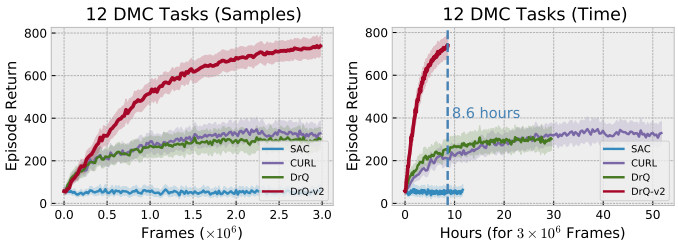
Mastering Visual Continuous Control: Improved Data-Augmented Reinforcement Learning

Denis Yarats NYU & FAIR
Rob Fergus NYU
Alessandro Lazaric FAIR
Lerrel Pinto NYU

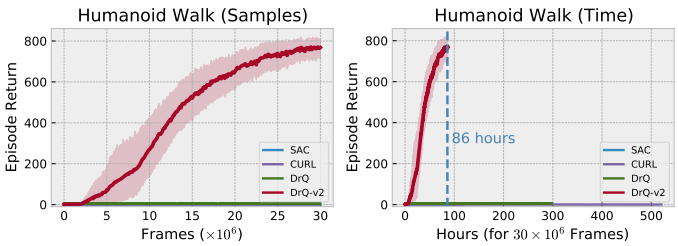
Code: `https://github.com/facebookresearch/drqv2`

Abstract

We present DrQ-v2, a model-free reinforcement learning (RL) algorithm for visual continuous control. DrQ-v2 builds on DrQ, an off-policy actor-critic approach that uses data augmentation to learn directly from pixels. We introduce several improvements that yield state-of-the-art results on the DeepMind Control Suite. Notably, DrQ-v2 is able to solve complex humanoid locomotion tasks directly from pixel observations, previously unattained by model-free RL. DrQ-v2 is conceptually simple, easy to implement, and provides significantly better computational footprint compared to prior work, with the majority of tasks taking just 8 hours to train on a single GPU. Finally, we publicly release DrQ-v2's implementation to provide RL practitioners with a strong and computationally efficient baseline.



(a) Average performance across 12 challenging DMC tasks.



(b) Performance on *Humanoid Walk*.

Figure 1: **DrQ-v2** demonstrates significantly better sample efficiency and computational footprint compared to state-of-the-art model-free methods for visual continuous control while being conceptually simple and easy to implement. (**a**) Average performance results across 12 challenging tasks from the DeepMind Control Suite (the set of tasks can be seen in Figure 4). (**b**) Performance on the *Humanoid Walk* task, previously unsolved by model-free methods. In both cases we report sample complexity and wall-clock time axes for evaluation, with time being measured on a single GPU machine and using official implementations for each method.

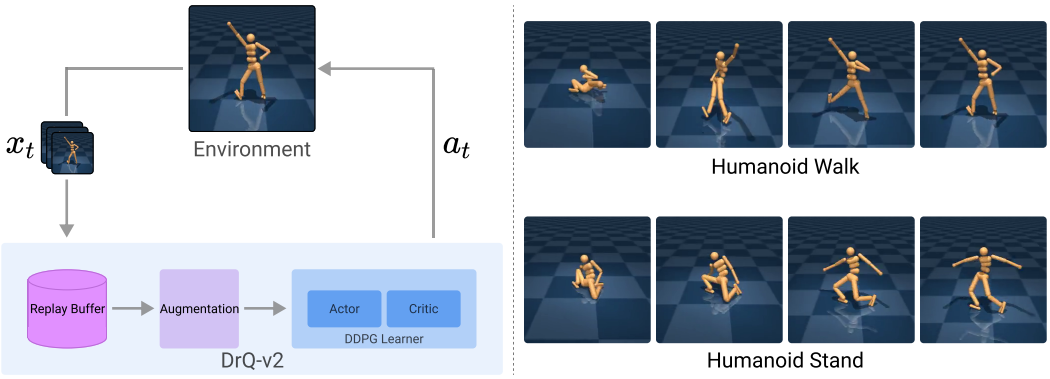


Figure 2: (Left): DrQ-v2 is an off-policy actor-critic algorithm for image-based RL. It alleviates encoder overfitting by applying random shift augmentation to pixel observations sampled from the replay buffer. (Right): Examples of walking and standing behaviors learned by DrQ-v2 for a complex humanoid agent from DMC [Tassa et al., 2018] with 21 and 54 dimensional action and state spaces, respectively. DrQ-v2 does not have access to the internal state of the environment, only observing three consecutive pixel frames at a time. Despite this imperfect observational channel, our agent still manages to solve the tasks. To the best of our knowledge, this is the first successful demonstration by a model-free method, using pixel-based inputs of these tasks.

1 Introduction

Creating sample-efficient continuous control methods that observe high-dimensional images has been a long standing challenge in reinforcement learning (RL) . Over the last three years, the RL community has made significant headway on this problem, improving sample-efficiency significantly. The key insight to solving visual control is the learning of better low-dimensional representations, either through autoencoders [Yarats et al., 2019, Finn et al., 2015], variational inference [Hafner et al., 2018, 2019, Lee et al., 2019], contrastive learning [Srinivas et al., 2020, Yarats et al., 2021a], self-prediction [Schwarzer et al., 2020b], or data augmentations [Yarats et al., 2021b, Laskin et al., 2020]. However, current state-of-the-art model-free methods are still limited in three ways. First, they are unable to solve the more challenging visual control problems such as quadruped and humanoid locomotion. Second, they often require significant computational resources, i.e. lengthy training times using distributed multi-GPU infrastructure. Lastly, it is often unclear how different design choices affect overall system performance.

In this paper we present **DrQ-v2**, a simple model-free algorithm that builds on the idea of using data augmentations [Yarats et al., 2021b, Laskin et al., 2020] to solve hard visual control problems. Most notably, it is the first model-free method that solves complex humanoid tasks directly from pixels. Compared to previous state-of-the-art model-free methods, DrQ-v2 provides significant improvements in sample efficiency across tasks from the DeepMind Control Suite [Tassa et al., 2018]. Conceptually simple, DrQ-v2 is also computationally efficient, which allows solving most tasks in DeepMind Control Suite in just 8 hours on a single GPU (see Figure 1). Recently, a model-based method, DreamerV2 [Hafner et al., 2020] was also shown to solve visual continuous control problems and it was first to solve the humanoid locomotion problem from pixels. While our model-free DrQ-v2 matches DreamerV2 in terms sample efficiency and performance, it does so $4\times$ faster in terms of wall-clock time to train. We believe this makes DrQ-v2 a more accessible approach to support research in visual continuous control and it reinforces the question on whether model-free or model-based is the more suitable approach to solve this type of tasks.

DrQ-v2, which is detailed in Section 3, improves upon DrQ [Yarats et al., 2021b] by making several algorithmic changes: (i) switching the base RL algorithm from SAC [Haarnoja et al., 2018b] to DDPG [Lillicrap et al., 2015a], (ii) this allows us straightforwardly incorporating multi-step return, (iii) adding bilinear interpolation to the random shift image augmentation, (iv) introducing an exploration schedule, (v) selecting better hyper-parameters including a larger capacity of the replay buffer. A careful ablation study of these design choices is presented in Section 4.3. Furthermore, we re-examine the original implementation of DrQ and identify several computational bottlenecks such

as replay buffer management, data augmentation processing, batch size, and frequency of learning updates (see Section 3.2). To remedy these, we have developed a new implementation that both achieves better performance and trains around 3.5 times faster with respect to wall-clock time than the previous implementation on the same hardware with an increase in environment frame throughput (FPS) from 28 to 96 (i.e., it takes $10^6/96/3600 \approx 2.9$ hours to train for 1M environment steps).

