MaDi: Learning to Mask Distractions for Generalization in Visual Deep Reinforcement Learning

Bram Grooten
Eindhoven University of Technology
Netherlands

Tristan Tomilin
Eindhoven University of Technology
Netherlands

Gautham Vasan
University of Alberta
Canada

Matthew E. Taylor
University of Alberta & Alberta Machine Intelligence Institute (Amii)
Canada

A. Rupam Mahmood
University of Alberta & Alberta Machine Intelligence Institute (Amii)
Canada

Meng Fang
University of Liverpool
United Kingdom

Mykola Pechenizkiy
Eindhoven University of Technology
Netherlands

Decebal Constantin Mocanu
University of Luxembourg
Luxembourg

ABSTRACT

The visual world provides an abundance of information, but many input pixels received by agents often contain distracting stimuli. Autonomous agents need the ability to distinguish useful information from task-irrelevant perceptions, enabling them to generalize to unseen environments with new distractions. Existing works approach this problem using data augmentation or large auxiliary networks with additional loss functions. We introduce *MaDi*, a novel algorithm that learns to **ma**sk **di**stractions by the reward signal only. In MaDi, the conventional actor-critic structure of deep reinforcement learning agents is complemented by a small third sibling, the Masker. This lightweight neural network generates a mask to determine what the actor and critic receive, such that they can focus on learning the task. We run experiments on the DeepMind Control Generalization Benchmark, the Distracting Control Suite, and a real UR5 Robotic Arm. Our algorithm improves the agent's focus with useful masks, while its efficient Masker network only adds 0.2% more parameters to the original structure, in contrast to previous work. MaDi consistently achieves generalization results better than or competitive to state-of-the-art methods.[1]

KEYWORDS

Deep Reinforcement Learning, Generalization, Robotics

ACM Reference Format:
Bram Grooten, Tristan Tomilin, Gautham Vasan, Matthew E. Taylor, A. Rupam Mahmood, Meng Fang, Mykola Pechenizkiy, and Decebal Constantin Mocanu. 2024. MaDi: Learning to Mask Distractions for Generalization in Visual Deep Reinforcement Learning. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 23 pages.

[1]Code: github.com/bramgrooten/mask-distractions and video: youtu.be/2oImF0h1k48.
Corresponding author: b.j.grooten@tue.nl

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

1 INTRODUCTION

Deep reinforcement learning (RL) has achieved remarkable success in a variety of complex tasks such as game playing [32, 37], robotics [2, 10, 17], nuclear fusion [7], and autonomous navigation [31, 60]. However, one of the major challenges faced by RL agents is their limited ability to generalize to unseen environments, particularly in the presence of distracting visual noise, such as a video playing in the background [22, 39]. These distractions can lead to significant degradation in the performance of deep RL agents, thereby hindering their applicability in the real world. To address this, we propose a novel algorithm, **Ma**sking **Di**stractions, which learns to filter out task-irrelevant visuals, enhancing generalization capabilities.

The key idea behind MaDi is to supplement the conventional actor-critic architecture with a third lightweight component, the Masker (see Figure 1). This small neural network generates a mask that dims the irrelevant pixels, allowing the actor and critic to focus on learning the task at hand without getting too distracted. Unlike previous approaches that have attempted to address this issue [4, 21, 22], our method increases generalization performance while introducing minimal overhead in terms of model parameters, thus preserving the efficiency of the original architecture.

Furthermore, no additional loss function is necessary for the Masker to optimize its parameters. To ensure that the Masker maintains visibility of the task-relevant pixels, it is trained on the critic's loss function. The Masker and critic networks are aligned in their objective, as pixels that are essential to determine the value of an observation should not be hidden. Figure 1 shows an example of a mask, visualizing the output produced by the Masker network corresponding to the current input frame. The Masker is able to learn such precise segmentations without any additional labels, bounding boxes, or other annotations. The reward alone is enough.

To evaluate the effectiveness of MaDi, we conduct experiments on multiple environments from three benchmarks: the DeepMind Control Generalization Benchmark [22], the Distracting Control Suite [39], and a real UR5 Robotic Arm for which we design a novel generalization experiment with visual distractions. Our results

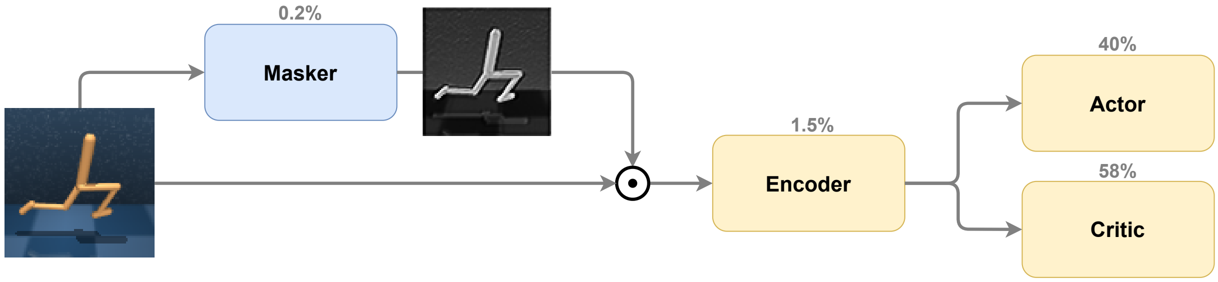

Figure 1: In the MaDi architecture, the Masker produces a soft mask (values between 0 and 1) for each frame, which subsequently gets multiplied element-wise with the observation. The encoder is a shared ConvNet updated only by the critic loss, which is also the objective function for the Masker. We show rounded percentages for the number of parameters used in the `walker-walk` environment. The actor and critic contain most of it (~ 98% together) as they consist of multiple fully-connected layers.

demonstrate that MaDi significantly improves the agent's ability to focus on relevant visual information by generating helpful masks, leading to enhanced generalization performance. Furthermore, MaDi achieves state-of-the-art performance on many environments, surpassing well-known methods in vision-based reinforcement learning [4, 18, 21, 22, 28, 50].

Our main contributions are:

- We introduce a novel algorithm, MaDi, which supplements the standard actor-critic architecture of deep RL agents with a lightweight Masker. This network learns to focus on the task-relevant pixels solely from the reward signal.
- We present a comprehensive set of experiments on the DeepMind Control Generalization Benchmark and the Distracting Control Suite. MaDi consistently achieves state-of-the-art or competitive generalization performance.
- We test MaDi on a physical robot, demonstrating that our algorithm increases the performance of the UR5 Robotic Arm in a challenging VisualReacher task, even when distracting videos are playing in the background.

The paper is structured as follows: Section 2 reviews related work, Section 3 formalizes our mathematical framework. Our algorithm MaDi is detailed in Section 4. We present simulation results in Section 5 and robotic experiments in Section 6. Section 7 concludes.

2 RELATED WORK

The problem of generalization in deep reinforcement learning has been an active area of research, with several approaches proposed to tackle the challenge of visual distractions. In this section, we review the most relevant literature, highlighting the differences between our proposed MaDi method and existing approaches.

Generalization in RL. In reinforcement learning, generalization refers to an agent's ability to perform well on unseen environments or tasks [27]. This can be challenging, as RL is prone to overfit to the training environment [6, 12, 20, 57]. Several works have focused on improving generalization capabilities by employing techniques such as domain adaptation [49], domain randomization [2, 42], meta-learning [9, 46], contrastive learning [1, 29], imitation learning [11], bisimulation metrics [13, 56], and data augmentation [22, 28, 34, 47, 50, 53]. Even using a ResNet [23] pretrained on ImageNet [36] as an encoder can improve generalization [54].

