19] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[20] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.

[21] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[22] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Sys-*

*tems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f7efa4f864ae9b88d43527f4b14f750f-Abstract.html.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[25] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[26] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32:11784–11794, 2019.

[27] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/0d2b2061826a5df3221116a5085a6052-Abstract.html.

[28] Vikash Kumar, Emanuel Todorov, and Sergey Levine. Optimal control with learned local models: Application to dexterous manipulation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 378–383. IEEE, 2016.

[29] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=SJJinbWRZ.

[30] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

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

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

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

[34] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[35] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip HS Torr. Manigan: Text-guided image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7880–7889, 2020.

[36] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1509.02971.

[37] Jae Hyun Lim and Jong Chul Ye. Geometric gan. *CoRR*, abs/1705.02894, 2017. URL http://arxiv.org/abs/1705.02894.

[38] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Provably good batch off-policy reinforcement learning without great exploration. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/0dc23b6a0e4abc39904388dd3ffadcd1-Abstract.html.

[39] Kendall Lowrey, Aravind Rajeswaran, Sham M. Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

URL https://openreview.net/forum?id=Byey7n05FQ.

[40] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020.

[41] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[42] Tung D Nguyen, Rui Shu, Tuan Pham, Hung Bui, and Stefano Ermon. Temporal predictive coding for model-based planning in latent space. In *International Conference on Machine Learning*, pages 8130–8139. PMLR, 2021.

[43] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[44] Masashi Okada and Tadahiro Taniguchi. Dreaming: Model-based reinforcement learning by latent imagination without reconstruction. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4209–4215. IEEE, 2021.

[45] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[46] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019.

[47] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *Learning for Dynamics and Control*, pages 1154–1168. PMLR, 2021.

[48] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.

[49] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[50] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[51] Max Schwarzer, Ankesh Anand, Rishab Goel, R. Devon Hjelm, Aaron C. Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=uCQfPZwRaUu.

[52] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[53] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[54] Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

[55] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

[56] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

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

[58] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=GY6-6sTvGaf.

[59] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y. Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: model-based offline policy optimization. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information*

*Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/a322852ce0df73e204b7e67cbbef0d0a-Abstract.html.

[60] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *arXiv preprint arXiv:2102.08363*, 2021.

[61] Wenxuan Zhou, Sujay Bajracharya, and David Held. PLAS: latent action space for offline reinforcement learning. In Jens Kober, Fabio Ramos, and Claire J. Tomlin, editors, *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, volume 155 of *Proceedings of Machine Learning Research*, pages 1719–1735. PMLR, 2020. URL https://proceedings.mlr.press/v155/zhou21b.html.

## Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 4
   (b) Did you describe the limitations of your work? [Yes] See Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] This work inherits the potential negative societal impacts of reinforcement learning. We do not anticipate any additional negative impacts that are unique to this work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We included code in the supplemental materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 3 and Appendix
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 4
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 4
   (b) Did you mention the license of the assets? [Yes] See Appendix.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include the code & model in the supplemental materials.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

