Adaptive Scheduling of Data Augmentation for Deep Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

We consider data augmentation technique to improve data efficiency and generalization performance in reinforcement learning (RL). Our empirical study on Open AI Procgen shows that the timing of when applying augmentation is critical, and to maximize test performance, an augmentation needs to be applied either during the entire RL training, or after the end of RL training. More specifically, if the regularization imposed by augmentation is helpful only in testing, it is better to procrastinate the augmentation after training than to use it during training in terms of sample and computation complexity since such an augmentation often disturbs the training process. Conversely, an augmentation providing regularization useful in training needs to be used during the whole training period to fully utilize its benefit in terms of not only generalization but also data efficiency. Based on our findings, we propose a mechanism to fully exploit a set of augmentations, which identifies an augmentation (including no augmentation) to maximize RL training performance, and then utilizes all the augmentations by network distillation to maximize test performance. Our experiment empirically justifies the proposed method compared to other automatic augmentation mechanism.

1 Introduction

Reinforcement Learning (RL) from visual observations is a fundamental problem since visual data is one of the most common observations available in practice, e.g., video game [20], board game [25, 26] , and robot [29, 15]. However, such a vision-based RL often suffers from the problem of sample efficiency and generalization capability due to the high-dimensional nature of images. As a part of overcoming this, regularization by data augmentation has been widely considered [17, 16], where visual data is augmented by transformation preserving the same meaning or context, e.g., cropping out unimportant part of images or randomizing colors. It not only resolves the data scarcity but also provides an explicit implementation of inductive bias for generalization performance.

Recent works show that using the right types of data augmentation significantly improves both data efficiency and generalization performance [22]. However, choosing the right data augmentation is found to be highly task-dependent, where poor choice can degenerate the generalization and destabilize the training [17, 22]. Hence, a variety of transformations have been developed to enlarge the set of data augmentation [18, 11]. A number of regularization methods using data augmentation, meanwhile, have been proposed to stabilize training process with data augmentation, e.g., self-supervised learning [22] and representation learning [27, 11] by reducing the interference between RL training and regularization. Although the previous works address *what* data augmentation to use and *how* to use it, but the understanding on *when* to apply it in the training process is limited. We test the hypothesis that applying augmentation method at different epochs can have different effects. This

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

is a non-trivial question since the timing of data augmentation is not critical in supervised learning (SL) [1, 9], whereas a curriculum learning can accelerate RL training [23]. [1]

To address our main question, we devise two frameworks with different timings of applying augmentation: InDA(**In**tra **D**istillation with **A**ugmented observations) and ExDA (**Ex**tra **D**istillation with **A**ugmented observations) (Section 3). Implementing the regularization via augmentation in a form of distillation to minimize interference in RL training, InDA interleave the distillation with RL training, while ExDA applies the distillation at the end of RL training. From experiments with InDA and ExDA, we find that:

Time does matter when using augmentation in RL

in contrast to the case of SL [9] where the effect of data augmentation is relatively insensitive to timing. The difference mainly comes from the fact that RL agent collects samples when training but SL uses a fixed data set. To be specific, our main findings from experiments are given in two folds:

(i) Augmentation *needs* to be applied as *early* as possible for sample efficiency and generalization if it can accelerate RL training, e.g., cropping out unnecessary part of image induces an efficient attention mechanism. It is obvious that we need to accelerate RL training from the beginning in order to maximize sample efficiency. However, interestingly, we observe that this kind of augmentation often connotes generalization which is transferable only through diverse experience in training process, i.e., ExDA does not fully exploit generalization gain whereas InDA does. Hence, it is indeed necessary to use such an augmentation during training to gain generalization.

(ii) Applying augmentation *needs* to be *postponed to the end* of RL training for sample efficiency if the regularization imposed by augmentation is helpful only in testing, e.g., augmentation by changing colors is useless when a single background is shown in training, but testing task has multiple backgrounds. We find that this type of augmentation possibly interferes with RL training, but there is no harm in generalization from delaying it. Hence, in this case, ExDA, which never disturbs RL training, is better than InDA.

(iii) Applying time of augmentations can be automatically *determined* by upper confidence bound [3] (UCB) based auto augmentation [22] algorithm for each task. We show that the necessity of *no augmentation* choice among the set of image transformations, because the augmentation can disturb the training. Thus, auto augmentation with identity function can be used as a discriminator to identify the benefit of augmentation during training.

In the above findings, we characterize and suggest effective timings of applying augmentation in RL. In addition to this, our contribution can include proposing InDA and ExDA algorithms, in particular, which is equipped with the distillation augmentation (DA) since it is of independent interest that developing a regularization method using augmentation with minimal interference with RL training. Compared to existing methods DrAC [22], RAD [17], Rand-FM [18], a potential advantage of the proposed one is discussed in Section 3.

