MLRC Report

Reinforcement Learning
Autumn Semester, 2022-23

by **Debaditya Mukhopadhyay** and **Rahul Aditya**

Under the supervision of
Prof. **Aritra Hazra**

1 Introduction

It has long been difficult to develop sample-efficient continuous control algorithms for reinforcement learning that can view high-dimensional images. The RL community has made a tremendous effort on this topic, boosting sample efficiency dramatically. However, most of these methods have limitations such as the inability to solve complex problems in the DeepMind Control Suite, which is the benchmark for continuous control. Secondly, most of these methods are computationally heavy and hence need distributed GPU support.

DrQ-v2 is a model-free reinforcement learning (RL) algorithm for visual continuous control. It is based on the idea of data augmentations to better capture pixels in hard visual problems. DrQ-v2 provides significant improvements in sample efficiency across tasks from the DeepMind Control Suite. It is by far the best model-free algorithm to have solved such hard visual control problems. It is computationally efficient and usually requires around 8 hours of training for most of the medium-sized visual control tasks in the DeepMind Control Suite.

DrQ-v2 is significantly better than DrQ, the previous version of the same algorithm. The salient changes made to the algorithm are switching the base RL algorithm from SAC to DDPG (Deep Deterministic Policy Gradient), and making use of muti-step returns. It also uses the novel technique of bilinear interpolation that is based on averaging the values of nearby pixels after data augmentation. In addition to that, it uses the popular concept of exploration to exploitation by introducing noise that decays with time.

The main ingredient of DrQ-v2 is how it handles various hyperparameters that make it significantly better than not just its previous version DrQ but also most of the state-of-the-art model-free and model-based algorithms known to solve visual control problems. One of the most important hyperparameters that improved the results of DrQ-v2 is the size of the replay buffer. The other bottlenecks when improved made the model both in terms of stable training as well as learning wall-clock time are replay buffer management, data augmentation processing, batch size, and frequency of learning updates. It turns out that the final implementation is almost 3.5 times faster than the previous implementation. It also performed better than the previous implementation in terms of the number of throughput frames.

2 Methodology

DrQ-v2 is a model-free reinforcement learning (RL) algorithm for visual continuous control. DrQ-v2 is a development of DrQ, an off-policy actor-critic methodology that learns directly from pixels through data augmentation.

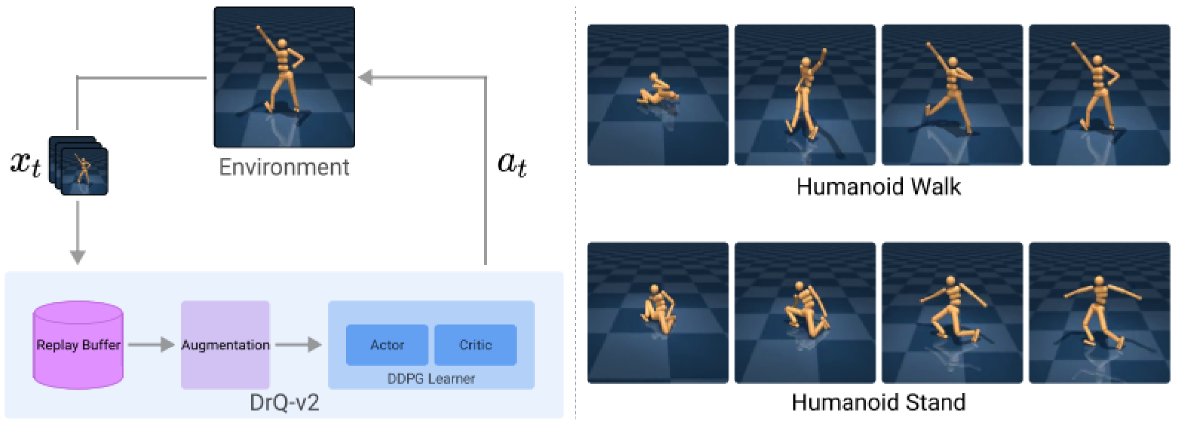

Figure 1: DrQ-v2 is displayed on the left as an image-based RL off-policy actor-critic algorithm. By adding random shift augmentation to pixel observations gathered from the replay buffer, it reduces encoder overfitting. Examples of the standing and walking motions that DrQ-v2 has mastered are shown on the right.