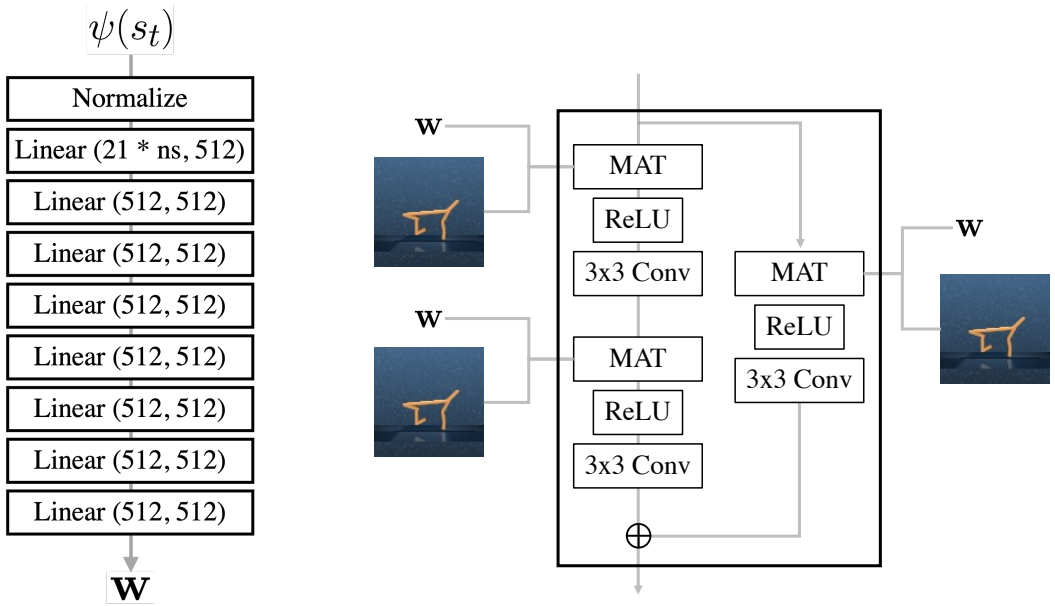
# A Image generation

## A.1 Architecture specification

The generator architecture is implemented with several MAT Residual Blocks followed by bilinear upsampling as shown in Figure 6(b). The architecture of the residual block is largely implemented from Park et al. [46] where we replace the SPADE module with MAT in SPADE Residual block. We also leverage the Spectral Normalization [41] to all the convolution layers in the generator. The latent mapping network $g$ for generating the latent code $\mathbf{w}$ is implemented with an 8-layer of MLP with channel-wise normalization at the first layer, which is the same as the style mapping function in StyleGAN [19]. The first MLP layer of the latent mapping function is different from the physical environment as the number of the input state, $ns$, is different. The dimension of the input state is increased $2L$ times by a high frequency positional encoding function $\psi$. We set the hyperparameter $L$ as 10 and concatenate the input and output of $\psi$, which leads to 21 times increase in the channel dimension. The specification of the entire architecture is shown in Figure 7.

| Environment | Cheetah run | Walker walk | Cartpole swingup | Finger spin | Ball-in-cup cath | Reacher easy |
|---|---|---|---|---|---|---|
| $ns$ | 17 | 24 | 5 | 9 | 8 | 6 |

Table 5: The number of the state according to the environment of DMControl.




(a) Latent mapping function $g$

(b) MAT residual block

Figure 6: An architecture of the sub-network in the generator.

## A.2 Training details

We implement our proposed generator architecture with the public deep learning platform PyTorch and train it on a single NVIDIA RTX A6000 GPU. We train 30 epochs for each task and the generated image size is set to $128 \times 128$. An Adam optimizer [23] with the learning rate of 0.0002 is utilized and the batch size is set to 16.

## A.3 Additional results of image synthesis

We report additional qualitative results on the multiple environments of DMControl in Figure 8. We show that our proposed S2P generates high-quality images regardless of the environment. We recurrently generate a single trajectory exploiting the current state and the previous images which are also the output of the generator from the previous state. Also, we evaluate the quality of generated

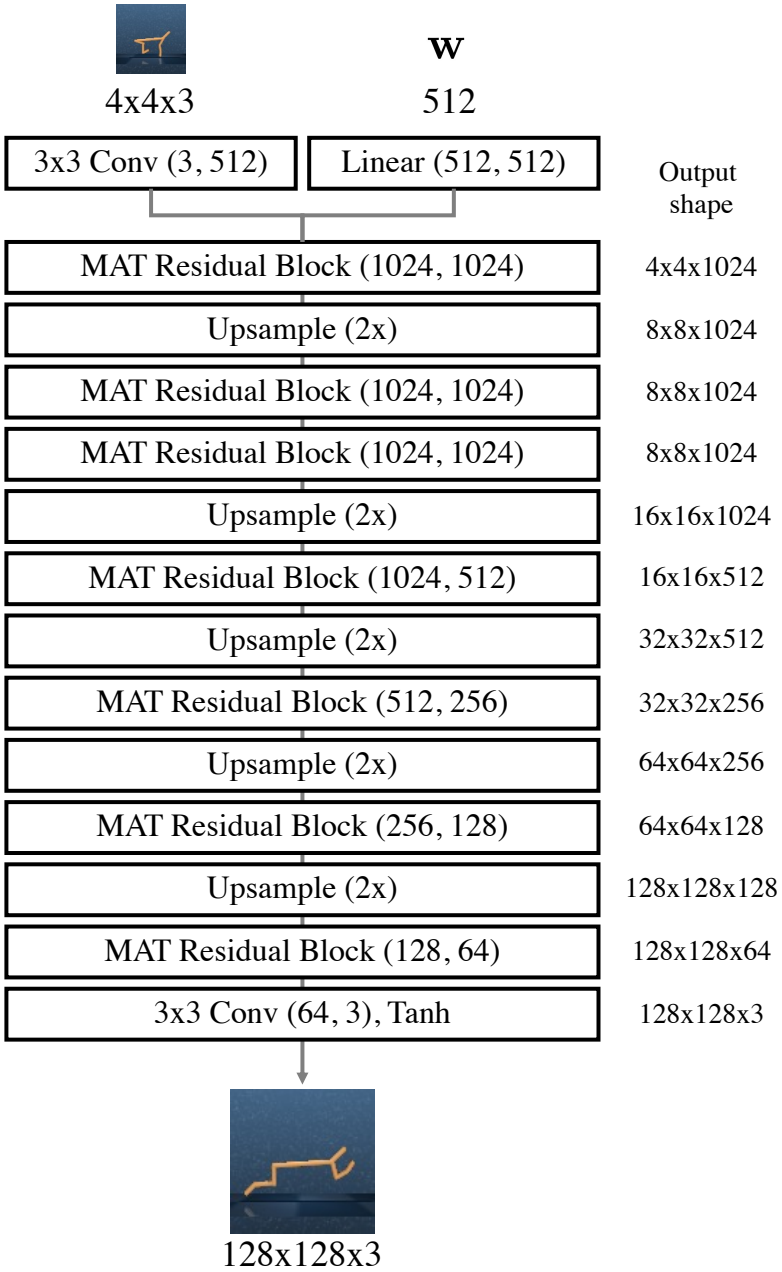



Figure 7: Specification of the generator.

