Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels

Ilya Kostrikov [*1] **Denis Yarats** [*1 2] **Rob Fergus** [1]

Abstract

We propose a simple data augmentation technique that can be applied to standard model-free reinforcement learning algorithms, enabling robust learning directly from pixels without the need for auxiliary losses or pre-training. The approach leverages input perturbations commonly used in computer vision tasks to transform input examples, as well as regularizing the value function and policy. Existing model-free approaches, such as Soft Actor-Critic (SAC) (Haarnoja et al., 2018), are not able to train deep networks effectively from image pixels. However, the addition of our augmentation method dramatically improves SAC's performance, enabling it to reach state-of-the-art performance on the DeepMind control suite, surpassing model-based (Hafner et al., 2019; Lee et al., 2019; Hafner et al., 2018) methods and recently proposed contrastive learning (Srinivas et al., 2020). Our approach, which we dub **DrQ**: **D**ata-**r**egularized **Q**, can be combined with any model-free reinforcement learning algorithm. We further demonstrate this by applying it to DQN (Mnih et al., 2013) and significantly improve its data-efficiency on the Atari 100k (Kaiser et al., 2019) benchmark.

1. Introduction

Sample-efficient deep reinforcement learning (RL) algorithms capable of directly training from image pixels would open up many real-world applications in control and robotics. However, simultaneously training a convolutional encoder alongside a policy network is challenging when given limited environment interaction, strong correlation between samples and a typically sparse reward signal. Naive attempts to use a large capacity encoder result in severe over-fitting (see Figure 1a) and smaller encoders produce impoverished representations that limit task performance.

Data augmentation methods have proven highly effective in vision and speech domains to deal with over-fitting, where output-invariant perturbations can easily be applied to the labeled input examples. Surprisingly, data augmentation has received relatively little attention in the RL community, and this is the focus of this paper. The key idea is to use standard image transformations to peturb input observations, as well as regularizing the Q-function learned by the critic so that different transformations of the same input image have similar Q-function values. No further modifications to standard actor-critic algorithms are required, obviating the need for additional losses, e.g. based on auto-encoders (Yarats et al., 2019), dynamics models (Hafner et al., 2018; 2019), or contrastive loss terms (Srinivas et al., 2020).

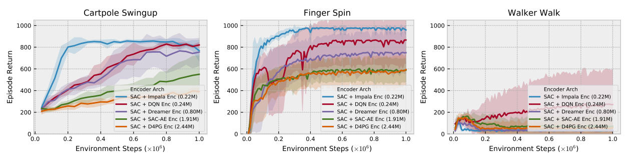
The paper makes the following contributions: (i) we demonstrate how straightforward image augmentation, applied to pixel observations, greatly reduces over-fitting in sample-efficient RL settings, without requiring any change to the underlying RL algorithm. (ii) exploiting MDP structure, we introduce two simple mechanisms for regularizing the value function which are generally applicable in the context of model-free off-policy RL. (iii) Combined with vanilla SAC (Haarnoja et al., 2018) and using hyper-parameters fixed across all tasks, the overall approach obtains state-of-the-art performance on the DeepMind control suite (Tassa et al., 2018). (iv) Combined with a DQN-like agent, the approach also obtains state-of-the-art performance on the Atari 100k benchmark. (v) It is thus the first effective approach able to train directly from pixels without the need for unsupervised auxiliary losses or a world model.

2. Sample Efficient RL from Pixels

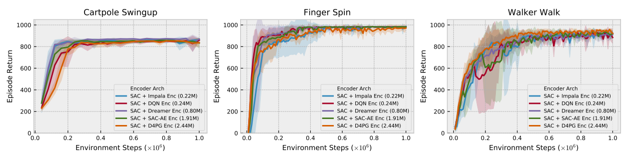
This work focuses on the data-efficient regime, seeking to optimize performance given limited environment interaction. In Figure 1a we show a motivating experiment that demonstrates over-fitting to be a significant issue in this scenario. Using three tasks from the DeepMind control

[*]Equal contribution, author ordering determined by coin flip. [1]New York Univeristy [2]Facebook AI Research. Correspondence to: Ilya Kostrikov <kostrikov@cs.nyu.edu>, Denis Yarats <denisyarats@cs.nyu.edu>.

Accepted at Workshop on Inductive Biases, Invariances and Generalization, International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).



(a) Unmodified SAC.



(b) SAC with image shift augmentation.

