Style-Agnostic Reinforcement Learning – *Supplementary Material* –

Juyong Lee*, Seokjun Ahn*, and Jaesik Park

Pohang University of Science and Technology (POSTECH), South Korea
{joy.lee, sdeveloper, jaesik.park}@postech.ac.kr

1 Implementation details

We explain the implementation details for both Procgen [1] and Distracting Control Suite [14] benchmark. We reproduce all the baseline results on top of implementation of PPO [6] for Procgen and implementation of SAC [15] for Distracting Control Suite.

1.1 Hyperparameters

Procgen. The baselines compared with our SAR model are PPO [13], RAD [9], UCB DrAC [12], Meta DrAC [12], MixStyle [16], and DARL [10]. We follow the settings of Cobbe et al. [2] in Procgen; the encoder in the actor-network is based on ResNet architecture [3], and the encoded features are shared to both actor and critic networks. The encoder is composed of three layer-blocks, where one layer-block is built with five convolutional layers with two skip connections. The hyperparameters for the model and environments are well described in Table 1.

For PPO, the basic baseline model, we use generalized advantage estimation [13] but no stacked observations [11].

For RAD, we apply the random translation and color cutout, where their results are shown in Figure 1, as they are reported as the best [9].

For UCB DrAC and Meta DrAC, we follow the setting in [12].

For MixStyle, the style mixing is done once after the feature passes through two layer-blocks, where the feature flow is divided into two branches in SAR.

For DARL, we not only follow the adaptive coefficient in gradient reversal layer [10] but also control the effect of the domain adversarial loss with the coefficient of d with the value reported in Table 1.

For SAR (Ours), we set the adversarial coefficients λ, λ' to be equal, but they can be optionally different. We perform a grid search to find the best hyperparameter pairs and report them as indices (1)∼(3) for the adversarial coefficient and (4)&(5) for the value similarity coefficient, where each index refers to the task of:

- (1) `starpilot`, `jumper`, `coinrun`
- (2) `climber`, `ninja`, `bigfish`
- (3) `maze`, `dodgeball`
- (4) `starpilot`, `ninja`, `coinrun`
- (5) `climber`, `jumper`, `maze`, `bigfish`, `dodgeball`

Distracting Control Suite We compare our SAR model with SAC [4], CURL [8] and DrQ [7] in Distracting Control Suite. The encoder network has 3 CNN layers with layer size 32 and kernel size 3. We set the stride of the first layer of the encoder to 2 and the stride of the remaining layers to 1. The encoder network is shared between actor and critic. In Table 2, we report the value of the hyperparameters.

For SAC and CURL, we reproduce the results based on the implementation in [15] and [8].

For DrQ, we apply the random translation as they are reported as the best [7]. We set the augmentation coefficients K=1 and M=1.

For SAR (Ours), we also set the adversarial coefficients λ, λ' to be equal. We search for the best hyperparameter pair using grid search and report them as indices (1)&(2) for the adversarial coefficient and (3)&(4) for the value similarity coefficient, where each index refers to the task of:

- (1) `walker:walk`, `cartpole:balance`
- (2) `reacher:easy`, `cheetah:run`
- (3) `walker:walk`, `reacher:easy`, `cheetah:run`
- (4) `cartpole:balance`

1.2 Visualization of images

Augmentation result in Procgen In our generalization performance experiment in Procgen, especially for RAD, we use two data augmentation methods: random translation and random color cutout. We show the results in Figure 1.

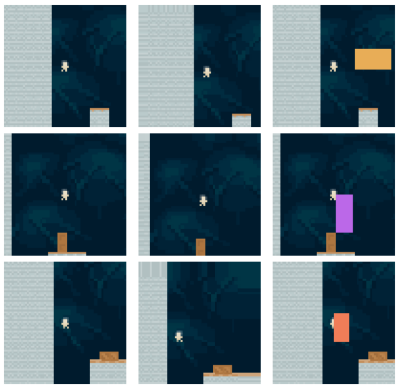

Fig. 1. (a) The original images, and augmentation results with (b) random translation and (c) random color cutout, of `coinrun` in Procgen. These methods are only applied to RAD in the generalization performance experiment. An additional experiment comparing SAR agents with and without these methods is conducted separately.

Distraction result in Distracting Control Suite. In Figure 2, we visualize more diverse examples in Distracting Control Suite. The noises, referring to the distractions we apply, are shifts of color, distortions in the camera angle, and changing the background image into videos. Especially, the camera angle noise intensity, i.e., β_{cam}, in the main text refers to camera angle distraction intensity.