2 Background

2.1 Reinforcement Learning from Images

We formulate image-based control as an infinite-horizon Markov Decision Process (MDP) [Bellman, 1957]. Generally, in such a setting, an image rendering of the system is not sufficient to perfectly describe the system's underlying state. To this end and per common practice [Mnih et al., 2013], we approximate the current state of the system by stacking three consecutive prior observations. With this in mind, such MDP can be described as a tuple $(\mathcal{X}, \mathcal{A}, P, R, \gamma, d_0)$, where $\mathcal{X}$ is the state space (a three-stack of image observations), $\mathcal{A}$ is the action space, $P : \mathcal{X} \times \mathcal{A} \to \Delta(\mathcal{X})$ is the transition function[1] that defines a probability distribution over the next state given the current state and action, $R : \mathcal{X} \times \mathcal{A} \to [0, 1]$ is the reward function, $\gamma \in [0, 1)$ is a discount factor, and $d_0 \in \Delta(\mathcal{X})$ is the distribution of the initial state $\boldsymbol{x}_0$. The goal is to find a policy $\pi : \mathcal{X} \to \Delta(\mathcal{A})$ that maximizes the expected discounted sum of rewards $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$, where $\boldsymbol{x}_0 \sim d_0$, and $\forall t$ we have $\boldsymbol{a}_t \sim \pi(\cdot|\boldsymbol{x}_t)$, $\boldsymbol{x}_{t+1} \sim P(\cdot|\boldsymbol{x}_t, \boldsymbol{a}_t)$, and $r_t = R(\boldsymbol{x}_t, \boldsymbol{a}_t)$.

2.2 Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient (DDPG) [Lillicrap et al., 2015a] is an actor-critic algorithm for continuous control that concurrently learns a Q-function Q_θ and a deterministic policy π_ϕ. For this, DDPG uses Q-learning [Watkins and Dayan, 1992] to learn Q_θ by minimizing the one-step Bellman residual $J_\theta(\mathcal{D}) = \mathbb{E}_{(\boldsymbol{x}_t, \boldsymbol{a}_t, r_t, \boldsymbol{x}_{t+1}) \sim \mathcal{D}}[(Q_\theta(\boldsymbol{x}_t, \boldsymbol{a}_t) - r_t - \gamma Q_{\bar{\theta}}(\boldsymbol{x}_{t+1}, \pi_\phi(\boldsymbol{x}_{t+1}))^2]$. The policy π_ϕ is learned by employing Deterministic Policy Gradient (DPG) [Silver et al., 2014] and maximizing $J_\phi(\mathcal{D}) = \mathbb{E}_{\boldsymbol{x}_t \sim \mathcal{D}}[Q_\theta(\boldsymbol{x}_t, \pi_\phi(\boldsymbol{x}_t))]$, so $\pi_\phi(\boldsymbol{x}_t)$ approximates $\operatorname{argmax}_{\boldsymbol{a}} Q_\theta(\boldsymbol{x}_t, \boldsymbol{a})$. Here, $\mathcal{D}$ is a replay buffer of environment transitions and $\bar{\theta}$ is an exponential moving average of the weights. DDPG is amenable to incorporate n-step returns [Watkins, 1989, eng and Williams, 1996] when estimating TD error beyond a single step [Barth-Maron et al., 2018]. In practice, n-step returns allow for faster reward propagation and has been previously used in policy gradient and Q-learning methods [Mnih et al., 2016b, Barth-Maron et al., 2018, Hessel et al., 2017].

2.3 Data Augmentation in Reinforcement Learning

Recently, it has been shown that data augmentation techniques, commonplace in Computer Vision, are also important for achieving the state-of-the-art performance in image-based RL [Yarats et al., 2021b, Laskin et al., 2020]. For example, the state-of-the-art algorithm for visual RL, DrQ [Yarats et al., 2021b] builds on top of Soft Actor-Critic [Haarnoja et al., 2018b], a model-free actor-critic algorithm, by adding a convolutional encoder and data augmentation in the form of random shifts. The use of such data augmentations now forms an essential component of several recent visual RL algorithms [Srinivas et al., 2020, Raileanu et al., 2020, Yarats et al., 2021a, Stooke et al., 2020, Hansen and Wang, 2021, Schwarzer et al., 2020b].

3 DrQ-v2: Improved Data-Augmented Reinforcement Learning

In this section, we describe **DrQ-v2**, a simple model-free actor-critic RL algorithm for image-based continuous control, that builds upon DrQ.

3.1 Algorithmic Details

Image Augmentation As in DrQ we apply random shifts image augmentation to pixel observations of the environment. In the settings of visual continuous control by DMC, this augmentation can be

[1]Here, $\Delta(\mathcal{X})$ denotes a distribution over the state space $\mathcal{X}$.

instantiated by first padding each side of 84×84 observation rendering by 4 pixels (by repeating boundary pixels), and then selecting a random 84×84 crop, yielding the original image shifted by ± 4 pixels. We also find it useful to apply bilinear interpolation on top of the shifted image (i.e, we replace each pixel value with the average of the four nearest pixel values). In our experience, this modification provides an additional performance boost across the board.

Image Encoder The augmented image observation is then embedded into a low-dimensional latent vector by applying a convolutional encoder. We use the same encoder architecture as in DrQ, which first was introduced introduced in SAC-AE [Yarats et al., 2019]. This process can be succinctly summarized as $\boldsymbol{h} = f_\xi(\mathrm{aug}(\boldsymbol{x}))$, where f_ξ is the encoder, aug is the random shifts augmentation, and $\boldsymbol{x}$ is the original image observation.

Actor-Critic Algorithm We use DDPG [Lillicrap et al., 2015a] as a backbone actor-critic RL algorithm and, similarly to Barth-Maron et al. [2018], augment it with n-step returns to estimate TD error. This results into faster reward propagation and overall learning progress [Mnih et al., 2016a]. While some methods [Hafner et al., 2020] employ more sophisticated techniques such as TD(λ) or Retrace(λ) [Munos et al., 2016], they are often computationally demanding when n is large. We find that using simple n-step returns, without an importance sampling correction, strikes a good balance between performance and efficiency. We also employ clipped double Q-learning [Fujimoto et al., 2018] to reduce overestimation bias in the target value. Practically, this requires training two Q-functions Q_{θ_1} and Q_{θ_2}. For this, we sample a mini-batch of transitions $\tau = (\boldsymbol{x}_t, \boldsymbol{a}_t, r_{t:t+n-1}, \boldsymbol{x}_{t+n})$ from the replay buffer $\mathcal{D}$ and compute the following two losses:

$$\mathcal{L}_{\theta_k, \xi}(\mathcal{D}) = \mathbb{E}_{\tau \sim \mathcal{D}}\big[(Q_{\theta_k}(\boldsymbol{h}_t, \boldsymbol{a}_t) - y)^2\big] \quad \forall k \in \{1, 2\}, \tag{1}$$

with the TD target y defined as:

$$y = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n \min_{k=1,2} Q_{\bar{\theta}_k}(\boldsymbol{h}_{t+n}, \boldsymbol{a}_{t+n}),$$

where $\boldsymbol{h}_t = f_\xi(\mathrm{aug}(\boldsymbol{x}_t))$, $\boldsymbol{h}_{t+n} = f_\xi(\mathrm{aug}(\boldsymbol{x}_{t+n}))$, $\boldsymbol{a}_{t+n} = \pi_\phi(\boldsymbol{h}_{t+n}) + \epsilon$, and $\bar{\theta}_1$,$\bar{\theta}_2$ are the slow-moving weights for the Q target networks. We note, that in contrast to DrQ, we do not employ a target network for the encoder f_ξ and always use the most recent weights ξ to embed $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t+n}$. The exploration noise ϵ is sampled from $\mathrm{clip}(\mathcal{N}(0, \sigma^2), -c, c)$ similar to TD3 [Fujimoto et al., 2018], with the exception of decaying σ, which we describe below. Finally, we train the deterministic actor π_ϕ using DPG with the following loss:

$$\mathcal{L}_\phi(\mathcal{D}) = -\mathbb{E}_{\boldsymbol{x}_t \sim \mathcal{D}}\big[\min_{k=1,2} Q_{\theta_k}(\boldsymbol{h}_t, \boldsymbol{a}_t)\big], \tag{2}$$

where $\boldsymbol{h}_t = f_\xi(\mathrm{aug}(\boldsymbol{x}_t))$, $\boldsymbol{a}_t = \pi_\phi(\boldsymbol{h}_t) + \epsilon$, and $\epsilon \sim \mathrm{clip}(\mathcal{N}(0, \sigma^2), -c, c)$. Similar to DrQ, we do not use actor's gradients to update the encoder's parameters ξ.

