Ferdowsi University of Mashhad

Project Report

Zahra Niazi

Abstract

In this document, a report on the idea and implementation of the article *"Stabilizing Deep Q-Learning with ConvNets under Data Augmentation"* will be given. Suggestions for improving this implementation will be reviewed and finally the results of their implementation will be monitored.

Key words: Reinforcement Learning, Deep Learning, OpenAi Gym, Deepmind Control Suite, Continuous Control, physics simulation, Mujoco, Distracting Control Suite, dmc2gym

Contents

1 Preliminaries

1.1 dm-control[1]

Controlling the physical world is an essential aspect and arguably a prerequisite for achieving general intelligence. Primates, which have been engaging in two-handed manipulation for millions of years, provide the only known example of general-purpose intelligence. In contrast to board games, language, and other symbolic domains, physical tasks possess a fundamental characteristic of continuity in terms of state, time, and action. The dynamics of physical systems adhere to second-order equations of motion, where the state is comprised of positions and velocities. Sensory signals, or observations, carry meaningful physical units and exhibit variations over corresponding timescales. These unique properties, along with their widespread occurrence and significance, distinguish control problems as a distinctive subset within the realm of general Markov Decision Processes. Notably, in many physical control tasks, there exists a fixed subset of degrees of freedom within the agent's body that can be directly manipulated, while the remaining degrees of freedom belong to the surrounding environment. These "embodied" tasks represent the central focus of dm_control .

1.1.1 Tasks

In recent years, there has been significant advancement in the application of Reinforcement Learning (RL) to challenging problem domains, including video games. The Arcade Learning Environment (ALE) played a crucial role in facilitating these advancements by providing a standardized set of benchmarks for evaluating and comparing learning algorithms. Similarly, in the field of control and robotics, there is a need for well-designed task suites that serve as a standardized platform where different approaches can compete and new methods can emerge. The OpenAI Gym has become a widely adopted benchmark in continuous RL, offering a collection of continuous control domains. To meet the growing demand for task suites that enable the study of algorithms related to multi-scale control, multi-task transfer, and meta-learning, various recent task suites have been introduced, such as Meta-world, SURREAL, RLbench and IKEA. These task suites aim to provide diverse environments for experimentation. Similarly, dm_control offers its own set of control tasks, categorized into three distinct categories:

Control Suite

The DeepMind Control Suite, is built directly with the MuJoCo wrapper, provides a set of standard benchmarks for continuous control problems. The unified reward structure offers interpretable learning curves and aggregated suite-wide performance measures. Furthermore, it emphasises high-quality, well-documented code using uniform design patterns, offering a readable, transparent and easily extensible codebase.

Locomotion

The Locomotion framework is designed to facilitate the implementation of a wide range of locomotion tasks for RL algorithms by introducing self-contained, reusable components which compose into different task variants.

Manipulation

We also provide examples of constructing robotic manipulation tasks. These tasks involve grabbing and manipulating objects with a 3D robotic arm. The set of tasks includes examples of reaching, placing, stacking, throwing, assembly and disassembly. The tasks are designed to be solved using a simulated 6 degree-of-freedom robotic arm based on the Kinova Jaco, though their modular design permit the use of other arms with minimal changes. These tasks make use of reusable components such as bricks that snap together, and provide examples of reward functions for manipulation. Tasks can be run using vision, low-level features, or combinations of both.

1.2 The Distracting Control Suite[2]

Robotic systems often encounter demanding perceptual conditions, such as variations in viewpoint, lighting, and background. However, existing simulated reinforcement learning benchmarks, like DM Control, lack such complexities in visual input. Consequently, the performance of well-established methods on these benchmarks may not translate effectively to real-world scenarios. Here this limitation is addressed by expanding DM Control to include three types of visual distractions: variations in background, color, and camera pose. This augmented benchmark serves as a challenging testbed for vision-based control, and the study evaluates state-of-the-art RL algorithms within these settings. The experimental results reveal that current RL methods for vision-based control exhibit subpar performance in the presence of distractions. Furthermore, the performance deteriorates as the complexity of distractions increases, highlighting the necessity for novel approaches capable of handling the visual intricacies encountered in real-world environments. Additionally, the research findings indicate that combinations of multiple distraction types pose greater difficulties than merely aggregating their individual effects.

A significant challenge in perception is the ability to extract task-relevant information from sensory input while filtering out distractions that may introduce misleading correlations in subsequent tasks. However, DM Control lacks such distractions, as the agent is presented with a consistent camera view, fixed lighting conditions, and a static background. Since any observation change in DM Control corresponds directly to a change in a task-relevant state variable, it does not allow for the measurement or development of the capability to filter out irrelevant variations through perception.

To address this limitation, the Distracting Control Suite is introduced, which is an expansion of DM Control specifically designed with real-world robot learning in mind. This extension incorporates three distinct types of distractions: random color changes applied to all objects in the scene, random video backgrounds, and random continuous variations in camera pose. Each distraction can be implemented in a static setting, where changes occur only during episode transitions, or in a dynamic setting, where distractions change smoothly between frames. Furthermore, the difficulty of each distraction can be adjusted, ranging from barely noticeable to highly distracting. Additionally, all three types of distractions can be combined in any desired manner, allowing for arbitrary combinations of distractions.