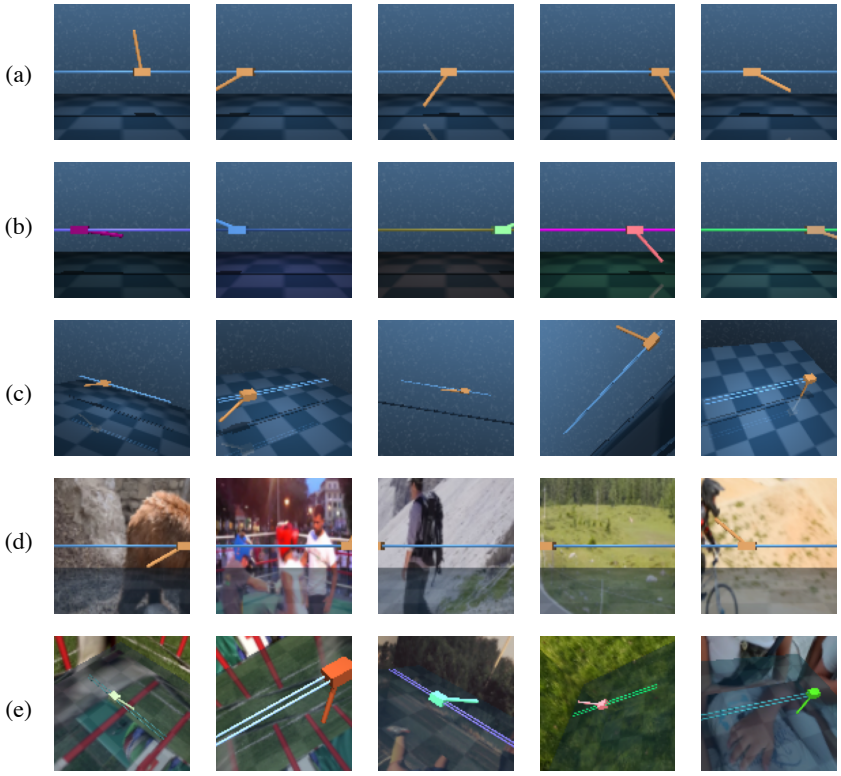

Fig. 2. Distracting environment examples of `cartpole:balance` task in Distracting Control Suite. Row (a) shows the task with zero noise. Row (b) shows the task with color shift $\beta_{\text{rgb}} = 0.5$. Row (c) shows the with camera angle distraction $\beta_{\text{cam}} = 0.5$. Row (d) shows the task with changed backgrounds. Row (e) shows the task with color shift $\beta_{\text{rgb}} = 0.5$, camera angle distraction $\beta_{\text{cam}} = 0.5$ and changed backgrounds.

Hyperparameter	**Value**
Input image resolution	(64,64)
Discount factor γ	0.999
Generalized advantage estimates	0.95
# timesteps per rollout	256
# epochs per rollout	3
# minibatches per epoch	8
Entropy bonus	0.01
PPO gradient clip range ϵ	0.2
Reward normlization	yes
Learning rate	5e-4
# workers	1
# environments per worker	64
# total timesteps	100M
Optimizer	Adam
Recurrent neural network	no
Frame stack k	no
Regularization coefficient α_r	0.1 (UCB DrAC, Meta DrAC)
Exploration coefficient c	0.1 (UCB DrAC)
Sliding window size K	10 (UCB DrAC)
Domain loss coefficient d	0.9 (DARL)
Meta gradient clip range	100 (Meta DrAC)
Meta # train steps	1 (Meta DrAC)
Meta # test steps	1 (Meta DrAC)
Adversarial coefficient λ	0.1 (1); 0.01 (2); 0.001 (3)
Value similarity coefficient κ	1.0 (4); 0.1 (5)

Table 1. Hyperparameters for SAR (Ours) and baselines in the Procgen experiment. The indices inside the parentheses for 'adversarial coefficient' and 'value similarity coefficient' indicate that the different values are used in different tasks.

