# Style-Agnostic Reinforcement Learning

Juyong Lee*, Seokjun Ahn*, and Jaesik Park

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

**Abstract.** We present a novel method of learning style-agnostic representation using both style transfer and adversarial learning in the reinforcement learning framework. The style, here, refers to task-irrelevant details such as the color of the background in the images, where generalizing the learned policy across environments with different styles is still a challenge. Focusing on learning style-agnostic representations, our method trains the actor with diverse image styles generated from an inherent adversarial style perturbation generator, which plays a min-max game between the actor and the generator, without demanding expert knowledge for data augmentation or additional class labels for adversarial training. We verify that our method achieves competitive or better performances than the state-of-the-art approaches on Procgen and Distracting Control Suite benchmarks, and further investigate the features extracted from our model, showing that the model better captures the invariants and is less distracted by the shifted style. The code is available at https://github.com/POSTECH-CVLab/style-agnostic-RL.

**Keywords:** Reinforcement Learning, Domain Generalization, Neural Style Transfer, Adversarial Learning

## 1 Introduction

Learning visual representation in reinforcement learning (RL) framework incorporated with deep convolutional neural networks enabled achieving remarkable performances in various control tasks, including video games [29,41], robot manipulation [23,38], and autonomous driving [46]. Unfortunately, however, generalization of the learned policies to unseen environments often results in failures, even with slight changes in the backgrounds [44,4,15].

Several methods have been proposed to overcome this limitation of RL agents, such as having an encoder with generative models [12,6,8,21] or training with auxiliary tasks [19,27,24]. Methods using generative models are designed to train the agents to understand the world environment, and auxiliary tasks enable the agent to extract better features that will lead to better performances. Due to its simplicity, the latter technique is gaining interest. For example, recent works have shown that representation learning with self-supervision objectives [19,9],

* These authors contributed equally to this work.

data randomization with feature matching [22], and data augmentation with additional regularization [17,11,10] result in high success.

The central concept of these approaches is to diversify training data so that the RL agents can learn invariants to the *different styles of environments*. Here, the style of the environment indicates too detailed or irrelevant elements in the observation. In an autonomous driving situation, for instance, detecting the road or pedestrians is key to success, while the texture of the road, the colors of the other cars, or the weather condition can be regarded as different styles, which distract the agent from abstract and understand the situation. Data augmentation, thus, might lead to better generalization capacity by mimicking natural style changes of observations. However, the results are inefficient or unstable without a careful choice of augmentation type and timing [20,16]. To tackle this issue, sounder training methods of adding more regularization terms can be applied [17,32,11,10], but this makes the training objectives much more complex.

In this work, we focus on learning style-agnostic representations and propose **SAR**: **S**tyle-**A**gnostic **R**L, which adopts the concept of both style transfer and adversarial learning. Style transfer has been applied in many computer vision tasks, including domain generalization in RL [14,42,45]. Here, we further examine how style transfer is used to train the agents via generating images of new styles. The generator module in our model generates *never-seen* styles and helps the actor generalize its learned policy to the unseen styles with various background images, including realistic images, without any heuristics or explicit environment class labels. Notably, the generator is trained with adversarial loss to perform adaptive style perturbation to the encoded feature representation. To our best knowledge, this attempt and success have not been presented anywhere before. An overview of our model is described in Figure 1.

In summary, the contributions of this paper are as follows:

- First, we introduce SAR, a novel method of learning style-agnostic representation for domain generalization in RL.
- Second, we conduct extensive empirical evaluations showing that the model better captures invariants between different styles of environment.
- Finally, we show that the SAR agents achieve competitive or better results on the Procgen [3] and Distracting Control Suite [36] benchmarks than the previous state-of-the-art algorithms.

## 2 Related work

### 2.1 Domain Generalization in RL

The main target of the domain generalization in RL can be summarized as training an agent to learn a robust policy that can be generalized to unseen environments. This allows RL algorithms to be applied in more realistic situations because agents are often tested in different environments from the training stage. One example is deploying a policy learned from the simulation to the real world in the robot manipulation task.

Data randomization is a promising technique for such domain generalization in many cases [38,2]. However, it is difficult to build an accurate and practical simulator that enables using data randomization. Visual augmentation, on the other hand, is much easier to apply as it is based on simple image transformations. Laskin & Lee et al. [20], for example, demonstrated that simply using data augmentation, such as random cropping or gray scaling, is indeed helpful in improving the generalization capacity of RL agents. Also, Yarats & Kostrikov et al. [17] suggested using regularization terms for stabilizing the model training when using data augmentation.

However, although data augmentation is potentially effective, it has several limitations. For example, a naïve choice of the augmentation type may degrade the generalization performance [20]. Applying cropping to an essential part of the image may confuse the agent, or training the model to produce the same action from a rotated image may be unreasonable. Here, we present a method for domain generalization by diversifying the training examples without requiring a complex strategy for data augmentation. The generator in the SAR model generates new feature examples having different styles and helps the agents with learning style-agnostic representations.

### 2.2 Adversarial Feature Learning

Adversarial feature learning has become popular for domain generalization in computer vision tasks [35,18,25,43,30]. Li et al. [25] showed that adversarial objectives help a model learn universal feature representations across different domains. Furthermore, Nam & Lee et al. [30] proposed a method of reducing the style gap for domain generalization in the image classification task. Inspired by this work, we investigate the adversarial feature learning for RL agents, but with a simpler training procedure, i.e., without dividing training phases or considering the environment style's *class*es.