1.3 DMControl Generalization Benchmark[3]

We assess the performance of our method on a set of tasks taken from the DeepMind Control Suite (DMControl), as well as in the domain of robotic manipulation. The DMControl Suite consists of a diverse range of challenging continuous control tasks and is widely recognized as a benchmark for vision-based RL. To measure the generalization capabilities of our method, a new benchmark called DMControl Generalization Benchmark (DMControl-GB) is introduced, which is built upon DMControl. In this benchmark, agents are trained in a fixed environment referred to as the training environment, and we evaluate their generalization performance on two distinct test distributions: (1) environments with randomized colors and (2) environments with natural videos serving as backgrounds. These test distributions represent the color hard and video easy benchmarks within DMControl-GB.

While DMControl-GB provides a solid platform for evaluating algorithm performance, our ultimate objective is to develop algorithms capable of solving real-world problems using vision-based RL. To better emulate real-world deployment scenarios, we also consider a robotic manipulation task involving a robotic arm in a simulated environment. Similar to DMControl-GB, agents are trained in a fixed environment and then evaluated in environments with randomized colors and video backgrounds. Additionally, we introduce random perturbations to camera settings, lighting conditions, and texture variations during testing to simulate real-world conditions.

2 Stabilized Q-Value Estimation under Augmentation[4]

Reinforcement Learning (RL) with visual observations has shown impressive success in various applications. However, the challenge lies in generalizing the learned skills to new environments, especially in high-dimensional observation spaces like images. To tackle this, researchers have explored domain randomization and data augmentation techniques to increase training data variability and promote invariant policies. Simple augmentations like cropping and translation have improved sample efficiency, but additional augmentation can decrease efficiency and lead to divergence. Balancing stability and generalization in RL requires careful trial and error, as the introduction of diverse data poses optimization challenges and risks instability, distinguishing it from supervised learning approaches.

To address these problems, **SVEA** (**S**tabilized Q-**V**alue **E**stimation under **A**ugmentation) is proposed. It is a simple yet effective framework for data augmentation in off-policy RL that greatly improves stability of Q-value estimation.

2.1 Method

The method introduced is called SVEA: **S**tabilized Q-**V**alue **E**stimation under **A**ugmentation, which is a comprehensive framework for achieving visual generalization in reinforcement learning (RL) through the application of data augmentation. SVEA utilizes a novel learning approach that incorporates two distinct data streams: one with augmented data and another without augmentation. This method seamlessly integrates with any standard off-policy RL algorithm without requiring modifications to the underlying neural network that parameterizes the policy. Additionally, it does not involve additional forward passes, auxiliary tasks, or learnable parameters. While SVEA in principle does not make any assumptions about the structure of states $\mathbf{s}_t \in \mathcal{S}$, this method is described within the context of image-based RL.

2.1.1 Architectural Overview

This approach utilizes common neural network architectures in off-policy RL without introducing additional learnable parameters. In this method, the neural network layers and corresponding learnable parameters of the state-action value function are divided into two subnetworks:

- f_θ referred to as the state encoder
- Q_θ referred to as the Q-function

This subdivision allows us to predict the Q-value, q_t, for a given state-action pair $(\mathbf{s}_t, \mathbf{a}_t)$ as:

$$q_t \triangleq Q_\theta(f_\theta(\mathbf{s}_t), \mathbf{a}_t)$$

Similarly, the target Q-value for $(\mathbf{s}_t, \mathbf{a}_t)$ is defined as:

$$q_t^{\text{tgt}} \triangleq r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \max_{\mathbf{a}'_t} Q_\psi^{\text{tgt}}(f_\psi^{\text{tgt}}(\mathbf{s}_{t+1}), \mathbf{a}')$$

where γ is the discount factor, and ψ represents parameters defined as an exponential moving average of θ. Depending on the chosen algorithm, we may also incorporate a parameterized policy, π_θ, that shares the encoder parameters with Q_θ and selects actions $\mathbf{a}_t \sim \pi_\theta(\cdot|f_\theta(\mathbf{s}_t))$.

To avoid incorrect bootstrapping resulting from augmented data, data augmentation is strictly applied only in the estimation of Q-values for the current state, $\mathbf{s}_t$, and not to the successor state, $\mathbf{s}_{t+1}$, used for bootstrapping with Q_ψ^{tgt} (and π_θ if applicable). To mitigate over-regularization during the optimization of f_θ and Q_θ, we utilize a modified Q-objective that incorporates both augmented and unaugmented data.