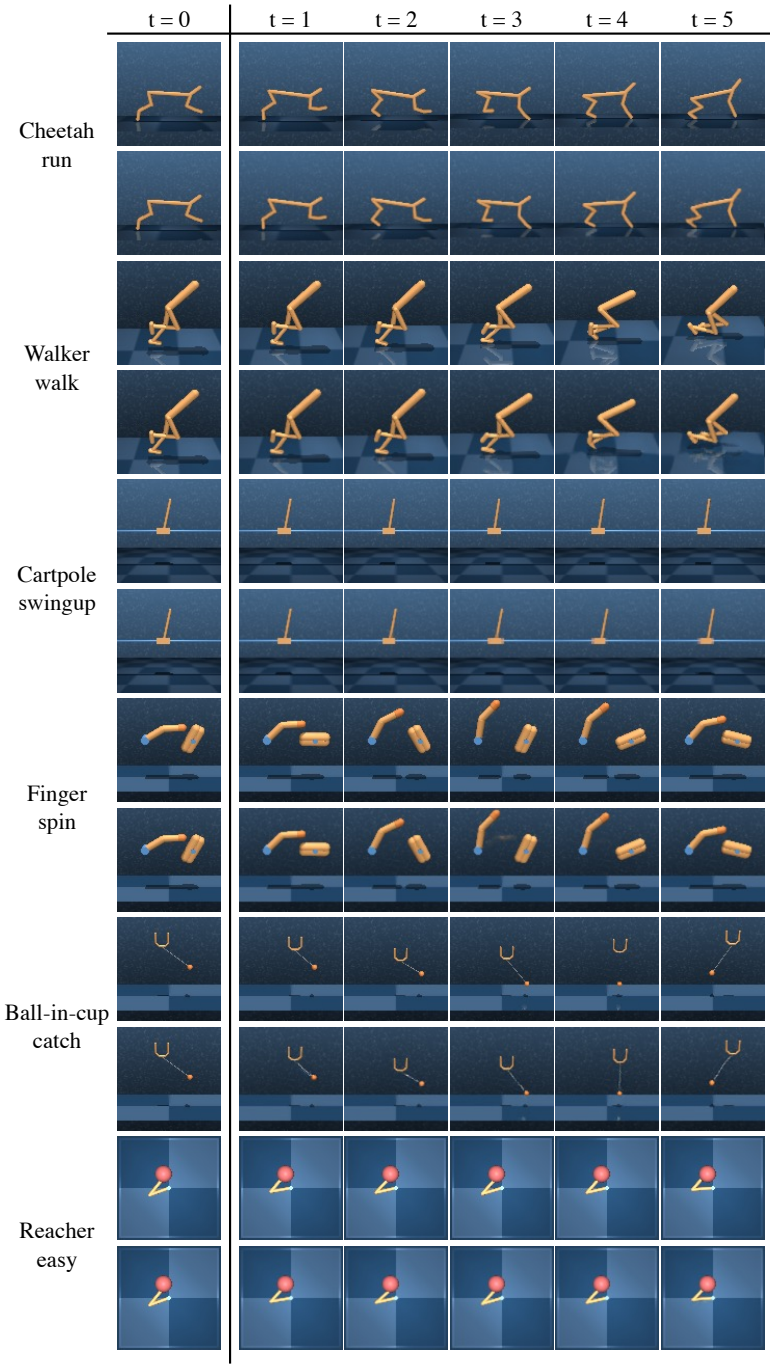



Figure 8: Additional qualitative results on the DMControl environment. The first row of each environment is ground truth images and the second row is the synthesized images from S2P.

| Environment | Method | FID (↓) | LPIPS (↓) | PSNR (↑) | SSIM (↑) |
|---|---|---|---|---|---|
| Cheetah run | Dreamer | 63.46 | 0.042 | 27.62 | 0.90 |
| | S2P | 47.70 | 0.028 | 33.40 | 0.94 |
| Walker walk | Dreamer | 209.99 | 0.30 | 18.38 | 0.69 |
| | S2P | 74.08 | 0.078 | 25.20 | 0.84 |
| Cartpole swingup | Dreamer | 82.02 | 0.129 | 28.05 | 0.94 |
| | S2P | 112.81 | 0.117 | 28.83 | 0.86 |
| Ball-in-cup catch | Dreamer | 112.45 | 0.055 | 33.25 | 0.93 |
| | S2P | 77.11 | 0.035 | 33.75 | 0.97 |
| Reacher easy | Dreamer | 171.92 | 0.115 | 27.64 | 0.95 |
| | S2P | 58.34 | 0.178 | 26.02 | 0.86 |
| Finger spin | Dreamer | 86.63 | 0.112 | 21.93 | 0.90 |
| | S2P | 18.86 | 0.025 | 39.54 | 0.99 |
| Mean value | Dreamer | 121.13 | 0.126 | 28.19 | 0.89 |
| | S2P | **64.82** | **0.077** | **31.13** | **0.91** |

Table 6: Quantitative results of generated images. S2P outperforms Dreamer in all the metrics for evaluating image quality.

images quantitatively with metrics (FID score, LPIPS, PSNR, SSIM) which are frequently adopted for evaluating image quality in Table 6. Our S2P outperforms Dreamer in all the quantitative results which indicates the generated image from S2P has better quality than from Dreamer.