Visual Learning in RL. Learning tasks from visual input, i.e., image-based or vision-based deep RL, is typically more demanding than learning from direct features in a vector. DQN [32] was the first to learn Atari games at human-level performance directly from pixels. However, it has been shown that these algorithms can be quite brittle to changes in the environment, as altering a few pixel values can significantly decrease DQN's performance [33, 58]. Using data augmentation proved to be the key in visual RL. DrQ [50] and RAD [28] use light augmentations such as random shifts or crops of the observation to increase the algorithm's robustness.

Distractions in RL. Several approaches have been proposed to deal with the presence of task-irrelevant noise and distractions in reinforcement learning environments. Automatic Noise Filtering (ANF) [16] works on noisy environments that provide states as feature vectors, such as the MuJoCo Gym suite [5, 43]. We focus our work on RL agents that need to learn from image-based observations, like the pixel-wrapped DeepMind Control Suite [41]. Two benchmarks that we use are extensions of this suite.

Several works [4, 21, 22, 53, 54] have tried to tackle the DeepMind Control Generalization Benchmark [22] and the Distracting Control Suite [39]. Many of these methods use stronger[2] data augmentations than the light shifting and cropping of DrQ and RAD. Usually, they apply one of two favored augmentation techniques: a randomly initialized convolution layer (`conv` augmentation) or overlaying the observation with random images from a large dataset, such as Places365 [59] (`overlay` augmentation).

Masking in Visual RL. There exist a few works that aim to improve the generalization ability of RL agents by masking parts of the input. Yu et al. [51] randomly mask parts of the inputs and use an auxiliary loss to reconstruct these pixels. SGQN [4] and the recent InfoGating [44] apply more targeted masking similar to MaDi. InfoGating has only experimented on offline RL, and it uses a large U-Net [35] to determine the appropriate masks, while MaDi uses a much smaller 3-layer convolutional neural network.

Baselines. We select the following set of six baselines, as these focus on online RL and do not use any pretrained models:

- *Soft Actor-Critic* [18, **SAC**] is an off-policy actor-critic algorithm that optimizes the trade-off between exploration and

[2] By *stronger*, we mean augmentations that alter an image significantly more.

exploitation by automatically tuning a temperature parameter for entropy regularization.

- *Data-regularized Q-learning* [50, **DrQ**] focuses on making Q-learning more stable and sample efficient by shifting the observations by a few pixels in a random direction.
- *RL with Augmented Data* [28, **RAD**] improves the data efficiency in visual RL by randomly cropping the images.
- *Soft Data Augmentation* [22, **SODA**] applies data augmentation in an auxiliary task that tries to minimize the distance of augmented and non-augmented images in its feature space.
- *Stabilized Value Estimation under Augmentation* [21, **SVEA**] stabilizes learning by using augmentation solely in the critic. It combines clean and augmented data in every batch used for a critic update. The actor only sees clean data.
- *Saliency Guided Q-Networks* [4, **SGQN**] is perhaps closest to our work, as it also uses masks to benefit learning. Its masks are not applied at the start of the architecture, but are learned by a third component after the encoder. This auxiliary model minimizes the difference between its masks and other masks generated by a saliency metric. By computing the gradient of the Q-function with respect to the input pixels, this saliency metric determines which pixels are important for the agent. A hyperparameter `sgqn_quantile` (often set to 95% − 98%) determines how many pixels are masked.

There are multiple significant differences between SGQN and MaDi. First of all, SGQN is sensitive to its quantile hyperparameter. MaDi is free from this hyperparameter tuning, as it automatically finds the right fraction of pixels to mask. Furthermore, SGQN needs to compute gradients with respect to the inputs and weights, while MaDi only requires gradients of the weights. The additional components of SGQN are heavier, adding about 1.6M parameters (an extra 25%) to the base architecture, MaDi roughly 10K (0.2%), reducing the memory requirements. MaDi does not introduce any additional auxiliary loss function, as it learns directly from the critic's objective. In essence, SGQN does not apply a mask to every input image, but uses them to learn better representations. MaDi tries to learn the most helpful masks such that the actor and critic receive only task-relevant information and are able to focus on the RL problem.

3 PRELIMINARIES

Problem formulation. We consider the problem of learning a policy for a Markov decision process (MDP) with the presence of visual distractions, similar to the formulation by Hansen et al. [21]. Our approach, MaDi, aims to learn a policy that generalizes well across MDPs with varying state spaces.

We formulate the interaction between the environment and policy as an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} \colon \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the state transition function, $r \colon \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, and γ is the discount factor. To address the challenges of partial observability [25], we define a state $\mathbf{s}_t$ as a sequence of k consecutive frames $(\mathbf{o}_t, \mathbf{o}_{t-1}, \ldots, \mathbf{o}_{t-(k-1)})$, $\mathbf{o}_i \in \mathcal{O}$, where $\mathcal{O}$ is the high-dimensional image space. In the particular benchmarks we employ for evaluation, $\mathcal{O} = \mathbb{R}^{84\times84\times3}$ for the simulation environments and $\mathcal{O} = \mathbb{R}^{160\times90\times3}$ for the robotic environment, as we receive RGB colored images as input with 84×84 and 160×90 pixels respectively.

Our goal is to learn a stochastic policy $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$, where $\Delta(\mathcal{A})$ denotes the space of probability distributions over the action space $\mathcal{A}$. This policy aims to maximize the discounted return $R_t = \mathbb{E}_{\Gamma\sim\pi,\mathcal{P}}\big[\sum_{t=0}^{T} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\big]$ along a trajectory $\Gamma = (\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_T)$. The policy π is parameterized by a collection of learnable parameters θ. We aim to learn parameters θ such that π_θ generalizes well across MDPs with perturbed observation spaces, denoted as $\overline{\mathcal{M}} = \langle \overline{\mathcal{S}}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$, where states $\overline{\mathbf{s}}_t \in \overline{\mathcal{S}}$ are constructed from observations $\overline{\mathbf{o}}_t \in \overline{\mathcal{O}}$. The original observation space $\mathcal{O}$ is a subset of the perturbed observation space $\overline{\mathcal{O}}$, which may contain distractions.

Distractions. We define a distraction to be any input feature that is irrelevant to the task of the MDP, meaning that an optimal policy π^* and value function Q^* remain invariant under alterations of the feature value. In our case, input features are pixels $p_i \in \mathbb{R}^3$, where a state $\boldsymbol{s}$ consists of n pixels: $\boldsymbol{s} = (p_1, p_2, \ldots, p_n)$.