Hyperparameter	**Value**
Input image resolution	$(84, 84)$
Discount factor γ	0.99
Frame stack k	3
Random shift	Up to 4 pixels
Action repeat	
cartpole	8
finger	2
else	4
Episode length	1000
Replay buffer size	100000
Optimizer	Adam
Learning rate	
actor, critic, attacker	10^{-3}
alpha	10^{-4}
Encoder feature dimension	50
Target smoothing coefficient τ	
actor	0.05
critic	0.01
alpha	0.5
Target update interval	2
Batch size	128
Latent dimension	128
Initial temperature	0.1
Initial steps	1000
Network update frequency	
attacker, critic	1
actor	2
Adversarial coefficient λ	0.01 (1); 0.1 (2)
Value similarity coefficient κ	0.1 (3); 1.0 (4)

Table 2. Hyperparameters for SAR (Ours) and baselines in the Distracting Control Suite experiment.

2 Learning curves

We plot the learning curve of SAR agents and baselines in both Procgen and Distracting Control Suite. For clear visualization, each graph is illustrated with smoothing, following the settings of Cobbe et al. [2].

Procgen. Figure 3 shows the learning curves of models with the best performances from each algorithm in the Procgen environment. We apply an exponential moving average smoothing with the smoothing coefficient value of 0.95.

Distracting Control Suite. Figure 4 and Figure 5 show the learning curves of agents in Distracting Control Suite. In Distracting Control Suite, the training environments do not present various styles, unlike Procgen. However, although the SAR agents are not trained with a wide enough range of training environments showing not the best performance in the training phase, they adapt to distracting environments. They show the best performances in three out of four tasks. Here, we apply exponential moving average smoothing only for the training curve with the smoothing coefficient value of 0.99.

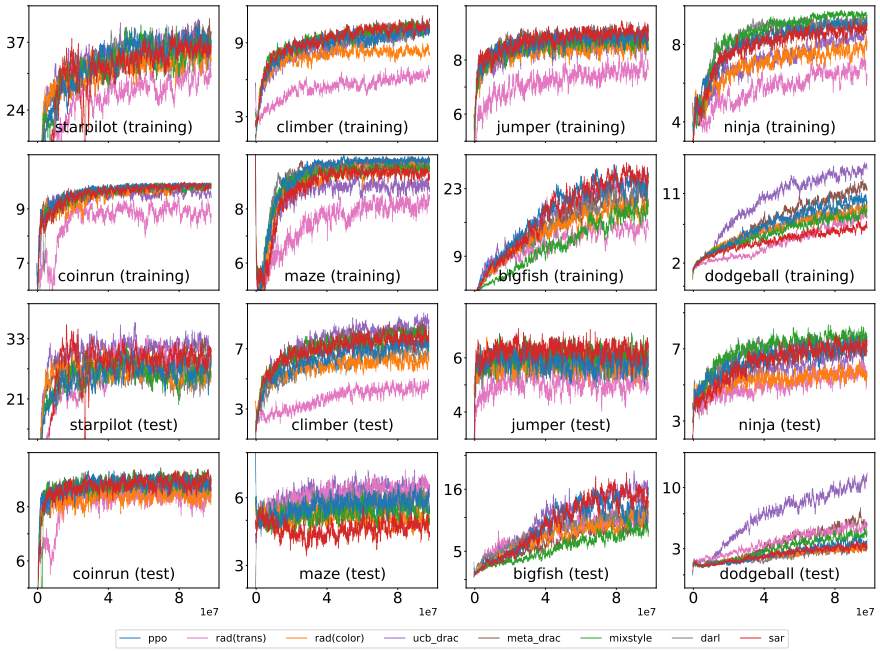

Fig. 3. The learning curves of training and test in Procgen.

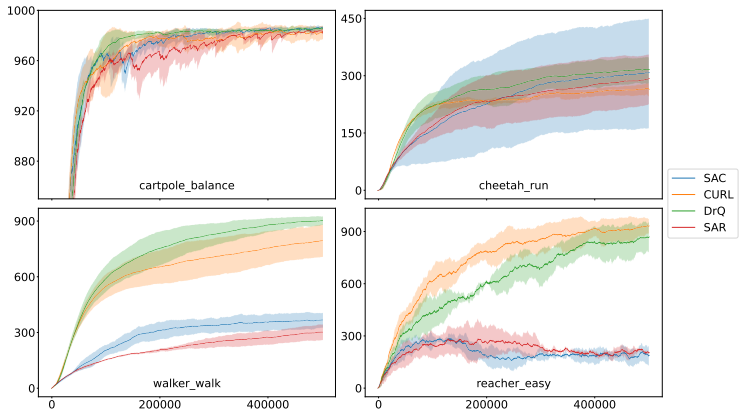

Fig. 4. The learning curves of training in Distracting Control Suite.

