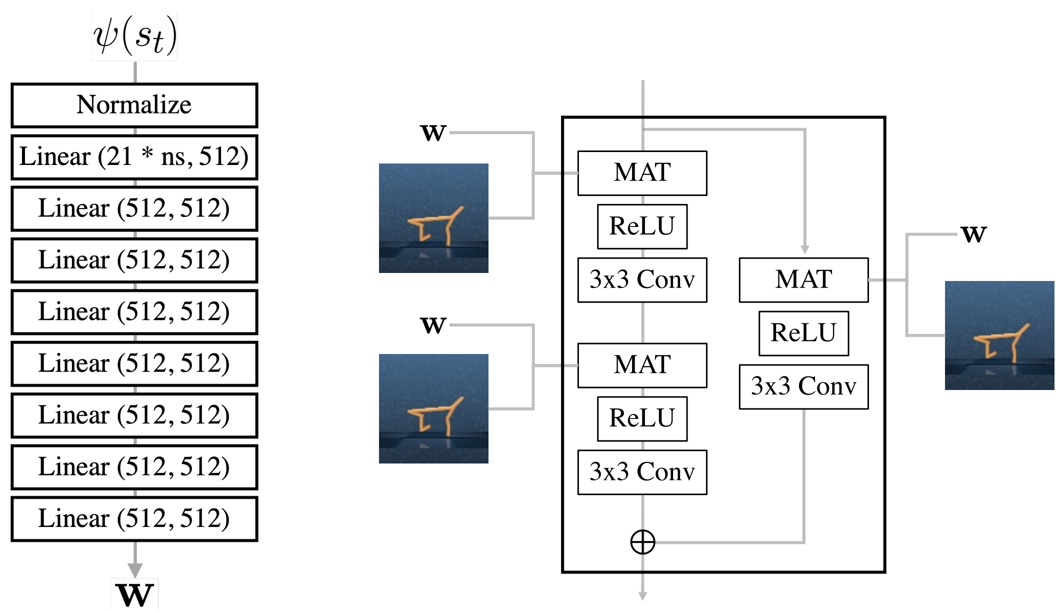
A Image generation

A.1 Architecture specification

The generator architecture is implemented with several MAT Residual Blocks followed by bilinear upsampling as shown in Figure 1(b). The architecture of the residual block is largely implemented from **?**] where we replace the SPADE module with MAT in SPADE Residual block. We also leverage the Spectral Normalization [**?**] 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 [**?**]. 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 ψ. We set the hyperparameter L as 10 and concatenate the input and output of ψ, which leads to 21 times increase in the channel dimension. The specification of the entire architecture is shown in Figure 2.

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

Table 1: The number of the state according to the environment of DMControl.



(a) Latent mapping function g

(b) MAT residual block

Figure 1: 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 128×128. An Adam optimizer [**?**] 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 3. We show that our proposed S2P generates high-quality images regardless of the environment. We