2.1 Image Augmentation

Image augmentation is done by applying random shifts. This is done by first padding the 84x84 pixels by 4 pixels on each side. A crop is then selected at random that gives an image with a +4 or -4 shift to the original image. It also uses the novel technique of bilinear interpolation that is based on averaging the values of nearby pixels after data augmentation. This change helped enhance the performance.

2.2 Image Encoder

A convolutional encoder is then used to embed the augmented picture observation into a low-dimensional latent vector. The encoder used is the same as that used in SAC-AE [Yarats et al., 2019]. This can be described by the function $\mathbf{h} = f(aug(\mathbf{x}))$. $aug(\mathbf{x})$ is the augmented image and f is the encoder.

2.3 Actor-Critic Algorithm

DDPG has been used as the foundational actor-critic RL algorithm. TD error is then calculated using n-step returns. This accelerates the spread of rewards and advances learning in general. n-step returns ensures performance and efficiency. Finally, double Q-learning has been used to reduce overestimation bias. Let's call the two functions to be learned are $\mathbf{Q}_{\theta_1}$ and $\mathbf{Q}_{\theta_2}$. From the replay buffer D, transitions are sampled as $\tau = (\mathbf{x}_{\mathrm{t}}, \mathbf{a}_{\mathrm{t}}, r_{\mathrm{t:t+n\text{-}1}}, \mathbf{x}_{\mathrm{t+n}})$. The losses are computed as $\mathrm{L}_{\theta_i} = \mathrm{E}_{\tau \sim D}$ $[\mathrm{Q}_{\theta_i}(\mathbf{x}_{\mathrm{t}}, \mathbf{a}_{\mathrm{t}})$ - y$]^2$, for i = 1, 2.

The TD target y in the above equation is $\mathrm{y} = \sum_{i=0}^{n-1} \gamma^{\mathrm{i}} \mathrm{r}_{\mathrm{t+i}} + \gamma^{\mathrm{n}} \min_{\mathrm{i=1,2}} \mathrm{Q}_{\theta_i}(\mathbf{x}_{\mathrm{t+n}}, \mathbf{a}_{\mathrm{t+n}})$. $\mathbf{h}_{\mathrm{t}}$ and $\mathbf{h}_{\mathrm{t+n}}$ are the encoder outputs as times t and t+n. $\mathbf{a}_{\mathrm{t+n}} = \pi_\phi(\mathbf{h}_{\mathrm{t+n}}) + \epsilon$ is the action as dictated by the policy with some additional noise to add the exploration component to the algorithm. ϵ is sampled from Normal Distribution with 0 mean and standard deviation that decays with time. Lastly, the actor is trained by the function: $\mathrm{L}_\phi(\mathrm{D}) = \text{-}\mathrm{E}_{\mathbf{x}_{\mathrm{t}} \sim D}$ $[\min_{\mathrm{i=1,\,2}} \mathrm{Q}_{\theta_i}(\mathbf{x}_{\mathrm{t}}, \mathbf{a}_{\mathrm{t}})]$.

2.4 Scheduled Exploration Noise

The noise in sampling action equation, textbfa$_{\mathrm{t+n}} = \pi_\phi(\mathbf{h}_{\mathrm{t+n}}) + \epsilon$, is such that $\epsilon \sim N(0, \sigma^2)$. σ is decayed with time. Intuitively, the agent is allowed to explore the possibility of fruitful returns. Later when it has learned significantly about the environment, it is made to deterministically optimize the return functions.

2.5 Key Parameters

The three most important hyperparameters that improved the performance are the size of the replay buffer, mini-batch size, and learning rate. The size of the replay buffer has been increased 10 times with the belief that it will allow the agent to learn more about the environment at once. The increased memory is adjusted with a smaller batch size. The mini-batch size has been set to 256. It does well against the standard algorithms in the domain that have the batch size set to 512. In doing so, there is no performance degradation. The learning rate has been reduced to 0.0001 from 0.001 with the belief that the learning will be more stable.

3 Results

3.1 Main Claim

We were able to reproduce most of the results by the authors (21 out of 24). We executed our codes on Param-Shakti which is based on V100 GPU. The entire DeepMind Control Suite was divided into easy (9), medium (12), and hard tasks (3).

Each Easy task took 3.5 hours whereas each medium one took around 10 hours. Which is in line with the author's claims of most tasks completing within 8 hours. In fact, we achieved over 110 frames per second when training our algorithms, which is higher than the 86 fps claimed by the authors, and much higher than the 28fps that DrQ had achieved.

We were not able to reproduce the 3 hard tasks because of restricted computational resources. Each hard tasks take around 86 hours of training according to the authors. As per our estimates, each would take around 100-105 hours for the required step count.

