Flexible Data Augmentation in Off-Policy Reinforcement Learning

Alexandra Rak[1(✉)], Alexey Skrynnik[1,2], and Aleksandr I. Panov[1,2]

[1] Moscow Institute of Physics and Technology, Moscow, Russia
rakalexandra@mail.ru
[2] Federal Research Center Computer Science and Control of the Russian Academy of Sciences, Moscow, Russia

Abstract. This paper explores an application of image augmentation in reinforcement learning tasks - a popular regularization technique in the computer vision area. The analysis is based on the model-free off-policy algorithms. As a regularization, we consider the augmentation of the frames that are sampled from the replay buffer of the model. Evaluated augmentation techniques are random changes in image contrast, random shifting, random cutting, and others. Research is done using the environments of the Atari games: Breakout, Space Invaders, Berzerk, Wizard of Wor, Demon Attack. Using augmentations allowed us to obtain results confirming the significant acceleration of the model's algorithm convergence. We also proposed an adaptive mechanism for selecting the type of augmentation depending on the type of task being performed by the agent.

Keywords: Reinforcement learning · Image augmentation · Rainbow · Regularization

1 Introduction

One of the best techniques that can significantly increase the generalizing property of a machine learning model is increasing the amount of data for the training. In practice, this approach often cannot be applied due to the limited amount of available data. One of the ways to solve this problem is to form an extension of the source data based on some knowledge about the problem being solved and some known requirements for model invariance. For example in the image classification problem, such requirement is model invariance to a wide variety of transformations, such as reflection, resizing, adding noise, or changing color. In speech recognition problems such properties are model invariance to adding noise, changing the volume, and changing the speed of the audio track. This approach is called data augmentation and it allows us to generate new data-target pairs using the described transformations and thus obtain more data for learning [15]. However, we should be careful about using different types of augmentations, since such data transformations may affect the label of the true

L. Rutkowski et al. (Eds.): ICAISC 2021, LNAI 12854, pp. 224–235, 2021.
https://doi.org/10.1007/978-3-030-87986-0_20

class of the object being classified. For example, vertical reflection can change the label of a true class for images that represent numbers or letters [5].

The described augmentation methods have proven their effectiveness in the fields of Computer Vision and speech recognition, where knowledge about model invariance can be easily used for data augmentation. However, such techniques weren't studied a lot in the field of reinforcement learning. At the same time image provides a representation of the agent's environment state, it is often one of the main information components received from the environment by the agent as an observation. This work is devoted to exploring the image augmentation influence on the quality of the off-policy model-free model in reinforcement learning. In this work, we investigate the effect of applying a simple idea of image augmentation representing the current and subsequent state of the environment, which are extracted from the replay buffer along with the rest of the data during the operation of the model algorithm. It is assumed that due to the limited size of the buffer this technique will work as regularization and will help to increase the generalization property of the model's algorithm.

Thus, the contribution of the work is as follows:

- studying the influence of popular Computer Vision augmentation techniques in model-free off-policy reinforcement learning algorithms,
- conducting experiments with several Atari environments and identifying augmentation techniques that affect the final quality and speed up the model's algorithm convergence,
- results generalization and interpretation of the augmentation effect for the off-policy algorithms.

2 Related Works

Image augmentation is actively researched in the field of Computer Vision. Successful attempts to solve image classification problem using image augmentation are made in [11]. The efficiency of training data extension using simple augmentation methods, such as cropping, rotating, and flipping input images is demonstrated widely. In this work, the authors restricted data access to a small subset of the ImageNet dataset and evaluated the results for each conversion method. One of the most successful strategies to increase the size of training data in the work is the traditional transformations mentioned above. This work also experiments with generative models for creating images of various styles and suggests a method called Neural Augmentation, which allows the neural network to study transformations that are most suitable for classification problems. In the article [2] a new algorithm was used to automatically search for the best image augmentation technique for each individual dataset. The proposed automatic augmentation selection mechanism showed high validation accuracy on the target dataset and achieved state-of-art accuracy on CIFAR-10, CIFAR-100, Svhn, and ImageNet (without using additional data). The augmentation algorithms obtained on the ImageNet dataset can be transferred to another algorithm and

can be used for other datasets such as Oxford Flowers, Caltech-101, Oxford-IIT Pets, FGVC Aircraft, and Stanford Cars.

