Environment Agnostic Representation for Visual Reinforcement learning - Supplementary material

Hyesong Choi[1], Hunsang Lee[2], Seongwon Jeong[1], Dongbo Min[1†]
[1]Ewha W. University [2]Hyundai Motor Company

1. Overview

In this supplementary material, we present more comprehensive results and analysis.

- Sample efficiency and computational efficiency of the proposed method (Section 2)
- Stability of the proposed method under strong augmentation (Section 3)
- Quantitative evaluation of the proposed method according to the asymmetric augmentation and the choice of augmentation (Section 4.1 and 4.2)
- Experimental details including hyperparameters, ESS module and network architecture (Section 5.1, 5.2 and 5.3)
- Visualized examples of test environments (Section 5.4)

2. Efficiency

2.1. Sample efficiency over state-of-the-arts

Aside from improving generalization capability, the method of EAR is closely related to learning good representations. Efficient policy learning is possible by extracting and learning only essential information of an agent, excluding unnecessary environment-related information. This argument is supported by the learning curves in Figure 1, which compares the training performance and sample efficiency of EAR against the strong baselines, RAD [7] and DrQ [6] over 5 random seeds. RAD and DrQ are the state-of-the-art methods that have achieved significant improvements on sample efficiency through various augmentations. In all tasks, the proposed method converged faster than the two methods with a smaller amount of data, at the same time, achieving higher or competitive episode returns. Our results indicate that learning only vital features through self-supervised learning enables the efficient policy learning, which in turn improves sample efficiency.

Table 1. Comparison on the number of model parameters and an average inference time per episode at evaluation time.

Method	EAR	VAI	SAC	PAD	CURL
Model parameter (M)	2.55	4.88	2.54	2.54	2.54
Time per episode (s)	0.72	0.72	0.71	4.13	0.71

2.2. Computational efficiency

Table 1 shows the specific values for Figure 2, which illustrates the comparison on the number of model parameters and an average speed per episode at evaluation time with VAI [14], SAC [2], PAD [3] and CURL [8]. EAR is computationally efficient in terms of model complexity and inference time while achieving superior performance (Table 1, 2 and Figure 7 of the original manuscript), compared to VAI [14] and PAD [3]. EAR even has the similar runtime and model complexity to the existing RL algorithms [2, 8] that do not consider the generalization capability.

3. Stability under strong augmentation

Determining the strength of the data augmentation is considered to be an important task among randomization methods, since strong augmentation makes optimization difficult, increasing instability and overall sample complexity. However, the proposed method effectively handles this problem, in that it suppresses all distractions and feeds only environment-agnostic features.

To compare the stability, we evaluated our method against VAI [14], which achieves the highest generalization performance among comparison methods, under the same degree of strong augmentations. To be more specific, VAI uses random box addition or brightness change in its experiment, and defined this degree as 'weak augmentation'. By increasing the degree of these augmentations up to 8 times (*e.g.*, number of random boxes, maximum size of boxes), we defined strong augmentations to conduct experiments on stability. Note that the same type of augmentation used in VAI [14] was adopted, and only the intensity was increased. Figure 3 and Table 2 show the training and test performance of EAR