The experiment results that we reproduced against what has been claimed by the authors have been given below for easy and medium tasks in the DeepMind Control Suite. The first 9 tasks [Figures 2-5] are easy and the next 12 are medium [Figures 5-11].

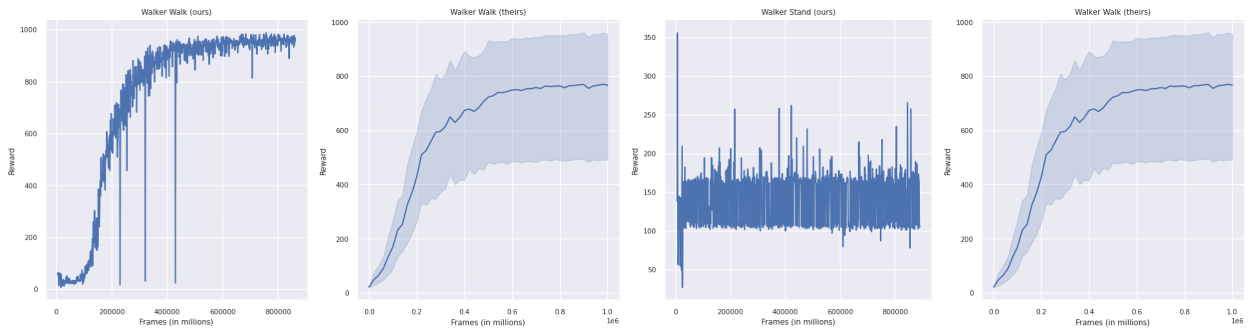

Figure 2: [Main Claim] Reward v/s Frames in millions for Walker Walk and Walker Stand tasks respectively. Our results are on the left and author's results on the right of each task.

3.2 Ablation Studies

As part of the Ablation Study, we implemented our own version of SAC (Soft Actor Critic), we studied the variation in the quality of the model when we varied n-step returns from 3 to 1 and 5, when we varied the buffer size from 1 Million Steps to 100k Steps and also by replacing the decaying exploration rate by a fixed learning rate. These results are available in the code base.

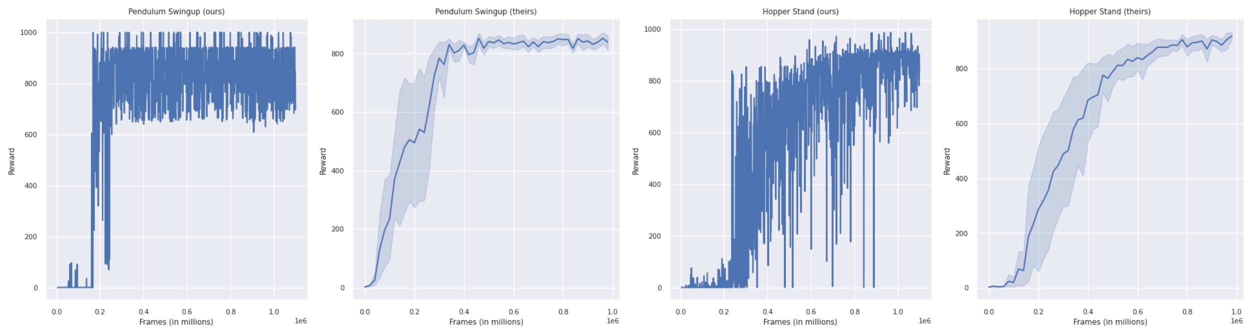

Figure 3: [Main Claim] Reward v/s Frames in millions for Pendulum Swingup and Hopper Stand tasks respectively. Our results are on the left and author's results on the right of each task.