Figure 1: The performance of SAC trained from pixels on the DeepMind control suite using image encoder networks of different capacity (network architectures taken from recent RL algorithms, with parameter count indicated). **(a)**: unmodified SAC. Task performance can be seen to get *worse* as the capacity of the encoder increases, indicating over-fitting. **(b)**: SAC combined with image augmentation in the form of random shifts. The task performance is now similar for all architectures, regardless of their capacity. There is also a clear performance improvement relative to (a).

suite (Tassa et al., 2018), SAC (Haarnoja et al., 2018) is trained with the same policy network architecture but using different image encoder architectures, taken from the following RL approaches: NatureDQN (Mnih et al., 2013), Dreamer (Hafner et al., 2019), Impala (Espeholt et al., 2018), SAC-AE (Yarats et al., 2019) (also used in CURL (Srinivas et al., 2020)), and D4PG (Barth-Maron et al., 2018). The encoders vary significantly in their capacity, with parameter counts ranging from 220k to 2.4M. The curves show that *performance decreases as parameter count increases*, a clear indication of over-fitting.

2.1. Image Augmentation

A range of successful image augmentation techniques to counter over-fitting have been developed in computer vision (Ciregan et al., 2012; Ciresan et al., 2011; Simard et al., 2003; Krizhevsky et al., 2012; Chen et al., 2020). These apply transformations to the input image for which the task labels are invariant, e.g. for object recognition tasks, image flips and rotations do not alter the semantic label. However, tasks in RL differ significantly from those in vision and in many cases the reward would not be preserved by these transformations. We examine several common image transformations from (Chen et al., 2020) in Appendix B and conclude that random shifts strike a good balance between simplicity and performance, we therefore limit our choice of augmentation to this transformation.

Figure 1b shows the results of this augmentation applied during SAC training. We apply data augmentation only to the images sampled from the replay buffer and not for samples collection procedure. The images from the DeepMind control suite are 84×84. We pad each side by 4 pixels (by repeating boundary pixels) and then select a random 84×84 crop, yielding the original image shifted by ± 4 pixels. This procedure is repeated every time an image is sampled from the replay buffer. The plots show overfitting is greatly reduced, closing the performance gap between the encoder architectures. These random shifts alone enable SAC to achieve competitive absolute performance, without the need for auxiliary losses.

2.2. Optimality Invariant Image Transformations

While the image augmentation described above is effective, it does not fully exploit the MDP structure inherent in RL tasks. We now introduce a general framework for regularizing the value function through transformations of the input state. For a given task, we define an optimality invariant state transformation $f : \mathcal{S} \times \mathcal{T} \rightarrow \mathcal{S}$ as a mapping that preserves the Q-values

$$Q(s,a) = Q(f(s,\nu),a) \text{ for all } s \in \mathcal{S}, a \in \mathcal{A} \text{ and } \nu \in \mathcal{T}.$$

where ν are the parameters of $f(\cdot)$, drawn from the set of all possible parameters $\mathcal{T}$. One example of such transformations are the random image translations successfully applied in the previous section.

For every state, the transformations allow the generation of several surrogate states with the same Q-values, thus providing a mechanism to reduce the variance of Q-function estimation. In particular, for an arbitrary distribution of states $\mu(\cdot)$ and policy π, instead of using a single sample $s^* \sim \mu(\cdot)$, $a^* \sim \pi(\cdot|s^*)$ estimation of the expectation

$$\mathop{\mathbb{E}}_{\substack{s\sim\mu(\cdot)\\ a\sim\pi(\cdot|s)}} [Q(s,a)] \approx Q(s^*,a^*)$$

we can instead generate K samples via random transformations and obtain an estimate with lower variance

$$\mathop{\mathbb{E}}_{\substack{s\sim\mu(\cdot)\\ a\sim\pi(\cdot|s)}} [Q(s,a)] \approx \frac{1}{K}\sum_{k=1}^{K} Q(f(s^*,\nu_k),a_k)$$

where $\nu_k \in \mathcal{T}$ and $a_k \sim \pi(\cdot|f(s^*,\nu_k))$. This suggests two distinct ways to regularize Q-function. First, we use the data augmentation to compute the target values for every transition tuple (s_i, a_i, r_i, s'_i) as

$$y_i = r_i + \gamma \frac{1}{K}\sum_{k=1}^{K} Q_\theta(f(s'_i,\nu'_{i,k}),a'_{i,k}) \qquad (1)$$

where $a'_{i,k} \sim \pi(\cdot|f(s'_i,\nu'_{i,k}))$ and $\nu'_{i,k} \in \mathcal{T}$ corresponds to a transformation parameter of s'_i. Then the Q-function is updated using these targets through an SGD update using learning rate λ_θ

$$\theta \leftarrow \theta - \lambda_\theta \nabla_\theta \frac{1}{N} \sum_{i=1}^{N} (Q_\theta(f(s_i, \nu_i), a_i) - y_i)^2. \quad (2)$$

In tandem, we note that the same target from Equation (1) can be used for different augmentations of s_i, resulting in the second regularization approach

$$\theta \leftarrow \theta - \lambda_\theta \nabla_\theta \frac{1}{NM} \sum_{i=1,m=1}^{N,M} (Q_\theta(f(s_i, \nu_{i,m}), a_i) - y_i)^2. \quad (3)$$

When both regularization methods are used, $\nu_{i,m}$ and $\nu'_{i,k}$ are drawn independently.

