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

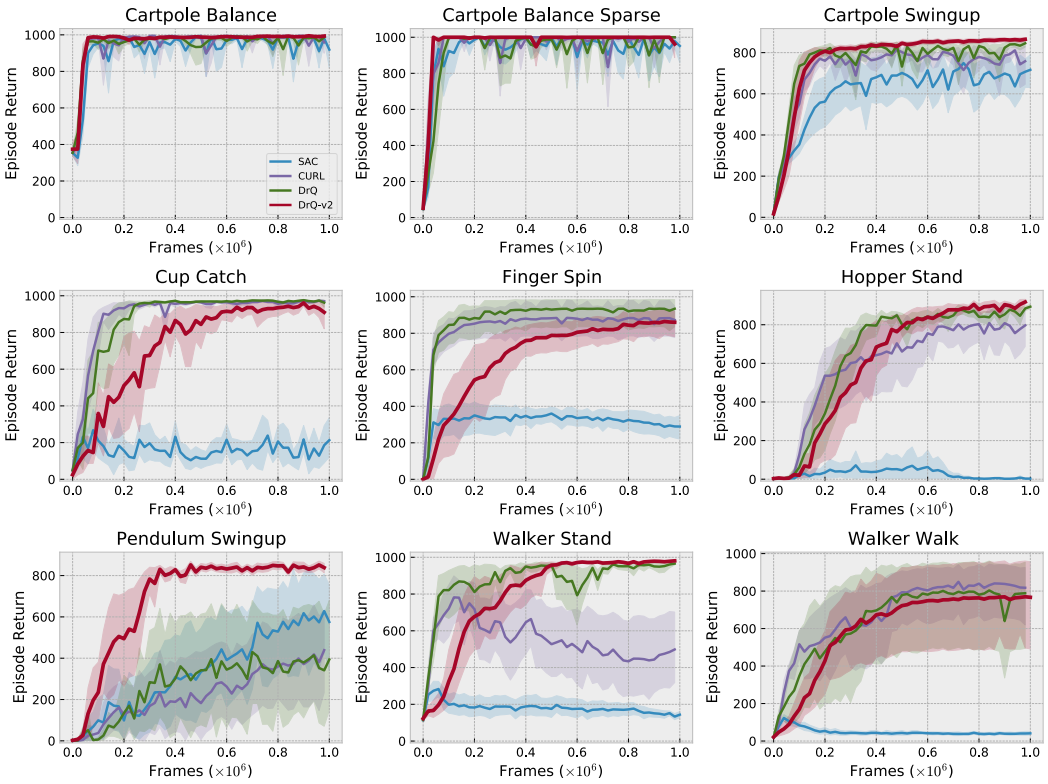

Figure 7: The *easy* benchmark consists of 9 tasks, where performance gains have been largely saturated by prior work. Still, DrQ-v2 is able to match sample complexity of the baselines. We note, that evaluation on these tasks is done for completeness reasons only, and encourage RL practitioners to refrain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