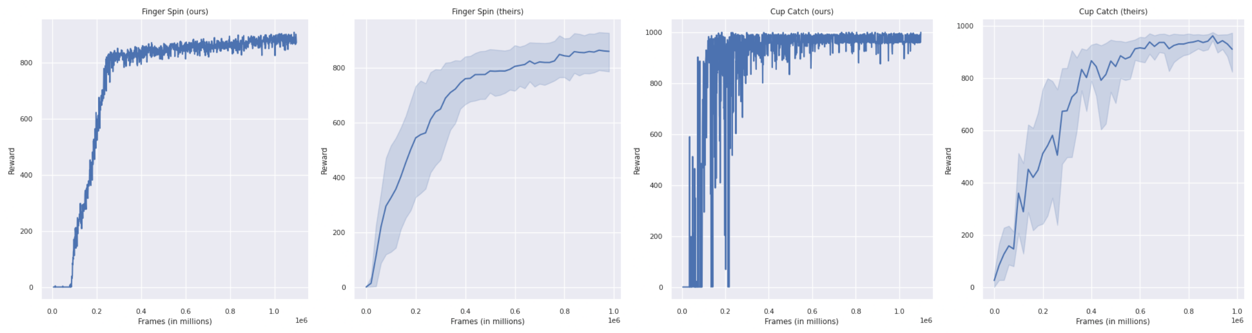

Figure 4: [Main Claim] Reward v/s Frames in millions for Finger Spin and Cup Catch tasks respectively. Our results are on the left and author's results on the right of each task.

3.2.1 Soft Actor Critic

The first version of DrQ used the Soft Actor Critic (SAC) algorithm to qualify the results of data augmentation. In DrQv2 the authors choose to replace the (SAC) Algorithm with Deep Deterministic Policy Gradient (DDPG). To find out the true advantage of this choice, we implement our own version of SAC (which was not available in their code-base), with an initial temperature of 0.1. We made the choice allow the agent to keep training with 3 step returns and the same encoder and augmenter that the DDPG algorithm enjoys. The authors on the other hand compared it with a much more vanilla SAC implementation. Here are the results.

in Figure 12, we observe the computational efficiency (the SAC algorithm took about 20% more time for the same number of frames), and the DDPG also gets a slightly better performance per frame than the SAC. However, we note that the benefits of using DDPG over SAC was overstated, and a SAC algorithm that is implemented with the same nuances as DrQv2 (as we do) actually performs very close to the DDPG performance. This opens up the possibility of other Actor Critic Algorithms having a better performance than even DDPG. This is left as a future task.

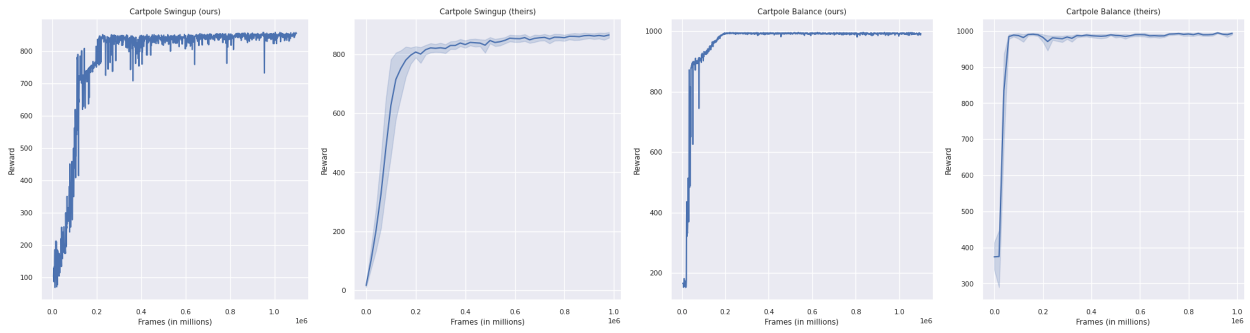

Figure 5: [Main Claim] Reward v/s Frames in millions for Cartpole Swingup and Cartpole Balance tasks respectively. Our results are on the left and author's results on the right of each task.

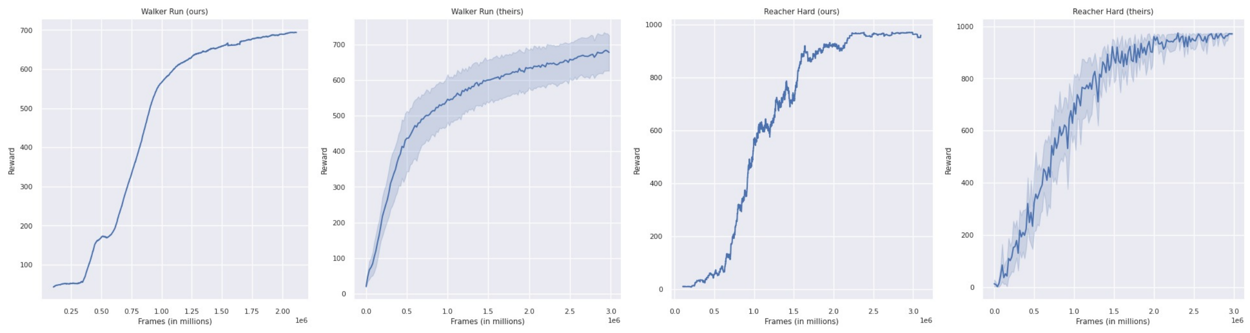

Figure 6: [Main Claim] Reward v/s Frames in millions for Walker run and Reacher Hard tasks respectively. Our results are on the left and author's results on the right of each task.