2.3. Our approach: Data-regularized Q (DrQ)

Our approach, **DrQ**, is the union of the three separate regularization mechanisms introduced above: (i) transformations of the input image (Section 2.1); (ii) averaging the Q target over K image transformations (Equation (1)) and (iii) averaging the Q function itself over M image transformations (Equation (3)). Note that if [K=1,M=1] then **DrQ** reverts to image transformations.

For the experiments, we pair **DrQ** with SAC (Haarnoja et al., 2018) and DQN (Mnih et al., 2013), popular model-free algorithms for control in continuous and discrete action spaces respectively. We select image shifts as the class of image transformations f, with $\nu \pm 4$, as explained in Section 2.1. For target Q and Q augmentation we use [K=2,M=2] respectively. Figure 2 shows **DrQ** and ablated versions, demonstrating clear gains over unaugmented SAC.

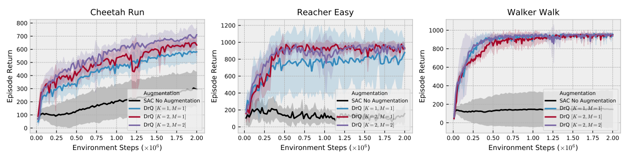


Figure 2: Different combinations of our three regularization techniques on tasks from (Tassa et al., 2018) using SAC. Black: standard SAC. Blue: **DrQ** [K=1,M=1], SAC augmented with random shifts. Red: **DrQ** [K=2,M=1], random shifts + Target Q augmentations. Purple: **DrQ** [K=2,M=2], random shifts + Target Q + Q augmentations.

3. Experiments

In this section we evaluate our algorithm (**DrQ**) on the two commonly used benchmarks based on the DeepMind control suite (Tassa et al., 2018), namely the PlaNet (Hafner et al., 2018) and Dreamer (Hafner et al., 2019) setups. Throughout these experiments all hyper-parameters of the algorithm are kept fixed: the actor and critic neural networks are trained using the Adam optimizer (Kingma & Ba, 2014) with default parameters and a mini-batch size of 512. For SAC, the soft target update rate τ is 0.01, initial temperature is 0.1, and target network and the actor updates are made every 2 critic updates (as in (Yarats et al., 2019)). We use the image encoder architecture from SAC-AE (Yarats et al., 2019) and follow their training procedure. Following (Henderson et al., 2018), the models are trained using 10 different seeds; for every seed the mean episode returns are computed every 10000 environment steps, averaging over 10 episodes. All figures plot the mean performance over the 10 seeds, together with $\pm$ 1 standard deviation shading. We compare our **DrQ** approach to leading model-free and model-based approaches: PlaNet (Hafner et al., 2018), SAC-AE (Yarats et al., 2019), SLAC (Lee et al., 2019), CURL (Srinivas et al., 2020) and Dreamer (Hafner et al., 2019).

3.1. DeepMind Control Suite Experiments

PlaNet benchmark (Hafner et al., 2018) consists of six challenging control tasks from (Tassa et al., 2018) with different traits. The benchmark specifies a different action-repeat hyper-parameter for each of the six tasks Following common practice (Hafner et al., 2018; Lee et al., 2019; Yarats et al., 2019; Mnih et al., 2013), we report the performance using true environment steps, thus are invariant to the action-repeat hyper-parameter. Aside from action-repeat, all other hyper-parameters of our algorithm are fixed across the six tasks, using the values previously detailed.

Figure 3 compares **DrQ** [K=2,M=2] to PlaNet (Hafner et al., 2018), SAC-AE (Yarats et al., 2019), CURL (Srinivas et al., 2020), SLAC (Lee et al., 2019), and an upper bound performance provided by SAC (Haarnoja et al., 2018) that directly learns from internal states. We use the version of SLAC that performs one gradient update per an environment step to ensure a fair comparison to other approaches. **DrQ** achieves state-of-the-art performance on this benchmark on all the tasks, despite being much simpler than other methods. Furthermore, since **DrQ** does not learn a model (Hafner et al., 2018; Lee et al., 2019) or any auxiliary tasks (Srinivas et al., 2020), the wall clock time also compares favorably to the other methods.