2.1.2 Learning Objective

This method redefines the temporal difference objective to better leverage data augmentation. First, recall that $q_t^{\text{tgt}} = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \max_{\mathbf{a}'_t} Q_\psi^{\text{tgt}}(f_\psi^{\text{tgt}}(\mathbf{s}_{t+1}), \mathbf{a}')$. Instead of learning to predict q_t^{tgt} only from state $\mathbf{s}_t$, it is proposed to minimize a nonnegative linear combination of $\mathcal{L}_Q$ over two individual data streams, $\mathbf{s}_t$ and $\mathbf{s}_t^{\text{aug}} = \tau(\mathbf{s}_t, \nu),\ \nu \sim \mathcal{V}$, which is then defined as the objective

$$\mathcal{L}_Q^{\text{SVEA}}(\theta, \psi) \triangleq \alpha \mathcal{L}_Q\left(\mathbf{s}_t, q_t^{\text{tgt}}; \theta, \psi\right) + \beta \mathcal{L}_Q\left(\mathbf{s}_t^{\text{aug}}, q_t^{\text{tgt}}; \theta, \psi\right) \tag{1}$$

$$= \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1} \sim \mathcal{B}}\left[\alpha \left\| Q_\theta(f_\theta(\mathbf{s}_t), \mathbf{a}_t) - q_t^{\text{tgt}} \right\|_2^2 + \beta \left\| Q_\theta(f_\theta(\mathbf{s}_t^{\text{aug}}), \mathbf{a}_t) - q_t^{\text{tgt}} \right\|_2^2\right], \tag{2}$$

where α, β are constant coefficients that balance the ratio of the unaugmented and augmented data streams, respectively, and q_t^{tgt} is computed strictly from unaugmented data. $\mathcal{L}_Q^{\text{SVEA}}(\theta, \psi)$ serves as a *data-mixing* strategy that **oversamples unaugmented data as an implicit variance reduction technique**. Data-mixing is a simple and effective technique for variance reduction that works well in tandem with the modifications proposed for bootstrapping. For $\alpha = \beta$, the objective in Eq. 2 can be evaluated in a single, batched forward-pass by rewriting it as:

$$\mathbf{g}_t = [\mathbf{s}_t, \tau(\mathbf{s}_t, \nu)]_{\text{N}} \tag{3}$$

$$h_t = \left[q_t^{\text{tgt}}, q_t^{\text{tgt}}\right]_{\text{N}} \tag{4}$$

$$\mathcal{L}_Q^{\text{SVEA}}(\theta, \psi) = \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1} \sim \mathcal{B},\ \nu \sim \mathcal{V}}\left[(\alpha + \beta) \left\| Q_\theta(f_\theta(\mathbf{g}_t), \mathbf{a}_t) - h_t \right\|_2^2\right], \tag{5}$$

2.2 Setup

The method and baselines are implemented using the Soft Actor-Critic (SAC) algorithm as the base. By default, random shift augmentation is incorporated into all methods. The base algorithm is referred to as the "unaugmented" version and we examine its stability under additional data augmentation. For all methods (when applicable), identical network architectures and hyperparameters are employed. Observations consist of stacked RGB frames, with a size of $84 \times 84 \times 3$. In the DMControl-GB and DistractingCS benchmarks, we train all methods for 500k frames and evaluate them on the same 5 tasks used in prior work. The same experimental setup is also applied to the robotic manipulation experiments.

2.3 Implementation Details

Network architecture. For experiments in DMControl, the network architecture from [3] is adopted, without any changes to the architecture nor hyperparameters. The shared encoder f_θ is implemented as an 11-layer CNN encoder that takes a stack of RGB frames rendered at $84 \times 84 \times 3$ and outputs features of size $32 \times 21 \times 21$, where 32 is the number of channels and 21×21 are the dimensions of the spatial feature maps. All convolutional layers use 32 filters and 3×3 kernels. The first convolutional layer uses a stride of 2, while the remaining convolutional layers use a stride of 1. Following previous work on image-based RL for DMControl tasks, the shared encoder is followed by independent linear projections for the actor and critic of the Soft Actor-Critic base algorithm used in our experiments, and the actor and critic modules each consist of three fully connected layers with hidden dimension 1024. Training takes approximately 24 hours on a single NVIDIA V100 GPU.