Scheduled Exploration Noise Empirically, we observe that it is helpful to have different levels of exploration at different stages of learning. At the beginning of training we want the agent to be more stochastic and explore the environment more effectively, while at the later stages of training, when the agent has already identified promising behaviors, it is better to be more deterministic and master those behaviors. Similar to Amos et al. [2020], we instantiate this idea by using linear decay $\sigma(t)$ for the variance σ^2 of the exploration noise defined as:

$$\sigma(t) = \sigma_{\mathrm{init}} + (1 - \min(\frac{t}{T}, 1))(\sigma_{\mathrm{final}} - \sigma_{\mathrm{init}}), \tag{3}$$

where σ_{init} and σ_{final} are the initial and final values for standard deviation, and T is the decay horizon.

Key Hyper-Parameter Changes We also conduct an extensive hyper-parameter search and identify several useful hyper-parameter modifications compared to DrQ. The three most important hyper-parameters are: (i) the size of the replay buffer, (ii) mini-batch size, and (iii) learning rate. Specifically, we use a 10 times larger replay buffer than DrQ. We also use a smaller mini-batch size of 256 without any noticeable performance degradation. This is in contrast to CURL [Srinivas et al., 2020] and DrQ [Yarats et al., 2021b] that both use a larger batch size of 512 to attain more stable training in the expense of computational efficiency. Finally, we find that using smaller learning rate of 1×10^{-4}, rather than DrQ's learning rate of 1×10^{-3}, results into more stable training without any loss in learning speed.

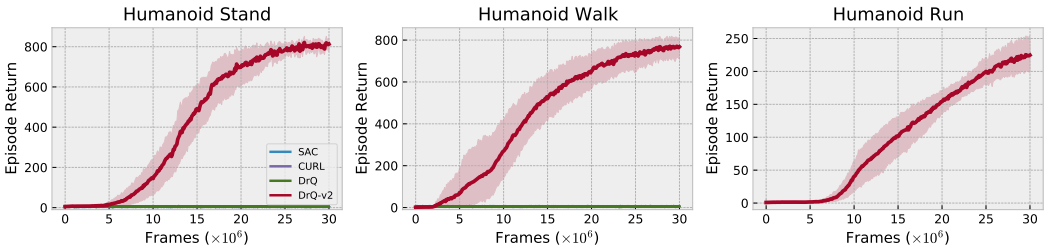


Figure 3: The *hard* benchmark consists of three humanoid locomotion tasks: *stand*, *walk*, and *run*. These three represent particularly hard exploration challenges, being previously unsolvable by model-free methods. The training speed of DrQ-v2 was key to solving the task, since it allowed for extensive investigation of different variations, resulting in the discovery of effective strategies.

3.2 Implementation Details

Faster Image Augmentation We replace DrQ's random shifts augmentation (i.e., `kornia.augmentation.RandomCrop`) by a custom implementation that uses flow-field image sampling provided in PyTorch (i.e., `grid_sample`). This is done for two reasons. First, we noticed that Kornia's implementation does not fully utilize GPU pipelining since it has some intermediate CPU to GPU data transferring which breaks the computational flow. Second, using `grid_sample` allows straightforward addition of bilinear interpolation. Our custom random shifts augmentation improves training throughput by a factor of 2.

Faster Replay Buffer Another computational bottleneck of DrQ was the replay buffer. The specific implementation had poor memory management which resulted in slow CPU to GPU data transfer, which also restricted the number of image-based transitions that could be stored. We reimplemented the replay buffer to address these issues which led to a ten-fold increase in storage capacity and faster data transfer. More details are available in our open-source release. We note that the improved training speed of DrQ-v2 was key to solving humanoid tasks as it enabled much faster experimentation.

4 Experiments

In this section we provide empirical evaluation of DrQ-v2 on an extensive set of visual continuous control tasks from DMC [Tassa et al., 2018]. We first present comparison to prior methods, both model-free and model-based, in terms of sample efficiency and wall-clock time. We then present a large scale ablation study that guided the final version of DrQ-v2.

4.1 Setup

Environments We consider a set of MuJoCo tasks [Todorov et al., 2012] provided by DMC [Tassa et al., 2018], a widely used benchmark for continous control. DMC offers environments of various difficulty, ranging from the simple control problems such as the single degree of freedom (DOF) pendulum and cartpool, to the control of complex multi-joint bodies such as the humanoid (21 DOF). We consider learning from pixels. In this setting, environment observations are stacks of 3 consecutive RGB images of size 84×84, stacked along the channel dimension to enable inference of dynamic information like velocity and acceleration. In total, we consider 24 different tasks, which we group into three buckets, *easy*, *medium*, and *hard*, according to the sample complexity to reach near-optimal performance (see Appendix B). Our motivation for this is to encourage RL practitioners to focus on the medium and hard tasks and stop using the easy tasks for evaluation, as they are mostly solved at this point and may no longer provide any valuable signal in comparing different methods.

Training Details For all tasks in the suite an episode corresponds to 1000 steps, where a per-step reward is in the unit interval $[0, 1]$. This upper bounds the episode return to 1000 making it easier to compute aggregated performance measures across tasks, as we do in Figure 1a. To facilitate fair wall-clock time comparison all algorithms are trained on the same hardware (i.e., a single NVIDIA V100 GPU machine) and evaluated with the same periodicity of 20000 environment steps. Each

evaluation query averages episode returns over 10 episodes. Per common practice [Hafner et al., 2019], we employ action repeat of 2 and measure sample complexity in the environment steps, rather than the actor steps. In all the figures we plot the mean performance over 10 seeds together with the shaded regions which represent 95% confidence intervals. A full list of hyper-parameters can be found in Appendix C.

Comparison Axes In many real-world applications, taking a step in the environment incurs significant computational cost making sample efficiency a critical feature of an RL algorithm. It is hence important to compare RL algorithms in terms of their sample efficiency. We facilitate this comparison by computing an algorithm's performance measured by episode return with respect to environment steps. On the other end, striving low sample complexity often comes at the cost of a poor computational efficiency. Unfortunately, recent deep RL literature has paid very little attention to this important axis which has led to skyrocketing hardware requirements. Such a trend has made it virtually impossible for an RL practitioner with modest hardware capacity to contribute to advancements in image-based RL, leaving research in this area to a few well-equipped labs. To democratize research in visual RL, we additionally propose to compare the agents in terms of wall-clock training time given the same single GPU hardware. We note that it is possible to adapt DrQ-v2 to a distributed setup, as has been done for DDPG in prior work [Barth-Maron et al., 2018, Hoffman et al., 2020].

4.2 Comparison to Model-Free Methods

Baselines We compare our method to several state-of-the-art model-free algorithms for visual RL including CURL [Srinivas et al., 2020], DrQ [Yarats et al., 2021b], and vanilla SAC [Haarnoja et al., 2018b] augmented with the convolutional encoder from SAC-AE [Yarats et al., 2019]. Vanilla SAC is a weak baseline and only included as a ground point to showcase the recent progress in visual RL.

Sample Efficiency Axis We present results on the hard (Figure 3), medium (Figure 4), and easy (Figure 7 in Appendix) subsets of the DMC tasks, where the maximum number of environment interactions is limited to *thirty*, *three*, and *one* million of steps respectively.Our empirical study reveals that DrQ-v2 outperforms prior model-free methods in terms of sample efficiency across the three benchmarks with different levels of difficulty. Importantly, DrQ-v2's advantage is more pronounced on harder tasks (i.e., acrobot, quadruped, and humanoid), where exploration is especially challenging. Finally, DrQ-v2 solves the DMC humanoid locomotion tasks directly from pixels, which, to the best of our knowledge, is the first successful demonstration of such feat by a model-free method.