In Reinforcement Learning (RL) the topic of augmentation has not been studied a lot, but more interesting papers on this topic have recently appeared. In [9] authors presented a plug-and-play method that can improve any RL algorithm. Authors demonstrated how random cutout, color jitter, patch cutout, and random convolution can allow simple algorithms to perform the same way or even outperform complex modern methods by common criteria in terms of efficiency and communication capability. It is argued that only a variety of data can cause agents to focus on meaningful information from multidimensional observations without any changes in the training method. The results are demonstrated in DeepMind environments, where state-of-art quality is shown for 15 environments.

The article [8] offers a simple method that can be applied to standard model-free reinforcement learning algorithms, allowing to get more stable learning of the model directly from images without introducing auxiliary loss functions or doing pre-training. The approach uses augmentations commonly used in Computer Vision tasks to transform input data and as a result dramatically improves the Soft Actor-Critic algorithm's performance, enabling it to reach state-of-the-art performance on the DeepMind control suite. The proposed algorithm, which they dub DrQ: Data-regularized Q, can be combined with any model-free reinforcement learning algorithm. It was also demonstrated by applying it to DQN algorithm and significantly improve its data-efficiency on the Atari 100k benchmark.

In other work [12] the method UCB-DrAC was proposed. This new method is used for automatically finding effective data augmentation for RL tasks. It enables the principled use of data augmentation with actor-critic algorithms by regularizing the policy and value functions with respect to state transformations. It was shown that UCB-DrAC avoids the theoretical and empirical pitfalls typical in naive applications of data augmentation in RL. The method improves training performance by 16% and test performance by 40% on the Procgen benchmark, relative to standard RL methods such as Proximal Policy Optimization [14].

We can also mention works in which task-oriented augmentation was carried out [16,17]. Episodes of solving one of the tasks available to the agent were replenished by episodes in which another goal was achieved. In our work, we focus on studying the impact of data augmentation specifically for off-policy algorithms, for which such research has not been conducted before.

3 Model Description

Q-Learning. Unlike classical algorithms and machine learning methods, Reinforcement Learning is a class of models that do not receive direct information about object-response pairs. Instead of this the agent learns to act in some environment in a way that maximizes some scalar value of the reward. At each discrete step $t = 0, 1, 2, ...$, the environment presents an observation S_t to the

agent, the agent reacts by selecting an action A_t, after which it receives a new reward value from the environment R_{t+1} and the next state S_{t+1}. This interaction is formalized by the concept of MDP or Markov decision - making process represented by a tuple $< S, A, T, r, \gamma >$, where S is a finite state space, A is a finite set of actions, $T(s, a, s') = P[S_{t+1} = s' | S_t = s, A_t = a]$ is a stochastic transition function between states, $r(s, a) = E[R_{t+1} | s_t = s, a_t = a]$ - reward function, and $\gamma \in [0, 1]$ - discount coefficient.

Reinforcement Learning uses the assumption that future rewards are discounted with a coefficient of γ for each step. Then the total discounted reward at time t is defined as

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$$

where T is understood as the moment when the game ends. Here it is important to define the so-called action-value function $Q^*(s, a)$ as the maximum expectation of the reward received for any policy π after the agent visits the state s and performs the action a:

$$Q^*(s, a) = \max_{\pi} E[R_t | S_t = s, A_t = a, \pi]$$

The π policy determines the probability distribution of actions for each of the states. The optimal Q-function, in this case, obeys the *equation of Bellman.*

$$Q^*(s, a) = E_{s' \sim \epsilon} \left[r + \gamma \max_{a'} Q^*(s', a') | s, a \right]$$