3.2. Atari 100k Experiments

We evaluate **DrQ** [K=1,M=1] on the recently introduced Atari 100k (Kaiser et al., 2019) benchmark – a sample-constrained evaluation for discrete control algorithms. The underlying RL approach to which **DrQ** is applied is a DQN, combined with double Q-learning (van Hasselt et al., 2015), n-step returns (Mnih et al., 2016), and dueling critic architecture (Wang et al., 2015). As per common practice (Kaiser et al., 2019; van Hasselt et al., 2019), we evaluate our agent

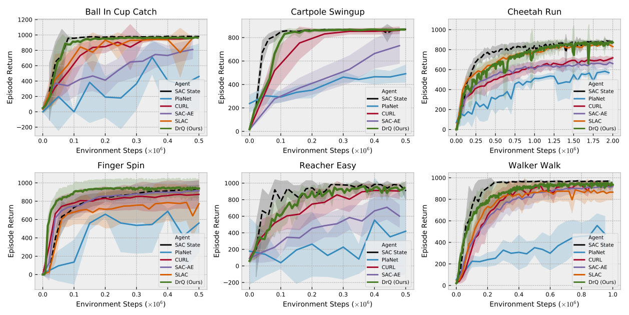


Figure 3: The PlaNet benchmark. Our algorithm (**DrQ** [K=2,M=2]) outperforms the other methods and demonstrates the state-of-the-art performance. Furthermore, on several tasks **DrQ** is able to match the upper-bound performance of SAC trained directly on internal state, rather than images. Finally, our algorithm not only shows improved sample-efficiency relative to other approaches, but is also faster in terms of wall clock time.

for 125k environment steps at the end of training and average its performance over 5 random seeds. Figure 4 shows the median human-normalized episode returns performance (as in (Mnih et al., 2013)) of the underlying model, which we refer to as Efficient DQN, in pink. When **DrQ** is added there is a significant increase in performance (cyan), surpassing OTRainbow (Kielak, 2020) and Data Efficient Rainbow (van Hasselt et al., 2019). **DrQ** is also superior to CURL (Srinivas et al., 2020) that uses an auxiliary loss built on top of a hybrid between OTRainbow and Efficient rainbow. **DrQ** combined with Efficient DQN thus achieves state-of-the-art performance, despite being significantly simpler than the other approaches.

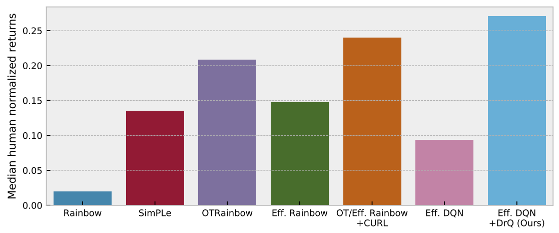


Figure 4: The Atari 100k benchmark. Compared to a set of leading baselines, our method (**DrQ** [K=1,M=1], combined with Efficient DQN) achieves the state-of-the-art performance, despite being much simpler. Note the large improvement that results from adding **DrQ** to Efficient DQN (pink vs cyan). By contrast, the gains from CURL, that utilizes tricks from both Data Efficient Rainbow and OTRainbow, are more modest over the underlying RL methods.

4. Related Work

Computer Vision Data augmentation via image transformations has been used to improve generalization since the inception of convolutional networks (Becker & Hinton, 1992; Simard et al., 2003; LeCun et al., 1989; Ciresan et al., 2011; Ciregan et al., 2012). Following AlexNet (Krizhevsky et al., 2012), they have become a standard part of training pipelines. For object classification tasks, the transformations are selected to avoid changing the semantic category, i.e. translations, scales, color shifts, etc. Perturbed versions of input examples are used to expand the training set and no adjustment to the training algorithm is needed. While a similar set of transformations are potentially applicable to control tasks, the RL context does require modifications to be made to the underlying algorithm.

Regularization in RL Some early attempts to learn RL function approximators used ℓ_2 regularization of the Q (Farahmand et al., 2008; Yan et al., 2017) function. Another approach is entropy regularization (Ziebart et al., 2008; Haarnoja et al., 2018; Nachum et al., 2017; Williams & Peng, 1991), where causal entropy is added to the rewards, making the Q-function smoother and facilitating optimization (Ahmed et al., 2018). Prior work has explored regularization of the neural network approximator in deep RL, e.g. using dropout (Farebrother et al., 2018) and cutout (Cobbe et al., 2018) techniques. See (Liu et al., 2019) for a comprehensive evaluation of different network regularization methods. In contrast, our approach directly regularizes the Q-function in a data-driven way that incorporates knowledge of task invariances, as opposed to generic priors.