Suppose we encounter a state $\hat{\mathbf{s}}$, and we wish to determine whether pixel p_i is a distraction in that particular state. Let $\mathcal{S}_i(\hat{\mathbf{s}})$ be the set of states where only pixel p_i is changed in comparison to $\hat{\mathbf{s}}$. Then pixel p_i is considered a distraction in state $\hat{\mathbf{s}}$ if π^* and Q^* remain invariant across the entire set $\mathcal{S}_i(\hat{\mathbf{s}})$. More formally:

Definition 3.1. *A pixel p_i is a **distraction** in state $\hat{\mathbf{s}}$ if, for an optimal policy π^* and value function Q^* it holds that, for an arbitrary but fixed action $\boldsymbol{a}$, we have $\forall\ \boldsymbol{s} \in \mathcal{S}_i(\hat{\mathbf{s}})$:*

$$\pi^*(\boldsymbol{a}|\boldsymbol{s}) = \rho^* \qquad \text{where probability } \rho^* \in \mathbb{R} \text{ is constant,}$$
$$Q^*(\boldsymbol{s}, \boldsymbol{a}) = q^* \qquad \text{where value } q^* \in \mathbb{R} \text{ is constant.}$$

In other words: pixel p_i can take on any value, but the optimal policy will not change. In that particular state $\hat{\mathbf{s}}$, the pixel p_i is irrelevant to the task and thus a distraction. From this definition we can derive that the partial derivative of π^* and Q^* with respect to the input feature p_i is zero.

Corollary 3.2. *If p_i is a distraction in state $\hat{\mathbf{s}}$, then for an arbitrary action $\boldsymbol{a}$ we have that*

$$\frac{\partial}{\partial p_i}\pi^*(\boldsymbol{a}|\hat{\mathbf{s}}) = 0 \qquad \text{and} \qquad \frac{\partial}{\partial p_i}Q^*(\hat{\mathbf{s}}, \boldsymbol{a}) = 0.$$

This follows from Definition 3.1 since π^* and Q^* remain constant for varying p_i. Optimal policies perfectly ignore distractions, while suboptimal policies (i.e., neural networks during training) may be hindered by distractions. As distractions have no effect on the optimal policy, they can be safely masked when using π^*. This suggests that when striving to approximate π^*, it may be advantageous to mask distractions as well, a concept at the core of MaDi.

Soft Actor-Critic. In this work, we build upon the model-free off-policy reinforcement learning algorithm Soft Actor-Critic (SAC; [18]). SAC aims to estimate the optimal state-action value function Q^* with its parameterized critic Q_{θ_Q}. The actor is represented by a stochastic policy π_{θ_π}, which aims to maximize the value outputted by the critic while simultaneously maintaining high entropy. The optional shared encoder f_{θ_f} is often used for SAC in image-based environments. The critic and shared encoder have target networks that start with the same parameters $\theta^{\text{tgt}} = \theta$. These are gradually updated throughout training by an exponential moving average: $\theta^{\text{tgt}} \longleftarrow (1-\tau)\theta^{\text{tgt}} + \tau\theta$. We will often omit the implied parameter θ_N in our notation of any network N.

Algorithm 1 MaDi based on SAC

randomly initialize all networks: π, Q, f, M
copy parameters to target networks: Q^{tgt}, f^{tgt}
1: **for** timestep $t = 1...T$ **do**
act:
2: $\mathbf{a}_t \sim \pi\left(\cdot|f(\mathbf{s}_t \odot M(\mathbf{s}_t))\right)$ — Sample action
3: $\mathbf{s}'_t, r_t \sim \mathcal{P}(\cdot|\mathbf{s}_t, \mathbf{a}_t)$ — Perform action in env
4: $\mathcal{B} \leftarrow \mathcal{B} \cup (\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}'_t)$ — Add to replay buffer
update:
5: $\{\mathbf{s}_b, \mathbf{a}_b, r_b, \mathbf{s}'_b\} \sim \mathcal{B}$ — Sample batch $b \subset \mathcal{B}$
6: $\theta_\pi \leftarrow \theta_\pi - \eta \nabla_{\theta_\pi} \mathcal{L}_\pi(\mathbf{s}_b)$ — Update π
7: $\mathbf{s}_b \leftarrow \text{concat}(\mathbf{s}_b, \delta(\mathbf{s}_b))$ — Apply augmentation
8: **for** network N in $[Q, f, M]$ **do**
9: $\theta_N \leftarrow \theta_N - \eta \nabla_{\theta_N} \mathcal{L}_Q\left(\mathbf{s}_b, \mathbf{a}_b, r_b, \mathbf{s}'_b\right)$ — Update Q, f, M
10: **for** network N in $[Q, f]$ **do**
11: $\theta_N^{\text{tgt}} \leftarrow (1-\tau)\theta_N^{\text{tgt}} + \tau\theta_N$ — Update Q^{tgt}, f^{tgt}

4 MADI

MaDi aims to mask distractions that hinder the agent from learning and performing well. We supplement the conventional actor-critic architecture of deep reinforcement learning agents by integrating a third, lightweight component, the Masker network M. The Masker adjusts the input by dimming irrelevant pixels, allowing the actor and critic networks to focus on learning the task at hand. The Masker and encoder compute internal representations using the Hadamard product and one forward call each: $f(\boldsymbol{s}_t \odot M(\boldsymbol{s}_t))$. Algorithm 1 indicates the few adjustments necessary to standard SAC (or SVEA, when using the augmentation on line 7). Note that MaDi does not need a target network for the Masker, reducing the additional number of parameters required.

4.1 The MaDi architecture

As shown in Figure 1, the Masker network is placed at the front of the agent's architecture. It produces a scalar multiplier for each pixel in the input image to determine the degree to which the pixel ought to be darkened. We refer to the output of this network as a *soft* mask.[3] These soft masks are applied element-wise to the observation frames, effectively reducing distractions (see Figure 2).

The Masker network is composed of three convolutional layers with ReLU non-linearities in between. The last layer outputs one channel with a Sigmoid activation function to squeeze values into the interval $[0, 1]$. The Masker receives three input channels, representing the RGB values of one frame $\boldsymbol{o}_t$. In Section 3, we defined a full state $\boldsymbol{s}_t$ to be a stack of k frames, which is indeed what the actor and critic receive as input. The Masker is the only network that processes each frame separately. However, we still require only one forward pass through the Masker network for each input $\boldsymbol{s}_t$ of k frames, as we efficiently reshape the channels into the batch dimension. See Appendix A for further implementation details.

[3]We also tried *hard* (i.e., *binary*) masks, but they proved more challenging to train.

4.2 How does the Masker learn?

One may expect that learning to output useful masks requires us to define a separate loss function, but this is not the case. The Masker can simply be updated via the critic's objective function[4] to update its parameters, as shown on line 9 of Algorithm 1. This means the masks are trained without any additional segmentation labels or saliency metrics. Our hypothesis on the surprising ability of MaDi to determine the task-relevant pixels solely from a scalar reward signal, pertains to the following:

- For *relevant* pixels, if the Masker network masks away essential pixels needed to determine an accurate Q-value, then the critic loss will presumably be high, and the Masker will thus be encouraged to leave these pixels visible.
- For *irrelevant* pixels, we believe (and empirically show in Section 5.3) that strong and varying data augmentation helps. It gives an irrelevant pixel in a particular state $\boldsymbol{s}$ a varying pixel-value each time state $\boldsymbol{s}$ is sampled from the replay buffer, while the pixel's contribution to the Q^*-value remains the same (none, because it is irrelevant). The Masker is thus incentivized to mask this pixel, such that the actor and critic networks always see the same pixel-value for state $\boldsymbol{s}$, no matter which augmentation is used.

The Masker is updated together with the critic, which happens once for every environment step in case of synchronous runs. The robotic experiments use an asynchronous version of each algorithm. In that case, the Masker still gets as many updates as the critic, but it is no longer equal to the number of environment steps.