3 Experiments

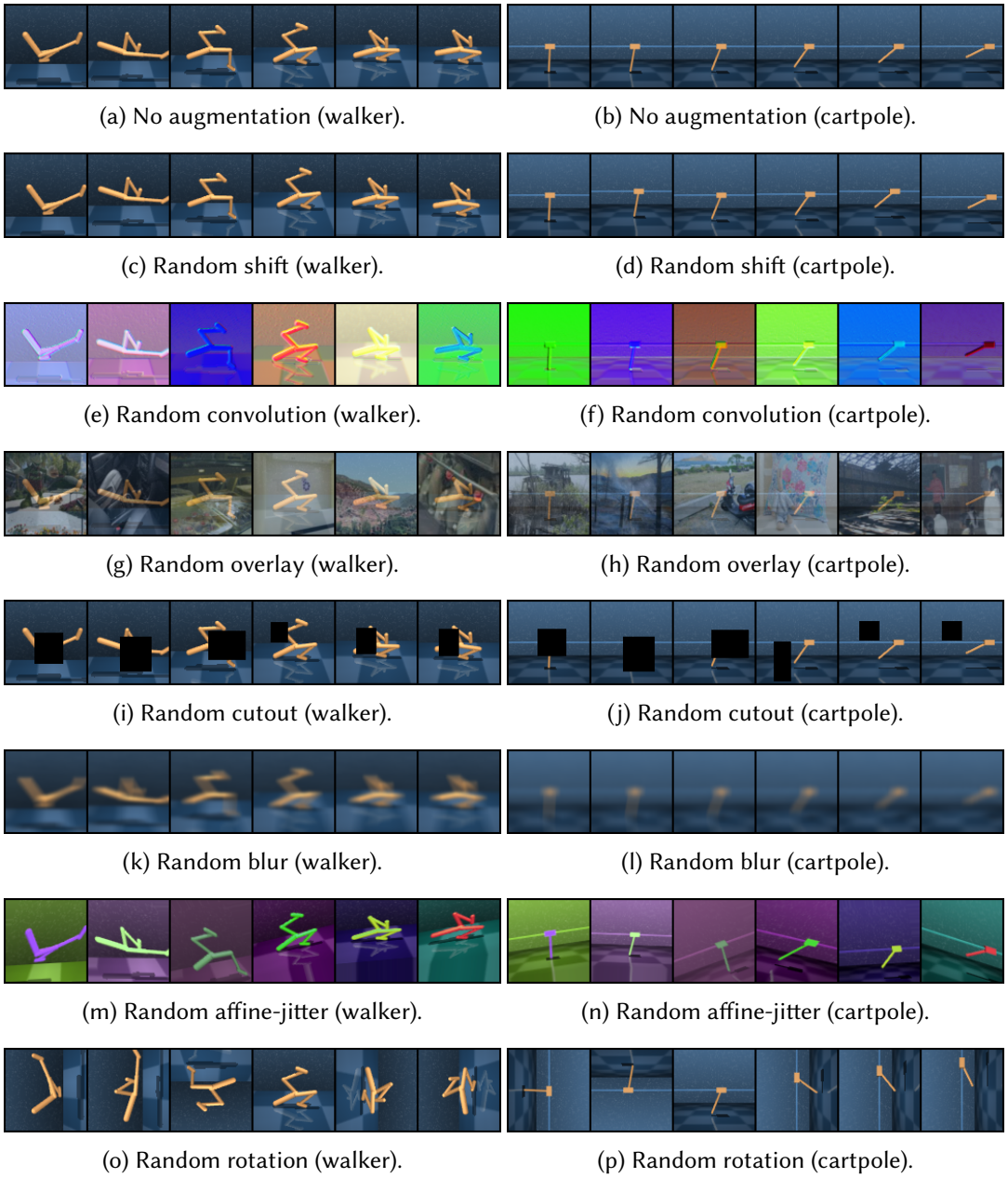
A comprehensive evaluation has been conducted of this method and a set of robust baselines, employing Convolutional Neural Networks (ConvNets), on tasks sourced from the DeepMind Control Suite (DMControl) and a collection of robotic manipulation tasks. DMControl provides a range of challenging and diverse continuous control tasks and is widely recognized as a benchmark for image-based RL. In order to assess the generalization capabilities of our method and the baselines, we subject them to rigorous distribution shifts from the **DMControl Generalization Benchmark** (DMControl-GB), the **Distracting Control Suite** (DistractingCS), as well as distribution shifts specific to the robotic manipulation environment. Our evaluation encompasses the analysis of *sample efficiency*, *asymptotic performance*, and *generalization performance* of the methods across these varied scenarios.

3.1 Data Augmentation in RL.

Figure 1 provides a comprehensive set of samples for each of the data augmentations considered in this study:

- random *shift*
- random convolution (denoted *conv*)
- random *overlay*
- random *cutout*
- Gaussian *blur*
- random *affine-jitter*
- random *rotation*

It is emphasized that the random convolution augmentation is not a convolution operation, but rather application of a randomly initialized convolutional layer as in the original proposal. As in previous work that applies data augmentation to image-based RL, we either clip values or apply a logistic function, whichever is more appropriate, to ensure that output values remain within the $[0, 1)$ interval that unaugmented observations are normalized to.
Each of the considered data augmentations are applied to the *walker* and *cartpole* environments and are representative of the *Walker, walk*, *Walker, stand*, *Cartpole, swingup*, and *Cartpole, balance* tasks. To illustrate the diversity of augmentation parameters associated with a given transformation, we provide a total of 6 samples for each data augmentation in each of the two environments.



(a) No augmentation (walker). (b) No augmentation (cartpole).

(c) Random shift (walker). (d) Random shift (cartpole).

(e) Random convolution (walker). (f) Random convolution (cartpole).

(g) Random overlay (walker). (h) Random overlay (cartpole).

(i) Random cutout (walker). (j) Random cutout (cartpole).

(k) Random blur (walker). (l) Random blur (cartpole).

(m) Random affine-jitter (walker). (n) Random affine-jitter (cartpole).

(o) Random rotation (walker). (p) Random rotation (cartpole).

Figure 1: **Data augmentation**. Visualizations of all data augmentations considered in this study. Left column contains samples from the *Walker, walk* and *Walker, stand* tasks, and right column contains samples from the *Cartpole, swingup* and *Cartpole, balance* tasks.