3.2.2 N Step Returns

The authors also incorporate n-step returns as a more sophisticated way to train the model. This is a step up from the 1 step returns DrQ used to enjoy. We compare 3 step returns with 1 step and 5 step returns to validate this choice.

in Figure 13, we see that their choice is validated, especially in acrobat. In the future however, more complex returns like TD(λ) can be used for better performance with the cost of computation time.

3.2.3 Buffer Size

The authors note that DrQ which was trained with a Experience Replay Buffer size of 100k steps suffers from a fate which afflicts many RL algorithms: Catastrophic interference. They believe that by increasing the buffer size to 10 times, the results would be much better. We compare the two.

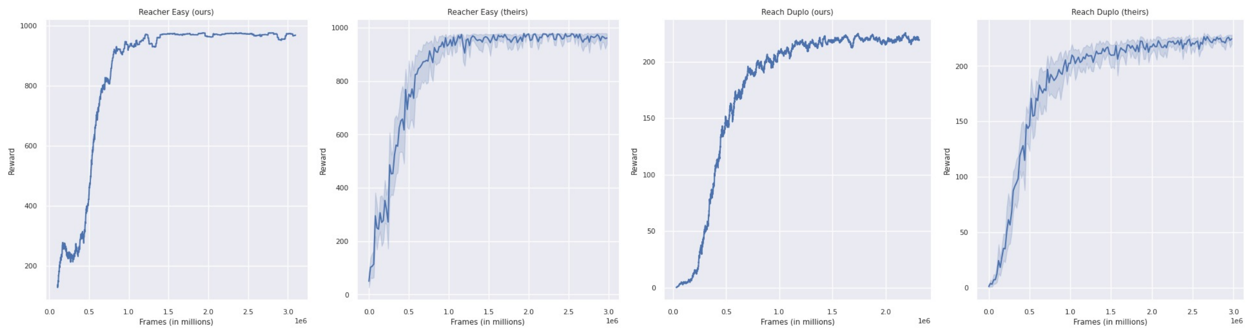

Figure 7: [Main Claim] Reward v/s Frames in millions for Reacher Hard and Reach Duplo tasks respectively. Our results are on the left and author's results on the right of each task.

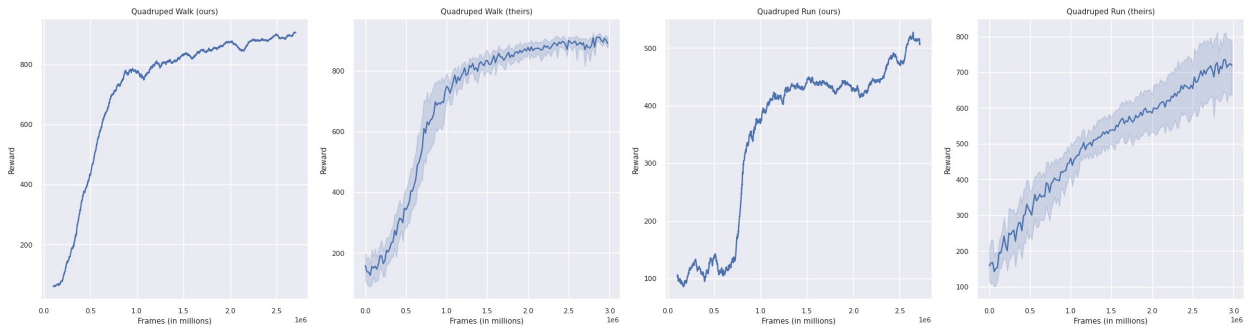

Figure 8: [Main Claim] Reward v/s Frames in millions for Quadruped Walk and Quadruped Run respectively. Our results are on the left and author's results on the right of each task.

in Figure 14, we see that this is true and it especially has an effect in tasks which are more reliant on memory.

3.2.4 Exploration Rate

For stability and convergence to an optimum solution, the authors use a linearly decaying schedule for the exploration rate, that goes from 1 to 0.1 linearly, where it reaches 0.1 on the final time step. We compare this with a constant exploration rate of $\epsilon = 0.2$, and we see the differences.