We note that adopting adversarial training for RL is not new [31,24]. To our best knowledge, however, exploiting adversarial learning to the latent features in RL framework and the min-max game scheme is not presented before. Especially, our method can be interpreted as domain randomization beyond pixel space. Mixing styles with linear interpolation for representation learning in RL setting has been proposed in the earlier work [45]. However, unlike in the previous study, the style perturbation generator in SAR produces new synthetic styles that will not be seen with a simple interpolation. The adversarial examples help the actor extract style-agnostic embeddings without any label of styles and, finally, learn a robust policy for unseen environments.

## 3 Backgrounds

### 3.1 Deep Reinforcement Learning

RL agents interact and get trained with the world environment within a Markov decision process, which is defined as a tuple of (state space $\mathcal{S}$, action space

$\mathcal{A}$, transition probability $P$, reward space $\mathcal{R}$, and discount factor $\gamma$); at every timestep $t$, the agent observes a state $s_t \in \mathcal{S}$ and takes an action $a_t \in \mathcal{A}$ from its policy $\pi(a_t|s_t)$ [1]. Then, the agent is rewarded with $r_t \in \mathcal{R}$, and moves to the next state $s_{t+1}$ sampled from the transition probability $P(s_{t+1}|s_t, a_t)$.

The policy of the agent is optimized to maximize the discounted sum of rewards $G_t = \sum_{k=t}^{\infty} \gamma^k r_k$. With given state $s_t$, the value of the state $V(s_t)$ is estimated as $\mathbb{E}_{\tau\sim\pi}[G_t|s_t]$ and the value of the state-action $Q(s_t, a_t)$ is computed as $\mathbb{E}_{\tau\sim\pi}[G_t|s_t, a_t]$, with trajectory $\tau$ sampled from the policy $\pi$.

With deep RL algorithms, the policy $\pi$ gets parameterized by a set of learnable parameters $\psi$, and value function $V$ or $Q$ is optimized with network parameter $\phi$. Also, especially for visual-based RL, since the images only offer partial observations, Mnih et al. [28] has proposed that defining the state $s_t$ as a stacked consecutive image frames $(o_{t-k}, o_{t-k+1}, \ldots, o_t)$, where $\mathcal{O}$ is a high-dimensional image space and $o \in \mathcal{O}$, is effective.

**Proximal policy optimization** (PPO) [33] is a state-of-the-art on-policy RL algorithm that is used for, in our setting, discrete control tasks. Here, on-policy refers to a situation in which the model is trained with trajectories collected from the current policy. With PPO, the actor is updated using policy gradients, where the gradients are computed by using (i) action-advantages $A_t$ to reduce the gradient variances and (ii) clipped-ratio loss to constraint the update region. The critic estimates the state-value $V_\phi$, and gets trained with mean-squared error loss toward a target state-value $V_t^{target}$ using generalized advantage estimation [33]. So, the objectives for the actor and critic network can be written as follows:

$$A_t = Q_\phi(s_t, a_t) - V_\phi(s_t) \tag{1}$$

$$\mathcal{L}_{\text{actor}}(\psi) = -\,\mathbb{E}_{s_t,a_t\sim\pi}\Big[\min\Big(\frac{\pi_\psi(a_t|s_t)}{\pi_{\psi_{\text{old}}}(a_t|s_t)}A_t, \text{clip}\big(\frac{\pi_\psi(a_t|s_t)}{\pi_{\psi_{\text{old}}}(a_t|s_t)}, \epsilon\big)A_t\Big)\Big] \tag{2}$$

$$\mathcal{L}_{\text{critic}}(\phi) = \mathbb{E}_{s_t\sim\pi}\Big[(V_\phi(s_t) - V_t^{target})^2\Big], \tag{3}$$

where $\epsilon$ is a coefficient for clipping function $\text{clip}(\cdot) \rightarrow [1-\epsilon, 1+\epsilon]$.

**Soft actor-critic** (SAC) [7] is an off-policy RL algorithm for continuous control tasks. Since off-policy algorithms can train the agent with trajectories collected from the different policies, other than the current one, it appears to be more flexible to alternative routes but may get slower. With SAC, the actor learns a policy $\pi_\psi$, with the guide of critic estimating the state-action value $Q_\phi$ to maximize an objective as a sum of the reward and the policy entropy $\mathbb{E}_{s_t,a_t\sim\pi}[\sum_t r_t + \alpha H(\pi(a_t|s_t))]$. Here, $\alpha$ is an entropy coefficient determining the priority of exploration over exploitation.

The actor, then, is trained by maximizing the expected return of its sampled actions where the objective can be denoted as follows:

$$\mathcal{L}_{\text{actor}}(\psi) = -\,\mathbb{E}_{a_t\sim\pi}\Big[Q_\phi(s_t, a_t) - \alpha \log \pi_\psi(a_t|s_t)\Big]. \tag{4}$$

The critic is updated to minimize the temporal difference. The objectives for the critic, with the estimated target value of the next state, are as follows:

$$V(s_{t+1}) = \mathbb{E}_{a_t \sim \pi}\left[Q_\phi(s_{t+1}, a_t) - \alpha \log \pi_\psi(a_t|s_{t+1})\right] \tag{5}$$

$$\mathcal{L}_{\text{critic}}(\phi) = \mathbb{E}_{s_t, a_t, r_t, s_{t+1} \sim \mathcal{D}}\left[\Big(Q_\phi(s_t, a_t) - \big(r_t + \gamma V(s_{t+1})\big)\Big)^2\right] \tag{6}$$

where $\mathcal{D}$ is the replay buffer.

In this work, we show that our method can be attached to *both on-policy and off-policy* RL algorithms, namely PPO and SAC. Also, our method can be applied to *both continuous and discrete* control tasks as tested with the Procgen and Distracting Control Suite benchmark.