The utilization of data augmentation in image-based reinforcement learning (RL) has demonstrated remarkable success by enhancing generalization and regularizing the network param-

eters of the Q-function and policy π. However, not all types of augmentations yield equal effectiveness. Studies show that small random crops and random shifts (image translations) significantly enhance the sample efficiency of image-based RL. However, they do not offer significant improvements in generalization to other environments. Conversely, augmentations such as random convolution exhibit substantial potential in improving generalization but are concurrently associated with instability and reduced sample efficiency. In this context, it is crucial to distinguish between *weak* augmentations, such as small random translations, which enhance *sample efficiency* through regularization, and *strong* augmentations, such as random convolution, which improve *generalization* at the cost of sample efficiency. This study specifically focuses on stabilizing deep Q-learning when applying strong data augmentation, aiming to enhance generalization capabilities.

Figure 2 shows training and test performance of SVEA implemented using each of the 6 data augmentations considered in this work. SVEA exhibits comparable stability and sample efficiency for all augmentations, but we find that generalization ability on the `color_hard` benchmark of DMControl-GB is highly dependent on the choice of augmentation. Generally, we observe that augmentations such as *conv*, *overlay*, and *affine-jitter* achieve the best generalization, but they empirically also cause the most instability in our *DrQ + aug* baseline as shown in Figure 5.

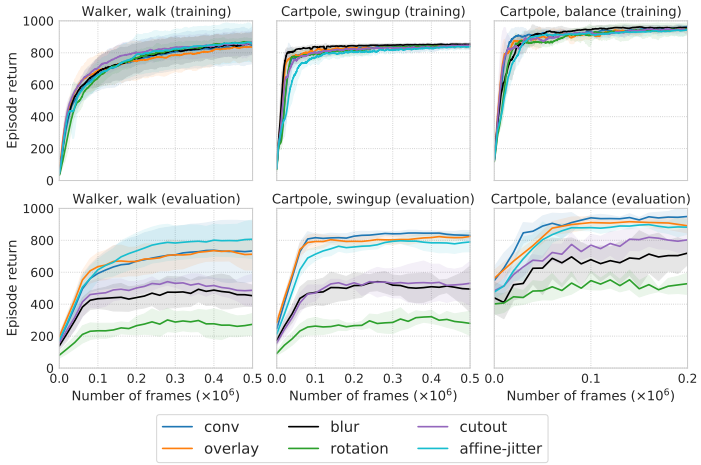


Figure 2: **Generalization depends on the choice of data augmentation.** A comparison of SVEA implemented using each of the 6 data augmentations considered in this work (using ConvNets). SVEA exhibits comparable stability and sample efficiency for all augmentations, but generalization ability is highly dependent on the choice of augmentation. *Top:* episode return on the training environment during training. *Bottom:* generalization measured by episode return on the `color_hard` benchmark of DMControl-GB. Mean of 5 seeds, shaded area is ± 1 std. deviation.

Stability

We evaluate the stability of SVEA under 6 common data augmentations; results are shown in Figure 3. SVEA is relatively unaffected by the **choice of data augmentation** and improves sample efficiency in **27** out of **30** instances. We further ablate each component of SVEA in Figure 4; we find that both components are key to SVEA's success. We observe that SVEA improves stability in all 27 instances where DrQ is impaired by data augmentation.

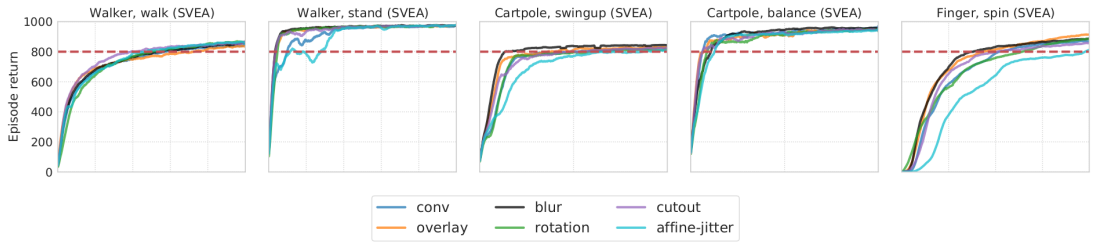


Figure 3: **Data augmentations.** Training performance of SVEA under 6 common data augmentations. Mean of 5 seeds. Red line at 800 return is for visual guidance only. We omit visualization of std. deviations for clarity, but provide per-augmentation comparisons to DrQ (including std. deviations) across all tasks in Figure 5, and test performances in Figure 1.