Compute Efficiency Axis To facilitate a fair comparison in terms of sheer wall-clock training time, besides employee the identical training protocol (see Section 4.1), we also use the same mini-batch size of 256 for each agent. In Figure 5, we evaluate DrQ-v2 on a subset of DMC tasks for the sake of brevity only, and note that the demonstrated results can be easily extrapolated to the other tasks given the linear dependency between training time and sample complexity. In our benchmarks, DrQ-v2 is able to achieve a throughput of 96 FPS, which favorably compares to DrQ's 28 FPS (a $3.4\times$ increase), and CURL's 16 FPS (a $6\times$ increase) throughputs. Practically, DrQ-v2 solves easy, medium, and hard tasks within 2.9, 8.6, and 86 hours respectively.

4.3 Ablation Study

In this section we present an extensive ablation study that guided us to the final version of DrQ-v2. Here, for brevity we only discuss experiments that were most impactful and omit others that did not pan out. For computational reasons, we only ablate on 3 different control tasks of various difficulty levels. Our findings are summarized in Figure 6 and detailed below.

Switching from SAC to DDPG DrQ [Yarats et al., 2021b] leverages SAC [Haarnoja et al., 2018b] as the backbone RL algorithm. While it has been demonstrated by many works, including the original manuscripts [Haarnoja et al., 2018b,a] that SAC is superior to DDPG [Lillicrap et al., 2015b], our careful examination identifies two shortcomings that preclude SAC (within DrQ) to solve hard exploration-wise image-based tasks. First, the automatic entropy adjustment strategy, introduced in Haarnoja et al. [2018a], is inadequate and in some cases leads to a premature entropy collapse. This prevents the agent from finding more optimal behaviors due to the insufficient exploration.

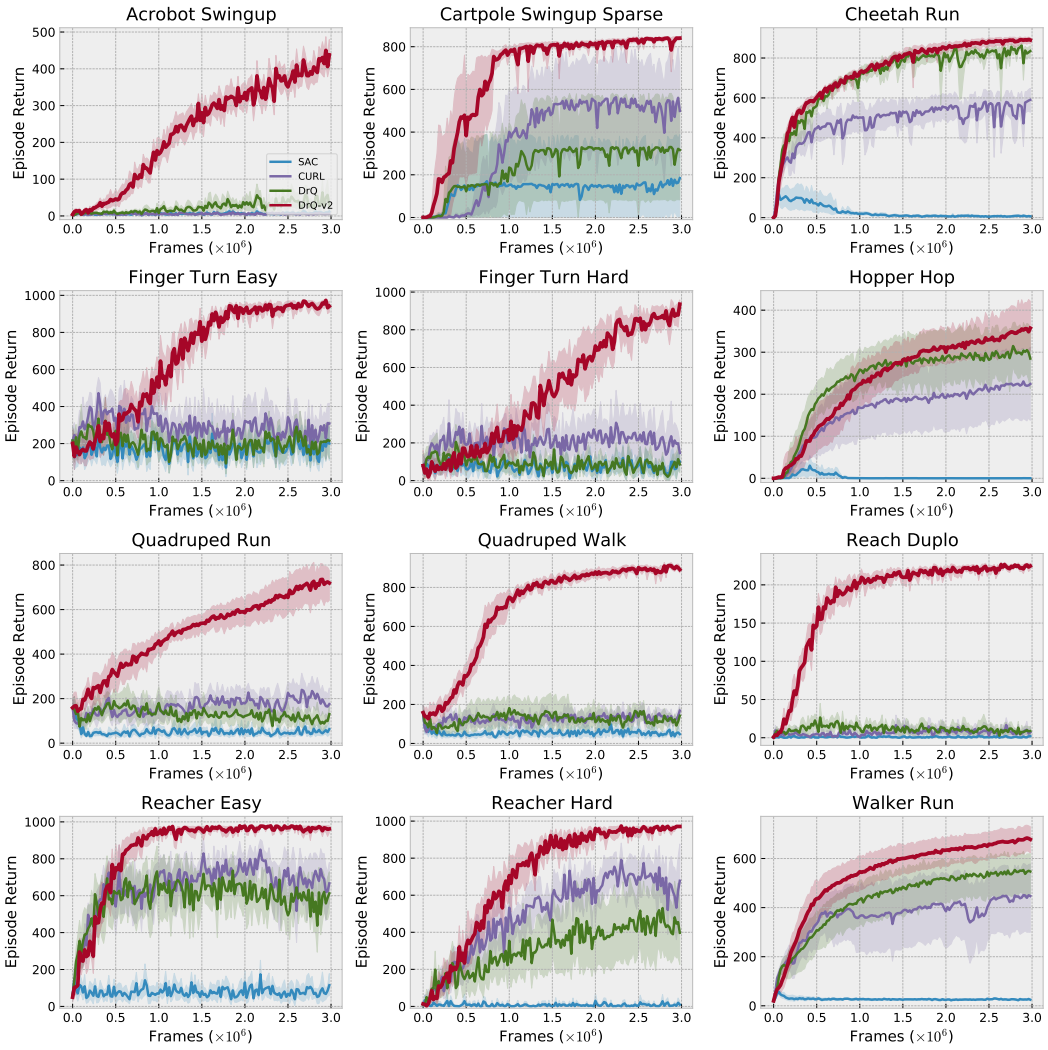


Figure 4: The *medium* benchmark consists of 12 complex control tasks that offer various challenges, including complex dynamics, sparse rewards, hard exploration, and more. DrQ-v2 demonstrates favorable sample efficiency and comfortably outperforms leading model-free baselines.

In Figure 6a, we empirically verify our intuition and, indeed, observe that DDPG demonstrates better exploration properties than SAC. Here, DDPG uses constant $\sigma = 0.2$ for the exploration noise.

N-step Returns The second issue concerns the inability of soft Q-learning to incorporate n-step returns to estimate TD error in a straightforward manner. The reason for this is that computing a target value for soft Q-function requires estimating per-step entropy of the policy, which is challenging to do for large n in the off-policy regime. In contrast, DDPG does not require estimating per-step entropy to compute targets and is more amenable for n-step returns. In Figure 6b we demonstrate that estimating TD error with n-step returns improves sample efficiency over vanilla DDPG. We select 3-step returns as a sensible choice for our method.

Replay Buffer Size We hypothesize that a larger replay buffer plays an important role in circumventing the catastrophic forgetting problem [Fedus et al., 2020]. This issue is especially prominent in tasks with more diverse initial state distributions (i.e., reacher or humanoid tasks), where the vast variety of possible behaviors requires significantly larger memory. We confirm this intuition by ablating the size of the replay buffer in Figure 6c, where we observe that a buffer size of 1M helps to improve performance on *Reacher Hard* considerably.