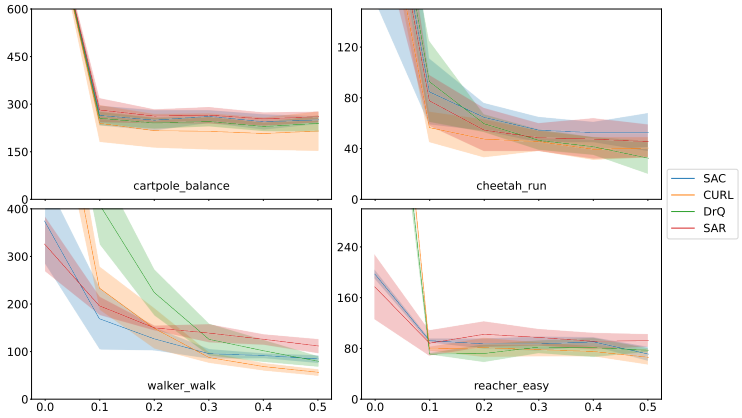

Fig. 5. Evaluation performances with respect to various distraction scales in Distracting Control Suite. β_{rgb} and β_{cam} are the same with x-axis distraction values. We use 4, 8 and 60 background videos in 0.1, 0.2 and >0.3 distraction values respectively. The SAR agents show better generalization performances with stronger noises. The observation images in distracting environments are depicted in Figure 2.

3 Curriculum Learning

We conduct an experiment on curriculum learning in both Procgen and Distracting Control Suite benchmarks. As addressed by Ko & Ok[5], the timing for the adoption of data augmentation may affect the test performance.

In Distracting Control Suite environment, while the SAR agents show no better performance with zero noise setting than its baseline SAC [4], they get improved on several tasks by adopting curriculum learning. The SAR agents also show better performances in several tasks in Procgen with a warm-up stage.

Procgen. The SAR agents are trained for 50M timesteps on Procgen with three different start times for applying the adversarial loss: from the beginning, after 10M timesteps of warm-up, and after 25M timesteps of warm-up. The results are averaged over three runs with different seeds.

Table 3. The generalization results on Procgen benchmark with different curriculum learning. The best result in the final is bold, and the second result in the final is underlined.

	from start	after 10M		after 25M	
	at final	at 10M	at final	at 25M	at final
startpilot	27.6±7.9	27.6±7.9	30.7±5.3	21.4±4.1	**34.0±13.9**
climber	**8.4±0.5**	5.6±2.7	7.1±0.1	6.3±2.8	6.5±2.6
jumper	6.0±1.0	6.3±0.6	6.0±1.0	4.0±2.7	**6.3±2.1**
ninja	6.0±1.0	5.0±2.7	6.7±0.6	6.3±0.6	**8.3±1.2**
coinrun	7.7±0.6	7.7±0.6	7.0±1.0	8.3±1.2	**8.3±0.6**
maze	6.3±2.1	6.3±0.6	5.0±2.0	7.0±1.0	**7.3±3.1**
bigfish	6.6±1.0	1.6±0.6	**12.3±5.4**	3.3±3.5	8.9±0.1
dodgeball	2.1±1.8	0.5±0.3	2.1±2.5	1.4±0.7	**3.4±2.3**

Distracting Control Suite. The generalization results in Distracting Control Suite with curriculum learning. The SAR agents have trained 500k timesteps with applying the adversarial loss after 300k of warm-up. The results are averaged over three runs with different seeds.

Table 4. The generalization results on Distracting Control Suite with curriculum learning. The best results are in bold.

		SAR from start	SAR after 300k
walker :walk	zero noise	325±57	**420±78**
	moderate	**139±19**	113±23
	hard	**112±15**	88±20
cartpole :balance	zero noise	990±5	**996±2**
	moderate	**266±26**	249±10
	hard	**261±17**	241±16
reacher :easy	zero noise	177±51	**211±33**
	moderate	**98±13**	85±19
	hard	**93±10**	77±14
cheetah :run	zero noise	**304±80**	277±47
	moderate	49±11	**57±16**
	hard	46±13	**53±13**

4 Learned Feature Analysis

Image reconstruction from embedded features. Figure 6 displays three consecutive original frames and their corresponding reconstructed images with embeddings from a trained SAR agent on two different levels of `jumper` in Procgen. This well describes that the SAR agent is extracting style-agnostic representation features while preserving important elements from the images.