2 Related Works

Augmented experience in RL. In order to resolve the problem of poor generalization and sparse data, it is a popular approach to generate diverse (virtual) experiences and let RL agent learn from them. Domain randomization is a technique to produce such experiences from a simulator of targeted system [29, 21, 23]. It is hard to obtain an accurate simulator of practical systems and thus this limits the spectrum of application. Visual augmentation, meanwhile, has no such limit since it is based on simple image transformations such as cropping, tilting, color jitters and so on, although we need a careful understanding on the targeted system to design appropriate image transformer. Reparthy *et al*. [23] propose a curriculum learning for domain randomization where the difficulty is gradually increasing and the insight coincides with some of our findings. However, we provide further understandings on which types of visual augmentation need to be used as sooner or later as possible.

The regularization from augmented data in vision-based RL has been implemented in various learning frameworks, including but not limited to representation [11, 28], self-supervised [22], and contrast

[27]. Raileanu *et al.* [22] further proposes an algorithm to automatically select the most effective augmentations over RL training based on UCB (Upper Confidence Bound) algorithm [3], where each augmentation is considered as an arm and is evaluated its effectiveness in sliding window. The idea of adapting augmentation shares with our main message regarding the timing of augmentation. In Raileanu *et al.* [22], not augmenting is not counted as an arm, while according to our findings, it needs to be. In addition, the post augmentation followed by RL training of ExDA is not considered.

Different time-sensitivity of augmentation than SL. In deep learning, we often observe that the early state of training impacts significantly [6, 1]. Hence, this motivates us to devise time-sensitive methods adapting to the progress of training such as learning rate decay [32] and curriculum learning [30]. Golatkar *et al*. [9] studied such a time-sensitivity of regularization techniques for SL, where the effect of data augmentation in different time does not change much. We find that the time-sensitivity of augmentation can be significant in RL. This contrast perhaps is because of the non-stationary nature of RL, which SL does not have. Although a set of techniques originally developed for SL such as convolutional neural network, weight decay, batch normalization, dropout and self-supervised learning improve deep RL [13, 4, 19, 7, 27, 31, 12], a thorough study needs to be in advance of introducing method from different learning framework as we find the contrasting time-sensitivities of data augmentation. This spirit is also shared with an application [14] of implicit bias in SL [10, 2, 8] to RL.

3 Method

Notation. We consider a standard agent-environment interface of *vision-based* reinforcement learning in discrete Markov decision process of state space $\mathcal{S}$, action space $\mathcal{A}$ and kernel $P = P(s_{t+1}, r_t | s_t, a_t)$ which determines the state transition and reward distribution. The goal of RL agent is to find policy maximizing the expectation of cumulative reward $\sum_{t=0}^{t'-1} \gamma^t r_t$, where t' is terminating time and $\gamma \in [0, 1]$ is discount factor. At each timestep t, the agent selects an action $a_t \in \mathcal{A}$ and receives reward r_t and an image $o_{t+1} = O(s_{t+1}) \in \mathbb{R}^{k \times k}$ as an (possibly partial) observation of the next state s_{t+1}. To augment observations, we consider image transformation function $\phi : \mathbb{R}^{k \times k} \mapsto \mathbb{R}^{k \times k}$ which maintains the dimension.

Baseline RL algorithm. As baseline of deep RL algorithm, we use Proximal Policy Optimization (PPO) [24] which is an on-policy actor-critic RL algorithm to learn policy $\pi_\theta(a \mid o)$ and value function V_θ with network parameter θ. Storing a set of recent transitions $\tau_t := (o_t, a_t, r_t, o_{t+1})$ in experience buffer $\mathcal{D}$, the network parameter θ is updated to maximize the following objective function:

$$L_{\text{PPO}}(\theta) = L_\pi(\theta) - \alpha L_V(\theta) \ , \tag{1}$$

where α is a hyperparameter and some regularization terms are omitted. The clipped policy objective function L_π and value loss function L_V are defined as:

$$L_\pi(\theta) = \hat{\mathbb{E}}\Big[\min(\rho_t(\theta)\hat{A}_t, \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\Big] \tag{2}$$

$$L_V(\theta) = \hat{\mathbb{E}}\Big[\big(V_\theta(o_t) - V_t^{\text{targ}}\big)^2\Big] \ , \tag{3}$$

where the expectation $\hat{\mathbb{E}}$ is taken with respect to $\tau \sim \mathcal{D}$, θ_{old} is the network parameter before the update, $\rho_t(\theta)$ is the importance ratio $\frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{\text{old}}}(a_t|o_t)}$, $\hat{A}_t$ is advantage from Generalized Advantage Estimator [24].