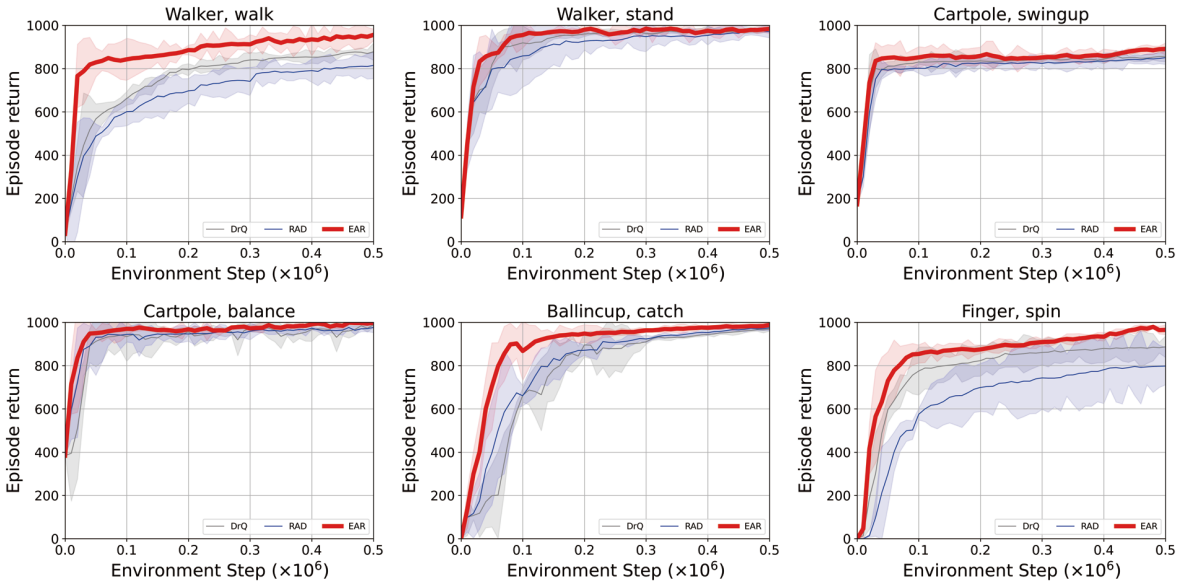


Figure 1. Learning curves of EAR compared to state-of-the-art methods including RAD [7] and DrQ [6] on DM Control Suite [12]. We report mean (line) and standard deviation (shaded area) over 5 random seeds. Aside from improving generalization capability, EAR competes favorably with the two methods in terms of sample efficiency.

Table 2. Generalization performance measured by episode return on randomized color test under the same degree of strong augmentations. We report mean and standard deviation over 5 random seeds. By excluding the environmental elements from the feature stage, EAR is always capable of achieving stable test performance regardless of the intensity of augmentation. Thus, EAR improves generalization stability under strong augmentations.

Methods	Walker, walk	Walker, stand	Cartpole, swingup	Cartpole, balance	Ball in cup, catch	Finger, spin
EAR	922±37	972±11	884±39	997±2	979±15	946±18
VAI [14]	651±63	959±28	798±22	947±31	810±78	915±17

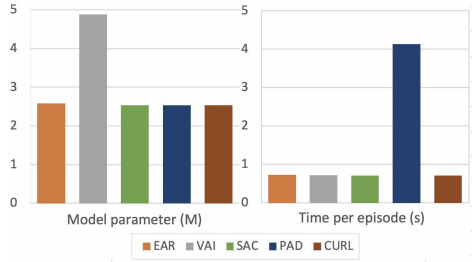


Figure 2. Comparison on the number of model parameters and an average inference time per episode.

and VAI [14] under strong augmentations, respectively. We reported mean and standard deviation over 5 random seeds for both training and test, and test performance was measured on randomized color test of DM Control Suite [12] Generalization Benchmark [4]. We empirically observed that strong augmentations could affect the training stability of VAI, since the RL policy network of VAI still needs to handle foreground variations remaining in the foreground mask. Namely, in the VAI that learns the policy network using a foreground object extracted from an estimated mask in an intensity domain (Figure 3 in [14]), the foreground variations still hampers the policy learning, especially in the presence of strong augmentations. Contrarily, EAR, which excludes the environmental elements from the feature stage, is always capable of conducting stable learning regardless of the intensity of augmentation. Therefore, as validated in the experiments, EAR improves stability significantly in both training (Figure 3) and test (Table 2) under strong augmentations.