In Figure 15, we see that While the lower initial exploration rate ensures that the constant ϵ leads to more exploitative and hence higher reward choices in the beginning, eventually due to greater exploration, the decaying schedule finds a more optimum solutions and achieves a higher ceiling.

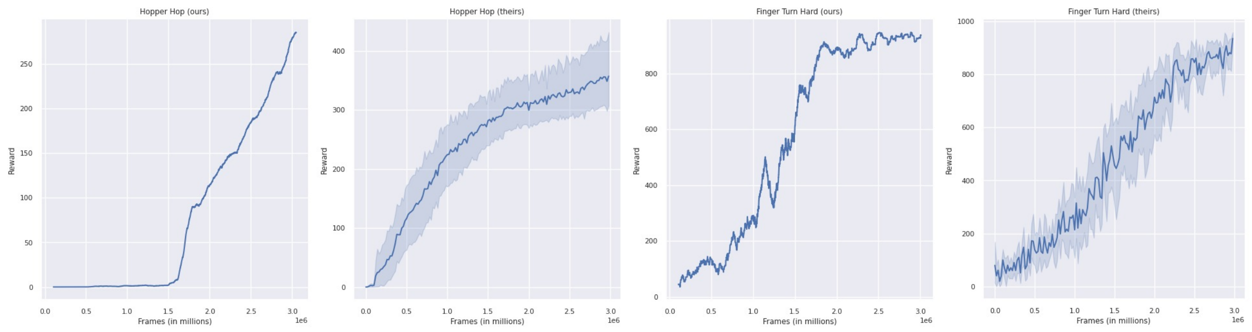

Figure 9: [Main Claim] Reward v/s Frames in millions for Hopper Hop and Finger Turn Hard tasks respectively. Our results are on the left and author's results on the right of each task.

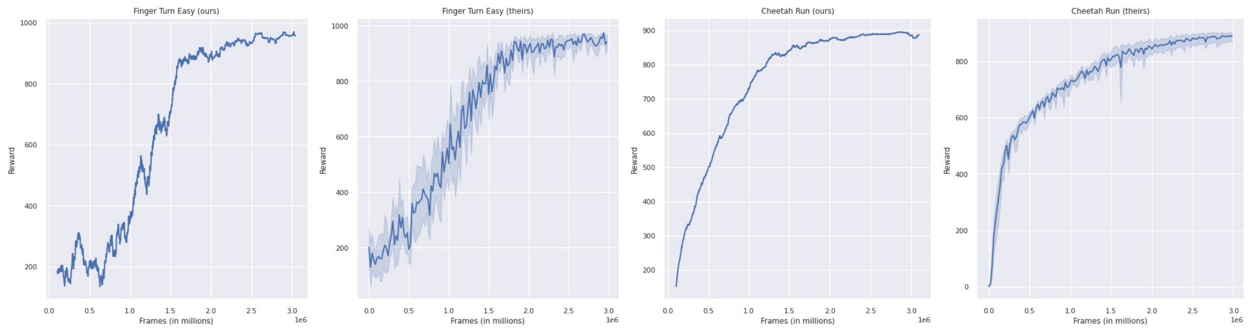

Figure 10: [Main Claim] Reward v/s Frames in millions for Finger Turn Easy and Cheetah Run tasks respectively. Our results are on the left and author's results on the right of each task.

4 Conclusion

We reproduced the code of DrQ-v2 which is by far the best implementation in model-free algorithms of hard visual tasks. The results that we got for easy and medium tasks of the DeepMind Suite closely resemble those of the authors. We developed our own implementation of the SAC algorithm to be able to do a more thorough analysis. We plan to implement further AC algorithms due to the encouraging results that we got. We performed multiple ablation studies to gain a grasp of which hyper-parameters make the model work well. This also gave us further exploratory ideas that can be explored, in terms of both hyper-parameters and algorithmic changes. We will use these leanings to further extend our analysis to these tasks in the future for the MLRC challenge.