Overall framework. We propose two frameworks: InDA (**In**tra **D**istillation with **A**ugmented observations) and ExDA (**Ex**tra **D**istillation with **A**ugmented observations). To be specific, both of them use PPO for RL and the DA (**D**istillation with **A**ugmented observation), described in Section 3.1, for regularization, although our frameworks can employ other RL algorithms and augmentation-based regularization. InDA, described in Section 3.2, interleaves PPO and DA, whereas ExDA, described in Section 3.3, performs PPO first then DA. As shown in Figure 1, we design InDA and ExDA to conduct either DA or PPO in each epoch.

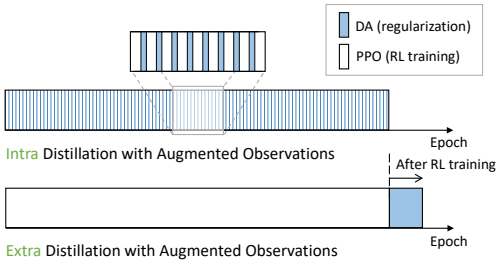


Figure 1: An illustration comparing InDA and ExDA

3.1 Distillation with Augmented observations (DA)

DA regularizes reinforcement learning using distillation with data augmentation, where we train the network to output the same policies and values for given both original and augmented observations. To do so, we fix the network θ_{old} to be distilled and store observation o_t, which is sampled from $\pi_{\theta_{\text{old}}}$, in $\mathcal{D}$. Their augmented observations are represented as $\phi(o_t)$, where $\phi : \mathbb{R}^{n\times n} \mapsto \mathbb{R}^{n\times n}$ is a transformation function. For given $\mathcal{D}$, θ_{old} and ϕ, we then train a network θ to minimize the following distillation loss function:

$$L_{\text{DA}}(\theta) = L_{\text{PD}}(\theta) + L_{\text{VD}}(\theta) \ , \tag{4}$$

where L_{PD} is Kullback–Leibler divergence between policies $\pi_{\theta_{\text{old}}}$ and π_θ and L_{VD} is the mean-squared deviation between value functions $V_{\theta_{\text{old}}}$ and V_θ, i.e.,

$$L_{\text{PD}}(\theta) = \hat{\mathbb{E}}_{o_t\sim\mathcal{D}}\left[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot|o_t), \pi_\theta(\cdot|o'_t)]\right] \ , \tag{5}$$

$$L_{\text{VD}}(\theta) = \hat{\mathbb{E}}_{o_t\sim\mathcal{D}}\left[(V_{\theta_{\text{old}}}(o_t) - V_\theta(o'_t))^2\right] \ . \tag{6}$$

Here o'_t is either the original observation o_t or the augmented one $\phi(o_t)$ with equal probability. The proposed method not only matches the outputs of θ for o_t and $\phi(o_t)$ but also conserves the behavior of θ for o_t to be identical to that of θ_{old} for o_t. By doing this, we can reduce the interference between reinforcement learning and distillation. Indeed, RL training performance can be degenerated by distillation without careful consideration on the interference, e.g., [22] with distillation loss of $\hat{\mathbb{E}}_{o_t\sim\mathcal{D}}\left[\text{KL}[\pi_\theta(\cdot|o_t), \pi_\theta(\cdot|\phi(o_t))]\right]$ and $\hat{\mathbb{E}}_{o_t\sim\mathcal{D}}\left[(V_\theta(o_t) - V_\theta(\phi(o_t)))^2\right]$ can change the behavior learned in RL training. In Section 4.1, we show the importance of separating distillation from RL training which provides substantial performance gain compared to other existing methods [22, 11], e.g., Table 1. In addition, it is worth to note that since the target behaviors ($\pi_{\theta_{\text{old}}}(\cdot|o_t)$ and $V_{\theta_{\text{old}}}(o_t)$), which will be used several times during the distillation, are fixed in DA, we can reduce the computational cost by pre-computing them in advance.

3.2 Intra Distillation with Augmented observations

InDA, described in Algorithm 1, iteratively optimizes PPO and DA, where we explicitly separate PPO and DA. Such a separation reduces their interference [11], while they are often optimized simultaneously in other methods [22]. This separation further robustifies our algorithm in addition to the conservative distillation loss functions in (5) and (6). We variate the timing of augmentation by S and T, which are the times of, respectively, starting and terminating DA. We can control the frequency and timing of applying distillation with hyperparameters I, S' and T', where we perform DA after each I rounds of RL training only if the number RL training rounds n is in the interval of $[S', T']$ (or equivalently, the number of timesteps which have been observed is in the range of $[S, T]$). We provide further details on InDA in the supplementary material.