4. Ablation studies on data augmentation

4.1. Asymmetry of augmentation

In the field of self-supervised learning, it is widely known that the degree of augmentation for source and target should be asymmetric [5, 15, 16, 13]. Inspired by this insight that keeping a relatively lower variance in the target encoder can help representation learning, we employ an asymmetric augmentation to the critic target encoder of soft actor critic (SAC) [2] algorithm as below, applying strong augmentation to the source input and weak augmentation to the target input, respectively. The source input is used for the actor encoder,

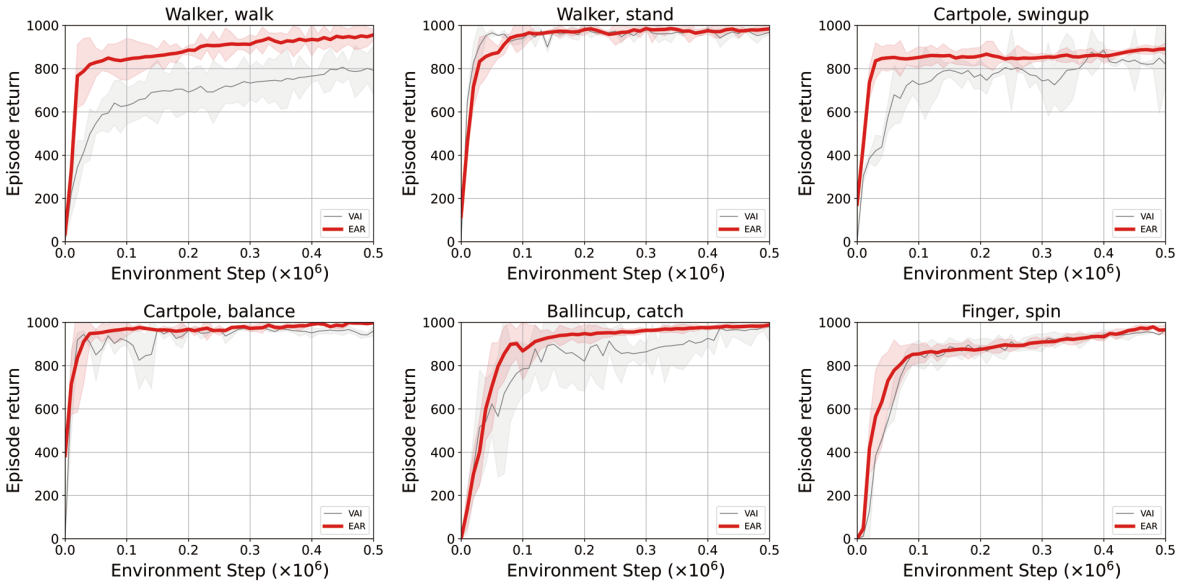


Figure 3. Learning curves of EAR compared to VAI under the same degree of strong augmentations on DM Control Suite [12]. We report mean and standard deviation over 5 random seeds. EAR improves training stability under strong augmentations.

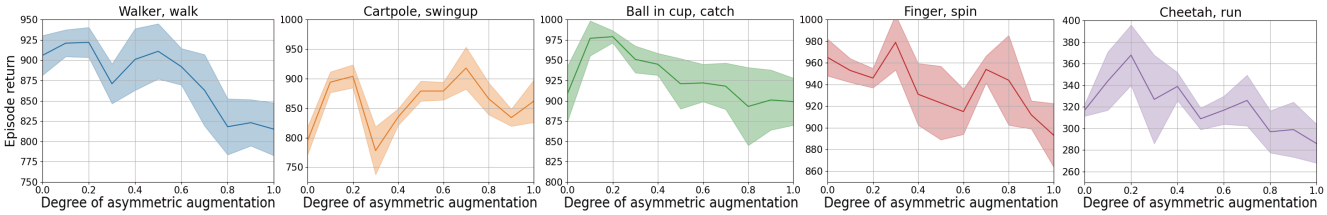
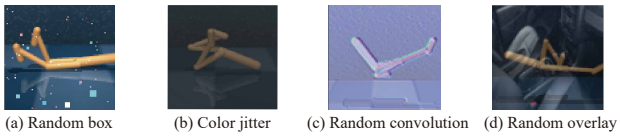


Figure 4. Episode returns according to the degree of asymmetric augmentation on the DM Control Suite [12] randomized color tests over 5 random seeds. The performance of each task is presented separately for clarity. We found that weaker target augmentation does not always result in better performance, but finding the appropriate threshold improves the performance of the proposed self-supervised learning approach.