The basic idea of many reinforcement learning algorithms is to evaluate the action-value function using the Bellman equation in an iterative algorithm:

$$Q_{i+1}(s, a) = E \left[r + \gamma \max_{a'} Q_i(s', a') | s, a \right]$$

Such iterative algorithms converge to the optimal action-value function $Q_i \rightarrow Q^*$ for $t \rightarrow \infty$ In practice, this basic approach is not very applicable, since the Q-function is evaluated separately for each sequence of steps, without any generalization. Instead, an approximation is used to estimate the Q-function:

$$Q(s, a; \theta) \approx Q^*(s, a)$$

This can be a linear approximator function, or a non-linear approximator can be used instead, including a neural network with θ-weights. Such a Q-neural network can be trained by minimizing the sequence of loss functions $L_i(\theta_i)$:

$$L_i(\theta_i) = E_{s,a \sim p(.)} \left[(y_i - Q(s, a; \theta_i))^2 \right]$$

where y_i is the value of the desired function on the $i$iteration:

$$y_i = E_{s' \sim \epsilon} \left[r + \gamma \max_{a'} Q^*(s', a'; \theta_{i-1}) | s, a \right]$$

The gradient of such a function will be equal to:

$$\nabla_{\theta_i} L_i(\theta_i) = E_{s,a\sim p(.);s'\sim\epsilon}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i)\right) \nabla_{\theta_i} Q(s, a; \theta_i)\right]$$

Instead of calculating the total expectation in the gradient above, it is often computationally appropriate to optimize the loss function using stochastic gradient descent. If the model weights update each step and replace the expectation with a sample from the distributions p and the emulator ϵ, respectively, then we come to the familiar Q-learning algorithm [20]. This algorithm is a model-free algorithm, as well as an off-policy - it learns using an ϵ-greedy strategy, choosing an action with the maximum value of the Q-function with probability ϵ and with probability $1 - \epsilon$ taking a random action. This strategy allows the model to adjust its own estimates if necessary [10].

Deep Q-Network. Deep Q-Network is a successful generalization of combining convolutional neural networks and reinforcement learning to approximate the Q-function for s_t in the t-step. In this case, the state is fed as input to the neural network in the form of a sequence of pre-processed pixel frames. At each step of the algorithm, depending on the current state, the agent selects the next action using the ϵ-greedy strategy described above and also adds a tuple $(s_t, a_t, r_{t+1}, s_{t+1})$ to a special playback buffer called the replay buffer. The parameters of the neural network are optimized using stochastic gradient descent to minimize the loss of the function:

$$L = \left(r + \gamma \max_{a'} Q_{\hat{\theta}}(s', a') - Q_{\theta}(s, a)\right)^2$$

In this case, the gradient of the loss function is considered only for *online* – a neural network that is also used to select the optimal action. The parameters $\hat{\theta}$ represent *target*, a neural network that is a periodic copy of the *online* network. *Target* – the network is not directly optimized. Using a replay buffer and a target network allows for greater stability of model training and leads to good results for many reinforcement learning tasks.

Rainbow. This algorithm is an extension of the DQN algorithm described above and uses the following 6 improvements: Double Q-Learning [18], Prioritized replay [13], Dueling networks [19], Multi-step learning [3], Distributed RL [1], Noisy Nets [4]. This modification of the DQN was used for research on the use of augmentation in reinforcement learning. The described algorithm was chosen because it exceeds the standard DQN in terms of convergence rate, and also preserves the model-free and off-policy properties.

4 Augmentation in Reinforcement Learning

Data augmentation in Computer Vision problems is considered as a good method not only for increasing the amount of source data but also for increasing the

diversity of these data [15]. The most popular image augmentation techniques are horizontal reflection, multi-pixel shifts, rotations, and other transformations. However, not all of these types of augmentation can be applied in reinforcement learning tasks. For example, rotation and horizontal reflection can significantly affect the optimal action that will maximize the agent's reward at a particular step of the game. Therefore, for the model under study, we have chosen augmentations with respect to which the model is invariant. This property means that after the data augmentation steps the optimal action at a particular step does not change.

4.1 Augmentation Types

Random Erase. To augment an arbitrary frame of the game a random point is selected within the borders of the image and a rectangle is constructed from it with side lengths distributed randomly from 0 to 20. The built shape is colored gray.

Random Crop. The original image is expanded by 4 pixels on each side and filled in with black. Then the resulting frame is randomly cut from the resulting image. The size of the final frame corresponds to the size of the initial image without expansion.