# B Offline RL Experiments Details

## B.1 Algorithm

**Algorithm 1** Offline RL with the S2P

**Input:** offline dataset $D$, state rollout distribution $\eta(\cdot|s)$.

Train the probabilistic dynamics models $\hat{T}_\theta(s', r|s, a) = \mathcal{N}(\mu_\theta(s, a), \Sigma_\theta(s, a))$ on $D$.
Train the image generator $G(s_t, I_{t-1})$ on $D$.
**for** $i = 1$ **to** $K$ **do**
  Randomly sample $I_0, s_0 \sim D$.
  Get $\tau_s \sim (s_0, a_0, r_0, s_1, ..., s_M)$ by using the $\eta(\cdot|s)$ and $\hat{T}_\theta(s', r|s, a)$.
  Generate $\tau_I \sim (I_0, a_0, r_0, I_1, ...I_M)$ from $\tau_s$ by $G$.
  Save $\tau_I$ in $D_{model}$.
**end for**
Apply any offline RL algorithm with the dataset sampled from $D, D_{model}$ with the ratio of $f$, $1 - f$.

## B.2 Ablation studies

### B.2.1 Uncertainty types

We conduct experiments on how the choice of the uncertainty affects the performance. We denote $u(s, a) = \max_{i=1,...N} ||\Sigma^i_\theta(s, a)||_F$ as **Max Var**, $u(s, a) = \max_{i=1,...N} ||\mu^i_\theta(s, a) - \frac{1}{N}\sum_{j=1}^{N} \mu^j_\theta(s, a)||_2$ as **Ens Var**, and average value of both uncertainty types as **Average Both**. In practice, we found that **Max Var** achieves better performance compared to other types of uncertainty in mixed, expert dataset, while **Ens Var** achieves better at random dataset. We hypothesize that the dynamics model is quite uncertain in the aspect of the **Max Var** on the random dataset due to the excessive randomness of the data. Thus, it could induce excessive penalty on the predicted reward when **Max Var** is used, and the agent could become too conservative.

| DATASET | METHOD | MAX VAR | AVERAGE BOTH | ENS VAR |
|---|---|---|---|---|
| CHEETAH RUN RANDOM | **IQL** | 12.64 | 16.61 | **19.27** |
| | **CQL** | 11.77 | 8.59 | **19.83** |
| | **SLAC-OFF** | 18.14 | 8.67 | **31.81** |
| CHEETAH RUN MIXED | **IQL** | 88.53 | **83.92** | 67.74 |
| | **CQL** | **93.16** | 83.93 | 87.5 |
| | **SLAC-OFF** | **26.39** | **26.39** | 24.41 |
| CHEETAH RUN EXPERT | **IQL** | **87.18** | 64.43 | 62.41 |
| | **CQL** | **96.28** | 89.38 | 90.36 |
| | **SLAC-OFF** | **14.41** | 9.85 | 11.92 |

Table 7: Different types of uncertainty quantification on cheetah-run environments.

### B.2.2 Rollout horizons

To validate the effectiveness of different rollout horizons, we conduct experiments with horizon length 1 (**+S2P (1step)**) and 5 (**+S2P (5step)**), while following the same rollout strategies in Section 4.2. For the 5 step case, as the proposed S2P generator is conditioned on the previous timestep's image to synthesize the next image, we recurrently generate the image transition data. That is, at the first timestep, the ground truth image is conditioned on the image generator, but after then, the generated image is conditioned on the image generator for generating the following timestep's image. The results shown in Table 8 represent that the augmentation with a longer horizon also has advantages in offline RL, but a short horizon is more effective overall. This is due to the uncertainty accumulation effect as shown in Figure 9. The average uncertainty grows as the rollout horizon increases due to the model bias, and it leads to more penalties on the predicted rewards, which can induce a too conservative agent.

| DATASET | METHOD | 50K DATASET | +S2P (1STEP) | +S2P (5STEP) |
|---|---|---|---|---|
| CHEETAH RUN RANDOM | **IQL** | 10.28 | **12.64** | 12.08 |
| | **CQL** | 4.89 | **11.77** | 11.46 |
| | **SLAC-OFF** | 16.37 | 18.14 | **19.09** |
| CHEETAH RUN MIXED | **IQL** | 41.68 | **88.53** | 66.38 |
| | **CQL** | 92.63 | **93.16** | 87.44 |
| | **SLAC-OFF** | 16.63 | 26.39 | **27.07** |
| CHEETAH RUN EXPERT | **IQL** | 79.89 | **87.18** | 81.01 |
| | **CQL** | 94.20 | **96.28** | 87.53 |
| | **SLAC-OFF** | 8.92 | 14.41 | **17.17** |

Table 8: Effect of rollout horizons in cheetah-run environment.