Continuous Control from Pixels There are a variety of methods addressing the sample-efficiency of RL algorithms that directly learn from pixels. The most prominent approaches for this can be classified into two groups, model-based and model-free methods. The model-based methods attempt to learn the system dynamics in order to acquire a compact latent representation of high-dimensional observations to later perform policy search (Hafner et al., 2018; Lee et al., 2019; Hafner et al., 2019). In contrast, the model-free methods either learn the latent representation indirectly by optimizing the RL objective (Barth-Maron et al., 2018; Abdolmaleki et al., 2018) or by employing auxiliary losses that provide additional supervision (Yarats et al., 2019; Srinivas et al., 2020; Sermanet et al., 2018; Dwibedi et al., 2018). Our approach is complementary to these methods and can be combined with them to improve performance.

5. Conclusion

We have introduced a simple regularization technique that significantly improves the performance of SAC trained directly from image pixels on standard continuous control tasks. Our method is easy to implement and adds a negligible computational burden. We compared our method to state-of-the-art approaches on both DeepMind control suite, where we demonstrated that it outperforms them on the majority of tasks, and Atari 100k benchmarks, where it outperforms other methods in the median metric. Furthermore, we demonstrate the method to be robust to the choice of hyper-parameters.

References

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

Ahmed, Z., Roux, N. L., Norouzi, M., and Schuurmans, D. Understanding the impact of entropy on policy optimization. *arXiv preprint arXiv:1811.11214*, 2018.

Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., TB, D., Muldal, A., Heess, N., and Lillicrap, T. Distributional policy gradients. In *International Conference on Learning Representations*, 2018.

Becker, S. and Hinton, G. E. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 1992.

Bellman, R. A markovian decision process. *Indiana Univ. Math. J.*, 1957.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

Ciregan, D., Meier, U., and Schmidhuber, J. Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3642–3649, 2012.

Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., and Schmidhuber, J. High-performance neural networks for visual object classification. *arXiv preprint arXiv:1102.0183*, 2011.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. Quantifying generalization in reinforcement learning. *arXiv preprint arXiv:1812.02341*, 2018.

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Dwibedi, D., Tompson, J., Lynch, C., and Sermanet, P. Learning actionable representations from visual observations. *CoRR*, 2018.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

Farahmand, A., Ghavamzadeh, M., Szepesvari, C., and Manor, S. Regularized policy iteration. In *NIPS*, 2008.

Farebrother, J., Machado, M. C., and Bowling, M. Generalization and regularization in dqn. *arXiv abs/1810.00123*, 2018.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmassan, Stockholm, Sweden, July 10-15, 2018*, 2018.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. *Thirty-Second AAAI Conference On Artificial Intelligence (AAAI)*, 2018.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 1998.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Sepassi, R., Tucker, G., and Michalewski, H. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

Kielak, K. P. Do recent advancements in model-based deep reinforcement learning really improve data efficiency? *openreview*, 2020. URL `https://openreview.net/forum?id=Bke9u1HFwB`.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989.

Lee, A. X., Nagabandi, A., Abbeel, P., and Levine, S. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv e-prints*, 2019.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *CoRR*, 2015.

Liu, Z., Li, X., Kang, B., and Darrell, T. Regularization matters in policy optimization. *arXiv abs/1910.09191*, 2019.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv e-prints*, 2013.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. *CoRR*, 2016.

Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, 2017.

Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., Levine, S., and Brain, G. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1134–1141. IEEE, 2018.

Simard, P. Y., Steinkraus, D., Platt, J. C., et al. Best practices for convolutional neural networks applied to visual document analysis. In *Icdar*, 2003.

Srinivas, A., Laskin, M., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020.

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.

van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. *arXiv e-prints*, 2015.

van Hasselt, H., Hessel, M., and Aslanides, J. When to use parametric models in reinforcement learning? *arXiv preprint arXiv:1906.05243*, 2019.

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., and De Freitas, N. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

Williams, R. J. and Peng, J. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 1991.

Yan, X., Choromanski, K., Boots, B., and Sindhwani, V. Manifold regularization for kernelized lstd. *arXiv abs/1710.05387*, 2017.

Yarats, D., Zhang, A., Kostrikov, I., Amos, B., Pineau, J., and Fergus, R. Improving sample efficiency in model-free reinforcement learning from images. *arXiv preprint arXiv:1910.01741*, 2019.

Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, 2008.