3.3 Extra Distillation with Augmented observations

As described in Algorithm 2, ExDA performs the distillation followed by the end of RL training, where the lengths of DA and RL training are parameterized by M and N, respectively. It is worth to note that one can lower computational cost by replacing L_{DA} with L_{PD} in DA as there is no need of value function after DA. We empirically check that this reduction wouldn't degenerate RL

Algorithm 1 InDA

Hyperparameter: N, I, ϕ and (S', T') in rounds (or (S, T) in time steps)
Initialize θ close to origin.
for $n = 1, 2, \ldots, N$ **do**
  // RL training
  Store sampled transitions to $\mathcal{D}$;
  Optimize RL objective $L_{\text{PPO}}(\theta)$ with $\mathcal{D}$;
  // Distillation
  if $n \in [S', T']$ and $\text{mod}(n-1, I) = 0$ **then**
    Store $\theta_{\text{old}} \leftarrow \theta$;
    Minimize $L_{\text{DA}}(\theta)$ for $\mathcal{D}$, θ_{old} and ϕ;
  end if
end for

Algorithm 2 ExDA

Hyperparameter: N, M, ϕ
Initialize θ close to origin.
//Pre-training phase with RL algorithm
for $n = 1, 2, \ldots, N$ **do**
  Store sampled transitions to $\mathcal{D}$;
  Optimize RL objective $L_{\text{PPO}}(\theta)$ with $\mathcal{D}$;
end for
Store $\theta_{\text{old}} \leftarrow \theta$;
// Distillation at the end of RL training
for $m = 1, 2, \ldots, M$ **do**
  Minimize $L_{\text{DA}}(\theta)$ for $\mathcal{D}$, θ_{old} and ϕ;
end for

performance. Besides, we consider the re-initialization after pre-training, because we expect that diminishing of non-stationarity can improve generalization as mentioned in [14]. However, training performance is not preserved after re-initialization because $\pi_{\theta_{old}}$ is not completely distillation by low data diversity. Thus, we do not use re-initialization for DA. We leave more interesting details in the supplementary material.

3.4 Auto Augmentation Discriminator

Since the training benefit by augmentation is different depending on the task, it is hard to decide a method to apply augmentation between InDA and ExDA. We inspire from an upper confidence bound [3] (UCB) based auto augmentation method, UCB-DrAC [22]. We make a small change to apply the auto augmentation method as a discriminator to check the necessity of augmentation during training. We combine the InDA with UCB instead of DrAC [22] and call it UCB-InDA. Also, we add an identity function, which returns the image without a transformation, to the set of image transformations. For a clear explanation, we describe the auto augmentation method as a multi armed bandit problem. The action space is the set of image transformations $\Phi = \{\phi_1, \ldots, \phi_k\}$ which contains the identity function. The gain of the augmentation $G(s)$ at sth sampling is the average return during the Interval I. The return is computed by the sum of estimated advantage $\hat{A}$ and predicted value V_θ. The general UCB algorithm uses a mean of rewards from the entire sampling, however in RL, the distribution of return is non-stationary as mentioned in [22], so we use the window average gain $\bar{G}_\phi(s)$ as a reward of each transformation ϕ. The UCB [3] selects actions each time using the sum of a window average gain $\bar{G}_\phi(s)$ and a degree of exploration, such as below:

$$\phi_t = \arg\max_{\phi \in \Phi} \left[\bar{G}_\phi(s) + c\sqrt{\frac{\log(s)}{N_\phi(s)}} \right] \tag{7}$$

where c is the UCB exploration coefficient and $N_\phi(s)$ is the selected number of each augmentation after s sampling. We find the UCB exploration coefficient c with an adaptive way, because the proper

c is different for each training by drastic change of return during the transient time. We will explain more details in the supplementary material.

4 Experiment

Setups. We evaluate the time-sensitivity of applying augmentation on OpenAI Procgen benchmark of 16 games, [5], where at each time t, visual observation o_t is given as an image of size 64×64, and contains full or partial information on the system state. A training or testing environment is defined by a pair of game and mode, where mode determines a set of levels and backgrounds shown in the environment. As training environment, we use one of two modes: *easy* and *easybg*. Easy mode provided by Cobbe *et al.* [5] contains a set of 200 levels, where an agent can learn basic dynamics of game and experience various backgrounds. To see clear advantage from visual augmentation, we further easicate easy mode and devise easybg mode of which only difference from easy mode is showing only a single background. To evaluate generalization capabilities, we use two modes: *test-bg* and *test-lv*, which contain unseen backgrounds and levels, respectively, in addition to the mode that we use for training. The details of modes in our evaluation is provided in the supplementary material.