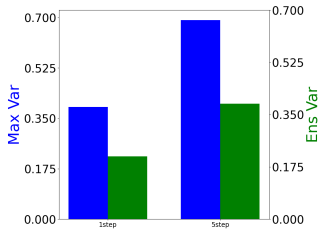



Figure 9: Average uncertainty of each different rollout horizon in cheetah-run environment.

### B.2.3 Results on policy constraint-based methods

We additionally test the proposed method on the policy constraint-based methods such as BEAR [26] and behavior cloning (BC) on the cheetah-run environment. As shown in Table 9, the performance of these methods with augmented data is worse than the results of non-augmented data. The poor performance is reasonable, because BEAR utilizes the action distribution's support matching by MMD, and BC is trained with maximizing the likelihood of the action. As the augmented action distributions are totally different from the behavior policy that induces the offline dataset, these two methods perform poorly because these methods try to clone or match both types of actions. That is, these methods try to clone or match the support of the given offline dataset and sampled actions that could have different distribution or support.

| DATASET | METHOD | 50K DATASET | +S2P (RANDOM $\eta$) | +S2P (OFFRL $\eta$) |
|---|---|---|---|---|
| CHEETAH RANDOM | BEAR | -1.15 | -1.39 | **1.14** |
| | BC | -1.41 | -1.3 | **2.54** |
| CHEETAH MIXED | BEAR | **10.64** | -0.29 | 6.57 |
| | BC | **51.81** | 0.06 | 5.17 |
| CHEETAH EXPERT | BEAR | **73.49** | 11.11 | 56.86 |
| | BC | 77.42 | 30.06 | **79.40** |

Table 9: Experiments on policy constraint-based methods.

### B.3 Additional experiments on Dreamer and conventional image augmentation technique

**Dreamer with larger dataset.** To validate whether the performance degradation by augmentation from Dreamer (Section 4.3) stems from the small dataset as the Dreamer requires more than 100k samples in online training, we collect an additional 250k dataset and augment the image transition amount of 250k by Dreamer (totally 500k datasets), and perform the same experiment. Despite the bigger size dataset, the performance does not increase overall (Table 10), and we could say that the inaccurate posture and quality of the images generated by the dreamer are attributed to the limited source of supervision from the uni-modal inputs rather than the size of the dataset, and it is not that proper for data augmentation in the offline setting.

**Comparison with the conventional image-augmentation technique.** To validate why S2P is needed instead of the conventional image-augmentation method, we additionally experiment with random crop and reflection padding, which is frequently used in image representation learning and online image RL. We perform the same experiment in Table 1, but replace the augmented images from S2P with randomly cropped images. Slightly better performance than Dreamer could be interpreted as it just manipulates the given true images (while maintaining the accurate posture of the agent) rather than generating new images like Dreamer (Table 10). But it still has difficulty in surpassing the S2P's results as it cannot deviate from the state-action distribution of the offline dataset.

| DATASET | METHOD | 50K DATASET | +S2P | +REFLECT RANDOMCROP | +DREAMER | +DREAMER 500K |
|---|---|---|---|---|---|---|
| CHEETAH RUN MIXED | IQL | 41.68 | **88.53** | 70.17 | 2.09 | 1.85 |
| | CQL | 92.63 | **93.16** | 79.78 | 58.93 | 82.55 |
| | SLAC-OFF | 16.63 | **26.39** | 3.10 | 4.68 | 0.14 |
| WALKER WALK MIXED | IQL | 96.07 | **95.49** | 95.39 | 1.28 | 7.21 |
| | CQL | 97.18 | **97.84** | 97.61 | 95.88 | 97.70 |
| | SLAC-OFF | 29.02 | **92.60** | 17.13 | 52.65 | 0.48 |
| CHEETAH RUN EXPERT | IQL | 79.89 | **87.18** | 68.39 | 73.23 | 3.50 |
| | CQL | 94.20 | **96.28** | 94.01 | 53.69 | 27.58 |
| | SLAC-OFF | 8.92 | **14.41** | 8.15 | 3.65 | 3.24 |
| WALKER WALK EXPERT | IQL | 94.34 | **94.97** | 92.46 | 34.95 | 37.92 |
| | CQL | 95.43 | **97.97** | 97.11 | 96.14 | 82.51 |
| | SLAC-OFF | 11.71 | **70.95** | 6.91 | 52.03 | 10.43 |

Table 10: Quantitative comparison of S2P and other data augmentation methods.