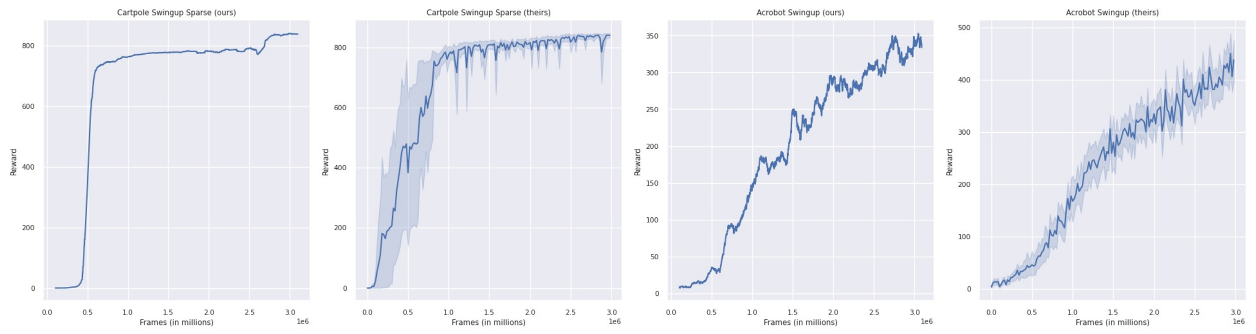

Figure 11: [Main Claim] Reward v/s Frames in millions for Cart-pole Swing-up Sparse and Acrobat Swing-up tasks respectively. Our results are on the left and author's results on the right of each task.

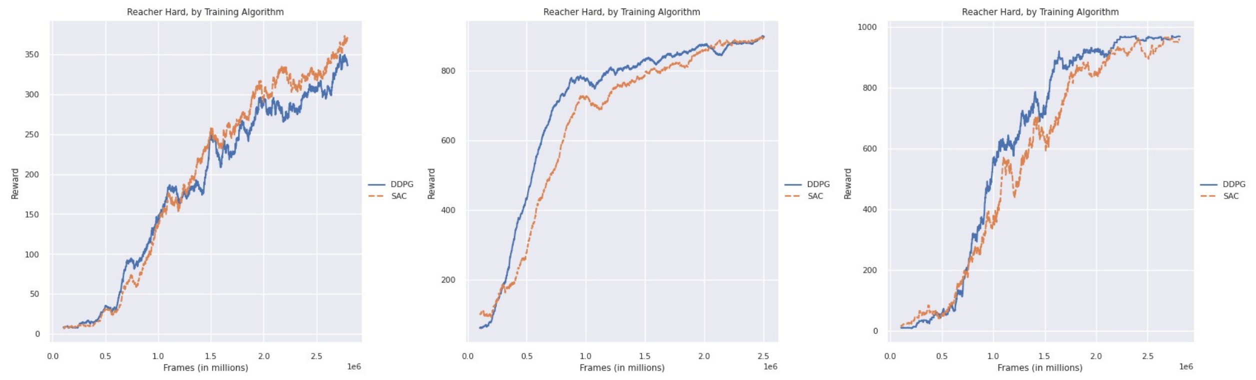

Figure 12: [Ablation] Comparing the performance of Soft Actor Critic and Deep Deterministic Policy Gradient in 3 tasks, Acrobat Swing Up, Quadrupled Walk and Reacher Hard