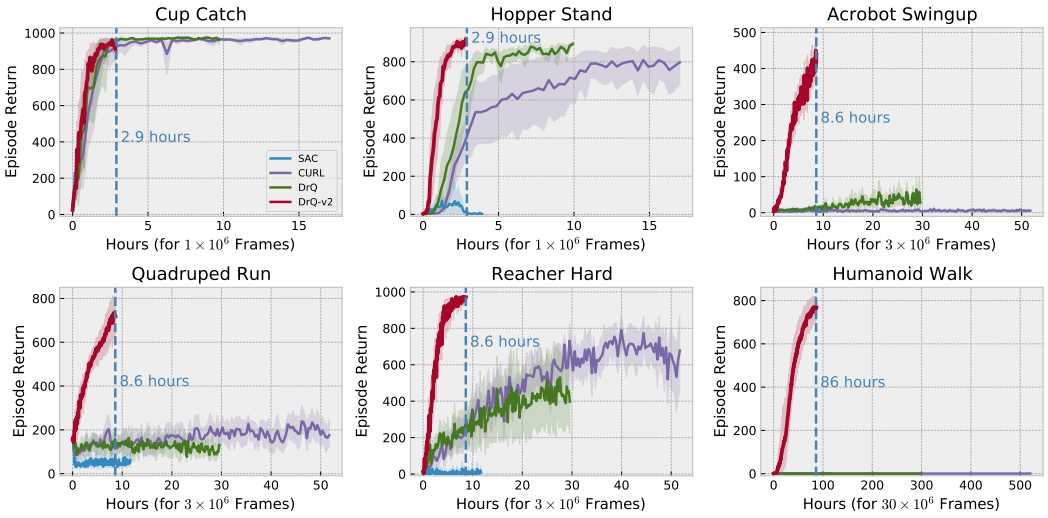


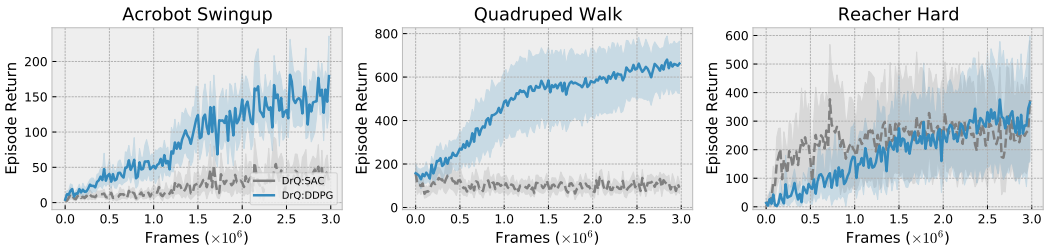
Figure 5: DrQ-v2 not only achieves superior sample efficiency than prior model-free methods, but it also requires less wall-clock training time to do so. Our benchmarking shows that DrQ-v2 can reach a throughput of 96 FPS on a single NVIDIA V100 GPU. Practically, this means that most of the task can be solved in 8 hours or less, which greatly speeds up the research process.

Scheduled Exploration Noise Finally, we demonstrate that it is useful to decay the variance of the exploration noise over the course of training according to Equation (3). In Figure 6d, we compare two versions of our algorithm, where the first variant uses a fixed standard deviation of $\sigma = 0.2$, while the second variant employes the decaying schedule $\sigma(t)$, with parameters $\sigma_{\text{init}} = 1.0$, $\sigma_{\text{final}} = 0.1$, and $T = 500000$. Having the exploration noise to decay linearly over time turns out to be helpful and provide an additional performance boost, which was especially useful for solving humanoid tasks.

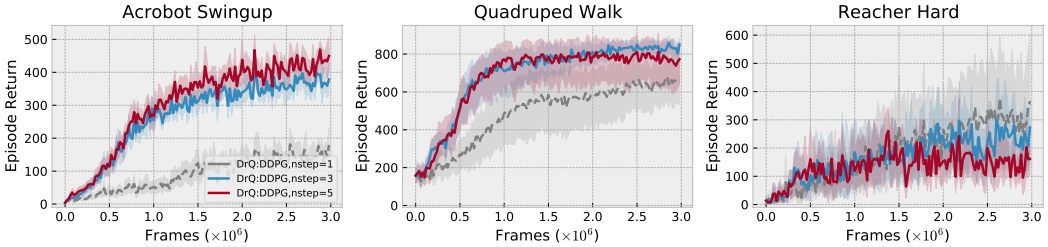
5 Related Work

Visual Reinforcement Learning Successes of visual representation learning in computer vision [Vincent et al., 2008, Doersch et al., 2015, Wang and Gupta, 2015, Noroozi and Favaro, 2016, Zhang et al., 2017, Gidaris et al., 2018] has inspired successes in visual RL, where coherent representations are learned alongside RL. Works such as SAC-AE [Yarats et al., 2019], PlaNet [Hafner et al., 2018], and SLAC [Lee et al., 2019], demonstrated how auto-encoders [Finn et al., 2015] could improve visual RL. Following this, other self-supervised objectives such as contrastive learning in CURL [Srinivas et al., 2020] and ATC [Stooke et al., 2020], self-prediction in SPR [Schwarzer et al., 2020a], contrastive cluster assignment in Proto-RL [Yarats et al., 2021a], and augmented data in DrQ [Yarats et al., 2021b] and RAD [Laskin et al., 2020], have significantly bridged the gap between state-based and image-based RL. Future prediction objectives [Hafner et al., 2018, 2019, Yan et al., 2020, Finn et al., 2015, Pinto et al., 2016, Agrawal et al., 2016] and other auxiliary objectives [Jaderberg et al., 2016, Zhan et al., 2020, Young et al., 2020, Chen et al., 2020] have shown improvements on a variety of problems ranging from gameplay, continuous control, and robotics. In the context of visual control settings, clever use of augmented data [Yarats et al., 2021b, Laskin et al., 2020] currently produces state-of-the-art results on visual tasks from DMC [Tassa et al., 2018].

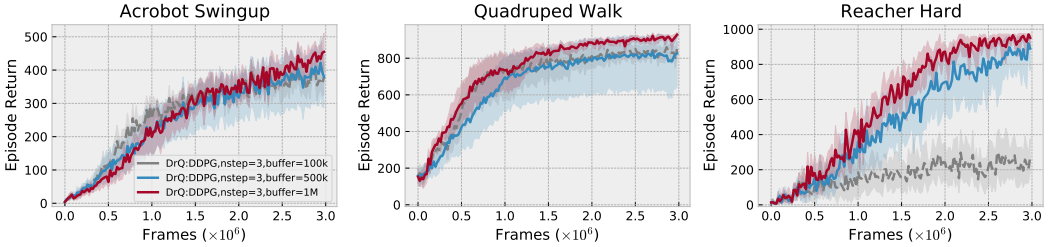
Humanoid Control The humanoid control problem first presented in Tassa et al. [2012], has been studied as one of the hardest control problems due to its large state and action spaces. The earliest solutions to this problem use ideas in model-based optimal control to generate policies given an accurate model of the humanoid . Subsequent works in RL have shown that model-free policies can solve the humanoid control problem given access to proprioceptive state observations. However, solving such a problem from visual observations has been a challenging problem, with leading RL algorithms making little progress to solve the task [Tassa et al., 2018]. Recently, Hafner et al. [2020] was able to solve this problem through a model-based technique in around 30M environment steps and 340 hours of training on a single GPU machine. DrQ-v2, presented in this paper, marks the first



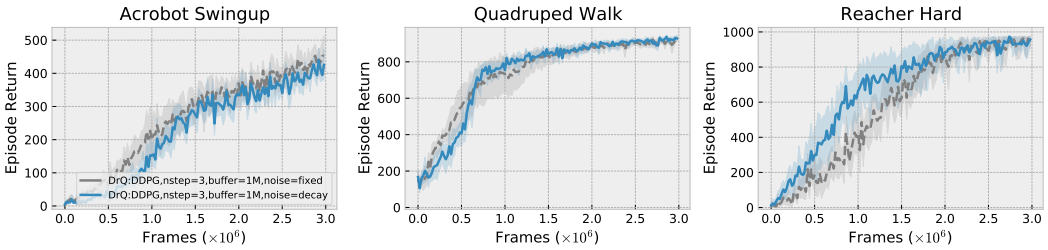
(a) DrQ (dotted silver) relies on SAC as a base RL algorithm. Replacing SAC with DDPG results in a significant performance gain (blue).



(b) DDPG straightforwardly incorporates n-step returns, a critical tool for exploration. We observe that the 3 (blue) and 5 (red) steps variants provide additional improvements to the previous version that uses single step TD-targets (silver). Going forward, we adopt 3-step returns (blue).



(c) Increasing the size of the replay buffer improves performance, over the original 100k used by DrQ (silver). Going forward, we use a buffer size of 1M (red).



