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

Matthew E. Taylor  
University of Alberta & Alberta  
Canada

A. Rupam Mahmood  
University of Alberta & Alberta  
Canada

Mykola Pechenizkiy  
Eindhoven University of Technology  
Netherlands

Gautham Vasan  
University of Alberta  
Canada

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 mask distractions 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 agent 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

¹Code: github.com/bramgrooten/mask-distractions and video: youtu.be/2odmF0hI4k48. Corresponding author: b.j.grooten@tue.nl

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, Masking Distractions, 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...
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 Disturbing 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 Disturbing Control Suite [39]. Many of these methods use stronger 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

\[\text{By stronger, we mean augmentations that alter an image significantly more.}\]
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, SQQN] 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 sqnn.quantile (often set to 95% – 98%) determines how many pixels are masked.

There are multiple significant differences between SQQN and MaDi. First of all, SQQN 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, SQQN needs to compute gradients with respect to the inputs and weights, while MaDi only requires gradients of the weights. The additional components of SQQN 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, SQQN 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 \( M = (S, A, P, r, \gamma) \), where \( S \) is the state space, \( A \) is the action space, \( P : S \times A \rightarrow S \) is the state transition function, \( r : S \times A \rightarrow \mathbb{R} \) is the reward function, and \( \gamma \) is the discount factor. To address the challenges of partial observability [25], we define a state \( s_i \) as a sequence of \( k \) consecutive frames \((o_1, o_1, \ldots, o_{i-(k-1)})\), \( o_i \in O \), where \( O \) is the high-dimensional image space. In the particular benchmarks we employ for evaluation, \( O = \mathbb{R}^{84\times84} \) for the simulation environments and \( O = \mathbb{R}^{160\times90\times3} \) 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 \( \pi : S \rightarrow \Delta(A) \), where \( \Delta(A) \) denotes the space of probability distributions over the action space \( A \). This policy aims to maximize the discounted return \( R_i = \mathbb{E}_{T} (\pi, P, r) \{ \sum_{t=0}^{T} \gamma^t r(s_t, a_t) \} \) along a trajectory \( \Gamma = (s_0, s_1, \ldots, s_T) \). The policy \( \pi \) is parameterized by a collection of learnable parameters \( \theta \). We aim to learn parameters \( \theta \) such that \( \pi_\theta \) generalizes well across MDPs with perturbed observation spaces, denoted as \( M = (\bar{S}, \bar{A}, \bar{P}, r, \gamma) \), where states \( \bar{s}_i \in \bar{S} \) are constructed from observations \( \bar{s}_i \in \bar{O} \). The original observation space \( O \) is a subset of the perturbed observation space \( \bar{O} \), 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 \( \pi^* \) 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 \( s \) consists of \( n \) pixels: \( s = (p_1, p_2, \ldots, p_n) \).

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

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

\[
\pi^*(a|s) = \rho^* \quad \text{where probability } \rho^* \in \mathbb{R} \text{ is constant},
\]

\[
Q^*(s, a) = q^* \quad \text{where value } q^* \in \mathbb{R} \text{ is constant}.
\]

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

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

\[
\frac{\partial}{\partial p_i} \pi^*(a|\hat{s}) = 0 \quad \text{and} \quad \frac{\partial}{\partial p_i} Q^*(\hat{s}, a) = 0.
\]

This follows from Definition 3.1 since \( \pi^* \) 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 \( \pi^* \). This suggests that when striving to approximate \( \pi^* \), 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 \( Q_{\theta_\pi} \). The actor is represented by a stochastic policy \( \pi_{\theta_{\pi}} \), which aims to maximize the value outputted by the critic while simultaneously maintaining high entropy. The optional shared encoder \( f_{\theta_\pi} \) is often used for SAC in image-based environments. The critic and shared encoder have target networks that start with the same parameters \( \theta_{\pi}^{\text{init}} = \theta \). These are gradually updated throughout training by an exponential moving average: \( \theta_{\pi}^{\text{init}} \overset{\text{new}}{\rightarrow} (1 - \tau) \theta_{\pi}^{\text{new}} + \tau \theta_{\pi}^{\text{init}} \). We will often omit the implied parameter \( \theta_N \) in our notation of any network \( N \).
Algorithm 1 MaDi based on SAC