For the sake of clarity, we mainly focus on two visual augmentations, each of which has clearly distinguishing inductive bias:

(a) *Random convolution* transforms an image by passing a single convolutional layer initialized randomly [18]. Augmentation with this can impose invariant behavior on color changes, and thus is anticipated to provide strong generalization on background changes.

(b) *Crop* leaves a randomly selected rectangle and puts zero-pads to the outside [22]. This augmentation is particularly useful in the fully observable scenarios as it imposes an efficient attention mechanism.

We also report the result with other visual augmentations including *color jitter, gray and cutout color* in the supplementary material, where the same main messages can be found. All results in the main paper are reported as averages over five runs.

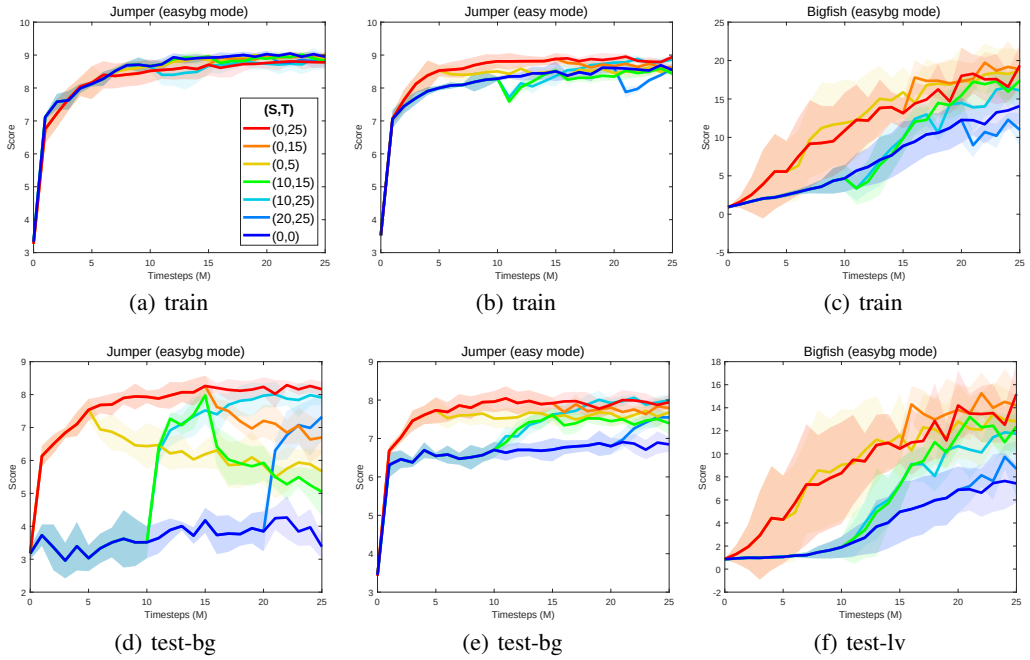
Augmentation		PPO	Oracle	DrAC	RAD	Rand-FM	InDA	ExDA
Rand conv	Train	1.00	0.85	0.88	**0.98**	0.88	0.88	**0.98**
	Test-bg	1.00	2.33	1.86	1.08	1.04	1.92	**2.11**
Color jitter	Train	1.00	0.85	0.95	0.94	-	0.96	**0.98**
	Test-bg	1.00	2.33	1.44	1.37	-	1.43	**1.48**
Grayscale	Train	1.00	0.85	0.93	0.94	-	0.95	**0.99**
	Test-bg	1.00	2.33	1.03	1.04	-	0.97	**1.13**
Cutout color	Train	1.00	0.82	0.82	0.72	-	0.76	**0.94**
	Test-bg	1.00	2.51	1.27	1.33	-	1.19	**1.53**
	Test-lv	1.00	-	0.83	0.69	-	0.69	**0.93**
Crop	Train	1.00	-	1.08	0.28	-	**1.25**	0.91
	Test-lv	1.00	-	1.52	0.46	-	**1.80**	1.09

Table 1: Performance on Open AI Procgen. We compare train and test performance with other baselines such as PPO, Oracle, Drac [22], RAD [17], Rand-FM [18]. Oracle is trained with test backgrounds. We indicate the best methods and Oracle using bold and red text. ExDA outperforms other baselines except when we use *crop*, which is evaluated on unseen levels.