Appendix

A. Background

Reinforcement Learning from Images We formulate image-based control as an infinite-horizon partially observable Markov decision process (POMDP) (Bellman, 1957; Kaelbling et al., 1998). An POMDP can be described as the tuple $(\mathcal{O}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{O}$ is the high-dimensional observation space (image pixels), $\mathcal{A}$ is the action space, the transition dynamics $p = Pr(o'_t | o_{\leq t}, a_t)$ capture the probability distribution over the next observation o'_t given the history of previous observations $o_{\leq t}$ and current action a_t, $r : \mathcal{O} \times \mathcal{A} \to \mathbb{R}$ is the reward function that maps the current observation and action to a reward $r_t = r(o_{\leq t}, a_t)$, and $\gamma \in [0, 1)$ is a discount factor. Per common practice (Mnih et al., 2013), throughout the paper the POMDP is converted into an MDP (Bellman, 1957) by stacking several consecutive image observations into a state $s_t = \{o_t, o_{t-1}, o_{t-2}, \ldots\}$. For simplicity we redefine the transition dynamics $p = Pr(s'_t | s_t, a_t)$ and the reward function $r_t = r(s_t, a_t)$. We then aim to find a policy $\pi(a_t|s_t)$ that maximizes the cumulative discounted return $\mathbb{E}_\pi[\sum_{t=1}^{\infty} \gamma^t r_t | a_t \sim \pi(\cdot|s_t), s'_t \sim p(\cdot|s_t, a_t), s_1 \sim p(\cdot)]$.

Soft Actor-Critic The Soft Actor-Critic (SAC) (Haarnoja et al., 2018) learns a state-action value function Q_θ, a stochastic policy π_θ and a temperature α to find an optimal policy for an MDP $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$ by optimizing a γ-discounted maximum-entropy objective (Ziebart et al., 2008). θ is used generically to denote the parameters updated through training in each part of the model. The actor policy $\pi_\theta(a_t|s_t)$ is a parametric $\tanh$-Gaussian that given s_t samples $a_t = \tanh(\mu_\theta(s_t) + \sigma_\theta(s_t)\epsilon)$, where $\epsilon \sim \mathcal{N}(0, 1)$ and μ_θ and σ_θ are parametric mean and standard deviation.

The policy evaluation step learns the critic $Q_\theta(s_t, a_t)$ network by optimizing a single-step of the soft Bellman residual

$$\begin{aligned} J_Q(\mathcal{D}) &= \mathbb{E}_{\substack{(s_t, a_t, s'_t) \sim \mathcal{D} \\ a'_t \sim \pi(\cdot|s'_t)}}[(Q_\theta(s_t, a_t) - y_t)^2] \\ y_t &= r(s_t, a_t) + \gamma[Q_{\theta'}(s'_t, a'_t) - \alpha \log \pi_\theta(a'_t|s'_t)], \end{aligned}$$

where $\mathcal{D}$ is a replay buffer of transitions, θ' is an exponential moving average of the weights as done in (Lillicrap et al., 2015). SAC uses clipped double-Q learning (van Hasselt et al., 2015; Fujimoto et al., 2018), which we omit from our notation for simplicity but employ in practice.

The policy improvement step then fits the actor policy $\pi_\theta(a_t|s_t)$ network by optimizing the objective

$$J_\pi(\mathcal{D}) = \mathbb{E}_{s_t \sim \mathcal{D}}[D_{\mathrm{KL}}(\pi_\theta(\cdot|s_t) || \exp\{\frac{1}{\alpha} Q_\theta(s_t, \cdot)\})].$$

Finally, the temperature α is learned with the loss

$$J_\alpha(\mathcal{D}) = \mathbb{E}_{\substack{s_t \sim \mathcal{D} \\ a_t \sim \pi_\theta(\cdot|s_t)}} [-\alpha \log \pi_\theta(a_t|s_t) - \alpha \bar{\mathcal{H}}],$$

where $\bar{\mathcal{H}} \in \mathbb{R}$ is the target entropy hyper-parameter that the policy tries to match, which in practice is usually set to $\bar{\mathcal{H}} = -|\mathcal{A}|$.

Deep Q-learning DQN (Mnih et al., 2013) also learns a convolutional neural net to approximate Q-function over states and actions. The main difference is that DQN operates on discrete actions spaces, thus the policy can be directly inferred from Q-values. The parameters of DQN are updated by optimizing the squared residual error

$$J_Q(\mathcal{D}) = \mathbb{E}_{(s_t,a_t,s'_t)\sim\mathcal{D}}[(Q_\theta(s_t,a_t) - y_t)^2]$$
$$y_t = r(s_t,a_t) + \gamma \max_{a'} Q_{\theta'}(s'_t,a').$$

In practice, the standard version of DQN is frequently combined with a set of tricks that improve performance and training stability, wildly known as Rainbow (van Hasselt et al., 2015).