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>for timestep ( t = 1 \ldots T ) do</td>
</tr>
<tr>
<td>2.</td>
<td>( a_t \sim \pi (\cdot</td>
</tr>
<tr>
<td>3.</td>
<td>( s_t', r_t \sim P (\cdot</td>
</tr>
<tr>
<td>4.</td>
<td>( \mathcal{B} \leftarrow \mathcal{B} \cup {s_t, a_t, r_t, s_t'} ) Add to replay buffer</td>
</tr>
<tr>
<td>5.</td>
<td>( {s_b, a_b, r_b, s_b'} \sim \mathcal{B} ) Sample batch ( b \subset \mathcal{B} )</td>
</tr>
<tr>
<td>6.</td>
<td>( \theta_\pi \leftarrow \theta_\pi - \eta \nabla_{\theta_\pi} \mathcal{L}_Q (s_b) ) Update ( \pi )</td>
</tr>
<tr>
<td>7.</td>
<td>( s_b \leftarrow \text{concat}(s_b, \delta(s_b)) ) Apply augmentation</td>
</tr>
<tr>
<td>8.</td>
<td>for network ( N ) in ( \mathcal{Q}, f, M ) do</td>
</tr>
<tr>
<td>9.</td>
<td>( \theta_N \leftarrow \theta_N - \eta \nabla_{\theta_N} \mathcal{L}_Q (s_b, a_b, r_b, s_b') ) Update ( Q, f, M )</td>
</tr>
<tr>
<td>10.</td>
<td>for network ( N ) in ( \mathcal{Q}, f ) do</td>
</tr>
<tr>
<td>11.</td>
<td>( \theta_N^{\text{tgt}} \leftarrow (1 - \tau) \theta_N^{\text{tgt}} + \tau \theta_N ) Update ( Q^{\text{tgt}}, f^{\text{tgt}} )</td>
</tr>
</tbody>
</table>

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: \( f(s_t \otimes M(s_t)) \). 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 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 \( o_t \). In Section 3, we defined a full state \( s_t \) 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 \( s_t \) of \( k \) frames, as we efficiently reshape the channels into the batch dimension. See Appendix A for further implementation details.

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 \( s \) a varying pixel-value each time state \( s \) is sampled from the replay buffer, while the pixel’s contribution to the Q\( ^\pi \)-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 \( s \), 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 \([22, \text{DMControl-GB}]\) and the Distracting Control Suite \([39, \text{DistractingCS}]\). 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 \text{easy} and video \text{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.

---

\(^3\) We also tried hard \( \text{(i.e., binary)} \) masks, but they proved more challenging to train.

\(^4\) 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.
500K steps on:
Training for clean (except SAC) use some form of data augmentation. MaDi is built in a clean training environment without distractions. We use the default hyperparameters for all baselines, as specified by the DMControl-Suite.

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.