5 SIMULATION EXPERIMENTS

We present the experiments done on generalization benchmarks based on the DeepMind Control Suite [41] in this section, while our robotic experiments are shown in Section 6. We describe our experimental setup and provide the results obtained from our method, MaDi, compared with state-of-the-art approaches. Our experiments are designed to demonstrate the effectiveness of MaDi in masking distractions and improving generalization in vision-based RL.

5.1 Experimental Setup

Benchmarks. We evaluate the performance of MaDi on the DeepMind Control Generalization Benchmark [22, DMControl-GB] and the Distracting Control Suite [39, DistractingCS]. These benchmarks consist of a range of environments with varying levels of complexity and noise, providing a comprehensive assessment of an agent's ability to generalize to unseen, distracting environments.

- **DMControl-GB** has two setups with task-irrelevant pixels in the background: `video_easy` and `video_hard`. For the easy setup, there are just 10 videos to randomly sample from, and the surface from the training environment is still shown. In the hard counterpart, the surface is no longer visible, and one of 100 videos is selected. Note that all the frames in these videos are unseen — they do not overlap with the images in the augmentation dataset we use.

[4]Future work could study whether the Masker network can also learn from the actor loss. In many SAC-based implementations with a shared ConvNet, the encoder is only updated by the critic loss, and it made sense to use these gradients for the Masker.

(a) `training_env (walker)` (b) `video_hard (ball_in_cup)` (c) `distracting_cs (cartpole)`

Figure 2: Examples of original observations (left) and their masked versions (right) generated by MaDi in training (a) and testing (b, c) environments. Masks from other benchmark and domain combinations are shown in Appendix C.

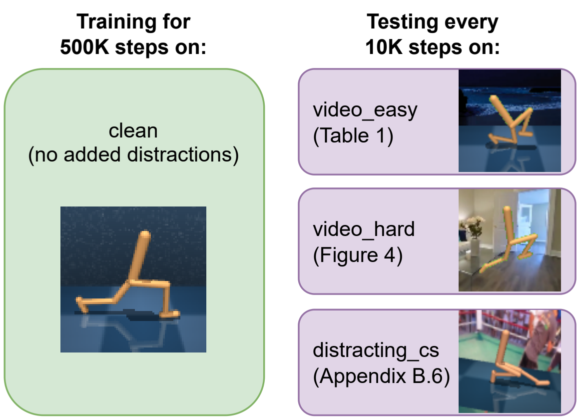

Figure 3: The training and testing setup.

- **DistractingCS** goes a step further, also adjusting the camera's orientation and the agent's color during an episode, while displaying a randomly selected video in the background. The surface remains visible, such that the agent can orient itself during a changing camera angle. The intensity of the DistractingCS determines the difficulty. With higher intensity, the environment (1) samples from a larger set of videos, (2) changes the agent's color faster and to more extreme limits, and (3) adjusts the camera's orientation faster and to more severe angles. We use the default intensity level of 0.1. See the Supplementary Material for example videos.

Environments. Within these two benchmarks, we run all algorithms on six distinct environments, listed in Table 1. From the `cartpole` and `walker` domains we select two tasks, which differ by their starting positions and reward functions. See Appendix F for a detailed description of each task.

Models & Training. For a fair comparison, we use the same base actor-critic architecture for all the methods considered in this study, including MaDi. All algorithms are trained for 500K timesteps on the clean training environment without distractions. We use the default hyperparameters for all baselines, as specified by the DMControl-GB [22]. See Appendix A for an overview of the hyperparameters.

Augmentation. As discussed in Section 2, all of our baselines (except SAC) use some form of data augmentation. MaDi is built on top of SVEA [21], which performs best with the `overlay` augmentation for distracting video backgrounds. Therefore, we choose

Table 1: Generalization performance of MaDi and various baseline algorithms on six different environments trained for $500K$ steps. We show undiscounted return on `video_easy` with mean and standard error over five seeds. MaDi outperforms or comes close to the state-of-the-art in all environments.

`video_easy`	SAC	DrQ	RAD	SODA	SVEA	SGQN	MaDi
`ball_in_cup catch`	602 ±91	714 ±131	561 ±147	750 ±98	757 ±138	761 ±171	**807 ±144**
`cartpole balance`	924 ±19	932 ±33	801 ±95	961 ±10	967 ±2	965 ±5	**982 ±4**
`cartpole swingup`	782 ±21	613 ±74	658 ±17	215 ±125	786 ±15	798 ±13	**848 ±6**
`finger spin`	227 ±26	543 ±50	479 ±65	429 ±100	645 ±39	592 ±11	**679 ±17**
`walker stand`	507 ±113	954 ±10	961 ±1	147 ±17	**977 ±3**	672 ±153	967 ±3
`walker walk`	334 ±37	821 ±38	726 ±42	479 ±168	**936 ±14**	882 ±26	895 ±24
`avg`	563	763	698	497	845	778	**863**

to apply `overlay` for MaDi as well. This strong augmentation combines an observation frame from the training environment, $\mathbf{o}_t$, with a random image $\boldsymbol{x}$ from a large dataset as follows:

$$\delta_{\boldsymbol{x}}(\mathbf{o}_t) = \alpha \cdot \mathbf{o}_t + (1 - \alpha) \cdot \boldsymbol{x}$$

where δ denotes the augmentation function. In the experiments we use the default overlay factor of $\alpha = 0.5$ and sample images from the same dataset as used in SVEA, namely Places365 [59].

Evaluation. To assess generalization, we evaluate the trained agents zero-shot on a set of unseen environments with different levels of distractions. Specifically, we test on `video_easy` and `video_hard` from DMControl-GB, and on the Distracting Control Suite. Every 10K steps we evaluate the current policy for 20 episodes on the test environments; see Figure 3. We report the average undiscounted return over five random seeds during the last 10% of training, a metric often used to reduce variance [15, 16]. We run statistical tests to verify significance, shown in Appendix B.

5.2 Generalization Results

In Table 1 we show the results of MaDi and its baselines when generalizing to the `video_easy` setup of the DMControl-GB benchmark. MaDi is able to achieve the best or competitive performance in all environments. The learning curves of Figure 4 show that MaDi also generalizes well to the more challenging `video_hard`