### B.4 Visualization of the image distribution

To validate whether the S2P really affects the distribution of the offline dataset, we visualized the cheetah-run-expert dataset and the S2P-based augmented dataset by the random policy by applying the t-sne (Figure 10).

As shown in Figure 10, the augmented dataset not only occupies a similar area to the original dataset but also encloses the original dataset, even connecting the clusters of the original dataset. It can be interpreted that S2P connects different modes of image distribution by deploying virtual exploration in the state space.

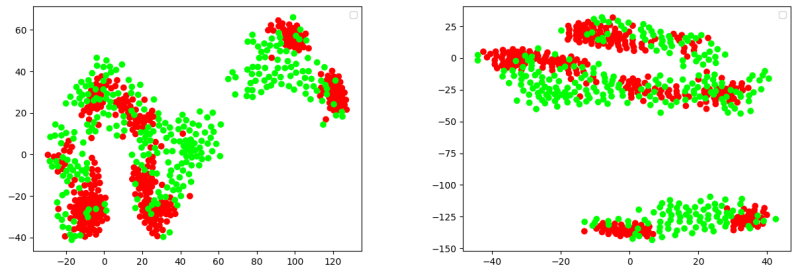


Figure 10: The t-sne visualizations examples of the cheetah-run-expert dataset (red) and the augmented dataset by the random policy (green).

### B.5 Model learning

In state space, we represent the dynamics, reward model as a probabilistic neural network that outputs a Gaussian distribution over the next state and reward given the current state and action.

$$T_\theta(s_{t+1}, r_t|s_t, a_t) = \mathcal{N}(\mu_\theta(s_t, a_t), \Sigma_\theta(s_t, a_t)) \tag{8}$$

We train an ensemble of $N$ dynamics models $\{\hat{T}_\theta = \mathcal{N}(\mu_\theta(s, a), \Sigma_\theta(s, a))\}_{i=1}^N$, with each model trained independently via maximum likelihood estimation on offline dataset $D$. We set the number of dynamics model $N$ as 7 following Yu et al. [60]. During model rollouts, we randomly pick one model. Each model in the ensemble is represented as 3 layers with 256 hidden units and relu activation function.

### B.6 Representation learning

For image-based offline RL process, we follow Lee et al. [33] that uses a variational model with the following components:

$$\begin{aligned}
\text{Image encoder} : h_t &\sim E_\theta(I_t) \\
\text{Posterior} : z_t &\sim q_\psi(z_t|h_t, z_{t-1}, a_{t-1}) \\
\text{Prior} : z_t &\sim p_\psi(z_t|z_{t-1}, a_{t-1}) \\
\text{Image decoder} : I_t &\sim D_\theta(I_t|z_t) \\
\text{Policy} : a_t &\sim \pi(a_t|I_{1:t}, a_{1:t-1})
\end{aligned}$$

Full derivation of those equations are in Lee et al. [33]. We train the representation model using the evidence lower bound :

$$\mathbb{E}_{z_{1:\tau+1}\sim q_\psi}\left[\sum_{t=0}^{\tau} -\log p_\psi(I_{t+1}|z_{t+1}) + D_{KL}(q_\psi(z_{t+1}|I_{t+1}, z_t, a_t)||p_\psi(z_{t+1}|z_t, a_t))\right]$$

where $\tau$ is the number of sequences, and z is the latent representation. We set $\tau = 8$ and the dimension of $z$ is 288, same as the original implementation. For image encoder, we use 6 convolutional neural network with kernel sizes [5, 3, 3, 3, 3, 4] and strides [2,2,2,2,2,2] respectively with leaky relu activation function. The decoder is constructed in a symmetrical manner to the encoder. We pre-train the image encoder by 300k steps and use the trained encoder's weights as initial weights when offline RL training proceeds.

### B.7 Training Detail

We use the batch size of 128 and the Adam optimizer for training the value function $Q$ and policy $\pi$ with the learning rates 0.0003, 0.0001, respectively. Each $Q$ and $\pi$ is represented as MLPs with

hidden layer sizes (1024, 1024), relu activation function. We apply tanh activation on the output of the policy with reparameterization trick same as Haarnoja et al. [10]. Also, we use the uncertainty penalty coefficient $\lambda = 2$ for all environments except the finger-spin, which use $\lambda = 1$. The sampling ratio of the offline dataset is $f = 0.5$. For offline RL implementation, we referred to the original implementations of each work and follow default parameters.

### B.8 Zero-shot Task Adaptation Detail

For the **cheetah-jump** task, we relabel the reward in the given offline dataset to the sparse reward that indicates whether the cheetah is jumping or not. That is, if z-init z $\geq$ 0.3, the reward is relabeled as 1, otherwise 0, where z denotes the z-position of the cheetah and init z denotes the initial z-position. For augmentation, we randomly select states whose z-position is greater than 0.2, and generate images from these states to encourage the jumping motion. Then, we augment these generated image transition data in the same way of Table 1, and evaluate the offline RL.