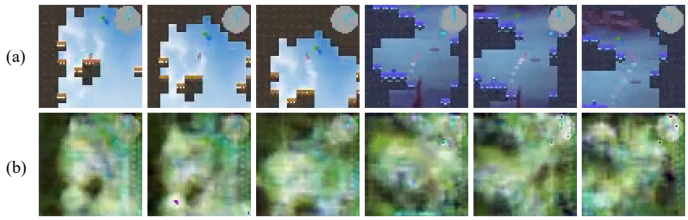

Fig. 6. (a) Images from two episodes of a trained SAR agent and (b) reconstructed images with the representation features from the agent with `jumper` in Procgen.

t-SNE Analysis. Figure 7 presents the t-SNE results from embeddings extracted from trained the SAR and PPO agents. While the representation features extracted from the PPO agent are well grouped concerning the styles of environments, those from the SAR agent are scattered without a certain pattern. The average distance between all the PPO sample pairs, where a sample is composed of two features five frames apart, is 1.21, while that of SAR sample pairs is 3.41. Also, with sample pairs composed of features 15 frames apart, that of PPO is 3.43, while that of SAR is 9.30.

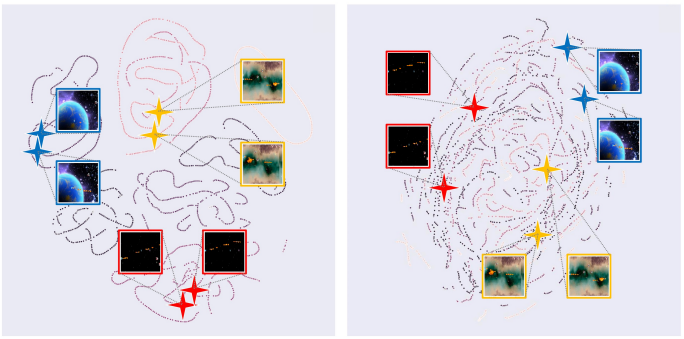

Fig. 7. t-SNE results of (left) PPO and (right) SAR with the `starpilot` in Procgen. Example image pairs within 15 frames apart from 3 different levels are marked.

References

1. Cobbe, K., Hesse, C., Hilton, J., Schulman, J.: Leveraging procedural generation to benchmark reinforcement learning. In: Proceedings of the International Conference on Machine Learning (ICML) (2020)
2. Cobbe, K., Hesse, C., Hilton, J., Schulman, J.: Leveraging procedural generation to benchmark reinforcement learning (2020)
3. Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al.: Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures (2018)
4. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Proceedings of the International Conference on Machine Learning (ICML) (2018)
5. Ko, B., Ok, J.: Time matters in using data augmentation for vision-based deep reinforcement learning. arXiv preprint arXiv:2102.08581 (2021)
6. Kostrikov, I.: Pytorch implementations of reinforcement learning algorithms. `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail` (2018)
7. Kostrikov, I., Yarats, D., Fergus, R.: Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. Proceedings of the International Conference on Learning Representations (ICLR) (2021)
8. Laskin, M., Srinivas, A., Abbeel, P.: CURL: Contrastive unsupervised representations for reinforcement learning. In: Proceedings of the International Conference on Machine Learning (ICML) (2020)
9. Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., Srinivas, A.: Reinforcement learning with augmented data. Advances in Neural Information Processing Systems (NIPS) (2020)
10. Li, B., François-Lavet, V., Doan, T., Pineau, J.: Domain adversarial reinforcement learning. arXiv preprint arXiv:2102.07097 (2021)
11. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
12. Raileanu, R., Goldstein, M., Yarats, D., Kostrikov, I., Fergus, R.: Automatic data augmentation for generalization in deep reinforcement learning. Advances in Neural Information Processing Systems (NIPS) (2021)
13. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
14. Stone, A., Ramirez, O., Konolige, K., Jonschkowski, R.: The distracting control suite – a challenging benchmark for reinforcement learning from pixels. arXiv preprint arXiv:2101.02722 (2021)
15. Yarats, D., Zhang, A., Kostrikov, I., Amos, B., Pineau, J., Fergus, R.: Improving sample efficiency in model-free reinforcement learning from images. arXiv preprint arXiv:1910.01741 (2019)
16. Zhou, K., Yang, Y., Qiao, Y., Xiang, T.: Domain generalization with mixstyle. In: Proceedings of the International Conference on Learning Representations (ICLR) (2021)