**Style transfer via instance normalization** For style transfer, many recent works adopt a method of using instance normalization (IN) [39,13,5,40,45]. The underlying idea is that the mean and standard deviations of feature maps, computed across the spatial dimension within each feature channel, reflect the images' style. For example, the color or texture of an image can be captured with these statistics, which may be irrelevant features for classifying or detecting an object. By using IN, the effect of styles can be normalized with the formula:

$$\text{IN}(z) = \gamma \cdot \frac{z - \mu(z)}{\sigma(z)} + \beta \tag{7}$$

where $z \in \mathbb{R}^{C \times H \times W}$ is a feature map with channel $C$, height $H$ and width $W$, and $\beta, \gamma \in \mathbb{R}^C$ refers to the affine transformation parameters.

Note that $\mu(z) \in \mathbb{R}^C$ and $\sigma(z) \in \mathbb{R}^C$ are denoted as:

$$\mu(z)_c = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} z_{c,h,w}, \quad \sigma(z)_c = \sqrt{\frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}(z_{c,h,w} - \mu(z)_c)^2} \tag{8}$$

with $c \in \{1, \ldots, C\}$.

Moreover, Huang & Belongie [13] proposed the method of adaptive instance normalization (AdaIN), which can be understood as replacing the style statistics of a target content image with those of a source style image with the definition below:

$$\text{AdaIN}(z, z') = \sigma(z') \cdot \frac{z - \mu(z)}{\sigma(z)} + \mu(z') \tag{9}$$

where $z'$ is the feature map extracted from the source style image.

This idea can be used for mixing styles between images within a mini-batch. Especially in the domain adaptation for image classification, this has been proved to be successful [30]. Zhou et al. [45] adopted style mixing for domain generalization in RL. However, the scope of mixing styles is restricted only to the training mini-batch as AdaIN is an interpolation. Here, our method enables the agents to observe unseen styles by generating new adversarial feature examples.

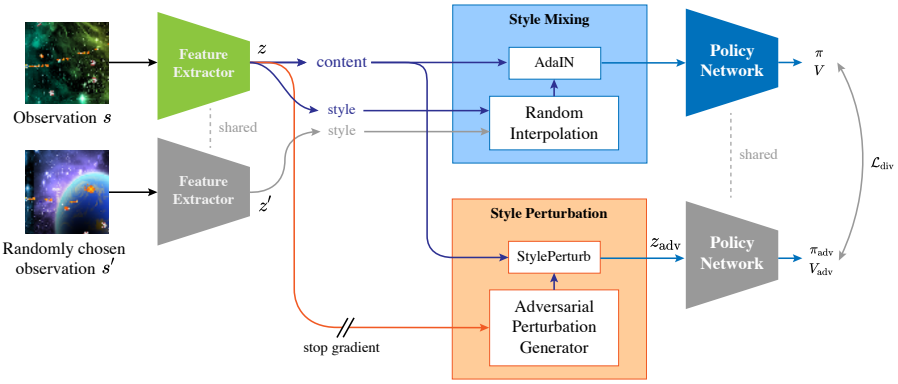


**Fig. 1.** Overview of the proposed Style-Agnostic Reinforcement learning (SAR) with the base model of PPO. The upper *Style Mixing* module makes the policy network focus on the critical content in the observations by mixing styles from randomly chosen states $s'$. We newly employ our *Style Perturbation* module, helping the agent with learning a robust policy by adversarially perturbing latent features.

## 4 Method

**Overview.** SAR is composed of an actor-critic module with RL objectives and a style perturbation generator helping the agents to observe more diverse styles of observations. While the generator is updated to produce more substantial perturbations for style transfer by *maximizing* the difference between the action predictions, the actor learns a more robust policy to the attack from the generator by *minimizing* the gap between predicted action distributions.

To perform this min-max game between actor and generator, we present a **style perturbation layer**, shown in Figure 1. Unlike the conventional approach using only style mixing within the mini-batch [45], the model in the training phase generates new styles and observes a broader range of feature examples. Note that this does not require explicit data augmentation that potentially degrades performance without a cautious choice of augmentation type.

### 4.1 Style Perturbation Layer

Our method is based on the concept of style transfer, which was proven to be successful in generating images with new styles [5,14]. The style perturbation layer shifts the style of observations $z$ with the generated perturbation mean $\beta_{\text{adv}}(z)$ and variance $\gamma_{\text{adv}}(z)$, to build style-perturbed feature map $z_{\text{adv}}$, or StylePerturb(z), with the following equation:

$$z_{\text{adv}} = \gamma_{\text{adv}}(z) \cdot \frac{z - \mu(z)}{\sigma(z)} + \beta_{\text{adv}}(z). \tag{10}$$

Then, the SAR agent should take the same action from $z_t$ and $z_{\text{adv},t}$ to be robust among different environments, as the perturbed feature indicates an observation with different styles but the same semantics, e.g., in Procgen, the same player, enemies, and items, but shifted texture of the background image, the colors of projectiles, and the shapes of obstacles. We will further explain the objectives to achieve this generalization.

### 4.2 SAR Objectives

Primarily, the policy network is updated via PPO or SAC objectives. Thus, the actor loss of SAR is adopted from Equation 2 with PPO baseline or from Equation 4 when using SAC. We will denote this loss be $\mathcal{L}^{\circ}_{\text{actor}}$. Also, for the critic loss, as suggested in RAD [20], we adopt the critic objective of PPO or SAC interchangeably, denoted as $\mathcal{L}^{\circ}_{\text{critic}}$.