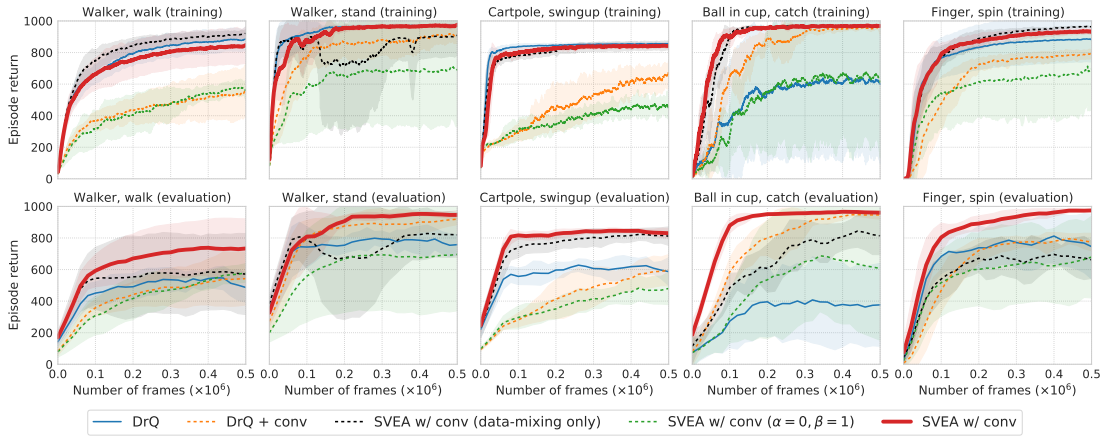


Figure 4: **Training and test performance.** We compare SVEA to DrQ with and without random convolution augmentation, as well as a set of ablations. *Data-mixing only* indiscriminately applies our data-mixing strategy to all data streams, and $(\alpha = 0, \beta = 1)$ only augments Q-predictions but without data-mixing. We find both components to contribute to SVEA's success. *Top:* episode return on the training environment during training. *Bottom:* generalization measured by episode return on the `color_hard` benchmark of DMControl-GB. Mean of 5 seeds, shaded area is ± 1 std. deviation.

Stability under data augmentation

Figure 5 compares the sample efficiency and stability of SVEA and DrQ under each of the 6 considered data augmentations for 5 tasks from DMControl. Stability of DrQ under data augmentation is found to be highly sensitive to both the choice of augmentation and the particular task. For example, the *DrQ + aug* baseline is relatively unaffected by a majority of data augmentations in the *Walker*, *stand* task, while we observe significant instability across all data augmentations in the *Cartpole*, *swingup* task. Our results therefore indicate that SVEA can be a highly effective method for eliminating the need for costly trial-and-error associated with application of data augmentation.