<table>
<thead>
<tr>
<th>Environment</th>
<th>SAC</th>
<th>DrQ</th>
<th>RAD</th>
<th>SODA</th>
<th>SVEA</th>
<th>SGQN</th>
<th>MaDi</th>
</tr>
</thead>
<tbody>
<tr>
<td>ball_in_cup</td>
<td>602 ±91</td>
<td>714 ±131</td>
<td>561 ±147</td>
<td>750 ±98</td>
<td>757 ±138</td>
<td>761 ±171</td>
<td>807 ±144</td>
</tr>
<tr>
<td>catch</td>
<td>924 ±19</td>
<td>932 ±33</td>
<td>801 ±95</td>
<td>961 ±10</td>
<td>967 ±2</td>
<td>965 ±5</td>
<td>982 ±4</td>
</tr>
<tr>
<td>cartpole balance</td>
<td>227 ±21</td>
<td>543 ±74</td>
<td>479 ±17</td>
<td>429 ±125</td>
<td>786 ±15</td>
<td>798 ±13</td>
<td>848 ±6</td>
</tr>
<tr>
<td>cartpole swingup</td>
<td>26 ±26</td>
<td>45 ±50</td>
<td>26 ±65</td>
<td>100 ±39</td>
<td>11 ±11</td>
<td>17 ±11</td>
<td>679 ±77</td>
</tr>
<tr>
<td>finger</td>
<td>507 ±113</td>
<td>954 ±10</td>
<td>961 ±17</td>
<td>479 ±147</td>
<td>977 ±3</td>
<td>672 ±97</td>
<td>967 ±13</td>
</tr>
<tr>
<td>walker</td>
<td>334 ±37</td>
<td>821 ±38</td>
<td>726 ±42</td>
<td>479 ±168</td>
<td>936 ±14</td>
<td>882 ±26</td>
<td>895 ±24</td>
</tr>
<tr>
<td>avg</td>
<td>563 ±56</td>
<td>763 ±69</td>
<td>698 ±49</td>
<td>497 ±845</td>
<td>778 ±77</td>
<td>863 ±24</td>
<td></td>
</tr>
</tbody>
</table>

**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-Suite [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 to apply overlay for MaDi as well. This strong augmentation combines an observation frame from the training environment, \( o_t \), with a random image \( x \) from a large dataset as follows:

\[
\delta_x(o_t) = \alpha \cdot o_t + (1 - \alpha) \cdot x
\]

where \( \delta \) 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
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.

<table>
<thead>
<tr>
<th>video_hard</th>
<th>SAC</th>
<th>MaDi-SAC</th>
<th>SVEA</th>
<th>MaDi</th>
</tr>
</thead>
<tbody>
<tr>
<td>ball_in_cup</td>
<td>176 ±38</td>
<td>190 ±52</td>
<td>327 ±59</td>
<td>758 ±135</td>
</tr>
<tr>
<td>catch</td>
<td>314 ±12</td>
<td>237 ±6</td>
<td>579 ±26</td>
<td>827 ±25</td>
</tr>
<tr>
<td>cartpole</td>
<td>140 ±10</td>
<td>132 ±9</td>
<td>453 ±26</td>
<td>619 ±24</td>
</tr>
<tr>
<td>balance</td>
<td>28 ±4</td>
<td>36 ±3</td>
<td>31 ±26</td>
<td>358 ±25</td>
</tr>
<tr>
<td>swingup</td>
<td>223 ±28</td>
<td>320 ±88</td>
<td>847 ±18</td>
<td>920 ±14</td>
</tr>
<tr>
<td>finger</td>
<td>168 ±19</td>
<td>95 ±24</td>
<td>526 ±55</td>
<td>504 ±33</td>
</tr>
<tr>
<td>avg</td>
<td>175 ±19</td>
<td>183 ±24</td>
<td>481 ±55</td>
<td>664 ±33</td>
</tr>
</tbody>
</table>

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.

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.
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

---

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

---

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

---

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

<table>
<thead>
<tr>
<th>Reward per step</th>
<th>SAC</th>
<th>RAD</th>
<th>SVEA</th>
<th>MaDi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training env.</td>
<td>1.38 ±0.09</td>
<td>1.32 ±0.11</td>
<td>1.18 ±0.37</td>
<td>1.95 ±0.09</td>
</tr>
<tr>
<td>Testing env.</td>
<td>0.32 ±0.04</td>
<td>0.24 ±0.07</td>
<td>0.47 ±0.14</td>
<td>0.74 ±0.07</td>
</tr>
</tbody>
</table>

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.

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 mask distractions 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