Another big goal of SAR is to be robust to different environments. Therefore, the agent should learn its policy by minimizing the difference between the distributions of actions from the style-perturbed features $z_{\text{adv},t}$ and the original ones $z_t$. By leveraging KL-divergence, we can calculate the objective as $\mathcal{L}_{\text{div}} = \text{KL}[\pi(\cdot|z_t)||\pi(\cdot|z_{\text{adv},t})]$. Integrating this with a weight coefficient $\lambda$, the objective for the SAR actor module can be written as:

$$\mathcal{L}_{\text{actor}}(\psi) = \mathcal{L}^{\circ}_{\text{actor}}(\psi) + \lambda \cdot \mathcal{L}_{\text{div}} \tag{11}$$

On the other hand, the generator participates in the min-max game in another manner: to maximize the differences between the action distributions. This module is trained with the objective of the same $\mathcal{L}_{\text{div}}$ but with a converted sign. Unlike the previous works using class *label* information of the environment style [24] or additional heavy background images [10], the objectives for the robust policy (i.e., adversarial loss) do not demand any secondary labors. Hence, the overall goals for the generator can be formalized as:

$$\mathcal{L}_{\text{gen}}(\theta) = -\lambda' \cdot \mathcal{L}_{\text{div}}, \tag{12}$$

where $\lambda'$ can be different coefficient from that of actor objective.

Finally, the critic gets updated to guide the actor to optimize its policy to maximize the value function. Meanwhile, we observed that the sharing critic network, for predicting the value for both style-perturbed features and the original ones, does not bring a huge difference in the performance from decoupling the critic network but lighter training computation. Instead, we add a regularization term $G_{\text{critic}}$ for the value function, to minimize the difference between the value predicted from the adversarial example, i.e., $(V_\phi(z_t) - V_\phi(z_{\text{adv},t}))^2$, which helps stabilization. Thus, the critic's objectives can be computed as follows:

$$\mathcal{L}_{\text{critic}}(\phi) = \mathcal{L}^{\circ}_{\text{critic}}(\phi) + \kappa \cdot G_{\text{critic}}, \tag{13}$$

with hyperparameter $\kappa$[1].

[1] The values used for each hyperparameters $\lambda$, $\lambda'$, $\kappa$ in the experiment are described in the supplementary material.

**On convergence.** When the SAR agents learn the optimal policy $\pi^*$, the KL divergence term, or $\mathcal{L}_{\text{div}}$, becomes zero. This is the situation where the actors infer the same actions from the features with different styles. This might be one of the two cases: (i) the generator produces the same style statistics for all images in the mini-batch, or more possibly, (ii) the actor well focuses on the invariant part of all observations.

Since the model should learn an additional generator module, the training procedure indeed demands more computations. However, the sample efficiency is not highly degraded even with limited training timesteps, e.g., the usual 25M timesteps in Procgen. Although the agent may not learn the optimal policy due to the limited number of epochs, we also empirically observed that the performances of the SAR agents converge as shown in Figure 3.

### 4.3 Pseudo-code

Here, we present the pseudo-code of the SAR algorithm. As depicted in Figure 1, to maximize the effect of style transfer, we design the $z_t$ to pass a *Style Mixing* module and a *Style Perturbation* module with two divided branches. In the *Style Mixing* module, the styles of observations in the mini-batch get interpolated with Equation 9. In *Style Perturbation* module, on the other hand, the styles of observations are shifted with new styles generated from the generator network with Equation 10.

With two different features $z_t$ and $z_{\text{adv},t}$, the SAR agent predicts two different action distributions $\pi_t$ and $\pi_{\text{adv},t}$. The difference between these predictions $\mathcal{L}_{\text{div}}$ is computed, and it gets interpreted in two different ways: by the generator to produce more unfamiliar styles and by the actor to make its policy more robust.

**Algorithm 1** SAR algorithm

Initialize rollout or replay buffer $\mathcal{D}$
Initialize parameters for policy $\psi$, generator $\theta$, and critic $\phi$
**for** every epoch **do**
  **for** every environment step **do**
    Sample $(s_t, a_t, r_t, s_{t+1})$
    Update $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1})\}$
  **end for**
  **for** each mini-batch sampled from $\mathcal{D}$ **do**
    $z_t \leftarrow \text{Encoder}(s_t)$ ▷ Encoder in the actor network
    Generate $\beta_{\text{adv}}(z_t), \gamma_{\text{adv}}(z_t)$ ▷ From the generator network
    $z_{\text{adv},t} \leftarrow \text{StylePerturb}(z_t)$ ▷ Use Equation 10
    $z_t \leftarrow \text{AdaIN}(z_t, z_t')$ ▷ $z_t'$ is permuted from $z_t$ within mini-batch
    Compute $\mathcal{L}_{\text{div}}$ from $z_t, z_{\text{adv},t}$
    Compute $\mathcal{L}_{\text{actor}}$, $\mathcal{L}_{\text{gen}}$, and $\mathcal{L}_{\text{critic}}$
    Update $\psi$, $\theta$, and $\phi$
  **end for**
**end for**

# 5 Results

## 5.1 Setup

In this section, we exhibit the experiment results for the generalization performance of our SAR model on Procgen [3] and Distracting Control Suite [36] benchmarks. Recently, these benchmarks have become a standard for measuring the generalization performance of visual-based RL algorithms [20,17,32,11,10]. These contain reasonably challenging and diverse tasks, which are highly relevant to real-world robot learning.

While the Procgen benchmark is with a *discrete* action space, the Distracting Control Suite presents *continuous* control tasks. We exploited PPO as the basic baseline on the Procgen, and SAC as the basic baseline on the Distracting Control Suite, showing that the SAR algorithm can be applied to both on-policy and off-policy algorithms. Figure 2 visualizes some examples of training and test environments in the two different benchmarks.