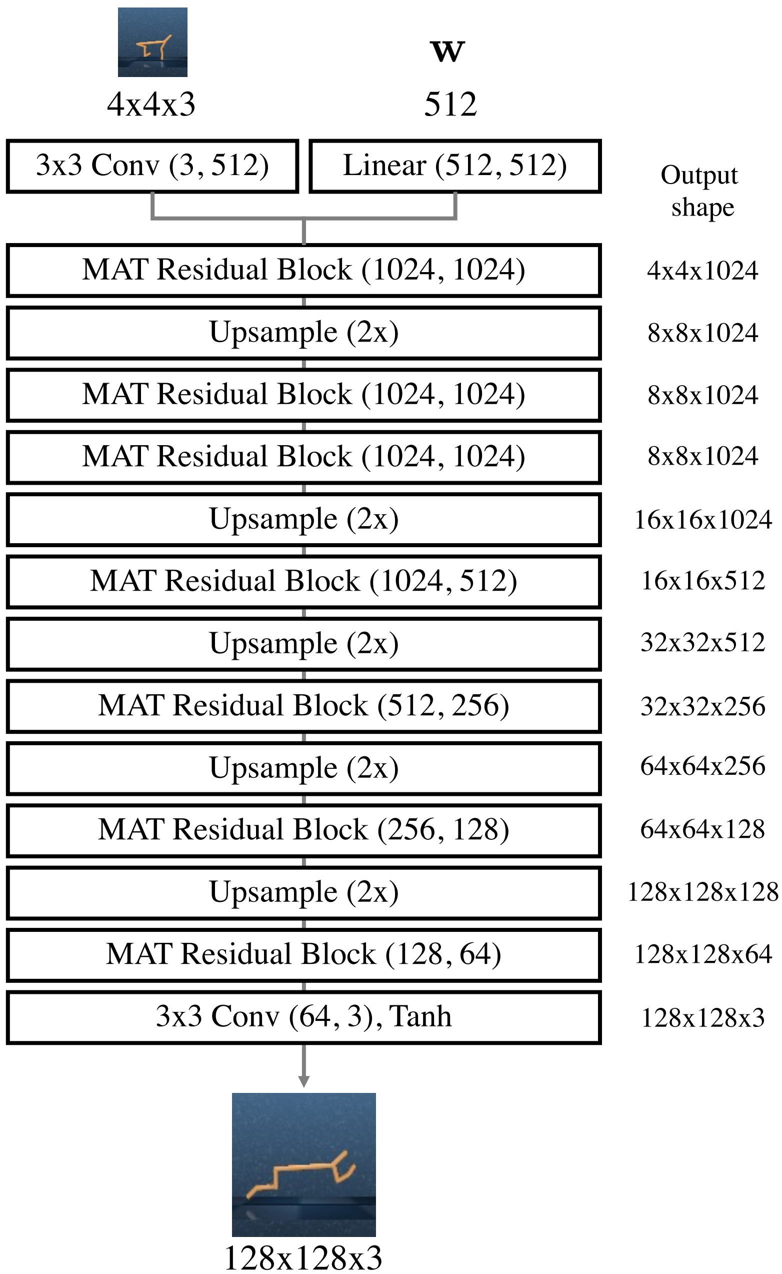


Figure 2: Specification of the generator.