For the **walker-run** task, we generate the dataset in the same manner as the mixed type in Section 4.1. That is, we train the state-based SAC until convergence, and we randomly sample trajectories from the replay buffer. Then, we generate $\hat{I}_1$ from $s_1$ and $I_0$, where $I_0$ is the initial image when the agent is reset, and recurrently generate images $\hat{I}_{t+1}$ from $s_{t+1}, \hat{I}_t$ $(t = 0, 1, \ldots, N)$ by the S2P generator trained with the **walker-walk-mixed** dataset. After then, we apply offline RL on these generated image transition data.

### B.9 Environment Detail

The DMControl's environment details and the random and expert scores obtained by training the state-based SAC are shown in Table 11. The normalized score is computed by $100*$(return - random score)/(expert score - random score), which is proposed in Fu et al. [7].

| Environment | Expert Score | Random Score | Action Repeat | Maximum steps per episode |
|---|---|---|---|---|
| Cheetah-run | 900 | 12.6 | 4 | 250 |
| Walker-walk | 970 | 43.2 | 2 | 500 |
| Ball in cup-catch | 976 | 25.1 | 4 | 250 |
| Cartpole-swingup | 979 | 61.8 | 8 | 125 |
| Reacher-easy | 906 | 2.1 | 4 | 250 |
| Finger-spin | 882 | 125.2 | 2 | 500 |
| Walker-run | 790 | 25.2 | 2 | 500 |

Table 11: The environment details including the expert and random scores for computing normalized scores.

### B.10 Results with standard deviation

The results in the manuscript with their standard deviation are included in this section (Table 12, Table 13, Table 14, Table 15).

Table 12: Offline RL results for DMControl. The numbers are the averaged normalized scores proposed in [7], where 100 corresponds to expert and 0 corresponds to the random policy.

| Environment | Dataset | IQL | IQL +S2P | CQL | CQL +S2P | SLAC-off | SLAC-off +S2P |
|---|---|---|---|---|---|---|---|
| cheetah, run | random | 10.28±1.0 | **12.64**±1.5 (16.21±2.5) | 4.89±1.2 | **11.77**±6.4 (7.52±6.6) | 16.37±6.7 | **18.14**±3.1 (35.38±10.1) |
| walker, walk | random | -0.28±1.7 | **4.03**±3.5 (0.83±1.7) | -0.43±1.4 | **10.44**±3.4 (1.99±4.6) | 18.23±3.1 | 17.38±3.4 (20.15±4.1) |
| ball in cup, catch | random | 74.77±12.4 | **82.28**±9.2 (80.39±14.7) | 84.87±9.5 | **92.81**±6.4 (91.61±9.1) | 70.04±8.1 | **85.57**±5.1 (52.81±7.6) |
| reacher, easy | random | 33.75±3.8 | **70.45**±4.1 (55.33±4.5) | 52.32±3.8 | **75.01**±5.0 (81.48±3.1) | 77.43±9.1 | **85.76**±7.1 (87.84±8.8) |
| finger, spin | random | -0.17±0.5 | **0.46**±1.1 (-0.11±0.3) | -0.01±0.6 | -0.11±0.4 (0.07±0.7) | 30.24±3.1 | 27.62±3.2 (32.65±3.6) |
| cartpole, swingup | random | 24.52±3.7 | **38.59**±5.0 (29.1±5.0) | 27.67±5.2 | **32.93**±4.9 (42.12±3.6) | 35.03±8.7 | 31.01±7.1 (52.22±10.6) |
| cheetah, run | mixed | 41.68±13.2 | **88.53**±6.1 (70.44±12.4) | 92.63±3.0 | **93.16**±2.8 (93.48±1.5) | 16.63±5.1 | **26.39**±4.3 (24.42±5.2) |
| walker, walk | mixed | 96.07±3.1 | 95.49±2.8 (97.80±2.2) | 97.18±3.7 | **97.84**±2.2 (98.70±3.7) | 29.02±8.3 | **92.60**±7.1 (67.09±7.4) |
| ball in cup, catch | mixed | 41.94±3.9 | 37.79±5.1 (40.65±5.0) | 30.82±4.5 | **51.28**±4.8 (37.21±4.9) | 28.54±5.9 | **40.41**±4.4 (32.88±5.8) |
| reacher, easy | mixed | 66.88±4.5 | **75.61**±3.2 (75.01±3.5) | 70.37±4.8 | **75.53**±3.0 (77.54±3.9) | 62.49±6.1 | **63.59**±5.3 (77.58±6.5) |
| finger, spin | mixed | 98.18±1.1 | 94.78±3.6 (98.65±1.5) | 98.54±2.1 | 87.17±6.2 (80.07±7.3) | 64.41±9.3 | **83.31**±8.8 (83.29±10.2) |
| cartpole, swingup | mixed | 14.49±6.6 | 14.04±6.3 (51.25±13.7) | 14.76±5.7 | -4.66±6.9 (36.94±13.3) | 14.51±8.1 | **16.36**±7.7 (25.41±10.3) |
| cheetah, run | expert | 79.89±7.5 | **87.18**±5.6 (88.89±8.2) | 94.20±4.3 | **96.28**±3.4 (95.54±5.5) | 8.92±5.1 | **14.41**±5.0 (8.42±5.7) |
| walker, walk | expert | 94.34±6.4 | **94.97**±4.0 (94.35±6.3) | 95.43±4.5 | **97.97**±2.9 (98.47±3.6) | 11.71±9.8 | **70.95**±6.9 (19.66±8.8) |
| ball in cup, catch | expert | 28.57±14.9 | **28.60**±4.7 (28.48±10.5) | 28.42±11.6 | **28.62**±4.8 (28.68±14.7) | 28.56±11.1 | **38.87**±10.1 (28.69±10.8) |
| reacher, easy | expert | 52.13±4.6 | **58.19**±4.4 (57.51±4.6) | 57.68±6.9 | 32.54±4.1 (48.46±4.9) | 26.61±7.1 | **42.85**±6.1 (34.49±6.8) |
| finger, spin | expert | 98.42±1.5 | 94.42±1.5 (99.19±1.2) | 73.07±3.2 | **97.25**±2.9 (99.51±0.5) | 24.75±5.1 | **81.05**±5.2 (52.21±7.6) |
| cartpole, swingup | expert | 20.43±3.2 | 18.37±8.8 (18.03±10.2) | 19.35±11.1 | 18.54±7.4 (30.22±12.4) | 14.11±12.6 | 11.18±10.1 (-3.80±13.3) |