Random Contrast. Before augmentation, the regularization coefficient for contrast is randomly selected as a sample from the normal distribution $N(1, 0.5)$. Then the image contrast is changed with this coefficient. In such transformation, the value $k = 1$ corresponds to the absence of changes, and $k = 2$ corresponds to a double increasing of the contrast.

Random Augmentation. One of the three augmentations described above is applied to the image with equal probability.

4.2 General Framework for Action-Value Function Regularization

In DQN-algorithm deep networks and reinforcement learning were successfully combined by using a convolutional neural net to approximate the action-value function Q for a given state S_t, which is fed as input to the network in the form of a stack of raw pixel frames. Applying any augmentation to such frames can be considered as some regularization of the value function. Here we can define the general framework for such regularization by adding augmentation function f, which will apply some exact type of state augmentation. Generalizing, we can say, that augmentation function f can apply to the exact state any transformation which preserves invariance property of the model. Invariant state transformation function can be defined:

$$f : S \times T \to S$$

as a mapping that preserves the Q-values

$$Q(s, a) = Q(f(s, \nu), a)$$

for all $s \in S, a \in A$ and $\nu \in T$. where ν are the parameters of f, drawn from the set of all possible parameters T.

In DQN-algorithm a non-linear approximator can be used for Q-value function approximation, including a neural network with θ-weights. Applying a new value-function regularization framework such Q-neural network can be trained by minimizing the sequence of loss functions $L_i(\theta_i)$:

$$L_i(\theta_i) = E_{s,a \in p(.)} \left[(y_i - Q(f_i(s, \nu), a; \theta_i))^2 \right]$$

where y_i is the value of the desired function on the i-iteration.

The choice of a particular regularization function strongly depends on the current problem being solved. The choice of function can be influenced by the specifics of the game, the agent's strategy for getting the maximum reward, and other features. Thus, the task of selecting the f function is non-trivial and requires additional experiments. We need to find such a function, and to choose its parameters to minimize the loss function described above:

$$\hat{f} = argmin_{f \in F, \nu \in T} L(\hat{\theta})$$

where F is a family of functions with mapping $f : S \times T \rightarrow S$ that preserves the property of model invariance.

Due to the complexity of the optimization problem for such a family of functions, this problem can be simplified and reduced to the problem of finding some approximation of a function $\hat{f}$. An important property of such an approximation function is adaptability, the ability of the function to take into account the features of the problem being solved.

In further experiments, we consider f as one of the image augmentation types described above. Set of all possible parameters T can be understood as all possible values of such augmentation parameters. For example among T will be all possible probability distributions of different augmentations for random augmentation.

Each state of the environment can be associated with a certain frame from the game. However, in this case, we can say little about the direction in which, for example, the ball is moving in a Breakout game and at what speed it does so. This raises the question of fulfilling the Markov property of the model, that is, if only one frame is used to represent a certain state of the environment, this property is violated. To solve this problem instead of a single frame of the game to characterize the state, several consecutive frames are used (in our case, 4). This allows you to get more information about the environment, and we can draw conclusions about the direction of the ball and its speed from two frames of the game, and about its acceleration – by three.

In all evaluated games, the source frames are reduced to the size of 84×84 and converted to the black and white format.

An augmentation procedure is added to the Rainbow algorithm, which is performed every time an image is sampled from the replay buffer. In this case, the randomness property should be performed only for different pairs of (s_t, s_{t+1}). Within each such pair, the augmentation is exactly the same for s_t and s_{t+1}. This is necessary in order to preserve the integrity of each observation of the environment, that is, the tuple $(s_t, a_t, r_{t+1}, s_{t+1})$ is an integral, indivisible unit for a reinforcement learning algorithm.

5 Experiments

5.1 Data Augmentation for Off-Policy RL

Currently, there is a lot of environments for solving RL tasks. We decided to explore the impact of augmentation regularization using the OpenAI Gym library, which provides APIs for simulating a large number of virtual environments, including Atari games. In this section, we compare four types of augmentation for the following environments: Berzerk, Breakout, Demon Attack, Space Invaders, Wizard of Wor. As a baseline model, we used the original Rainbow algorithm with the parameters specified in the original paper [6]. Table 1 reports summary results for whole 5 games. Learning curves for Berzerk and Space Invaders environments presented in Fig. 1.