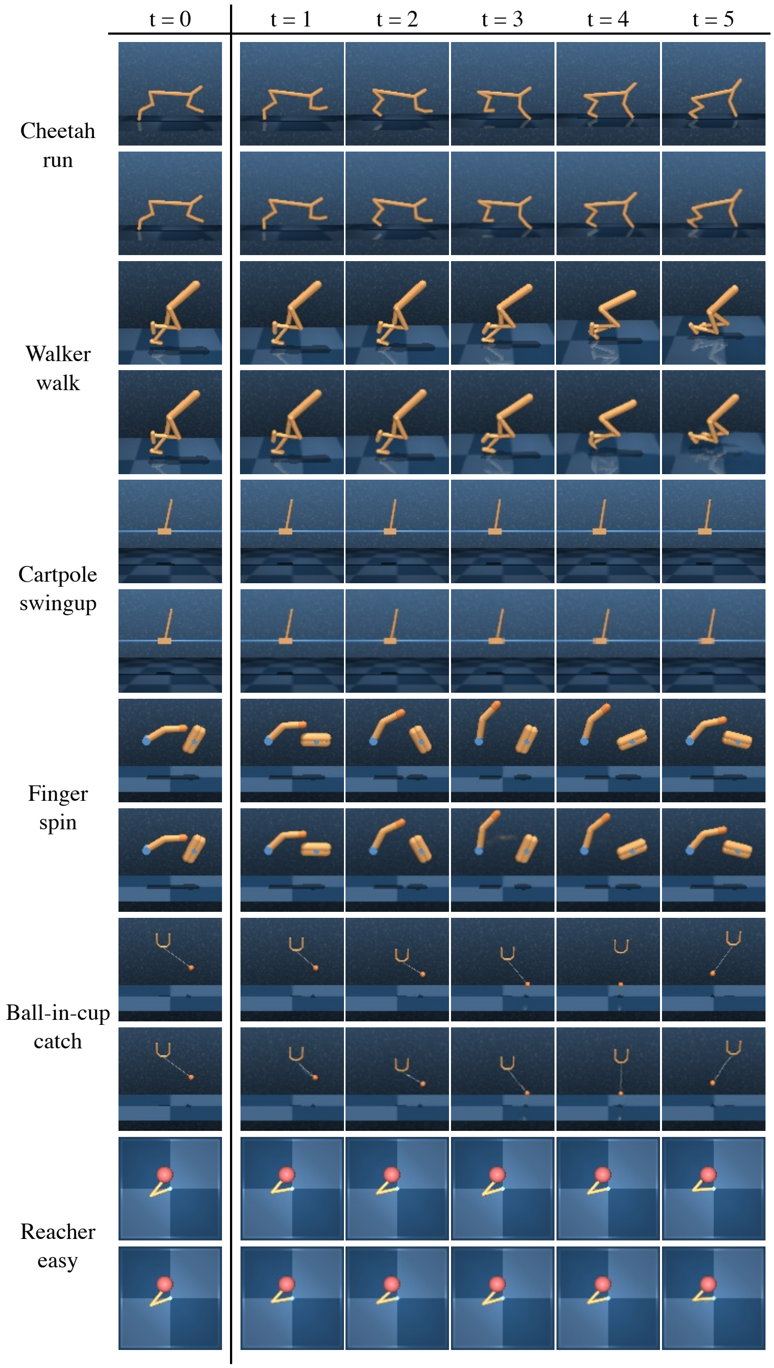


Figure 3: Additional qualitative results on the DMControl environment. The first row of each environment is ground truth images and the second row is the synthesized images from S2P.

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

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 images quantitatively with metrics (FID score, LPIPS, PSNR, SSIM) which are frequently adopted for evaluating image quality in Table 2. 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 τ_s by G.
  Save τ_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 3: 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 **??**. 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 4 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 4. 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 4: Effect of rollout horizons in cheetah-run environment.

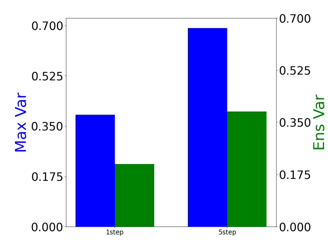


Figure 4: Average uncertainty of each different rollout horizon in cheetah-run environment.

B.2.3 Results on policy constraint-based methods

We additionally test the proposed method on the policy constraint-based methods such as BEAR [**?**] and behavior cloning (BC) on the cheetah-run environment. As shown in Table 5, 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 η)	+S2P (OFFRL η)
CHEETAH	BEAR	-1.15	-1.39	**1.14**
RANDOM	BC	-1.41	-1.3	**2.54**
CHEETAH	BEAR	**10.64**	-0.29	6.57
MIXED	BC	**51.81**	0.06	5.17
CHEETAH	BEAR	**73.49**	11.11	56.86
EXPERT	BC	77.42	30.06	**79.40**

Table 5: 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 (**??**) 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 6), 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 **??**, 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 6). 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	IQL	41.68	**88.53**	70.17	2.09	1.85
RUN	CQL	92.63	**93.16**	79.78	58.93	82.55
MIXED	SLAC-OFF	16.63	**26.39**	3.10	4.68	0.14
WALKER	IQL	96.07	**95.49**	95.39	1.28	7.21
WALK	CQL	97.18	**97.84**	97.61	95.88	97.70
MIXED	SLAC-OFF	29.02	**92.60**	17.13	52.65	0.48
CHEETAH	IQL	79.89	**87.18**	68.39	73.23	3.50
RUN	CQL	94.20	**96.28**	94.01	53.69	27.58
EXPERT	SLAC-OFF	8.92	**14.41**	8.15	3.65	3.24
WALKER	IQL	94.34	**94.97**	92.46	34.95	37.92
WALK	CQL	95.43	**97.97**	97.11	96.14	82.51
EXPERT	SLAC-OFF	11.71	**70.95**	6.91	52.03	10.43

Table 6: 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 5).

As shown in Figure 5, 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

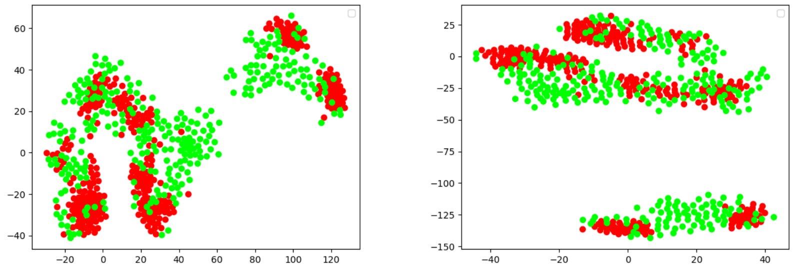


Figure 5: The t-sne visualizations examples of the cheetah-run-expert dataset (red) and the augmented dataset by the random policy (green).

interpreted that S2P connects different modes of image distribution by deploying virtual exploration in the state space.

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.

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

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 **?**]. 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 **?**] that uses a variational model with the following components:

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

Full derivation of those equations are in **?**]. 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 τ 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 π with the learning rates 0.0003, 0.0001, respectively. Each Q and π 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 **?**]. 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 **??**, and evaluate the offline RL.

For the **walker-run** task, we generate the dataset in the same manner as the mixed type in **??**. 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 7. The normalized score is computed by $100*$(return - random score)/(expert score - random score), which is proposed in **?**].

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