(d) Finally, a decaying schedule for the variance of the exploration noise (blue) helps on hard exploration tasks, versus the fixed variance variant (silver).

Figure 6: An ablation study that led us to the final version of DrQ-v2. We incrementally show each of the four key improvements to DrQ that collectively form DrQ-v2. The silver dotted curves in the first row show the original DrQ. In subsequent rows they show progressive improvements, using the optimal choice from the previous rows (i.e., the silver curve in the third row shows DrQ with a DDPG base RL algorithm and 3-step returns). The red and blue curves show the effect of individual modifications. In the last row the blue curve corresponds to DrQ-v2.

model-free RL method that can solve humanoid control from visual observations, taking also around 30M steps and 86 hours of training on the same hardware.

References

Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: experiential learning of intuitive physics. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 5092–5100, 2016.

Brandon Amos, Samuel Stanton, Denis Yarats, and Andrew Gordon Wilson. On the model-based stochastic value gradient for continuous reinforcement learning. *CoRR*, 2020.

Gabriel Barth-Maron, Matthew W. Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributional policy gradients. In *International Conference on Learning Representations*, 2018.

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

Bryan Chen, Alexander Sax, Gene Lewis, Iro Armeni, Silvio Savarese, Amir Roshan Zamir, Jitendra Malik, and Lerrel Pinto. Robust policies via mid-level visual representations: An experimental study in manipulation and navigation. *CoRR*, 2020.

Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015.

Jing eng and Ronald J. Williams. Incremental multi-step q-learning. *Machine Learning*, 1996.

William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. *CoRR*, 2020.

Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Learning visual feature spaces for robotic manipulation with deep spatial autoencoders. *CoRR*, 2015.

Scott Fujimoto, Herke van Hoof, and David Meger. 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.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations, 2018.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta a nd Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *CoRR*, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *CoRR*, 2020.

Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *International Conference on Robotics and Automation*, 2021.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Daniel Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. *arXiv preprint arXiv:1710.02298*, 2017.

Matt Hoffman, Bobak Shahriari, John Aslanides, Gabriel Barth-Maron, Feryal Behbahani, Tamara Norman, Abbas Abdolmaleki, Albin Cassirer, Fan Yang, Kate Baumli, Sarah Henderson, Alex Novikov, Sergio Gómez Colmenarejo, Serkan Cabi, Caglar Gulcehre, Tom Le Paine, Andrew Cowie, Ziyu Wang, Bilal Piot, and Nando de Freitas. Acme: A research framework for distributed reinforcement learning, 2020.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks, 2016.

Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data, 2020.

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

Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, 2015a.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015b.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv e-prints*, 2013.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, 2016a.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016b.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning. *CoRR*, 2016.

Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016.

Lerrel Pinto, Dhiraj Gandhi, Yuanfeng Han, Yong-Lae Park, and Abhinav Gupta. The curious robot: Learning visual representations via physical interactions. *CoRR*, 2016.

Roberta Raileanu, Max Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in deep reinforcement learning. *CoRR*, 2020.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with momentum predictive representations. *arXiv preprint arXiv:2007.05929*, 2020a.

Max Schwarzer, Ankesh Anand, Rishab Goel, R. Devon Hjelm, Aaron C. Courville, and Philip Bachman. Data-efficient reinforcement learning with momentum predictive representations. *CoRR*, 2020b.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020.

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. *arXiv preprint arXiv*, 2020.

Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 1992.

Christopher John Cornish Hellaby Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, 1989.

Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. Learning predictive representations for deformable objects using contrastive estimation. *arXiv preprint arXiv:2003.05436*, 2020.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. *arXiv preprint arXiv:1910.01741*, 2019.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. *CoRR*, 2021a.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *9th International Conference on Learning Representations, ICLR 2021*, 2021b.

Sarah Young, Dhiraj Gandhi, Shubham Tulsiani, Abhinav Gupta, Pieter Abbeel, and Lerrel Pinto. Visual imitation made easy. *CoRR*, 2020.

Albert Zhan, Philip Zhao, Lerrel Pinto, Pieter Abbeel, and Michael Laskin. A framework for efficient robotic manipulation. *CoRR*, 2020.

Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1058–1067, 2017.

Appendix

A DrQ-v2: Improved Data-Augmented Reinforcement Learning

Algorithm 1 DrQ-v2: Improved data-augmented RL.

Inputs:
f_ξ, π_ϕ, Q_{θ_1}, Q_{θ_2}: parametric networks for encoder, policy, and Q-functions respectively.
aug: random shifts image augmentation.
$\sigma(t)$: scheduled standard deviation for the exploration noise defined in Equation (3).
T, B, α, τ, c: training steps, mini-batch size, learning rate, target update rate, clip value.
Training routine:
for each timestep $t = 1..T$ **do**
  $\sigma_t \leftarrow \sigma(t)$ ▷ Compute stddev for the exploration noise
  $\boldsymbol{a}_t \leftarrow \pi_\phi(f_\xi(\boldsymbol{x}_t)) + \epsilon$ and $\epsilon \sim \mathcal{N}(0, \sigma_t^2)$ ▷ Add noise to the deterministic action
  $\boldsymbol{x}_{t+1} \sim P(\cdot|\boldsymbol{x}_t, \boldsymbol{a}_t)$ ▷ Run transition function for one step
  $\mathcal{D} \leftarrow \mathcal{D} \cup (\boldsymbol{x}_t, \boldsymbol{a}_t, R(\boldsymbol{x}_t, \boldsymbol{a}_t), \boldsymbol{x}_{t+1})$ ▷ Add a transition to the replay buffer
  UPDATECRITIC($\mathcal{D}, \sigma_t$)
  UPDATEACTOR($\mathcal{D}, \sigma_t$)
end for
procedure UPDATECRITIC($\mathcal{D}, \sigma$)
  $\{(\boldsymbol{x}_t, \boldsymbol{a}_t, r_{t:t+n-1}, \boldsymbol{x}_{t+n})\} \sim \mathcal{D}$ ▷ Sample a mini batch of B transitions
  $\boldsymbol{h}_t, \boldsymbol{h}_{t+n} \leftarrow f_\xi(\mathrm{aug}(\boldsymbol{x}_t)), f_\xi(\mathrm{aug}(\boldsymbol{x}_{t+n}))$ ▷ Apply data augmentation and encode
  $\boldsymbol{a}_{t+n} \leftarrow \pi_\phi(\boldsymbol{h}_{t+n}) + \epsilon$ and $\epsilon \sim \mathrm{clip}(\mathcal{N}(0, \sigma^2))$ ▷ Sample action
  Compute $\mathcal{L}_{\theta_1,\xi}$ and $\mathcal{L}_{\theta_2,\xi}$ using Equation (1) ▷ Compute critic losses
  $\xi \leftarrow \xi - \alpha\nabla_\xi(\mathcal{L}_{\theta_1,\xi} + \mathcal{L}_{\theta_2,\xi})$ ▷ Update encoder weights
  $\theta_k \leftarrow \theta_k - \alpha\nabla_{\theta_k}\mathcal{L}_{\theta_k,\xi} \quad \forall k \in \{1, 2\}$ ▷ Update critic weights
  $\bar{\theta}_k \leftarrow (1 - \tau)\bar{\theta}_k + \tau\theta_k \quad \forall k \in \{1, 2\}$ ▷ Update critic target weights
end procedure
procedure UPDATEACTOR($\mathcal{D}, \sigma$)
  $\{(\boldsymbol{x}_t)\} \sim \mathcal{D}$ ▷ Sample a mini batch of B observations
  $\boldsymbol{h}_t \leftarrow f_\xi(\mathrm{aug}(\boldsymbol{x}_t))$ ▷ Apply data augmentation and encode
  $\boldsymbol{a}_t \leftarrow \pi_\phi(\boldsymbol{h}_t) + \epsilon$ and $\epsilon \sim \mathrm{clip}(\mathcal{N}(0, \sigma^2))$ ▷ Sample action
  Compute $\mathcal{L}_\phi$ using Equation (2) ▷ Compute actor loss
  $\phi \leftarrow \phi - \alpha\nabla_\phi\mathcal{L}_\phi$ ▷ Update actor's weights only
end procedure

B Benchmarks

We classify a set of 24 continuous control tasks from DMC [Tassa et al., 2018] into *easy*, *medium*, and *hard* benchmarks and provide a summary for each task in Table 1.