B. Image Augmentations Ablation

Following (Chen et al., 2020), we evaluate popular image augmentation techniques, namely random shifts, cutouts, vertical and horizontal flips, random rotations and image-wise intensity jittering. Below, we provide a comprehensive overview of each augmentation. Furthermore, we examine effectiveness of these techniques in Figure 5.

Random Shift We bring our attention to random shifts that are commonly used to regularize neural networks trained on small images (Becker & Hinton, 1992; Simard et al., 2003; LeCun et al., 1989; Ciresan et al., 2011; Ciregan et al., 2012). In our implementation of this method images of size 84×84 are padded each side by 4 pixels (by repeating boundary pixels) and then randomly cropped back to the original 84×84 size.

Cutout Cutouts introduced in (DeVries & Taylor, 2017) represent a generalization of Dropout (Hinton et al., 2012). Instead of masking individual pixels cutouts mask square regions. Since image pixels can be highly correlated, this technique is proven to improve training of neural networks.

Horizontal/Vertical Flip This technique simply flips an image either horizontally or vertically with probability 0.1.

Rotate Here, an image is rotated by r degrees, where r is uniformly sampled from $[-5, -5]$.

Intensity Each $N \times C \times 84 \times 84$ image tensor is multiplied by a single scalar s, which is computed as $s = \mu + \sigma \cdot \text{clip}(r, -2, 2)$, where $r \sim \mathcal{N}(0, 1)$. For our experiments we use $\mu = 1.0$ and $\sigma = 0.05$.

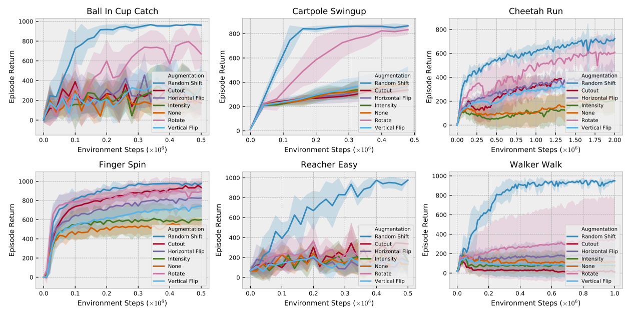


Figure 5: Various image augmentations have different effect on the agent's performance. Overall, we conclude that using image augmentations helps to fight overfitting. Moreover, we notice that random shifts proven to be the most effective technique for tasks from the DeepMind control suite.

C. Algorithm

In this section we present our algorithm in Algorithm 1.

D. Data-efficiency Experiments

In Table 1 we compare performance of our method given at a fixed number of environment interactions (e.g. 100k and 500k).

E. Additional DeepMind Control Suite Experiments

Dreamer Benchmark is a more extensive testbed that was introduced in Dreamer (Hafner et al., 2019), featuring a diverse set of tasks from the DeepMind control suite. Tasks involving sparse reward were excluded (e.g. Acrobot and Quadruped) since they require modification of SAC to incorporate multi-step returns (Barth-Maron et al., 2018), which is beyond the scope of this work. We evaluate on the remaining 15 tasks, fixing the action-repeat hyper-parameter to 2, as in Dreamer (Hafner et al., 2019).

We compare **DrQ** [K=2,M=2] to Dreamer (Hafner et al., 2019) and the upper-bound performance of SAC (Haarnoja et al., 2018) from states[1]. Again, we keep all the hyper-parameters of our algorithm fixed across all the tasks. In Figure 6, **DrQ** demonstrates the state-of-the-art results by collectively outperforming Dreamer (Hafner et al., 2019), al-

[1]No other publicly reported results are available for the other methods due to the recency of the Dreamer (Hafner et al., 2019) benchmark.