4.1 Improving generalization on Procgen

In Table 1, we compare with other baselines about train and test performance. Every method are trained on 200 levels *easybg* mode which contain a single background. InDA and other baselines are trained for 25M time steps and ExDA is trained for 30 epochs with 0.5M time steps, after training with PPO for 20M time steps. *Test-bg* is used for *random convolution, color jitter, grayscale*

and *cutout color*, which give information of color diversity. *Test-lv* is used for *cutout color* and *crop*, which give a consistency of partial observation. Further details about implementation and hyperparameter are described in the supplementary material. ExDA outperforms other baselines with most augmentation on *test-bg*. Especially, the ExDA with *random convolution* has a comparable performance to Oracle, despite it is trained on a single background. Moreover, it consumes only 0.5M time steps to inject knowledge from augmented images when others use whole training data. We describe the computation issue about both ExDA and InDA in the supplementary material. However, InDA have a better train and test performance with *crop* on *test-lv*. From this result, we can know that each combination of environment and augmentation have suitable timing to apply augmentation. Thus, we analyze the proper condition to use InDA or ExDA in the next section.



(a) train (b) train (c) train

(d) test-bg (e) test-bg (f) test-lv

Figure 2: Comparison according to usage period of augmentation with InDA: (S, T) is hyperparameters of InDA for the start and terminal time step of distillation. InDA uses random convolution on Jumper and crop on Bigfish as an augmentation method. Test results are evaluated on test-bg (unseen backgrounds) and test-lv (unseen levels). Shade region is a standard deviation of five runs. The difference between Jumper (easybg) and Jumper (easy) shows that the benefits of data efficiency and the maintenance of generalization can be differed by the diversity of factors in observations. The augmentation can accelerate the training, such as Bigfish. Furthermore, delayed augmentation commonly improves generalization as much as fully used cases.

4.2 Time dependency of augmentation in RL

In this section, we study *when* and *how* the agent is particularly helped by augmentation in RL training. We make a difference in the times to start and terminate distillation, denoted by S and T, respectively. We experiment by changing the distillation start time S and distillation terminal time T of InDA to see how generalization's effect depends on the time to use augmentation. We evaluate InDA with seven different pairs of start and terminal time of distillation (S, T), where the number of entire timesteps used is 25M: (0, 25), (0, 5), (0, 15), (10, 0), (20, 0), (10, 15), (0, 0). Note that InDA with (S, T) = (0, 0) means RL training only without augmentation, i.e., vanilla PPO. We explain with Jumper and Bigfish in the main paper, and experiments on other environments are described in the supplementary material. Figure 2 represents three environments, Jumper with easy and easybg mode and Bigfish with easybg mode. In the following, we call the curve using a parameter (S,T).

Interrupted augmentation . We wonder how generalization would change after regularization stopped. Thus, we stop the DA during training, such as (0, 5), (0, 15). When we compare the

graph Figure 2(d) and Figure 2(e), the generalization performance in Figure 2(d) rapidly decreases after interrupted on both (0, 5) and (0, 15). In contrast, the curves (0, 5) and (0, 15) in Figure 2(e) maintain the generalization in spite of interrupting augmentation. In training performance, InDA, which uses augmentation throughout training, performs better than PPO in Figure 2(b); however, augmentation does not improve the training performance in Figure 2(a). These results mean that the random convolution alleviates the difficulty by various backgrounds.

On the contrary, random convolution can induce a growing difficulty by increasing the number of factors on a single background. Therefore, the generalization rapidly decreases after augmentation interrupted when training with a single background because the learning direction toward generalization about various backgrounds is not helpful to train. On the other hand, the training can have help when their difficulty is solved by augmentation, such as Figure 2(b) and Figure 2(c). Thus, in deep RL, neural networks maintain the regularization when augmentation helps the training.

Similarly, Golatkar *et al*. [9] argues that the regularization biases toward regions of loss landscape have several equivalent generalized solutions. For the same reason, augmentation regularizes a neural network model by the bias toward generalization in deep RL. Moreover, (0, 5) and (0, 15) increase the training performance and generalization, similar to (0, 25), although they use augmented observations only for a while in Bigfish. Therefore, the augmentation can be not necessary during whole training in some tasks.

Delayed augmentation . We experiment on the generalization when we start to use augmentation lately as 10M, 20M. As shown in Figure 2(d) and Figure 2(e), the generalization rapidly increases after using augmentation at 10M and 20M. Although we use augmentation lately, the augmentation helps the generalization regardless of the usage timing. Golatkar *et al*. [9] shows that delayed augmentation cannot achieve as much as using augmentation during whole training in supervised-learning. However, (10, 25) improves the generalization comparable with (0, 25), which use augmentation throughout training, unlike supervised learning. However, when augmentation noticeably helps the training, such as Figure 2(e), delayed augmentation struggles to follow earlier one in Figure 2(f), because the RL gradually improves the policy and trajectory by Markov property. Furthermore, RL has a limited number of samples unlike supervised learning, so using augmentation from the initial time is more critical than supervised learning if augmentation helps the training.