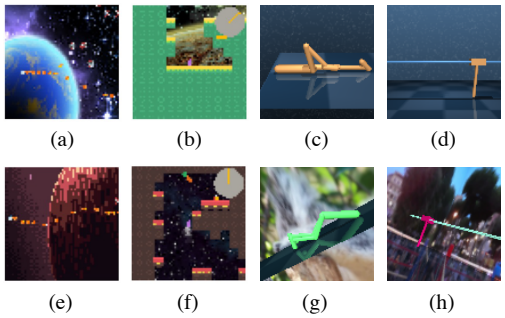


**Fig. 2.** Examples of seen training environments from (a) `starpilot` and (b) `jumper` in Procgen, (c) `walker:walk` and (d) `cartpole:balance` task in Distracting Control Suite, with examples of unseen test environments from (e) `starpilot` and (f) `jumper` in Procgen, (g) `walker:walk` and (h) `cartpole:balance` task in Distracting Control Suite.

**OpenAI Procgen.** One key reason for choosing this benchmark is that this presents different styles between test and training environments. We train the agents on the first 200 levels in the Procgen environment. Then, we test the generalization performance of the agents on the environment levels sampled from the full distribution of unseen levels, with *easy* distribution mode. Among 16 tasks, we selected four tasks demonstrating comparably more considerable differences (`starpilot, climber, jumper, ninja`) and four tasks showing comparably less significant differences (`coinrun, maze, bigfish, dodgeball`) between the training and test environments style.

**Distracting Control Suite.** DeepMind Control Suite [37] presents various continuous control tasks where RL agents can be tested. On top of the DMC, Stone et al. [36] proposed Distracting Control Suite that distracts the agents by applying a color shift, changing the background images into videos, and rotating the camera angle. We test our model and other baselines with different noise coefficient values and show how these models generalize to unseen situations.

### 5.2 Generalization Performance

**Procgen.** First, Table 1 shows the result of the generalization test of SAR with six other baselines. The SAR agent achieved high and robust performances in the zero-shot generalization test: 3 *top-1 score*s and 7 *top-3 score*s out of 8 tasks.

The baselines are six visual-learning RL algorithms showing state-of-the-art results on Procgen. **PPO** [33] is the vanilla on-policy RL baseline, and **RAD** [20] uses data augmentation on top of PPO. We performed random translation (denoted as 'trans') and random color cutout (denoted as 'color') for RAD, as they are reporting the best performance. Among many advanced algorithms on RAD, **UCB DrAC**[32], and **Meta DrAC** [32] are chosen to be compared with our method among three variants of DrAC; the former one presents the best performance among the variants. **Mixstyle** [45] exploits the style mixing, and **DARL** [24] uses an adversarial objective for regularization with style *label*s. [2]

**Table 1.** The generalization scores of SAR and baseline methods on Procgen. The results are averaged over three runs with 100M training timesteps without smoothing. The ranking stands for the average rank among all tasks. The *top-1 score* is bold.

| | PPO [33] | RAD [20] (trans) | RAD [20] (color) | UCB DrAC [32] | Meta DrAC [32] | MixStyle [45] | DARL [24] | **SAR** (Ours) |
|---|---|---|---|---|---|---|---|---|
| Starpilot | 30.37 ±11.14 | 29.57 ±7.52 | 27.03 ±7.51 | 33.17 ±6.37 | 29.40 ±4.61 | 25.70 ±8.13 | 21.97 ±10.66 | **35.87 ±9.13** |
| Climber | 6.73 ±1.27 | 4.87 ±1.31 | 7.23 ±2.05 | **9.43 ±1.35** | 7.77 ±0.68 | 7.37 ±2.71 | 7.03 ±1.37 | 7.93 ±1.10 |
| Jumper | 6.00 ±2.65 | 4.67 ±0.58 | 5.67 ±1.53 | 5.67 ±0.58 | 7.33 ±2.52 | 6.00 ±2.65 | **7.67 ±1.53** | 6.33 ±1.15 |
| Ninja | 6.00 ±2.83 | 5.33 ±2.52 | 5.33 ±2.08 | 6.33 ±1.53 | 7.33 ±0.58 | **8.67 ±1.53** | 7.33 ±0.58 | 8.33 ±1.15 |
| Coinrun | 8.67 ±1.15 | 8.33 ±1.15 | **9.33 ±1.15** | 9.00 ±1.00 | 8.33 ±0.58 | **9.33 ±0.58** | **9.33 ±1.15** | 9.00 ±1.00 |
| Maze | 4.67 ±0.58 | 5.33 ±1.53 | 5.33 ±0.58 | **7.33 ±1.53** | 4.67 ±0.58 | 5.33 ±0.58 | 3.67 ±1.15 | 5.00 ±1.00 |
| Bigfish | 10.37 ±3.27 | 6.03 ±2.06 | 10.13 ±1.84 | 9.37 ±3.16 | 12.03 ±4.38 | 9.00 ±2.94 | 9.07 ±4.05 | **13.20 ±6.16** |
| Dodgeball | 4.13 ±1.75 | 4.93 ±1.53 | 3.20 ±1.56 | **8.13 ±1.33** | 2.40 ±2.46 | 3.60 ±2.31 | 4.47 ±2.73 | 3.60 ±2.23 |
| Avg. Rank | 4.9 | 5.8 | 4.8 | 3.1 | 4.5 | 3.9 | 4.5 | **2.9** |

**Distracting Control Suite.** As Table 2 demonstrates, SAR again showed robust performances in selected four tasks in Distracting Control suite compared to the baselines. This experiment implies that our method can also be applied in continuous control tasks and is attachable to the off-policy RL algorithms.

In this experiment, we purposely tested different baselines from Procgen to compare SAR with various algorithms. **SAC** [7] is the vanilla off-policy RL algorithm, and **CURL** [19] uses a contrastive objective for representation learning

[2] We reproduced all the results of the baselines. The results showed better than the reported performance in several tasks as more training steps [20,32,45].