(a) Random box (b) Color jitter (c) Random convolution (d) Random overlay

Figure 5. Visualizations of data augmentations used in this ablation. For all experiments in manuscript, we used (a) Random box addition and (b) Color jitter. Note that '(a)+(b)' is the experiment presented in the original manuscript and VAI [14]. To ablate the type of augmentations, we additionally applied two recently proposed augmentation: (c) Random convolution and (d) Random overlay in this experiment.

critic encoder, and our self-supervised encoder (Figure 2 of original manuscript), while the target input is for the critic target encoder.

Implementation. We provide Python-like implementation for the asymmetry of augmentation. Specifically, in the case of EAR, the strength of augmentation indicates the size and number of random boxes, or the degree of change in brightness and saturation.

```
ratio = 0.2 if target else 1

random_boxes = gen_random_boxes(
                    boxes=num_boxes * ratio,
                    mu=mu, sigma=sigma * ratio,
                    size_min=size_min,
                    size_max=size_max * ratio)

# [brightness, contrast, saturation, hue]
adjustment_params = random_sample()
adjustment_params =
    [ap * ratio for ap in adjustment_params]
```

Figure 4 measured the episode returns according to the asymmetric augmentation on the DM Control [12] randomized color tests over 5 random seeds. The X-axis represents the degree of asymmetric augmentation related to the target, in other words, the degree to which target augmentation is

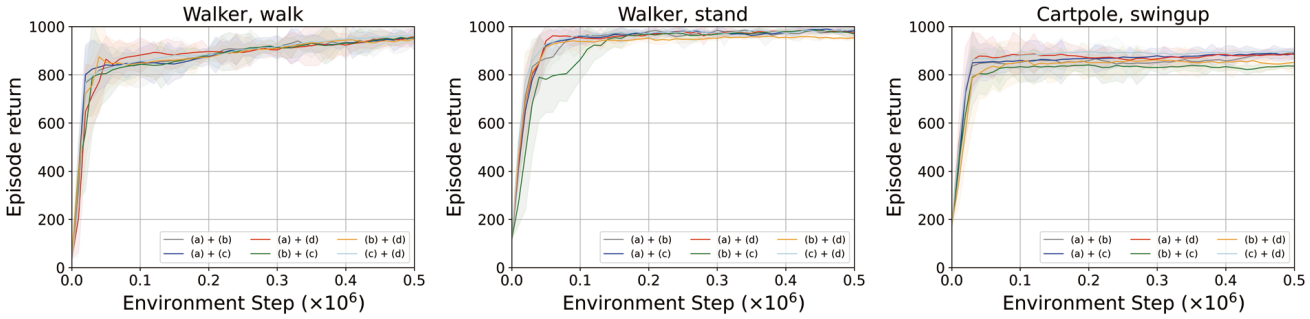


Figure 6. Learning curves of EAR according to the choice of augmentations.

weaker than source augmentation. We found that simply applying higher or lower intensity variations does not always result in a performance gain. It is necessary to appropriately control the strength of the augmentation for the source and target images, respectively. Empirically, assuming that the augmentation strength for the source input is 1, it was most desirable to apply the strength of 0.2 to the target image in EAR.

4.2. Choice of augmentation

For all experiments, similar to VAI [14] we applied random box addition and color jitter (*e.g.*, changes in brightness and saturation) as augmentations. Merely, we used asymmetric augmentation by increasing the degree of these augmentations. However, the operation of EAR is regardless of the choice of augmentation as its ultimate purpose is to learn features in which environmental variations are removed, as mentioned in Section 3.1 of the original manuscript. To ensure this, we further ablated by using the two recently proposed augmentation methods (random convolution [9] and random overlay [4]) as A_1 or A_2 in the proposed method. Figure 5 shows the visualizations of all data augmentations used in this ablation.

Figure 6 provides training performance and sample efficiency of EAR implemented using the combination of 4 augmentations presented in Figure 5. For instance, '(a)+(c)' corresponds to A_1 = Random box, A_2 = Random convolution. Most of the curves showed a similar distribution, which reinforces the argument that the implementation of EAR is not affected by the choice of augmentation. Meanwhile, '(b)+(c)' achieves low sample efficiency and relatively slow convergence tendency in most of the tasks, which is considered to be due to the similar attributes of two augmentations (b) and (c). From this fact, we found that effective learning of EAR is possible when A_1 and A_2 have sufficiently different disposition.