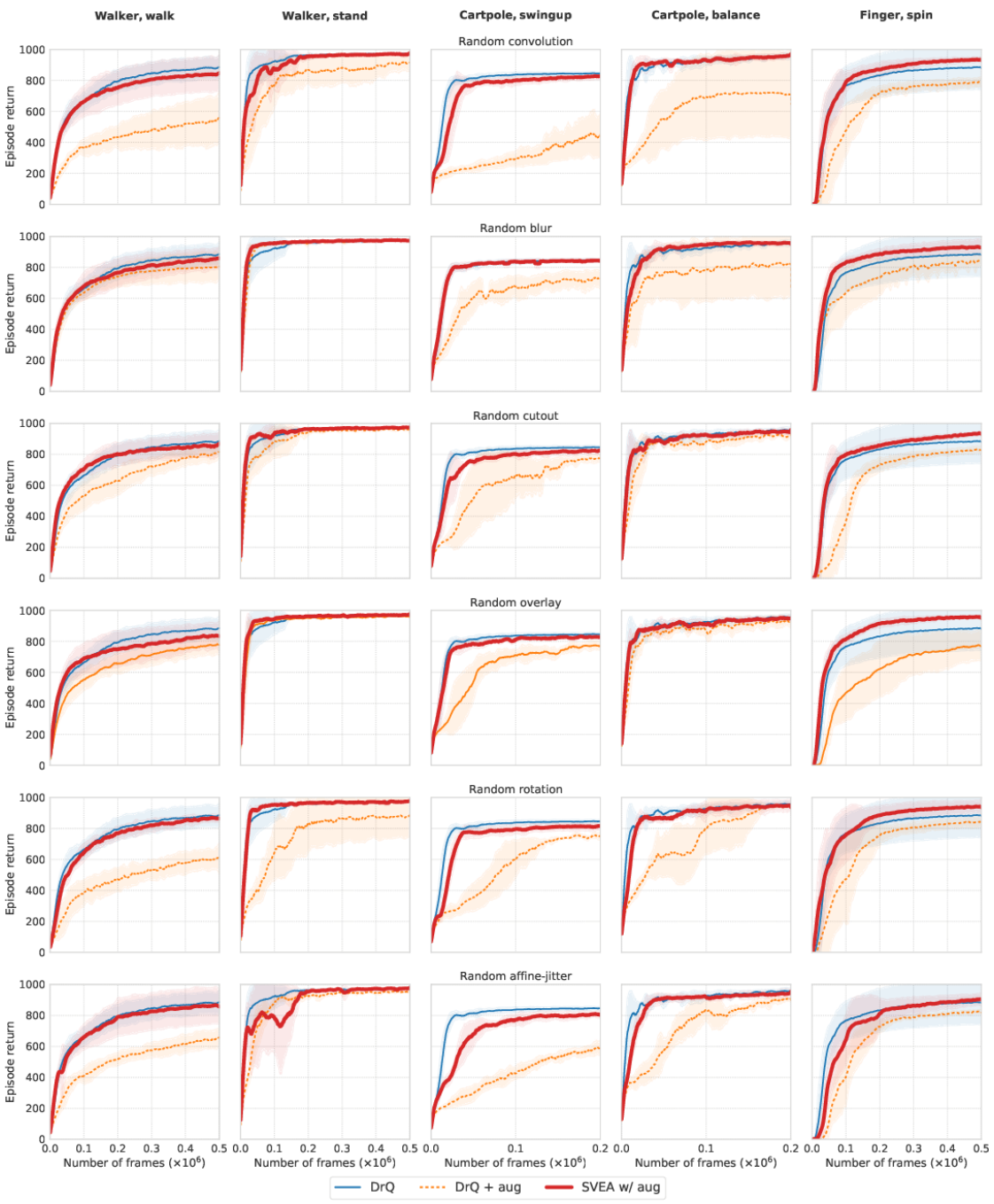


Figure 5: **Stability under data augmentation.** Training performance measured by episode return of SVEA and DrQ under 6 common data augmentations (using ConvNets). We additionally provide reference curves for DrQ without additional augmentation. Mean of 5 seeds, shaded area is ± 1 std. deviation. SVEA obtains similar sample efficiency to DrQ without augmentation, while the sample efficiency of *DrQ + aug* is highly dependent on the task and choice of augmentation.

3.2 Generalization.

Here the test performance of SVEA is compared to 5 recent state-of-the-art methods for image-based RL on the `color_hard` and `video_easy` benchmarks from DMControl-GB, as well as the extremely challenging DistractingCS benchmark, where camera pose, background, and colors are continually changing throughout an episode. Here *conv* and *overlay* augmentations are used and we report additional results on the `video_hard` benchmark. SVEA outperforms all methods considered in $\mathbf{12}$ out of $\mathbf{15}$ instances on DMControl-GB, and at a lower computational cost than CURL, PAD, and SODA that all learn auxiliary tasks. On DistractingCS, we observe that SVEA improves generalization by $\mathbf{42}\%$ at low intensity, and its generalization degrades significantly slower than DrQ for high intensities. While generalization depends on the particular choice of data augmentation and test environments, this is an encouraging result considering that SVEA enables efficient policy learning with stronger augmentations than previous methods.

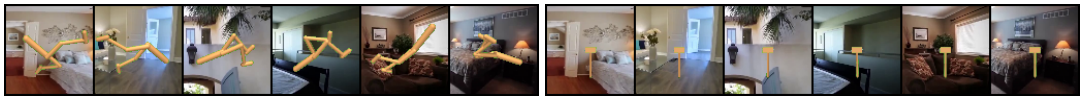
3.3 Test Environments

Figure 6 provides visualizations for each of the two generalization benchmarks, DMControl Generalization Benchmark and Distracting Control Suite, used in our experiments. Agents are trained in a fixed training environment with no visual variation, and are expected to generalize to novel environments of varying difficulty and factors of variation. The `color_hard`, `video_easy`, and `video_hard` benchmarks are from DMControl Generalization Benchmark, and we further provide samples from the Distracting Control Suite (DistractingCS) benchmark for intensities $I = \{0.1, 0.2, 0.5\}$. While methods are evaluated on a larger set of intensities, we here provide samples deemed representative of the intensity scale. We note that the DistractingCS benchmark has been modified to account for action repeat (frame-skip). Dynamically changing the environment at each simulation step makes the benchmark disproportionally harder for tasks that use a large action repeat, e.g. *Cartpole* tasks. Therefore, we choose to modify the DistractingCS benchmark and instead update the distractors every second simulation step, corresponding to the lowest action repeat used (2, in *Finger, spin*). This change affects both SVEA and baselines equally. Figure 7 shows generalization results on DistractingCS for each task individually. We find that the difficulty of DistractingCS varies greatly between tasks, but SVEA consistently outperforms DrQ in terms of generalization across all intensities and tasks.

(a) Training environment (walker). (b) Training environment (cartpole).

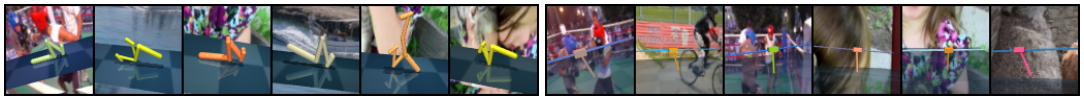
(c) `color_hard` environment (walker). (d) `color_hard` environment (cartpole).

(e) `video_easy` environment (walker). (f) `video_easy` environment (cartpole).

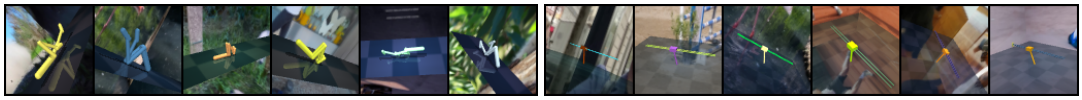


(g) `video_hard` environment (walker). (h) `video_hard` environment (cartpole).