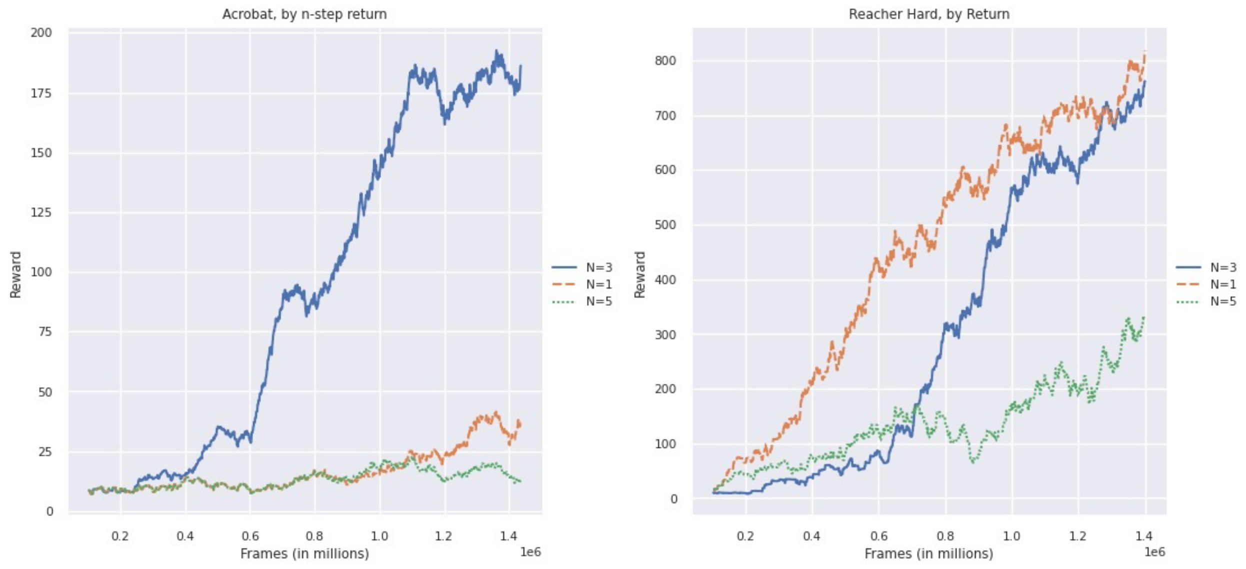

Figure 13: [Ablation] Comparing the performance of different n step returns, n=1,3,5 in 3 tasks, Acrobat Swing Up, Quadrupled Walk and Reacher Hard

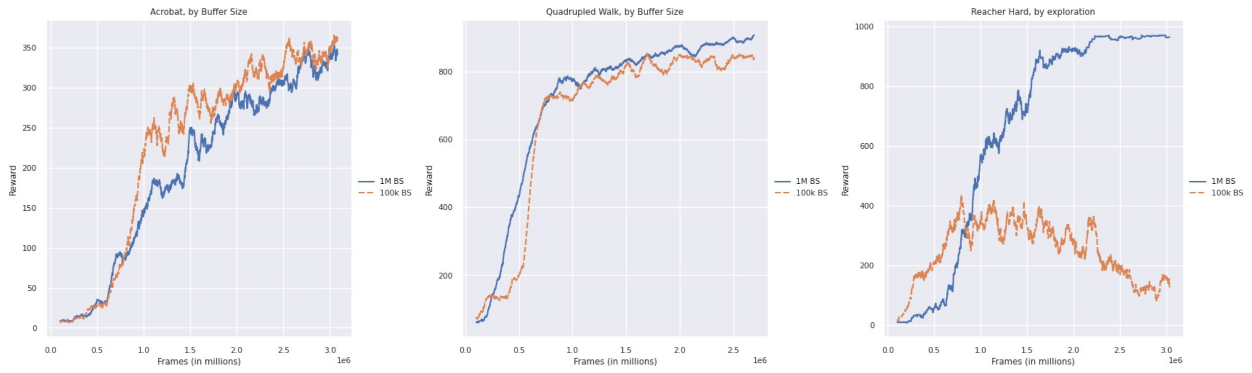

Figure 14: [Ablation] Comparing the performance of Experience Replay Size of 100k and 1 million time steps in 3 tasks, Acrobat Swing Up, Quadrupled Walk and Reacher Hard

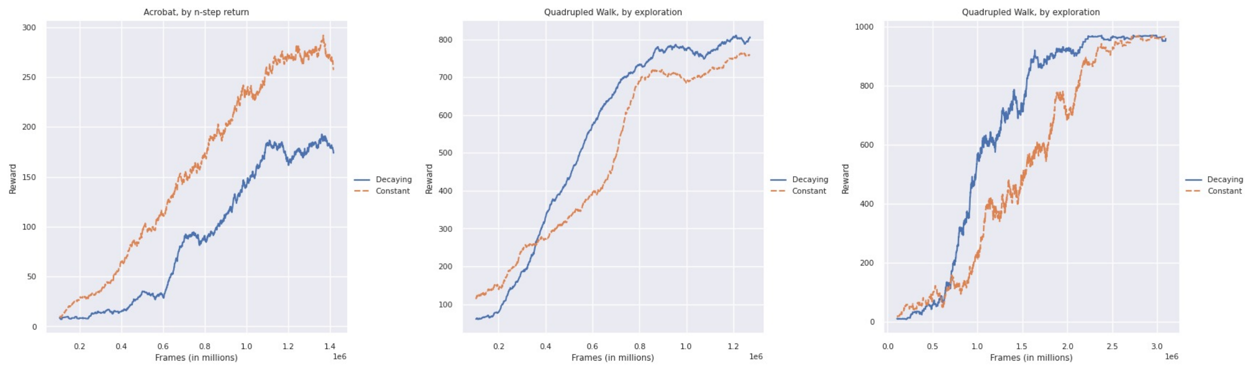

Figure 15: [Ablation] Comparing the performance of Linearly decaying exploration rate and constant ϵ in 3 tasks, Acrobat Swing Up, Quadrupled Walk and Reacher Hard