Breakout. For Breakout, several augmentations showed a result that exceeds the result of the original algorithm. Augmentations not only increased the convergence rate of the algorithm but also resulted in a higher reward after convergence. The best result was shown by Random Augmentation, which increased the model's reward by 17%. Random Crap and Random Erase augmentations also accelerated convergence and were able to increase the reward by 10% and 8%, respectively. Random Contrast augmentation did not show results significantly different from baseline.

Space Invaders. The game requires more training time compared to other tested games, so we got significant results only after a large number of training steps. We obtained higher reward results for all types of augmentations in this game. The best type of augmentation is Random Erase, which increases the model's reward by rather 400%. In our experiment, Random Augmentation and Random Contrast are also successful with the result of more than 250% reward increasing. The model with Random Crop augmentation got 47% higher score than a Baseline model.

Wizard of Wor. For Wizard of Wor the best types of augmentation were Random Augmentation and Random Erase with 45% and 20% higher reward result. These two augmentations also increased the convergence rate of the algorithm. For this environment, we got worse results for Random Crop and Random Contrast augmentation with reward decreasing by 17% and 32%.

Berzerk. For this game, we obtained the highest reward results for Random Crop augmentation, which increases the model's reward after 30M training steps

by 202%. In our experiment, Random Augmentation is also successful with a result of more than 173% reward increasing. The model with Random Erase on average shows the same score as a baseline model. The worse results were provided by the model with Random Contrast augmentation with 20% reward decreasing.

Demon Attack. For Demon Attack all augmentation types provided good results. But here we got more acceleration of the convergence rate and less increase of models reward in the end. The best augmentations for this game are Random Crop and Random Augmentation with over 15% reward increasing after 30M training steps. For these two augmentation types, we got over 160% increasing of convergence from 10M to 20M training steps. Random Erase provided 90% increasing of convergence from 10M to 20M training steps and 6% reward increasing in the end. For Random Contrast we got 15% increasing in convergence and nearly the same results as a baseline model in the end.

Table 1. Final results for each type of the augmentation for five Atari games. The column for each augmentation shows the final cumulative reward and its percentage relative to the Rainbow without augmentations (Original).

	Steps	Original		Random		Erase		Crop		Contrast	
Environment	$\times 10^6$	Score	%	Score	%	Score	%	Score	%	Score	%
Berzerk	40	2527	0	7301	189	2821	12	**11841**	**369**	2298	−9
Breakout	20	353	0	**389**	**10**	370	5	368	4	338	−4
Demon Attack	40	107648	0	**124237**	**15**	115004	7	124057	**15**	106936	−1
Space Invaders	40	2663	0	9406	253	**13253**	**398**	3926	47	10021	276
Wizard of Wor	30	8261	0	**12746**	**54**	10196	23	6175	−25	5027	−39

5.2 Behavioral Cloning

Augmentations work fine for such tasks as classification. In the following series of experiments, we decided to study the effects of different augmentation types on the behavioral cloning model. The main idea of this section is to explore how different proportions of image transformations can influence the final quality of a model trained to copy the behavior of an expert. We took the model with the best results for the Wizard of Wor game from the previous section. The selected model was trained during 50 m steps using the Rainbow algorithm with Random Augmentation regularization. We launched inference for this model and saved 10,000 state-action transitions to use it as input data for the behavioral cloning model. The input data was augmented using the Random Augmentation method with different proportions of Random Crop, Random Erase, Random Contrast, and Original Data augmentations (data without changes). As a model, we took a simple classifier with Cross-Entropy Loss. To iterate through various augmentation ratios for each of the augmentations in Random Augmentation we used discretization of [0, 33, 66, 100] followed by normalization to obtain the final discrete