On the other hand, we confirm that delayed augmentation can induce bias toward generalization after interrupted, although it is not used in initial transient time. For example, curves (10, 15) at Figure 2(e), Figure 2(f) equivalently perform with (10, 25) even after 15M time step. Through this result, the bias toward generalization can occur regardless of timing by augmentation, but usage from the start is essential when the augmentation gives important knowledge during training.

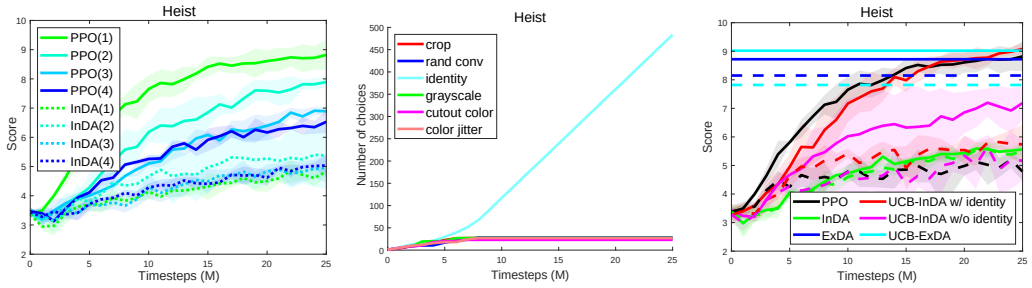
When ExDA needs to be used. ExDA have different with other methods in the timing of applying augmentation. Most methods combined with data augmentation in RL use augmentation throughout the entire RL training. We show that augmentation rapidly increases the generalization performance in spite of the late usage of augmentation in Figure 2. These results motivate us to consider that the possibility to improve generalization by augmentation after training in RL.

Especially in Figure 3(a), the performance of PPO is closer to InDA as the background becomes more diverse. It means that the various background increase the difficulty of training, and also random convolution make similar effect to diverse background image. In contrast, the ExDA preserves the training score after the pure RL training. These gap in training are reflected in the test performance.

Based on the above two analyses, ExDA is proper to environments which suffer difficulty of training by various factors such as background, object color in image. Thus, for generalization, it is possible to train with ExDA in a single background rather than various backgrounds.

When InDA needs to be used. ExDA does not always provide a guarantee of improving generalization. In Table 1, InDA is well generalized to unseen levels with *crop* than ExDA. There are two reasons why ExDA cannot overcome InDA in some environments.

First, the diversity of data is important to generalize about the unseen levels [5]. InDA is trained with various observations during training, while ExDA apply the augmentation on only pre-trained policy's trajectories. Thus, augmentation after training is hard to get over the limitation of data diversity when



(a) Background diversity (b) Augmentation selection (c) Performance

Figure 3: Comparison to the train performance according to background diversity on Heist *easybg* when InDA use random convolution in 3(a). We describe the degree of background diversity in numbers, e.g., PPO(1). As the diversity of the background increases, the performance of the PPO approaches InDA. Figure 3(b) show the selected number of augmentations by UCB on Heist. We compare two UCB-InDAs, w/ and w/o identity function with PPO, InDA, ExDA, UCB-ExDA on Heist *easybg* in Figure 3(c). UCB-InDA is trained after UCB-InDA w/ identity, we use *random convolution* as a data augmentation in InDA, ExDA. The solid line is the train performance and the dotted line is the test performance. ExDA make a large gain in the test performance than InDA by preserved train performance. Moreover, UCB-InDA w/ identity outperforms UCB-InDA w/o identity in the training.

the generalization needs the diversity of data distribution. Second, InDA can accelerate the training such as 2(c), so ExDA cannot overcome gap in training performance. Thus, InDA needs to be used in both cases.

How to choose between InDA and ExDA. We explain that ExDA is proper when the augmentation methods increase the difficulty to train, and InDA is proper when the augmentation methods help to train. However, we cannot know that certain augmentation helps the train or not. Thus, we use UCB-InDA to automatically decide the necessity of the augmentation during training. As shown in Figure 3(b), UCB selects identity function the most. It means that other augmentations are not helpful to train on Heist(*easybg*) and proposes ExDA rather than InDA. Furthermore, UCB-InDA w/ identity performs better than w/o identity. It suggests that identity should be included in the UCB action, because of the environments, which do not need the augmentation during training. In the other hands, PPO is same with InDA with identity, which is the best transformation on Heist(*easybg*). Thus, we can use it as pre-trained method for ExDA, e.g., UCB-ExDA, because UCB-InDA w/ identity performs comparable train performance with PPO in Figure 3(c). As the result, we can automatically select the appropriate method between InDA and ExDA for each task.