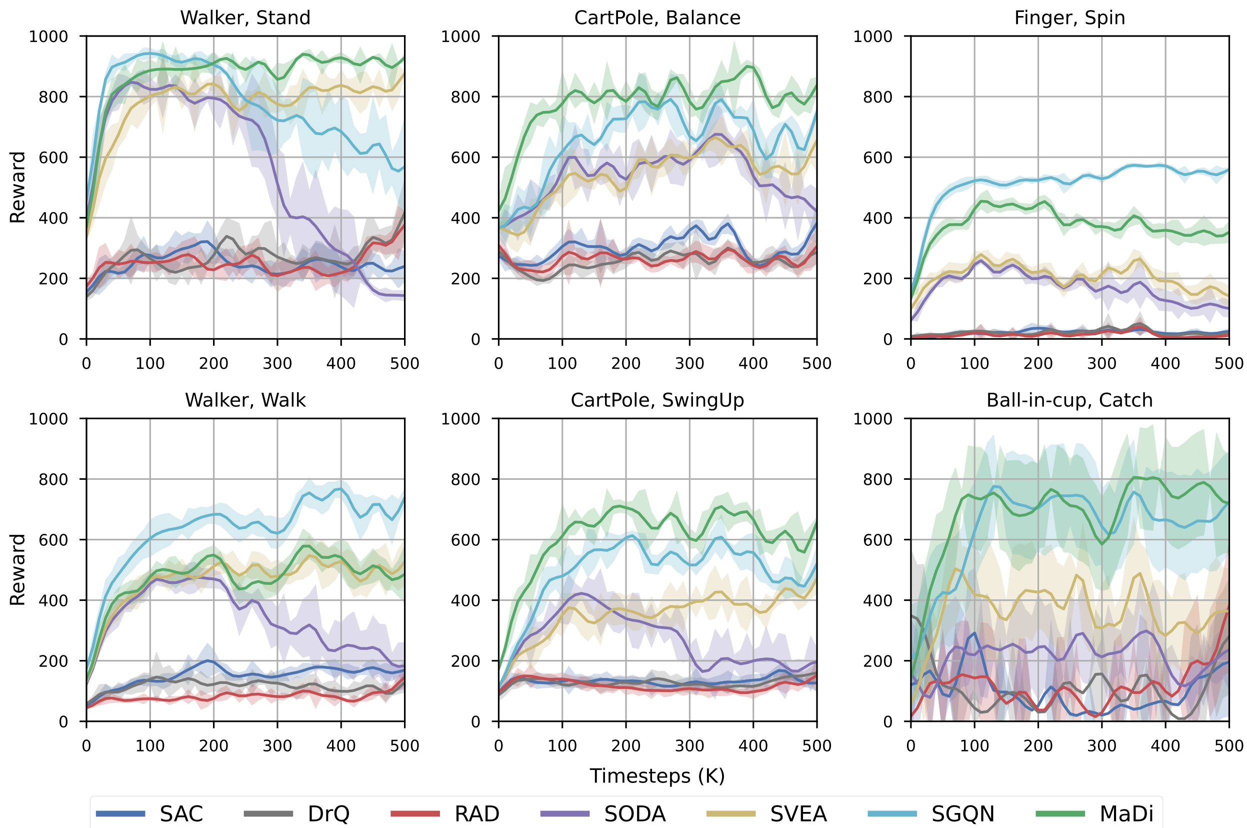

Figure 4: Learning curves of MaDi and six baselines on `video_hard`. Agents are trained on clean data for $500K$ steps and tested on `video_hard` every $10K$ steps. MaDi often reaches the top of the class, while some baselines can overfit to the training environment and decrease in generalizability. The curves show the mean over five seeds with standard error shaded alongside.

environments. Furthermore, the curves show that MaDi has a high sample efficiency, often reaching adequate performance in just 100K environment steps, by its ability to focus on the task-relevant pixels.

We present tables with results on the DistractingCS benchmark and the original training environments in Appendix B. MaDi also shows competitive performance in these additional settings. Note that the environments `ball_in_cup-catch` and `finger-spin` have *sparse* rewards, but even in this setting MaDi performs well and generates useful masks. See Appendix C for examples of masks.

5.3 Ablation on Augmentation

In Section 4.2 we described the expectation that MaDi would perform better with augmentations, as that can help it to recognize which pixels are irrelevant. To verify whether this intuition holds, we run five seeds of MaDi without the `overlay` augmentation on all six environments. We call this variant MaDi-SAC, as it now builds on top of SAC [18] instead of SVEA. The results are shown in Table 2. The generalization performance of the algorithms that use augmentation is much better than those without. MaDi indeed benefits from data augmentation to become more robust against previously unseen distractions.

Table 2: Ablation study showing the effect of the `overlay` augmentation. SVEA and MaDi both use it, while SAC and MaDi-SAC do not. All algorithms are trained for $500K$ steps. We show mean undiscounted return and standard error over five seeds evaluated on `video_hard`. The results reveal that MaDi benefits from data augmentation.

video_hard	SAC	MaDi-SAC	SVEA	MaDi
ball_in_cup catch	176 ±38	190 ±52	327 ±59	758 ±135
cartpole balance	314 ±12	237 ±6	579 ±26	827 ±25
cartpole swingup	140 ±10	132 ±9	453 ±26	619 ±24
finger spin	21 ±4	121 ±36	154 ±31	358 ±25
walker stand	233 ±28	320 ±88	847 ±18	920 ±14
walker walk	168 ±19	95 ±24	526 ±55	504 ±33
avg	175	183	481	664

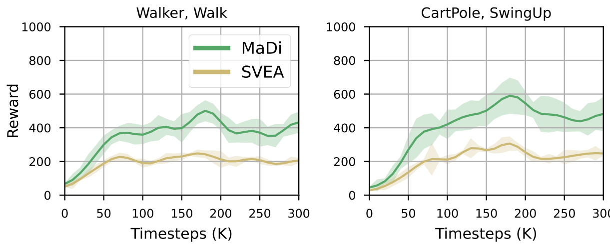

Figure 5: Generalization performance of MaDi and SVEA on `video_hard` when trained with a ViT encoder for $300K$ steps. We show the mean and standard error over five seeds.

5.4 Does MaDi work with Vision Transformers?

In image-based RL, the use of Vision Transformers [8, **ViT**] has recently gained in popularity [21, 40]. We set up a small experiment to verify whether MaDi still works in this setting. The shared encoder (see Figure 1), which is originally an 11-layer ConvNet, is now replaced by a ViT of 4 blocks with 8 attention heads each. We maintain the same architecture for the Masker network. More implementation details on the ViT encoder are in Appendix A.1.

We train the ViT-based versions of SVEA and MaDi for 300K timesteps on the clean environments of `walker-walk` and `cartpole-swingup`, and test on `video_hard` at every 10K steps. The results presented in Figure 5 show that not only does MaDi work well with a ViT encoder, but it even boosts the generalization performance.

6 ROBOTIC EXPERIMENTS

In this section, we describe our experiments on a robotic arm visual reacher task, showing that MaDi can learn to generalize when facing distracting backgrounds, even in real-world environments.

6.1 Experimental Setup

Robotic Arm. The UR5 industrial robot arm consists of six joints that can rotate to move the tip to a desired position. It uses the conventional TCP/IP protocol to transmit the state of the arm to the host computer and receive actuation packets in return at an interval of 8 milliseconds. A state packet includes the angles, velocities, target accelerations, and currents of all six joints. We configured the UR5 to use the velocity control mode. Since the UR5 does not come with a camera, we attached a Logitech RGB camera to the tip of the arm to facilitate vision-based tasks. We use *SenseAct*, a computational framework for robotic learning experiments to communicate with the robot [30]. This setup enables robust and reproducible learning across different locations and conditions.

Training environment. We train on the UR5-VisualReacher task [48, 52] for real-world robot experiments. Figure 6 depicts the experimental setup. This task involves using a camera to guide a UR5 robotic arm to reach a red target on a monitor. We ensure that the arm stays within a safe bounding box to avoid collisions. The available actions are represented by the desired angular velocities for five[5] joints, ranging from −0.7 to 0.7 radians per second. The full observation that the learning agent receives includes three consecutive RGB images from the Logitech camera of dimensions 160×90 pixels, joint angles, joint angular velocities, and the previous action taken. The reward function is defined as follows [52]:

$$r_t = \frac{c}{hw} M_t \odot W$$

where c is a scaling coefficient, h and w are the height and width of the image in pixels, M_t is a binary mask[6] of shape $h \times w$ that detects for each pixel whether it is currently red, and W is a weighting matrix of shape $h \times w$ with values decreasing from 1 at its center to 0 near the edges. These are multiplied element-wise by the Hadamard product $\odot$. The reward incentivizes the robot to move its camera closer to the target and keep the target at the center of the frame. We set the coefficient $c = 800$ for all experiments and clip the rewards to be between 0 and 4. An episode lasts 150 timesteps of 40 ms each, for 6 seconds in total. The agent sends an action at every timestep, which is repeated by the SenceAct system five times at every 8 ms.