Table 3 illustrates the quantitative evaluation of episode returns according to the combination of augmentations on randomized color test of DM Control Suite [12] Generalization Benchmark [4]. We report mean and standard deviation over 5 random seeds. Generally, regardless of the choice of augmentation applied, EAR achieves high generalization performance. However, similar to the evaluation of sample efficiency, the types of augmentation should be different enough to lead to high generalization capability. Moreover, the two newly proposed augmentations were effective, achieving the highest test performance.

5. Experimental details

5.1. Hyperparameter

In table 4 we provide full hyperparameters for DM Control Suite [12] and DrawerWorld [14] experiments. In both benchmarks, we followed the setting of PAD [3] and VAI [14], which are empirically shown to be effective. For detailed experimental environment, EAR was implemented in Pytorch [10], and the speed in Table 1 was measured on a single Titan RTX GPU.

5.2. Target feature for ESS

We followed the self-supervised literature [1] to compute the target feature $\tilde{f}_{t+1}^{\mathrm{EAF}}$ of ESS module using the network updated via the exponential moving average (EMA) as mentioned in Section 3.2 of the original manuscript. When the base RL algorithm is soft actor critic (SAC) [2], the feature of the critic target encoder can be reused as the target representation for this constraint, instead of performing additional computation for deriving the target feature.

Implementation. In order to express above explanation in an easy-to-understand manner, we provide Python-like implementation.

```
def ss_constraint(f_x1s, f_x2s):
    f_x1 = F.normalize(f_x1s.float(), p=2.,
                       dim=-1, eps=1e-3)
    f_x2 = F.normalize(f_x2s.float(), p=2.,
                       dim=-1, eps=1e-3)
    loss = F.mse_loss(f_x1, f_x2,
                       reduction="none").mean()

    return loss

```

Table 3. Test performance according to the combination of augmentations on randomized color test.

Augmentations (a)	(b)	(c)	(d)	Walker, walk	Walker, stand	Cartpole, swingup
✓	✓			922±37	972±11	884±39
✓		✓		917±12	**979±7**	881±17
✓			✓	894±51	964±30	841±45
	✓	✓		835±68	973±19	812±40
	✓		✓	857±65	973±28	856±34
		✓	✓	**929±26**	975±9	**892±27**

Table 4. Hyperparameters used for DM Control Suite [12] and DrawerWorld [14] experiments.

Parameter	**Value**
Observation Size	(84, 84)
Observation Rendering	(100, 100)
Stacked Frames	3
Weight parameter (α, β, γ)	(0.01, 1, 10)
Action Repeat	2 (DMC-finger, DrawerWorld) 8 (DMC-cartpole), 4 (otherwise)
Discount Factor	0.99
Optimizer	Adam
$(\beta_1, \beta_2) \rightarrow (\pi^e, \pi^a, \pi^d)$	(0.9, 0.999)
$(\beta_1, \beta_2) \rightarrow (\alpha)$	(0.5, 0.999)
Learning Rate (π^e, π^a, π^d)	$3e-4$ (DMC-cheetah), $1e-3$ (otherwise)
Learning Rate (α)	$1e-4$
Batch Size	128
Replay Buffer Size	500000
Number of training steps	500000
Episode length	1000
π^e, π^d Update Frequency	2