5 Discussion

We have studied the time-sensitivity of applying visual augmentation for RL, which is in contrast to that for SL, where such a difference can be explained with non-stationary data generation in RL. In particular, if the regularization imposed by augmentation is useful only for testing, it is better to procrastinate the augmentation to the end of RL training than to use it for the entire learning in terms of sample and computation complexity since it can disturb the RL training. However, an augmentation providing regularization useful in training obviously needs to be used during the whole training period to fully utilize its benefit in terms of not only generalization but also data efficiency. We believe that our findings provide useful insights not only for auto-augmentation adjusting the use of augmentation round-by-round, where DA at the end of RL training would provide substantial gains as ExDA does. However, it is still open to design auto-augmentation for RL as the gain from augmentation has highly non-stationary, and thus its evaluation is challenging.

References

[1] A. Achille, M. Rovere, and S. Soatto. Critical learning periods in deep networks. In *International Conference on Learning Representations*, 2018.

[2] S. Arora, N. Cohen, W. Hu, and Y. Luo. Implicit regularization in deep matrix factorization. In *Advances in Neural Information Processing Systems*, pages 7411–7422, 2019.

[3] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[4] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman. Quantifying generalization in reinforcement learning, 2019.

[5] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR, 2020.

[6] D. Erhan, A. Courville, Y. Bengio, and P. Vincent. Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 201–208. JMLR Workshop and Conference Proceedings, 2010.

[7] J. Farebrother, M. C. Machado, and M. Bowling. Generalization and regularization in dqn, 2020.

[8] G. Gidel, F. Bach, and S. Lacoste-Julien. Implicit regularization of discrete gradient dynamics in linear neural networks. In *Advances in Neural Information Processing Systems*, pages 3196–3206, 2019.

[9] A. S. Golatkar, A. Achille, and S. Soatto. Time matters in regularizing deep networks: Weight decay and data augmentation affect early learning dynamics, matter little near convergence. In *Advances in Neural Information Processing Systems*, pages 10678–10688, 2019.

[10] S. Gunasekar, B. E. Woodworth, S. Bhojanapalli, B. Neyshabur, and N. Srebro. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems*, pages 6151–6159, 2017.

[11] N. Hansen and X. Wang. Generalization in reinforcement learning by soft data augmentation. *arXiv preprint arXiv:2011.13389*, 2020.

[12] N. Hansen, Y. Sun, P. Abbeel, A. A. Efros, L. Pinto, and X. Wang. Self-supervised policy adaptation during deployment. *arXiv preprint arXiv:2007.04309*, 2020.

[13] I. Higgins, A. Pal, A. A. Rusu, L. Matthey, C. P. Burgess, A. Pritzel, M. Botvinick, C. Blundell, and A. Lerchner. Darla: Improving zero-shot transfer in reinforcement learning, 2018.

[14] M. Igl, G. Farquhar, J. Luketina, W. Boehmer, and S. Whiteson. The impact of non-stationarity on generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826*, 2020.

[15] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

[16] I. Kostrikov, D. Yarats, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

[17] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020.

[18] K. Lee, K. Lee, J. Shin, and H. Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. *arXiv*, pages arXiv–1910, 2019.

[19] Z. Liu, X. Li, B. Kang, and T. Darrell. Regularization matters in policy optimization – an empirical study on continuous control, 2020.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[21] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017.

[22] R. Raileanu, M. Goldstein, D. Yarats, I. Kostrikov, and R. Fergus. Automatic data augmentation for generalization in deep reinforcement learning. *arXiv preprint arXiv:2006.12862*, 2020.

[23] S. C. Raparthy, B. Mehta, F. Golemo, and L. Paull. Generating automatic curricula via self-supervised active domain randomization, 2020.

[24] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[25] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550 (7676):354–359, 2017.

[26] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[27] A. Srinivas, M. Laskin, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning, 2020.

[28] A. Stooke, K. Lee, P. Abbeel, and M. Laskin. Decoupling representation learning from reinforcement learning. *arXiv preprint arXiv:2009.08319*, 2020.

[29] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017.

[30] X. Wu, E. Dyer, and B. Neyshabur. When do curricula work?, 2020.

[31] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus. Improving sample efficiency in model-free reinforcement learning from images, 2020.

[32] K. You, M. Long, J. Wang, and M. I. Jordan. How does learning rate decay help modern neural networks?, 2019.