(a) Training environment **(b) Testing environment**

Figure 6: The UR5-VisualReacher(-VideoBackgrounds) benchmark used in our experiments. The agent is rewarded for getting its camera as close as possible to the red circle randomly located on the screen. For visual demonstrations of the robot's performance with each algorithm, please refer to the videos in the Supplementary Material.

Testing environment. We test the generalization performance on a similar task, but now with videos playing in the background on the screen. We define this new generalization benchmark as UR5-VisualReacher-VideoBackgrounds (see Figure 6b), selecting five videos from the DMControl-GB [22] to use as backgrounds. We test on this benchmark after every 4500 timesteps in the training environment. This evaluation is always done for 10 episodes, twice on each video. See Appendix F for examples of frames.

Baselines. We compare the performance of SAC [18], RAD [28], SVEA [21], and MaDi. The base architecture is the same for all algorithms, but it differs somewhat from the simulation experiments of Section 5. See Appendix A for more details.

6.2 Results

In Figure 7, we present the results on the testing environment with video backgrounds. Without being trained on these distracting visuals, MaDi is able to generalize well in this challenging task.

The algorithms are trained asynchronously in this robotic environment, which means that MaDi will make fewer updates as it uses the additional Masker network. However, as Figure 7 shows, this

[5] We do not move the sixth joint because its sole purpose is to control a gripper.

[6] Note that this is a preprogrammed mask based on RGB thresholds, not made by MaDi.

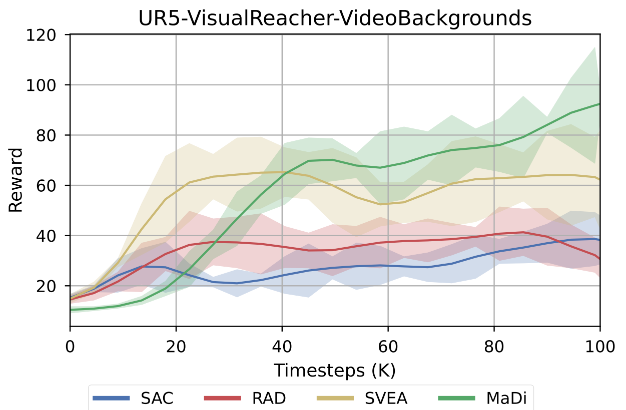

Figure 7: Performance of MaDi and three baselines on the UR5-VisualReacher task with random videos playing in the background. Agents are trained on the clean environment for $100K$ steps. We display the mean and standard error over five seeds. MaDi already generalizes best after $50K$ timesteps.

Table 3: The average reward per timestep on the UR5-VisualReacher task during the last 10% of steps in an episode. MaDi receives higher rewards in the final position of an episode in both training and testing environments, showing that it finds the red target with higher accuracy.

Reward per step	SAC	RAD	SVEA	MaDi
Training env.	1.38 ±0.09	1.32 ±0.11	1.18 ±0.37	1.95 ±0.09
Testing env.	0.32 ±0.04	0.24 ±0.07	0.47 ±0.14	0.74 ±0.07

only incurs a small delay in learning while gaining superior generalization capability through the increased focus on task-relevant pixels. A similar pattern is present on the original training environment, as shown in Figure 10 of Appendix B.1.

In these experiments, both MaDi and SVEA use the `overlay` augmentation. See Appendix D.1 for results showing the effect of the `conv` augmentation in this environment. MaDi also outperforms the baselines with the `conv` augmentation but scores slightly lower, despite the arguably clearer masks it generates in that setting.

In Appendix B.2, we present an additional experiment with a *sparse reward* function. In this UR5-VisualReacher-SparseRewards task, the agent is only rewarded with a +1 whenever its camera is close enough to the red target (according to a predefined threshold), and receives zero reward otherwise. Even in this challenging setting, MaDi is able to surpass the baselines in generalization performance.

6.3 Analysis

In Figure 8 we show that MaDi can learn to recognize the task-relevant features even in this real-world robotic task. It generalizes to the unseen testing environment and produces helpful masks. There seem to be fewer pixels dimmed in this environment compared to the other benchmarks, which may be because it can be useful for the agent to know where the entire screen is positioned.

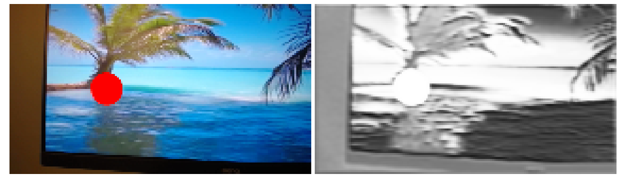

Figure 8: Observation and the corresponding mask generated by MaDi in the UR5-VisualReacher-VideoBackgrounds task. The mask is subtle, but clearly leaves the red dot's pixels intact while dimming other areas of the frame. See Appendix C for more examples of masks.

MaDi performs well in both the training and test environments, but there is quite a large discrepancy between the total rewards received. For all algorithms, the reward in the testing environment is substantially lower than in the training environment. Taking a qualitative look at the behavior of the robotic arm, it seems this is mostly because the robotic arm moves slower toward the target in the testing environment than in the training environment. The agents encounter unseen observations that significantly deviate from the training environment, likely driving them to select different actions that do not match well in sequence, causing the arm to slow down. The SAC and RAD agents rarely complete the task at all when videos are playing in the background. Even though the movement towards the goal is slower in the testing environment for all algorithms, MaDi does often reach a (near) optimal state at the end of its trajectory, similar to training. See Table 3 for an overview of rewards in the last 10% of steps. MaDi shows a higher accuracy in finding the red target near the end of an episode.

7 CONCLUSION

In the domain of vision-based deep reinforcement learning, we formalize the problem setting of distracting task-irrelevant features. We propose a novel method, MaDi, which learns directly from the reward signal to *ma*sk *di*stractions with a lightweight Masker network, without requiring any additional segmentation labels or loss functions. Our experiments show that MaDi is competitive with state-of-the-art algorithms on the DeepMind Control Generalization Benchmark and the Distracting Control Suite, while only using 0.2% additional parameters. The masks generated by MaDi enhance the agent's focus by dimming visual distractions. Even in the sparse reward setting, the Masker network is able to learn where the task-relevant pixels are in each state. Furthermore, we test MaDi on a real UR5 Robotic Arm, showing that it can outperform the baselines not only in simulation environments but also on our newly defined UR5-VisualReacher-VideoBackgrounds generalization benchmark.

Limitations & Future Work. The algorithms in this work build on the model-free off-policy deep RL algorithm SAC, while other options remain open for investigation. In future work, we seek to apply MaDi to other reinforcement learning algorithms such as PPO [38] or DQN [32] and improve the clarity of its masks. We have experimented with MaDi on one robotic arm, it would be interesting to see whether the Masker network can also produce useful masks on a diverse set of robots. Lastly, in future research, we aim to explore the possibility of using MaDi for transfer learning.

ETHICS STATEMENT

Our research aims to contribute to the development of reinforcement learning algorithms to facilitate their application in practical scenarios that positively impact society. We believe our work with MaDi has the potential to contribute to, for instance, enhancing a household robot's ability to focus on relevant visual information amidst a clutter of distractions. Furthermore, we aim to minimize the computational footprint of our algorithms by adding a negligible amount of parameters to the original structure, supporting the development of sustainable and energy-efficient AI.