Table 5. Detailed description of the proposed network architecture

Encoder ($\mathrm{E}^{\mathrm{base}}$ + $\mathrm{E}^{\mathrm{dis}}$)			
Layer	Operations	Input	Output
$\mathrm{E}^{\mathrm{base}}$			
1	Conv(N32, K3, S2) - ReLU	$o_t \sim o_{t+M}$	$en1_t$
2	Conv(N32, K3, S1) - ReLU	$en1_t$	$en2_t$
3	Conv(N32, K3, S1) - ReLU	$en2_t$	$en3_t$
4	Conv(N32, K3, S1) - ReLU	$en3_t$	$en4_t$
5	Conv(N32, K3, S1) - ReLU	$en2_t$	$en5_t$
6	Conv(N32, K3, S1) - ReLU	$en2_t$	$en6_t$
7	Conv(N32, K3, S1) - ReLU	$en2_t$	$en7_t$
8	Conv(N32, K3, S1) - ReLU	$en7_t$	f_t
$\mathrm{E}^{\mathrm{fac}}$			
9	Conv(N16, K1, S1) - BN - ReLU - DropOut	f_t	$f1_t$
10	Conv(N16, K1, S1) - BN - ReLU - DropOut	$f1_t$	$f2_t$
11	Conv(N32, K1, S1) - BN	$f2_t$	f_t^{EAF}
Action Conditioned Prediction (ACP)			
Layer	Operations	Input	Output
1	Concatenate	$f_t^{\mathrm{EAF}}, \hat{a}_t$	conc
2	Conv(N96, K3, S1) - ReLU - BN	conc	conv1t
3	Conv(N32, K3, S1) - ReLU	conv1t	$\hat{f}_{t+1}^{\mathrm{EAF}}$

```
target_feature = critic_target.encoder.outputs["latent"]
query_feature = ACP(obs, action)

target_proj = target_projection(target_feature)
query_proj = query_projection(query_feature)
query_pred = query_prediction(query_proj)
L_ss = ss_constraint(query_pred, target_proj)
```

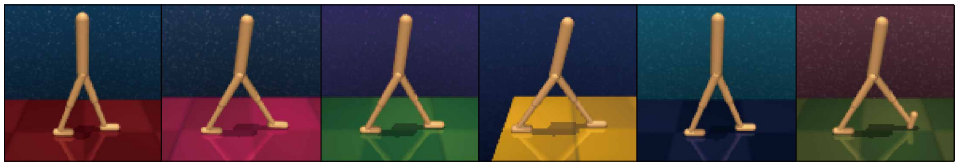
5.3. Network architecture

We provide detailed network architecture in Table 5. In the case of encoder, we follow the structure of encoders commonly used by off-the-shelf RL algorithms [8, 3, 7]. However, in order to separate an environment-agnostic feature, we split the network into a base encoder $\mathrm{E}^{\mathrm{base}}$ and a feature factorization encoder $\mathrm{E}^{\mathrm{fac}}$. This allows the model complexity to remain unchanged as in Figure 2 by avoiding the use of additional layers for the feature factorization.

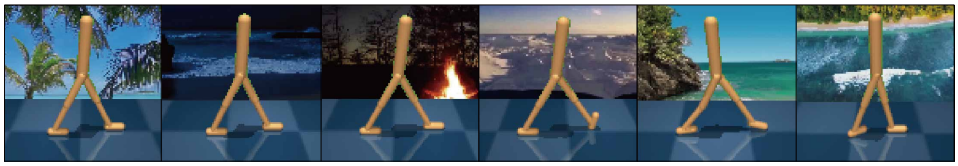
Implementation. Since the layer of $\mathrm{E}^{\mathrm{fac}}$ has batch normalization and dropout added to the layer of $\mathrm{E}^{\mathrm{base}}$, we provide Python-like implementation of $\mathrm{E}^{\mathrm{fac}}$ for explicit explanation. As validated in Figure 2, this modification causes little change in the overall model complexity.

```

E_fac = nn.Sequential(
    nn.Conv2d(32, 16, 3, stride=1, padding=1),
    nn.BatchNorm2d(16),
    nn.ReLU(inplace=False),
    nn.Dropout(p=0.5),
    nn.Conv2d(16, 16, 3, stride=1, padding=1),
    nn.BatchNorm2d(16),
    nn.ReLU(inplace=False),
```