Task	Traits	Difficulty	Allowed Steps	$\dim(\mathcal{S})$	$\dim(\mathcal{A})$
Cartpole Balance	balance, dense	easy	1×10^6	4	1
Cartpole Balance Sparse	balance, sparse	easy	1×10^6	4	1
Cartpole Swingup	swing	dense	1×10^6	4	1
Cup Catch	swing, catch, sparse	easy	1×10^6	8	2
Finger Spin	rotate, dense	easy	1×10^6	6	2
Hopper Stand	stand, dense	easy	1×10^6	14	4
Pendulum Swingup	swing, sparse	easy	1×10^6	2	1
Walker Stand	stand, dense	easy	1×10^6	18	6
Walker Walk	walk, dense	easy	1×10^6	18	6
Acrobot Swingup	diff. balance, dense	medium	3×10^6	4	1
Cartpole Swingup Sparse	swing, sparse	medium	3×10^6	4	1
Cheetah Run	run, dense	medium	3×10^6	18	6
Finger Turn Easy	turn, sparse	medium	3×10^6	6	2
Finger Turn Hard	turn, sparse	medium	3×10^6	6	2
Hopper Hop	move, dense	medium	3×10^6	14	4
Quadruped Run	run, dense	medium	3×10^6	56	12
Quadruped Walk	walk, dense	medium	3×10^6	56	12
Reach Duplo	manipulation, sparse	medium	3×10^6	55	9
Reacher Easy	reach, dense	medium	3×10^6	4	2
Reacher Hard	reach, dense	medium	3×10^6	4	2
Walker Run	run, dense	medium	3×10^6	18	6
Humanoid Stand	stand, dense	hard	30×10^6	54	21
Humanoid Walk	walk, dense	hard	30×10^6	54	21
Humanoid Run	run, dense	hard	30×10^6	54	21

Table 1: A detailed description of each tasks in our *easy*, *medium*, and *hard* benchmarks.

C Hyper-parameters

The full list of hyper-parameters is presented in Table 2. While we tried to keep the settings identical for each of the task, there are a few specific deviations for some tasks.
Walker Stand/Walk/Run For all three tasks we use mini-batch size of 512 and n-step return of 1.
Cartpole Swingup Sparse stddev. schedule is set to 1.0 to facilitate stronger exploration in the sparse reward setting.
Quadruper Run We set the replay buffer size to 10^5.
Humanoid Stand/Walk/Run We set learning rate to 8×10^{-5} and increase features dim. to 100.

Table 2: A default set of hyper-parameters used in our experiments.

Parameter	Setting
Replay buffer capacity	10^6
Action repeat	2
Seed frames	4000
Exploration steps	2000
n-step returns	3
Mini-batch size	256
Discount γ	0.99
Optimizer	Adam
Learning rate	10^{-4}
Agent update frequency	2
Critic Q-function soft-update rate τ	0.01
Features dim.	50
Hidden dim.	1024
Exploration stddev. clip	0.3
Exploration stddev. schedule	easy: $\text{linear}(1.0, 0.1, 100000)$ medium: $\text{linear}(1.0, 0.1, 500000)$ hard: $\text{linear}(1.0, 0.1, 2000000)$

D Comparison to Model-Free Methods

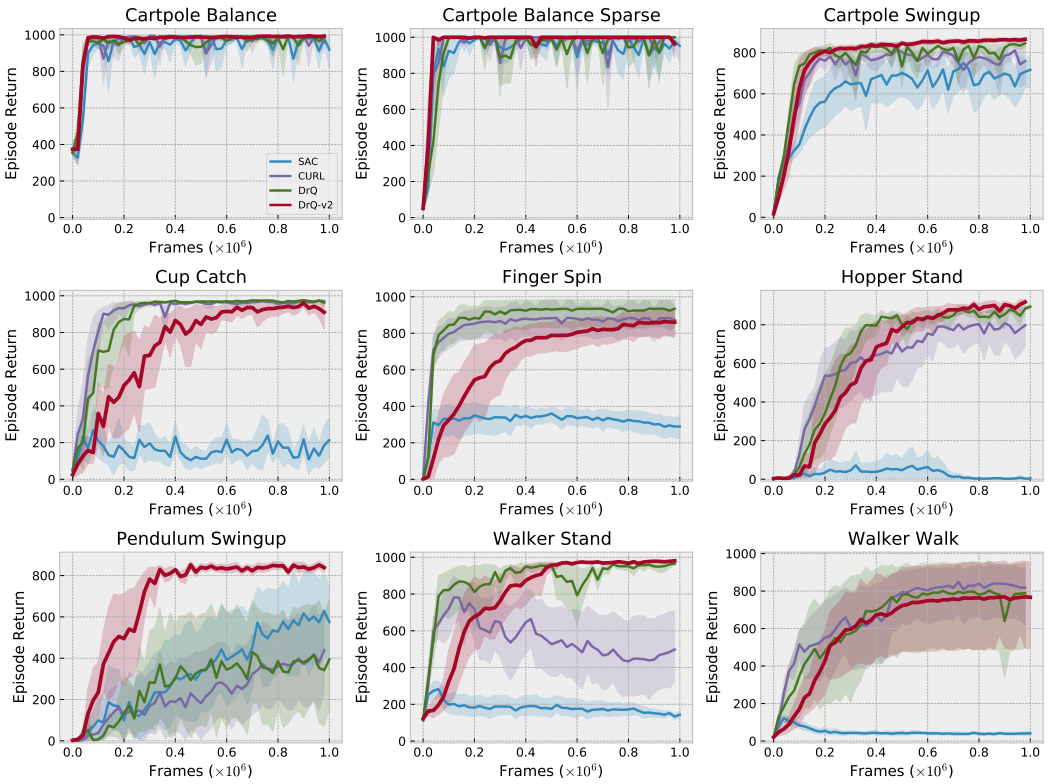


Figure 7: The *easy* benchmark consists of 9 tasks, where performance gains have been largely saturated by prior work. Still, DrQ-v2 is able to match sample complexity of the baselines. We note, that evaluation on these tasks is done for completeness reasons only, and encourage RL practitioners to refrain from using them for benchmarking purposes in future research.

E Comparison to Model-Based Methods

Baseline To see how DrQ-v2 stacks up against model-based methods, which tend to achieve better sample complexity in expense of a larger computational footprint, we also compare to recent and unpublished[2] improvements to Dreamer-v2 [Hafner et al., 2020], a leading model-based approach for visual continuous control. The recent update shows that the model-based approach can solve the DMC humanoid tasks directly from pixel inputs. The open-source implementation of Dreamer-v2 (`https://github.com/danijar/dreamerv2`) only provides learning curves for *Humanoid Walk*. For this reason we run their code to obtain results on other DMC tasks. To limit hardware requirements of compute-expensive Dreamer-v2, we only run it on a subset of 12 out of 24 considered tasks. This subset, however, overlaps with all the three (i.e. easy, medium, and hard) benchmarks.

Sample Efficiency Axis Our empirical study in Figure 8 reveals that in many cases, DrQ-v2, despite being a model-free method, can rival sample efficiency of state-of-the-art model-based Dreamer-v2. We note, however, that on several tasks (for example *Acrobot Swingup* and *Finger Turn Hard*) Dreamer-v2 outperforms DrQ-v2. We leave investigation of such discrepancy for future work.