**Table 2.** The generalization results on Distracting Control Suite after training 500k timesteps. The models are evaluated in two distraction settings: `moderate` setup with the noise coefficient $\beta_{\text{cam}} = \beta_{\text{rgb}} = 0.3$ and 60 background videos, and `hard` setup with the noise coefficient $\beta_{\text{cam}} = \beta_{\text{rgb}} = 0.5$ and 60 background videos, where $\beta_{\text{cam}}$ and $\beta_{\text{rgb}}$ mean camera angle noise intensity and color noise intensity. The results are averaged over 3 runs with different seeds, and rank is calculated within distracted environments.

| | | SAC [7] | CURL [19] | DrQ [17] | PAD [9] | SAR (Ours) |
|---|---|---|---|---|---|---|
| `walker :walk` | zero noise | 373±89 | 828±99 | **930±23** | 838±47 | 325±57 |
| | `moderate` | 96±10 | 88±11 | 126±33 | 125±27 | **139±19** |
| | `hard` | 85±8 | 57±7 | 80±11 | 71±8 | **112±15** |
| `cartpole :balance` | zero noise | **996±1** | 995±3 | **996±3** | 992±6 | 990±5 |
| | `moderate` | 262±20 | 215±57 | 246±15 | 236±17 | **266±26** |
| | `hard` | 251±12 | 216±62 | 240±26 | 238±22 | **261±17** |
| `reacher :easy` | zero noise | 197±7 | **960±24** | 844±63 | 671±285 | 177±51 |
| | `moderate` | 88±11 | 79±11 | 83±10 | 75±19 | **98±13** |
| | `hard` | 72±11 | 67±12 | 78±3 | 71±8 | **93±10** |
| `cheetah :run` | zero noise | 316±159 | 280±12 | **332±21** | 285±29 | 304±80 |
| | `moderate` | **55±10** | 46±8 | 47±8 | 49±5 | 49±11 |
| | `hard` | **53±15** | 41±8 | 33±13 | 41±10 | 46±13 |
| Avg. Rank | | 2.125 | 4.625 | 3.125 | 3.625 | **1.25** |

on top of SAC. **DrQ** [17] was chosen as the representative baseline using the data augmentation with additional regularization terms. **PAD** [9] adapts to a new test environment using self-supervision.[3][4]

**Model behavior.** Figure 3 provides the learning curve of the SAR agents. They exhibit competitive sample efficiency compared to the baselines. A quantitative comparison of the models' computational complexity is in Table 3. Although the SAR model requires more parameters, it does not sacrifice much training and test time in comparison with methods using data augmentation.

**With augmentation.** Training the SAR agents can be integrated with other techniques. For example, Table 4 presents the result of the SAR agents with data augmentation. Both the use of random translation and color cutout improved the performance. This result implies that the SAR agents can potentially be improved using other auxiliary tasks or regularization terms.

**On curriculum learning.** Choice of timing for adopting the min-max game, i.e., curriculum learning, can improve final generalization performances for SAR. See supplementary for the results of the experiment.

[3] We reproduced all the results of the baselines and applied 'trans' to DrQ. The results with zero noise well match the reported performances in most cases [7,19,17].

[4] The performance of PAD differs from the reported value because of the simultaneous application of natural video backgrounds, color noise, and camera angle noise.

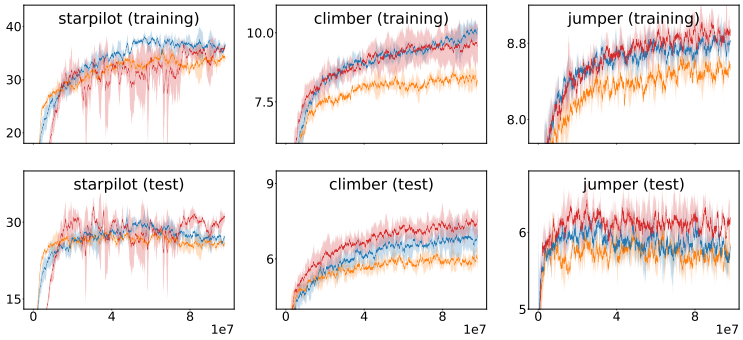


**Fig. 3.** The learning curve of SAR with baselines. For better visualization, we selected three models and three tasks: PPO (blue), RAD (orange), and SAR (red). Here, we applied exponential moving average smoothing with a coefficient value of 0.98.

**Table 3.** A comparison between the number of parameters, training time, and test time. The training time refers to the time consumed for 256 timesteps and an update, and the test time is for running ten episodes in Procgen.

| | PPO [33] | RAD [20] (color) | UCB DrAC[32] | MixStyle [45] | DARL [24] | SAR (Ours) |
|---|---|---|---|---|---|---|
| Parameters ($\times 10^6$) | 0.626 | 0.626 | 0.626 | 0.626 | 0.678 | 1.151 |
| Training Time (s) | 6.507 | 11.605 | 12.841 | 6.735 | 6.542 | 13.377 |
| Test Time (s) | 2.983 | 2.656 | 3.154 | 2.349 | 2.521 | 3.969 |

**Table 4.** Results on generalization performances of SAR with the application of data augmentation and ablation study in `starpilot`. SAR ($\lambda = 0$) refers to the setting without adversarial loss, and SAR ($\kappa = 0$) refers to the setting without regularization loss. We apply two different data augmentation: trans and color. The results are averaged over three runs.