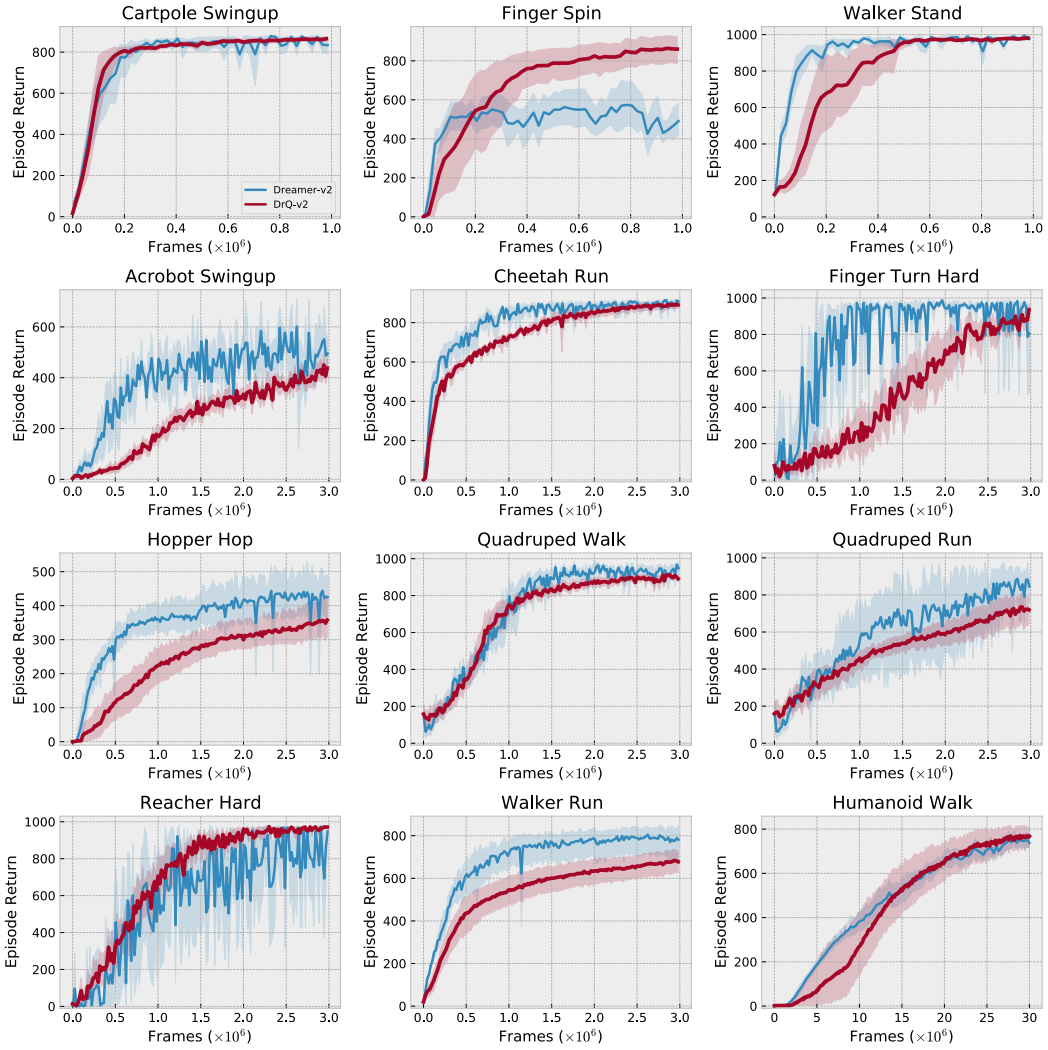

Figure 8: In many cases model-free DrQ-v2 can rival model-based Dreamer-v2 in sample efficiency. There are several tasks, however, where Dreamer-v2 performs better.

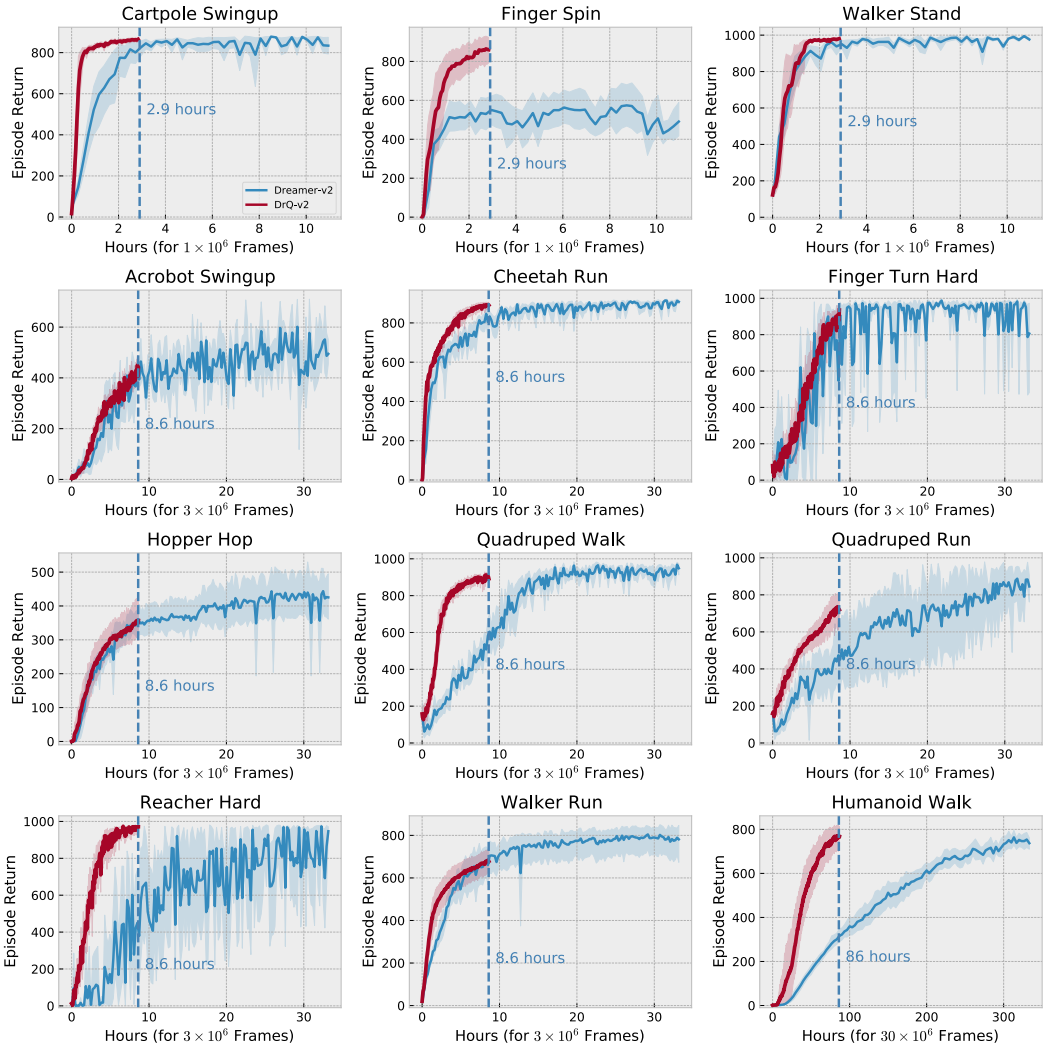

Figure 9: Model-based Dreamer-v2 performs more computations than model-free DrQ-v2. This allows DrQ-v2 to train faster in terms of wall-clock time and outperform Dreamer-v2 in this aspect.