Compute Efficiency Axis A different picture emerges if comparison is done with respect to wall-clock training time. Dreamer-v2, being a model-based method, performs significantly more floating point operations to reach its sample efficiency. In our benchmarks, Dreamer-v2 records a throughput of 24 FPS, which is $4\times$ less than DrQ-v2's throughput of 96 FPS, measured on the same hardware. In Figure 9 we plot learning curves against wall-clock time and observe that DrQ-v2 takes less time to solve the tasks.

[2] ArXiv v3 revision from May 3, 2021 introduces a new result on the *Humanoid Walk* task in Appendix A.

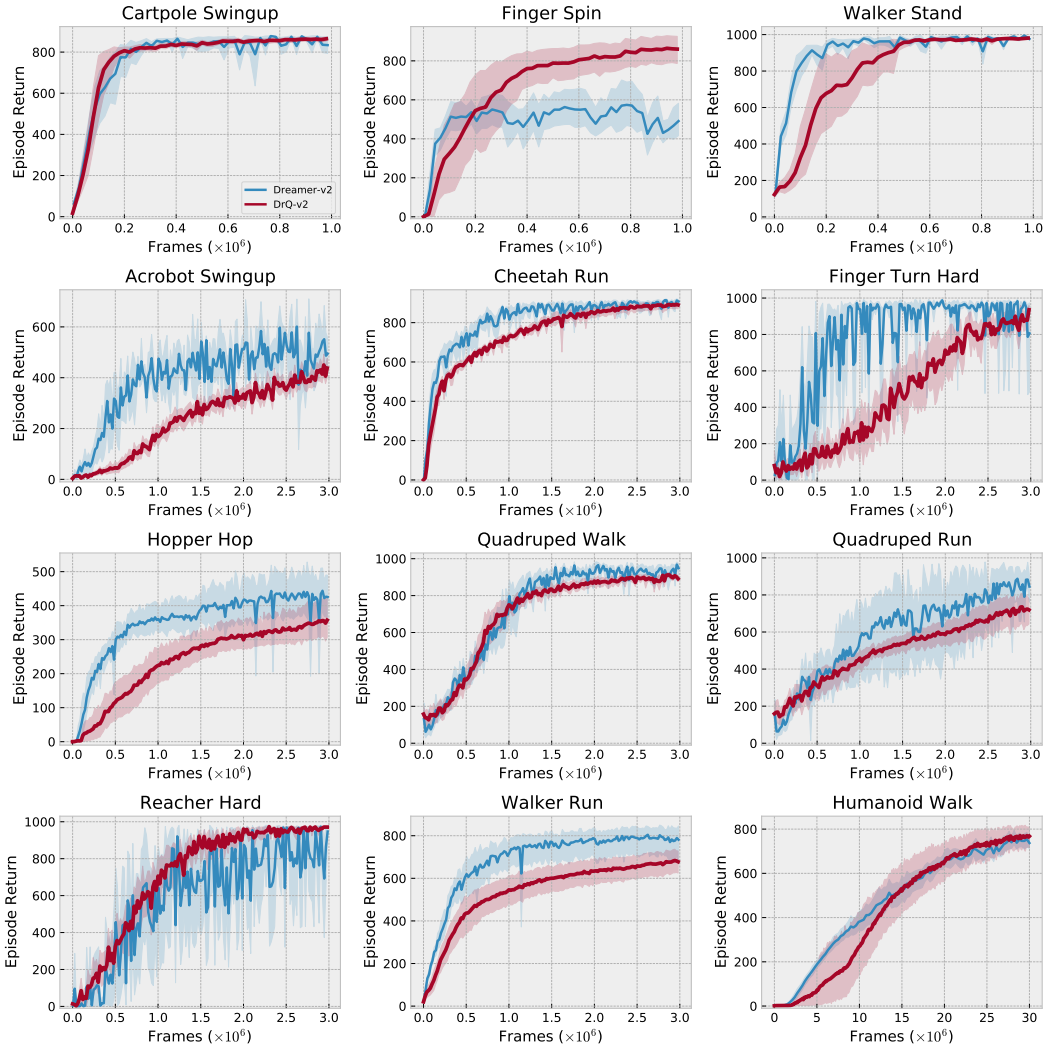


Figure 8: In many cases model-free DrQ-v2 can rival model-based Dreamer-v2 in sample efficiency. There are several tasks, however, where Dreamer-v2 performs better.

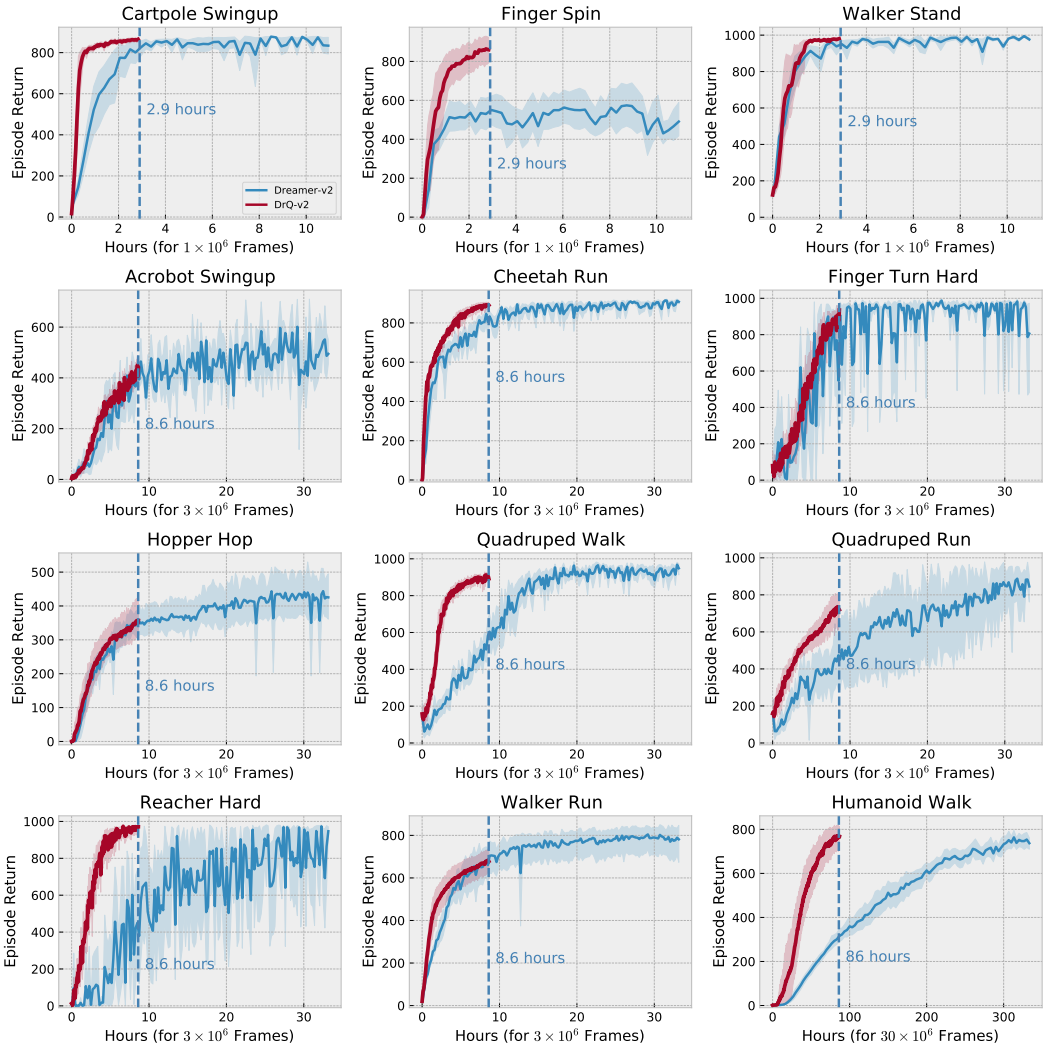


Figure 9: Model-based Dreamer-v2 performs more computations than model-free DrQ-v2. This allows DrQ-v2 to train faster in terms of wall-clock time and outperform Dreamer-v2 in this aspect.