REFERENCES

[1] Rishabh Agarwal, Marlos C Machado, Pablo Samuel Castro, and Marc G Bellemare. 2020. Contrastive Behavioral Similarity Embeddings for Generalization in Reinforcement Learning. In *International Conference on Learning Representations*. URL: https://arxiv.org/abs/2101.05265. (Cited in Section 2)

[2] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. 2019. Solving Rubik's Cube with a Robot Hand. *arXiv preprint arXiv:1910.07113* (2019). URL: https://openai.com/research/solving-rubiks-cube. (Cited in Section 1, 2)

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer Normalization. *Advances in Neural Information Processing Systems, Deep Learning Symposium* (2016). URL: https://arxiv.org/abs/1607.06450. (Cited in Section A)

[4] David Bertoin, Adil Zouitine, Mehdi Zouitine, and Emmanuel Rachelson. 2022. Look where you look! Saliency-guided Q-networks for generalization in visual Reinforcement Learning. *Advances in Neural Information Processing Systems* 35 (2022), 30693–30706. URL: https://arxiv.org/abs/2209.09203. (Cited in Section 1, 2, 2, 2, 4)

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *arXiv preprint arXiv:1606.01540* (2016). URL: https://www.gymlibrary.dev/. (Cited in Section 2)

[6] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. 2019. Quantifying Generalization in Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 1282–1289. URL: https://proceedings.mlr.press/v97/cobbe19a.html. (Cited in Section 2)

[7] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. 2022. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature* 602, 7897 (2022), 414–419. URL: https://www.nature.com/articles/s41586-021-04301-9. (Cited in Section 1)

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations* (2021). URL: https://arxiv.org/abs/2010.11929. (Cited in Section 5.4, A.1)

[9] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. 2016. RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning. *arXiv preprint arXiv:1611.02779* (2016). URL: https://arxiv.org/abs/1611.02779. (Cited in Section 2)

[10] Open X-Embodiment Collaboration et al. 2023. Open X-Embodiment: Robotic Learning Datasets and RT-X Models. URL: https://robotics-transformer-x.github.io. (Cited in Section 1)

[11] Linxi Fan, Guanzhi Wang, De-An Huang, Zhiding Yu, Li Fei-Fei, Yuke Zhu, and Anima Anandkumar. 2021. SECANT: Self-Expert Cloning for Zero-Shot Generalization of Visual Policies. *International Conference on Machine Learning* (2021). URL: https://arxiv.org/abs/2106.09678. (Cited in Section 2)

[12] Jesse Farebrother, Marlos C Machado, and Michael Bowling. 2018. Generalization and Regularization in DQN. *NeurIPS'18 Deep Reinforcement Learning Workshop* (2018). URL: https://arxiv.org/abs/1810.00123. (Cited in Section 2)

[13] Norm Ferns, Prakash Panangaden, and Doina Precup. 2011. Bisimulation Metrics for Continuous Markov Decision Processes. *SIAM J. Comput.* 40, 6 (2011), 1662–1714. URL: https://doi.org/10.1137/10080484X. (Cited in Section 2)

[14] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. 2016. Deep Spatial Autoencoders for Visuomotor Learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 512–519. URL: https://arxiv.org/abs/1509.06113. (Cited in Section A)

[15] Laura Graesser, Utku Evci, Erich Elsen, and Pablo Samuel Castro. 2022. The State of Sparse Training in Deep Reinforcement Learning. In *International Conference on Machine Learning*. PMLR, 7766–7792. URL: https://arxiv.org/abs/2206.10369. (Cited in Section 5.1)

[16] Bram Grooten, Ghada Sokar, Shibhansh Dohare, Elena Mocanu, Matthew E. Taylor, Mykola Pechenizkiy, and Decebal Constantin Mocanu. 2023. Automatic Noise Filtering with Dynamic Sparse Training in Deep Reinforcement Learning. *The 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2023). URL: https://arxiv.org/abs/2302.06548. (Cited in Section 2, 5.1)

[17] Tuomas Haarnoja, Ben Moran, Guy Lever, Sandy H Huang, Dhruva Tirumala, Markus Wulfmeier, Jan Humplik, Saran Tunyasuvunakool, Noah Y Siegel, Roland Hafner, et al. 2023. Learning Agile Soccer Skills for a Bipedal Robot with Deep Reinforcement Learning. *arXiv preprint arXiv:2304.13653* (2023). URL: https://sites.google.com/view/op3-soccer. (Cited in Section 1)

[18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning*. PMLR, 1861–1870. URL: https://arxiv.org/abs/1801.01290. (Cited in Section 1, 2, 3, 5.3, 6.1, A)

[19] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. 2020. Dream to Control: Learning Behaviors by Latent Imagination. *International Conference on Learning Representations* (2020). URL: https://arxiv.org/abs/1912.01603. (Cited in Section F.1)

[20] Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. 2020. Self-Supervised Policy Adaptation during Deployment. *International Conference on Learning Representations* (2020). URL: https://arxiv.org/abs/2007.04309. (Cited in Section 2)

[21] Nicklas Hansen, Hao Su, and Xiaolong Wang. 2021. Stabilizing Deep Q-Learning with ConvNets and Vision Transformers under Data Augmentation. *35th Conference on Neural Information Processing Systems* (2021). URL: https://arxiv.org/abs/2107.00644. (Cited in Section 1, 2, 2, 3, 5.1, 5.4, 6.1, 4, A.1, F.1)

[22] Nicklas Hansen and Xiaolong Wang. 2021. Generalization in Reinforcement Learning by Soft Data Augmentation. In *International Conference on Robotics and Automation*. URL: https://arxiv.org/abs/2011.13389. (Cited in Section 1, 1, 2, 2, 2, 5.1, 5.1, 6.1, A, 4, A, B, B.5, D, F.1, F.2)

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf. (Cited in Section 2)

[24] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415* (2016). URL: https://arxiv.org/abs/1606.08415. (Cited in Section A.1)

[25] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101, 1-2 (1998), 99–134. URL: https://www.sciencedirect.com/science/article/pii/S000437029800023X. (Cited in Section 3)

[26] Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *International Conference for Learning Representations* (2015). URL: https://arxiv.org/abs/1412.6980. (Cited in Section 4)

[27] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. 2023. A Survey of Zero-shot Generalisation in Deep Reinforcement Learning. *Journal of Artificial Intelligence Research* 76 (2023), 201–264. URL: https://arxiv.org/abs/2111.09794. (Cited in Section 2)

[28] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. 2020. Reinforcement Learning with Augmented Data. *Advances in Neural Information Processing Systems* 33 (2020), 19884–19895. URL: https://arxiv.org/abs/2004.14990. (Cited in Section 1, 2, 2, 2, 6.1, 4)

[29] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. 2020. CURL: Contrastive Unsupervised Representations for Reinforcement Learning. In *International Conference on Machine Learning*. PMLR, 5639–5650. URL: https://arxiv.org/abs/2004.04136. (Cited in Section 2)

[30] A Rupam Mahmood, Dmytro Korenkevych, Gautham Vasan, William Ma, and James Bergstra. 2018. Benchmarking Reinforcement Learning Algorithms on Real-World Robots. In *Conference on robot learning*. PMLR, 561–591. URL: https://arxiv.org/abs/1809.07731. (Cited in Section 6.1)