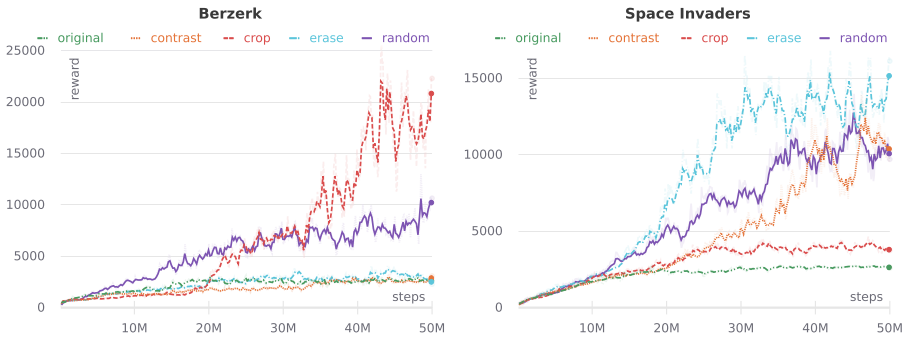

Fig. 1. Learning curves for Berzerk **(left)** and Space Invaders **(right)** environments. The correct choice of augmentation leads to a significant improvement in the convergence of the algorithm. The Crop augmentation shows superior results for Berzerk environment, but for the Space Invaders environment, the result is almost the same as the original Rainbow.

distribution $\{RandomCrop, RandomErase, RandomContrast, OriginalData\}$. This distribution is used to sample an augmentation for each item of a batch during the training process.

We tested model results with augmentation discretization above. Figure 2 shows the results for the top 30 and worst 30 sets from 256 experiments. The trained model for each ratio set tested over 100 evaluation episodes in the Wizard of Wor environment. The best cumulative reward 8650 showed the following augmentation ratio: contrast 33%, crop 100%, erase 100%, original 66% (normalized: 0.11, 0.33, 0.33, 0.22). From these results, one can conclude that a properly augmented model can achieve a better reward score than the original model while testing in the environment. Also, different proportions of augmentation types lead to different model quality. For example, for the Wizard of Wor environment, we got that the higher rate of Random Contrast and low rates of other augmentations leads to lower model quality. We also inspect the correlation between the obtained reward score and model accuracy. For all 256 experiments model accuracy varies not much (in the interval: 0.83 .. 0.86). Despite this, the model reward varies a lot, so the small perturbation in model accuracy leads to a significant difference in the reward score. Thus in our experiment, the mixture of all discussed augmentations types with a higher proportion of Erase and Crop Augmentations and a lower proportion of Contrast showed the best result.

Behavioral cloning experiments can be considered as a part of the search for an approximation of an adaptable regularization function described in Sect. 4.2. The experiment can be continued with subsequent model training using Reinforcement Learning algorithms and applying a new adaptable regularization function. For example, the best augmentations distribution found in this section can be applied to DQN from Demonstrations (DQfD) algorithm [7], where some expert's demonstrations produced by human or another well-trained agent also can be used in DQN algorithm and could significantly speed up the training

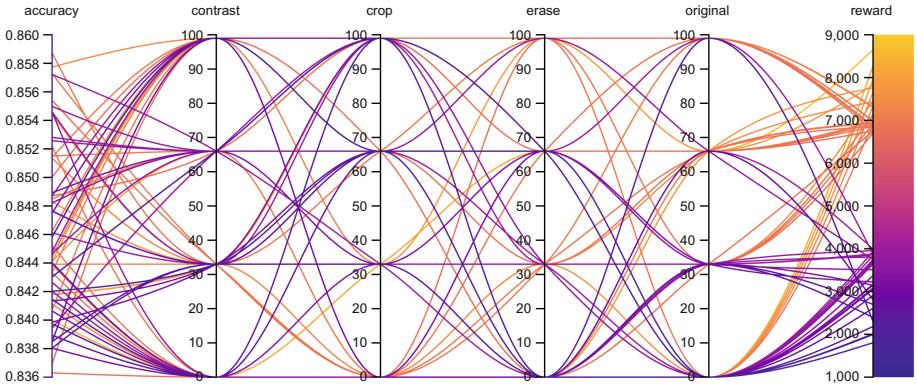

Fig. 2. Hyperparameter search for behavioral cloning on Wizard of Wor environment. We grid search over the ratio of frames for different augmentation type in training batch, considering the following percentages: 0%, 33%, 66%, 100%. For visibility purposes, this chart shows the results only for the top 30 and worst 30 sets from 256 experiments. The trained model for each ratio set tested over 100 evaluation episodes in the environment. The best cumulative reward 8650 showed the following augmentation ratio: contrast 33%, crop 100%, erase 100%, original 66% (normalized: 0.11, 0.33, 0.33, 0.22). The leftmost column reports validation accuracy on test data consisting of 20,000 frames.

process. Such experiments can include applying adaptable regularization function during pre-train or fine-tuning stages and will be the next step of our research.