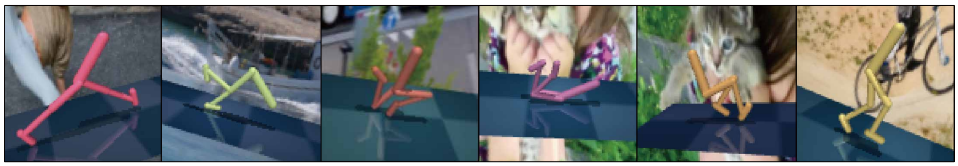


(a) DeepMind Control Generalization Benchmark [4]: Randomized colors



(b) DeepMind Control Generalization Benchmark [4]: Video backgrounds

(c) Distracting Control Suite [11]: Camera poses (intensity=0.1)

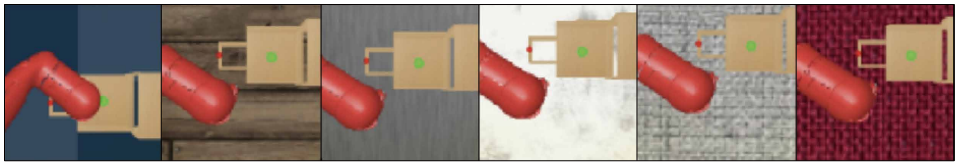


(d) Distracting Control Suite [11]: Camera poses (intensity=0.2)

(e) Distracting Control Suite [11]: Camera poses (intensity=0.3)

(f) Distracting Control Suite [11]: Camera poses (intensity=0.4)

(g) Distracting Control Suite [11]: Camera poses (intensity=0.5)



(h) DrawerWorld robotic manipulation benchmark [14]: Texture backgrounds

Figure 7. Samples from the test environments used in our experiments, including DeepMind Control Generalization Benchmark [4], Distracting Control Suite [11], and DrawerWorld [14] robotic manipulation tasks.

```
    nn.Dropout(p=0.5)
    nn.Conv2d(16, 32, 3, stride=1, padding=1),
    nn.BatchNorm2d(32),
)
```

5.4. Examples for test environments

Figure 7 shows the samples from the test environment used in our experiments, including DeepMind Control Generalization Benchmark [4], Distracting Control Suite [11], and DrawerWorld [14] robotic manipulation tasks. DeepMind Control Generalization Benchmark [4] provides two distinct benchmarks for visual generalization, (a) Randomized colors and (b) Video backgrounds. Environment (a) randomizes the color of floor, and background. Environment (b) replaces the background with videos from real-life scenarios. For environments (c)-(g), we used Distracting Control Suite [11] of DM Control Generalization Benchmark [4], where camera pose, background, and colors continually change throughout episodes. The intensity indicates the degree of variations, and we provide sample images with different intensities $I = \{0.1, 0.2, 0.3, 0.4, 0.5\}$. In particular, when the intensity is $I = \{0.5\}$, the camera pose changes significantly so that the camera point is often positioned vertically above the agent, as in the third example of (g). Under this strong intensity change, most of the methods cannot work well, as shown in Figure 7 of the original manuscript. We also used DrawerWorld robotic manipulation benchmark [14] for the environment (h), which measures generalization performance on six different types of new texture environments: Black, Blanket, Fabric, Metal, Marble and Wood including both color change and texture change of background, following the setup of [14].

References

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

[2] 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.

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

[4] Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13611–13617. IEEE, 2021.

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

[6] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

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

[8] 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.

[9] Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. *arXiv preprint arXiv:1910.05396*, 2019.

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[11] Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. The distracting control suite–a challenging benchmark for reinforcement learning from pixels. *arXiv preprint arXiv:2101.02722*, 2021.

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

[13] Xiao Wang, Haoqi Fan, Yuandong Tian, Daisuke Kihara, and Xinlei Chen. On the importance of asymmetry for siamese representation learning. *arXiv preprint arXiv:2204.00613*, 2022.

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

[15] Xiao Wang and Guo-Jun Qi. Contrastive learning with stronger augmentations. *arXiv preprint arXiv:2104.07713*, 2021.

[16] Haohang Xu, Xiaopeng Zhang, Hao Li, Lingxi Xie, Hongkai Xiong, and Qi Tian. Seed the views: Hierarchical semantic alignment for contrastive representation learning. *arXiv preprint arXiv:2012.02733*, 2020.