(i) DistractingCS for intensity 0.1 (walker). (j) DistractingCS for intensity 0.1 (cartpole).



(k) DistractingCS for intensity 0.2 (walker). (l) DistractingCS for intensity 0.2 (cartpole).



(m) DistractingCS for intensity 0.5 (walker). (n) DistractingCS for intensity 0.5 (cartpole).

Figure 6: **Test environments**. Samples from each of the two generalization benchmarks, DM-Control Generalization Benchmark and Distracting Control Suite, considered in this study. In our experiments, agents are trained in a fixed training environment with no visual variation, and are expected to generalize to novel environments of varying difficulty and factors of variation.

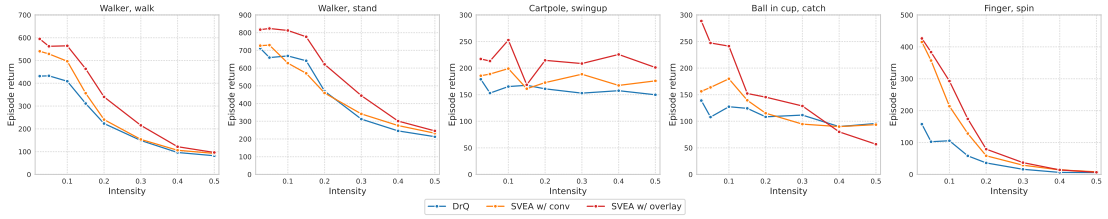


Figure 7: **DistractingCS.** Episode return as a function of randomization intensity, for each of the 5 tasks from DMControl-GB (using ConvNets). Mean of 5 seeds. We find that the difficulty of DistractingCS varies greatly between tasks, but SVEA consistently outperforms DrQ in terms of generalization across all intensities and tasks, except for *Ball in cup, catch* at the highest intensity.

4 My Experiments

When exploring enhancements for SVEA, there are several factors to consider. Given the limitations imposed by hardware, we need to scale our suggestions to align with the available GPUs provided by Google Collaboratory. Within this context, we focus on two aspects of the problem formulation:

1. Introducing non-determinism in the test environment. This entails considering how this change should be reflected in the applied data augmentations.

2. Analyzing the relationship between adjustments made to the test domain's parameters and the corresponding modifications required for data augmentation parameters.

4.1 Non-Determinism

Training an RL algorithm for continuous control tasks in a deterministic environment can simplify the learning process as the agent can directly associate actions with specific outcomes. However, there are a few considerations to keep in mind. First, deterministic environments may not fully capture the complexity and uncertainty of real-world scenarios. If the goal is to develop RL algorithms that can generalize to uncertain or stochastic environments, training solely in a deterministic environment may limit the algorithm's ability to adapt to such conditions. Ultimately, in some cases, training in a deterministic environment may result in the agent overfitting to that specific environment. The learned policy may not generalize well to different variations or perturbations of the environment. To address this, it can be beneficial to introduce randomization or variability during training, such as through the use of stochastic dynamics or adding noise to the actions or observations.

The DMControl environments are, in fact, deterministic by default. To make them non-deterministic, you can introduce stochasticity into the environments by adding random factors. Here's how you can achieve this:

- **Perturb the dynamics:** You can introduce random noise or variations in the dynamics of the environment. This can be done by modifying the physics parameters, such as friction, mass, or joint constraints, with random values.

- **Add sensory noise:** Introduce random noise to the observations received by the agent. This can simulate uncertainty or variability in the sensor measurements.
- **Randomize initial conditions:** Vary the initial states of the environment to provide different starting points for each episode. This can be done by sampling initial positions, velocities, or other relevant variables from a distribution.
- **Modify reward functions:** Randomize or introduce stochasticity in the rewards given to the agent. This can create variability in the feedback received and affect the agent's behavior.

By applying these techniques, we can introduce non-determinism into the DMControl environments, making them more realistic and challenging for reinforcement learning algorithms.

In DM Control Suite we have the following parameters that can be used for this purpose:

- $random$: Optional, either a 'numpy.random.RandomState' instance, an integer seed for creating a new 'RandomState', or None to select a seed automatically (default).

We can choose the same seed for the training environment and the testing environment and therefore we will have a non-deterministic test environment.

4.2 Adjusting the augmentations

The application of weak augmentation, such as random shift, has been shown to improve sample efficiency without causing significant issues. However, it has been observed that strong augmentation, such as random convolution, can lead to instability and poor sample efficiency. The use of stronger and more varied augmentations, including random convolution, random overlay, and affine-jitter, has the potential to enhance generalization to a broader range of Markov Decision Processes (MDPs). In SVEA, we add random convolution to make the augmented image. Here we will attempt to use random overlay alongside random convolution.

References

[1] Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control : Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

[2] Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. The distracting control suite – a challenging benchmark for reinforcement learning from pixels. *arXiv preprint arXiv:2101.02722*, 2021.

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

[4] Nicklas Hansen, Hao Su, and Xiaolong Wang. Stabilizing deep q-learning with convnets and vision transformers under data augmentation. 2021.