6 Conclusion

Data augmentation methods have proven to be effective in image analysis. In this paper, we have applied a number of well-known augmentation techniques to the problem of Reinforcement Learning with image-based observations. We have developed an adaptive version of data augmentation for off-policy algorithms that use replay buffer as temporary memory. We have shown that augmentation improves the quality of one of the well-known state-of-the-art Rainbow algorithm. We conducted an experimental study on the selection of hyperparameters for our method of augmented data mixing. In future work, we plan to develop this method and conduct experiments with algorithms that use demonstrations.

Acknowledgements. The reported study was supported by the Ministry of Education and Science of the Russian Federation, project No. 075-15-2020-799.

References

1. Bellemare, M.G., Dabney, W., Munos, R.: A distributional perspective on reinforcement learning. arXiv preprint arXiv:1707.06887 (2017)

2. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation policies from data. arXiv preprint arXiv:1805.09501 (2018)
3. De Asis, K., Hernandez-Garcia, J.F., Holland, G.Z., Sutton, R.S.: Multi-step reinforcement learning: A unifying algorithm. arXiv preprint arXiv:1703.01327 (2017)
4. Fortunato, M., et al.: Noisy networks for exploration. arXiv preprint arXiv:1706.10295 (2017)
5. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016). http://www.deeplearningbook.org
6. Hessel, M., et al.: Rainbow: Combining improvements in deep reinforcement learning. arXiv preprint arXiv:1710.02298 (2017)
7. Hester, T., et al.: Deep q-learning from demonstrations. arXiv preprint arXiv:1704.03732 (2017)
8. Kostrikov, I., Yarats, D., Fergus, R.: Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. arXiv preprint arXiv:2004.13649 (2020)
9. Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., Srinivas, A.: Reinforcement learning with augmented data. arXiv preprint arXiv:2004.14990 (2020)
10. 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)
11. Perez, L., Wang, J.: The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621 (2017)
12. Raileanu, R., Goldstein, M., Yarats, D., Kostrikov, I., Fergus, R.: Automatic data augmentation for generalization in deep reinforcement learning. arXiv preprint arXiv:2006.12862 (2020)
13. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. arXiv preprint arXiv:1511.05952 (2015)
14. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
15. Shorten, C., Khoshgoftaar, T.M.: A survey on image data augmentation for deep learning. J. Big Data **6**(1), 60 (2019)
16. Skrynnik, A., Staroverov, A., Aitygulov, E., Aksenov, K., Davydov, V., Panov, A.I.: Forgetful experience replay in hierarchical reinforcement learning from expert demonstrations. Knowledge-Based Systems 218, 106844 (2021), https://linkinghub.elsevier.com/retrieve/pii/S0950705121001076 https://arxiv.org/abs/2006.09939
17. Skrynnik, A., Staroverov, A., Aitygulov, E., Aksenov, K., Davydov, V., Panov, A.I.: Hierarchical Deep Q-Network from imperfect demonstrations in Minecraft. Cognitive Syst. Res. **65**, 74–78 (2021). https://arxiv.org/pdf/1912.08664.pdfwww.sciencedirect.com/science/article/pii/S1389041720300723?via%3Dihubwww.scopus.com/record/display.uri?eid=2-s2.0-85094320898&origin=resultslistlinkinghub.elsevier.com/retrieve/pii/S138904172
18. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. arXiv preprint arXiv:1509.06461 (2015)
19. Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., Freitas, N.: Dueling network architectures for deep reinforcement learning. In: International Conference on Machine Learning, pp. 1995–2003 (2016)
20. Watkins, C.J., Dayan, P.: Q-learning. Mach. Learn. **8**(3–4), 279–292 (1992)