| | PPO [33] | MixStyle [45] | SAR ($\lambda = 0$) | SAR ($\kappa = 0$) | SAR | SAR (trans) | SAR (color) |
|---|---|---|---|---|---|---|---|
| `starpilot` | 27.09 ±0.83 | 26.81 ±0.89 | **27.44** ±2.59 | **29.28** ±7.79 | 28.92 ±4.60 | **30.76** ±0.90 | **33.72** ±1.16 |

### 5.3 Ablation Study

This ablation study answers two questions regarding (1) whether the generator module helps generalization performance and (2) generalization term $G_V$ is important for stabilization. Comparing SAR to PPO baseline, MixStyle using only style mixing, and SAR ($\lambda = 0$), Table 4 shows that the adversarial objective helps improve the mean of the performances. Comparing SAR to SAR ($\kappa = 0$), Table 4 shows that $G_V$ helps stabilize the variance of the performances. [5]

[5] Note that the results in Table 4 are slightly different from Table 1, as we applied exponential moving average smoothing before averaging with coefficient value 0.98.

### 5.4 Learned Feature Analysis

Furthermore, we qualitatively examine the features extracted from the encoder learned with the SAR objectives. The feature $z$ we analyzed is from encoded features before entering the AdaIN layer to exclude the effect of explicit style mixing.

We demonstrate three analyses on the embedding:

- GradCAM [34] visualization for the high-level understanding interpretation.
- Reconstruction images from the feature maps.
- t-SNE [26] for analyzing the latent representations.

**Visualization of model decision** We use GradCAM [34] to visualize where the trained agents are focusing with respect to the decisions. GradCAM can be computed by averaging the activation scores across the channels of the target convolutional layer and weighting by their gradients. Both the agent trained by the vanilla PPO and our agents predict their actions as focused on similar objects in the training environment; in the case of `starpilot`, they are focusing on the shooter and the projectiles from enemies. In the unseen test environment shown in Figure 4, however, the vanilla PPO agent gets more distracted by the changed backgrounds and focuses on irrelevant areas in the images.

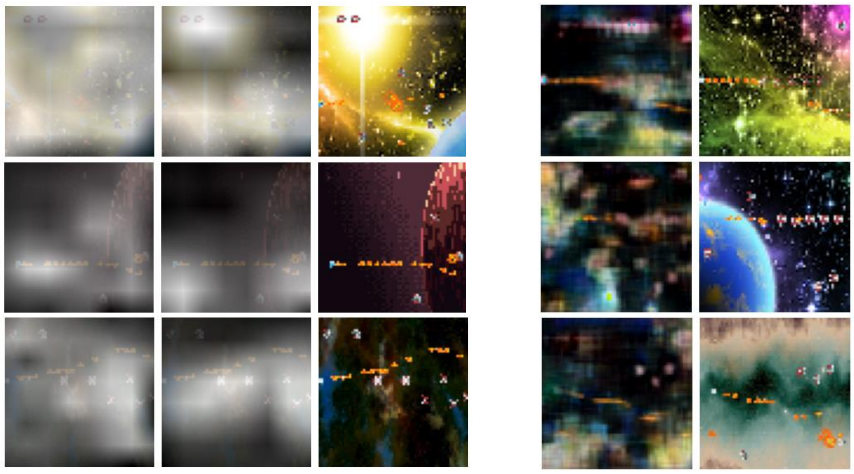


**Fig. 4.** GradCAM results of (a) PPO and (b) SAR, overlaid on (c) the original images from `starpilot` in Procgen. The highlighted regions represent where the agent is focusing. The SAR model better focuses on what is important with the style shifts. (d) Image reconstruction results from features extracted with the SAR agents, and (e) the original observations of `starpilot` in Procgen are displayed.

**Image reconstruction from embedded features.** Reconstructing images from the feature maps displays a more straightforward visualization of the characteristics of the learned features. We trained a new decoder network that converts the feature maps into the original images from training environments. In Figure 4, we show the reconstructed and original images. While the meaningful semantics, e.g., shooters or enemies, are remained, the reconstructed background seems invariant to the different original styles.

**t-SNE Analysis.** Li et al. [24] addressed that the distance between the embedding in the latent space may reflect the dissimilarities between the features. Thus, by observing the t-SNE [26] of the embedding from different environments, how the feature maps are correlated with the style of images can be visualized. While the features extracted from the PPO encoder are patterned with respect to the level *label*s, i.e., the styles, the SAR encoder extracts invariant embedding regardless of them. The visualization result can be seen in supplementary materials.

## 6 Limitation

We address the limitation of the SAR agents, mainly shown in the noise-free setting in Distracting Control Suite in Table 2, although they could well adapt to heavy noise. The additional terms in learning objectives may negatively affect the performance when there is zero noise in the test environment. Not enough styles of training environments would have also affected the actors, as they could not observe a sufficient amount of styles of training features to compete well with the generator. The generator would have taken the wrong direction for generating the new styles, and a failure in the min-max game may happen. Curriculum learning may help alleviate such concerns.

## 7 Conclusion

The SAR agents learn style-agnostic representations by observing features with a wide range of styles by (i) mixing with style randomization and (ii) producing from an adversarial style perturbation generator. In both Procgen and Distracting Control Suite benchmark experimentation, the SAR agents show the best generalization performances in terms of rank. The qualitative analysis reveals that the model helps to learn style-agnostic representations. We hope that the progress made here provides a broader view bringing out more techniques for many other tasks as well, as the SAR agents do.

**Acknowledgement** This work was supported by IITP grant funded by the Korea government(MSIT) (No.2019-0-01906, Artificial Intelligence Graduate School Program(POSTECH) and No.2022-0-00290, Visual Intelligence for Space-Time Understanding and Generation based on Multi-layered Visual Common Sense).