[31] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. 2016. Learning to Navigate in Complex Environments. *arXiv preprint arXiv:1611.03673* (2016). URL: https://arxiv.org/abs/1611.03673. (Cited in Section 1)

[32] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533. URL: https://www.nature.com/articles/nature14236. (Cited in Section 1, 2, 7)

[33] Xinghua Qu, Zhu Sun, Yew-Soon Ong, Abhishek Gupta, and Pengfei Wei. 2020. Minimalistic Attacks: How Little it Takes to Fool a Deep Reinforcement Learning Policy. *IEEE Transactions on Cognitive and Developmental Systems* 13, 4 (2020), 806–817. URL: https://arxiv.org/abs/1911.03849. (Cited in Section 2)

[34] Roberta Raileanu, Maxwell Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. 2021. Automatic Data Augmentation for Generalization in Reinforcement Learning. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 5402–5415. URL: https://proceedings.neurips.cc/paper/2021/hash/2b38c2df6a49b97f706ec9148ce48d86-Abstract.html. (Cited in Section 2)

[35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 234–241. URL: https://arxiv.org/abs/1505.04597. (Cited in Section 2)

[36] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252. https://doi.org/10.1007/s11263-015-0816-y URL: https://link.springer.com/article/10.1007/s11263-015-0816-y. (Cited in Section 2)

[37] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. 2020. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature* 588, 7839 (2020), 604–609. URL: https://www.nature.com/articles/s41586-020-03051-4. (Cited in Section 1)

[38] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347* (2017). URL: https://arxiv.org/abs/1707.06347. (Cited in Section 7)

[39] Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. 2021. The Distracting Control Suite – A Challenging Benchmark for Reinforcement Learning from Pixels. *arXiv preprint arXiv:2101.02722* (2021). URL: https://arxiv.org/abs/2101.02722. (Cited in Section 1, 1, 2, 5.1, B.6, F.1)

[40] Tianxin Tao, Daniele Reda, and Michiel van de Panne. 2022. Evaluating Vision Transformer Methods for Deep Reinforcement Learning from Pixels. *arXiv preprint arXiv:2204.04905* (2022). URL: https://arxiv.org/abs/2204.04905. (Cited in Section 5.4)

[41] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. 2018. Deepmind Control Suite. *arXiv preprint arXiv:1801.00690* (2018). URL: https://arxiv.org/abs/1801.00690. (Cited in Section 2, 5, F.1)

[42] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 2017. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 23–30. URL: https://arxiv.org/abs/1703.06907. (Cited in Section 2)

[43] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5026–5033. URL: https://mujoco.org/. (Cited in Section 2)

[44] Manan Tomar, Riashat Islam, Sergey Levine, and Philip Bachman. 2023. Ignorance is Bliss: Robust Control via Information Gating. *arXiv preprint arXiv:2303.06121* (2023). URL: https://arxiv.org/abs/2303.06121. (Cited in Section 2)

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *Advances in Neural Information Processing Systems* 30 (2017). URL: https://arxiv.org/abs/1706.03762. (Cited in Section A.1)

[46] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. 2016. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763* (2016). URL: https://arxiv.org/abs/1611.05763. (Cited in Section 2)

[47] Kaixin Wang, Bingyi Kang, Jie Shao, and Jiashi Feng. 2020. Improving Generalization in Reinforcement Learning with Mixture Regularization. *Advances in Neural Information Processing Systems* 33 (2020), 7968–7978. URL: https://arxiv.org/abs/2010.10814. (Cited in Section 2)

[48] Yan Wang, Gautham Vasan, and A Rupam Mahmood. 2023. Real-Time Reinforcement Learning for Vision-Based Robotics Utilizing Local and Remote Computers. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 9435–9441. URL: https://arxiv.org/abs/2210.02317. (Cited in Section 6.1, A, F.2)

[49] Jinwei Xing, Takashi Nagata, Kexin Chen, Xinyun Zou, Emre Neftci, and Jeffrey L Krichmar. 2021. Domain Adaptation In Reinforcement Learning Via Latent Unified State Representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 10452–10459. URL: https://arxiv.org/abs/2102.05714. (Cited in Section 2)

[50] Denis Yarats, Ilya Kostrikov, and Rob Fergus. 2021. Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels. *International Conference on Learning Representations* (2021). URL: https://openreview.net/forum?id=GY6-6sTvGaf. (Cited in Section 1, 2, 2, 2, 4, F.1)

[51] Tao Yu, Zhizheng Zhang, Cuiling Lan, Yan Lu, and Zhibo Chen. 2022. Mask-based Latent Reconstruction for Reinforcement Learning. *Advances in Neural Information Processing Systems* 35 (2022), 25117–25131. URL: https://arxiv.org/abs/2201.12096. (Cited in Section 2)

[52] Yufeng Yuan and A Rupam Mahmood. 2022. Asynchronous Reinforcement Learning for Real-Time Control of Physical Robots. In *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 5546–5552. URL: https://arxiv.org/abs/2203.12759. (Cited in Section 6.1, F.2)

[53] Zhecheng Yuan, Guozheng Ma, Yao Mu, Bo Xia, Bo Yuan, Xueqian Wang, Ping Luo, and Huazhe Xu. 2022. Don't Touch What Matters: Task-Aware Lipschitz Data Augmentation for Visual Reinforcement Learning. *International Joint Conference on Artificial Intelligence* (2022). URL: https://arxiv.org/abs/2202.09982. (Cited in Section 2, 2)

[54] Zhecheng Yuan, Zhengrong Xue, Bo Yuan, Xueqian Wang, Yi Wu, Yang Gao, and Huazhe Xu. 2022. Pre-Trained Image Encoder for Generalizable Visual Reinforcement Learning. *Advances in Neural Information Processing Systems* 35 (2022), 13022–13037. URL: https://arxiv.org/abs/2212.08860. (Cited in Section 2, 2)

[55] Dylan Yung, Andrew Szot, Prithvijit Chattopadhyay, Judy Hoffman, and Zsolt Kira. 2023. Augmentation Curriculum Learning For Generalization in RL. (2023). URL: https://openreview.net/forum?id=Fj1S0SV8p3U. (Cited in Section D)

[56] Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. 2021. Learning Invariant Representations for Reinforcement Learning without Reconstruction. *International Conference on Learning Representations (ICLR)* (2021). URL: https://arxiv.org/abs/2006.10742. (Cited in Section 2)

[57] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. 2018. A Study on Overfitting in Deep Reinforcement Learning. *arXiv preprint arXiv:1804.06893* (2018). URL: https://arxiv.org/abs/1804.06893. (Cited in Section 2)

[58] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. 2020. Robust Deep Reinforcement Learning against Adversarial Perturbations on State Observations. *Advances in Neural Information Processing Systems* 33 (2020), 21024–21037. URL: https://arxiv.org/abs/2003.08938. (Cited in Section 2)

[59] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Places: A 10 million Image Database for Scene Recognition. *IEEE transactions on pattern analysis and machine intelligence* 40, 6 (2017), 1452–1464. URL: https://ieeexplore.ieee.org/document/7968387. (Cited in Section 2, 5.1)

[60] Y Zhu, R Mottaghi, E Kolve, JJ Lim, A Gupta, L Fei-Fei, and A Farhadi. 2017. Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. *International Conference on Robotics and Automation* (2017). URL: https://ieeexplore.ieee.org/document/7989381. (Cited in Section 1)