Algorithm 1 DrQ: **D**ata-**r**egularized **Q** applied to a generic off-policy actor critic algorithm.
Black: unmodified off-policy actor-critic.
Orange: image transformation.
Green: target Q augmentation.
Blue: Q augmentation.

```
**Hyperparameters:** Total number of environment steps $T$, mini-batch size $N$, learning rate $\lambda_\theta$, target network update rate $\tau$, image transformation $f$, number of target $Q$ augmentations $K$, number of $Q$ augmentations $M$.
for each timestep $t = 1..T$ do
    $a_t \sim \pi(\cdot|s_t)$
    $s'_t \sim p(\cdot|s_t, a_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup (s_t, a_t, r(s_t, a_t), s'_t)$
    UPDATECRITIC($\mathcal{D}$)
    UPDATEACTOR($\mathcal{D}$)                ▷ Data augmentation is applied to the samples for actor training as well.
end for
procedure UPDATECRITIC($\mathcal{D}$)
    $\{(s_i, a_i, r_i, s'_i)\}_{i=1}^N \sim \mathcal{D}$                ▷ Sample a mini batch
    $\left\{\nu'_{i,k} \middle| \nu'_{i,k} \sim \mathcal{U}(\mathcal{T}), i = 1..N, k = 1..K\right\}$                ▷ Sample parameters of target augmentations
    for each $i = 1..N$ do
        $a'_i \sim \pi(\cdot|s'_i)$ or $a'_{i,k} \sim \pi(\cdot| f(s'_i, \nu'_{i,k})), k = 1..K$
        $\hat{Q}_i = Q_{\theta'}(s'_i, a'_i)$ or $\hat{Q}_i = \frac{1}{K}\sum_{k=1}^K Q_{\theta'}(f(s'_i, \nu'_{i,k}), a'_{i,k})$
        $y_i \leftarrow r(s_i, a_i) + \gamma \hat{Q}_i$
    end for
    $\{\nu_{i,m}|\nu_{i,m} \sim \mathcal{U}(\mathcal{T}), i = 1..N, m = 1..M\}$                ▷ Sample parameters of Q augmentations
    $J_Q(\theta) = \frac{1}{N}\sum_{i=1}^N (Q_\theta(s_i, a_i) - y_i)^2$ or $J_Q(\theta) = \frac{1}{NM}\sum_{i,m=1}^{N,M}(Q_\theta(f(s_i, \nu_{i,m}), a_i) - y_i)^2$
    $\theta \leftarrow \theta - \lambda_\theta \nabla_\theta J_Q(\theta)$                ▷ Update the critic
    $\theta' \leftarrow (1-\tau)\theta' + \tau\theta$                ▷ Update the critic target
end procedure
```

500k step scores	DrQ (Ours)	CURL	PlaNet	SAC-AE	SLAC	SAC State
Finger Spin	**938±103**	874±151	418±382	914±107	771±203	927±43
Cartpole Swingup	**868±10**	861±30	464±50	730±152	-	870±7
Reacher Easy	**942±71**	904±94	351±483	601±135	-	975±5
Cheetah Run	**660±96**	500±91	321±104	544±50	629±74	772±60
Walker Walk	**921±45**	906±56	293±114	858±82	865±97	964±8
Ball In Cup Catch	**963±9**	958±13	352±467	810±121	959±4	979±6
100k step scores						
Finger Spin	**901±104**	779±108	95±164	747±130	680±130	672±76
Cartpole Swingup	**759±92**	592±170	303±71	276±38	-	812±45
Reacher Easy	**601±213**	517±113	140±256	225±164	-	919±123
Cheetah Run	344±67	307±48	165±123	240±38	**391±47***	228±95
Walker Walk	**612±164**	344±132	125±57	395±58	428±74	604±317
Ball In Cup Catch	**913±53**	772±241	198±442	338±196	607±173	957±26

Table 1: The PlaNet benchmark at 100k and 500k environment steps. Our method (**DrQ** [K=2,M=2]) outperforms other approaches in both the data-efficient (100k) and asymptotic performance (500k) regimes. *: SLAC uses 100k exploration steps which are not counted in the reported values. By contrast, **DrQ** only uses 1000 exploration steps which are included in the overall step count.

though Dreamer is superior on 3 of the 15 tasks (Walker Run, Cartpole Swingup Sparse and Pendulum Swingup). On many tasks **DrQ** approaches the upper-bound performance of SAC (Haarnoja et al., 2018) trained directly on states.

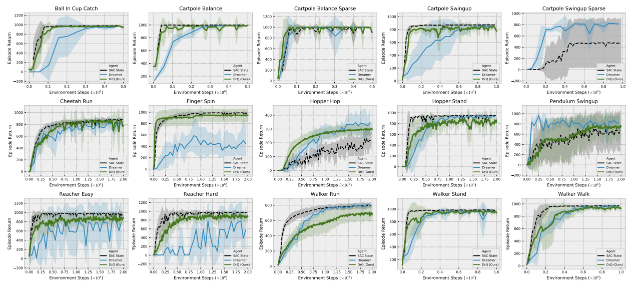


Figure 6: The Dreamer benchmark. Our method **DrQ** [K=2,M=2] again demonstrates superior performance over Dreamer on 12 out 15 selected tasks. In many cases it also reaches the upper-bound performance of SAC that learns directly from states.