## References


1. Bellman, R.: A markovian decision process. Journal of mathematics and mechanics (1957)
2. Bousmalis, K., Irpan, A., Wohlhart, P., Bai, Y., Kelcey, M., Kalakrishnan, M., Downs, L., Ibarz, J., Pastor, P., Konolige, K., et al.: Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In: Proceedings of the International Conference on Robotics and Automation (ICRA) (2018)
3. 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)
4. Cobbe, K., Klimov, O., Hesse, C., Kim, T., Schulman, J.: Quantifying generalization in reinforcement learning. In: Proceedings of the International Conference on Machine Learning (ICML) (2019)
5. Dumoulin, V., Shlens, J., Kudlur, M.: A learned representation for artistic style. Proceedings of the International Conference on Learning Representations (ICLR) (2017)
6. Ha, D., Schmidhuber, J.: World models. arXiv preprint arXiv:1803.10122 (2018)
7. 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)
8. Hafner, D., Lillicrap, T., Ba, J., Norouzi, M.: Dream to control: Learning behaviors by latent imagination. Proceedings of the International Conference on Learning Representations (ICLR) (2020)
9. Hansen, N., Jangir, R., Sun, Y., Alenyà, G., Abbeel, P., Efros, A.A., Pinto, L., Wang, X.: Self-supervised policy adaptation during deployment. In: Proceedings of the International Conference on Learning Representations (ICLR) (2021)
10. Hansen, N., Su, H., Wang, X.: Stabilizing deep q-learning with convnets and vision transformers under data augmentation. Advances in Neural Information Processing Systems (NIPS) (2021)
11. Hansen, N., Wang, X.: Generalization in reinforcement learning by soft data augmentation. In: Proceedings of the International Conference on Robotics and Automation (ICRA) (2021)
12. Higgins, I., Pal, A., Rusu, A., Matthey, L., Burgess, C., Pritzel, A., Botvinick, M., Blundell, C., Lerchner, A.: Darla: Improving zero-shot transfer in reinforcement learning. In: Proceedings of the International Conference on Machine Learning (ICML) (2017)
13. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: Proceedings of the International Conference on Computer Vision (ICCV) (2017)
14. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
15. Kirk, R., Zhang, A., Grefenstette, E., Rocktäschel, T.: A survey of generalisation in deep reinforcement learning. arXiv preprint arXiv:2111.09794 (2021)
16. Ko, B., Ok, J.: Time matters in using data augmentation for vision-based deep reinforcement learning. arXiv preprint arXiv:2102.08581 (2021)
17. 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)

18. Laidlaw, C., Feizi, S.: Functional adversarial attacks. Advances in Neural Information Processing Systems (NIPS) (2019)
19. Laskin, M., Srinivas, A., Abbeel, P.: CURL: Contrastive unsupervised representations for reinforcement learning. In: Proceedings of the International Conference on Machine Learning (ICML) (2020)
20. 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)
21. Lee, A.X., Nagabandi, A., Abbeel, P., Levine, S.: Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. Advances in Neural Information Processing Systems (NIPS) (2020)
22. Lee, K., Lee, K., Shin, J., Lee, H.: Network randomization: A simple technique for generalization in deep reinforcement learning. Proceedings of the International Conference on Learning Representations (ICLR) (2020)
23. Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-end training of deep visuomotor policies. Journal of Machine Learning Research (2016)
24. Li, B., François-Lavet, V., Doan, T., Pineau, J.: Domain adversarial reinforcement learning. arXiv preprint arXiv:2102.07097 (2021)
25. Li, H., Pan, S.J., Wang, S., Kot, A.C.: Domain generalization with adversarial feature learning. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
26. van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. Journal of Machine Learning Research (2008)
27. Mazoure, B., Ahmed, A.M., MacAlpine, P., Hjelm, R.D., Kolobov, A.: Cross-trajectory representation learning for zero-shot generalization in rl. arXiv preprint arXiv:2106.02193 (2021)
28. 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)
29. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature (2015)
30. Nam, H., Lee, H., Park, J., Yoon, W., Yoo, D.: Reducing domain gap by reducing style bias. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
31. Pinto, L., Davidson, J., Sukthankar, R., Gupta, A.: Robust adversarial reinforcement learning. In: Proceedings of the International Conference on Machine Learning (ICML) (2017)
32. 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)
33. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
34. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the International Conference on Computer Vision (ICCV) (2017)
35. Song, Y., Shu, R., Kushman, N., Ermon, S.: Constructing unrestricted adversarial examples with generative models. In: Advances in Neural Information Processing Systems (NIPS) (2018)

36. 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)
37. Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D.d.L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al.: Deepmind control suite. arXiv preprint arXiv:1801.00690 (2018)
38. Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P.: Domain randomization for transferring deep neural networks from simulation to the real world. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2017)
39. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 (2016)
40. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
41. Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., et al.: Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature (2019)
42. Wang, Y., Qi, L., Shi, Y., Gao, Y.: Feature-based style randomization for domain generalization. arXiv preprint arXiv:2106.03171 (2021)
43. Xu, Q., Tao, G., Cheng, S., Zhang, X.: Towards feature space adversarial attack. arXiv preprint arXiv:2004.12385 (2020)
44. Zhang, C., Vinyals, O., Munos, R., Bengio, S.: A study on overfitting in deep reinforcement learning. arXiv preprint arXiv:1804.06893 (2018)
45. Zhou, K., Yang, Y., Qiao, Y., Xiang, T.: Domain generalization with mixstyle. In: Proceedings of the International Conference on Learning Representations (ICLR) (2021)
46. Zhu, Y., Mottaghi, R., Kolve, E., Lim, J.J., Gupta, A., Fei-Fei, L., Farhadi, A.: Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: Proceedings of the International Conference on Robotics and Automation (ICRA) (2017)