| DATASET | METHOD | 50K DATASET | +S2P (RANDOM $\eta$) | +S2P (OFFRL $\eta$) |
|---|---|---|---|---|
| CHEETAH RUN RANDOM | IQL | 10.28±1.0 | -0.107±0.27 | **12.64**±1.5 |
| | CQL | 4.89±1.2 | -0.69±0.4 | **11.77**±6.4 |
| | SLAC-OFF | 16.37±6.7 | 11.94±3.9 | **18.14**±3.1 |
| CHEETAH RUN MIXED | IQL | 41.68±13.2 | **88.53**±6.1 | 58.67±10.3 |
| | CQL | 92.63±3.0 | **93.16**±2.8 | 89.6±2.3 |
| | SLAC-OFF | 16.63±5.1 | 26.39±4.3 | **26.53**±5.9 |
| CHEETAH RUN EXPERT | IQL | 79.89±7.5 | **87.18**±5.6 | 79.20±16.0 |
| | CQL | 94.20±4.3 | **96.28**±3.4 | 93.69±2.0 |
| | SLAC-OFF | 8.92±5.1 | **14.41**±5.0 | 7.79±6.6 |

Table 13: Experiments on each different rollout distribution $\eta(\cdot|s)$ in cheetah-run environment.

| DATASET | METHOD | 50K DATASET | +S2P | +DREAMER |
|---|---|---|---|---|
| CHEETAH RUN MIXED | IQL | 41.68±13.2 | **88.53**±6.1 | 2.09±1.27 |
| | CQL | 92.63±3.0 | **93.16**±2.8 | 58.93±27.6 |
| | SLAC-OFF | 16.63±5.1 | **26.39**±4.3 | 4.68±1.3 |
| WALKER WALK MIXED | IQL | 96.07±3.1 | **95.49**±2.8 | 1.28±6.3 |
| | CQL | 97.18±3.7 | **97.84**±2.2 | 95.88±3.6 |
| | SLAC-OFF | 29.02±8.3 | **92.60**±7.1 | 52.65±8.6 |
| CHEETAH RUN EXPERT | IQL | 79.89±7.5 | **87.18**±5.6 | 73.23±15.6 |
| | CQL | 94.20±4.3 | **96.28**±3.4 | 53.69±21.3 |
| | SLAC-OFF | 8.92±5.1 | **14.41**±5.0 | 3.65±2.8 |
| WALKER WALK EXPERT | IQL | 94.34±6.4 | **94.97**±4.0 | 34.95±24.7 |
| | CQL | 95.43±4.5 | **97.97**±2.9 | 96.14±3.4 |
| | SLAC-OFF | 11.71±9.8 | **70.95**±6.9 | 52.03±9.9 |

Table 14: Quantitative comparison of S2P and Dreamer.

| METHOD | WALKER-RUN | METHOD | CHEETAH-JUMP |
|---|---|---|---|
| **BATCH MEAN** | 572.94±159.5 | **BATCH MEAN** | 2.05±18.7 |
| **IQL** | N/A | **IQL** | 36.6±15.2 |
| **IQL+S2P** | **659.25**±54.3 | **IQL+S2P** | **54.4**±10.1 |
| **CQL** | N/A | **CQL** | 40.8±12.2 |
| **CQL+S2P** | **652.19**±58.2 | **CQL+S2P** | **47.2**±10.4 |

Table 15: Average returns of the walker-run and cheetah-jump tasks. We include mean undiscounted return of the episodes in the batch data for comparison.