Attention-Privileged Reinforcement Learning

Sasha Salter [1] **Dushyant Rao** [2] **Markus Wulfmeier** [2] **Raia Hadsell** [2] **Ingmar Posner** [1]

Abstract

Reinforcement learning is known to suffer from poor sample efficiency and generalisation to unseen environments. Domain randomisation encourages transfer by training over factors of variation that may be encountered in the target domain. This increases learning complexity, can negatively impact learning rate and performance, and requires knowledge of potential variations during deployment. In this paper, we introduce Attention-Privileged Reinforcement Learning (APRiL) which uses a self-supervised attention mechanism to significantly alleviate these drawbacks: by focusing on task-relevant aspects of the observations, attention provides robustness to distractors as well as significantly increased learning efficiency. APRiL trains two attention-augmented actor-critic agents: one purely based on image observations, available across training and transfer domains; and one with access to privileged information (such as environment states) available only during training. Experience is shared between both agents and their attention mechanisms are aligned. The image-based policy can then be deployed without access to privileged information. We experimentally demonstrate accelerated and more robust learning on a diverse set of domains, leading to improved final performance for environments both within and outside the training distribution [3] .

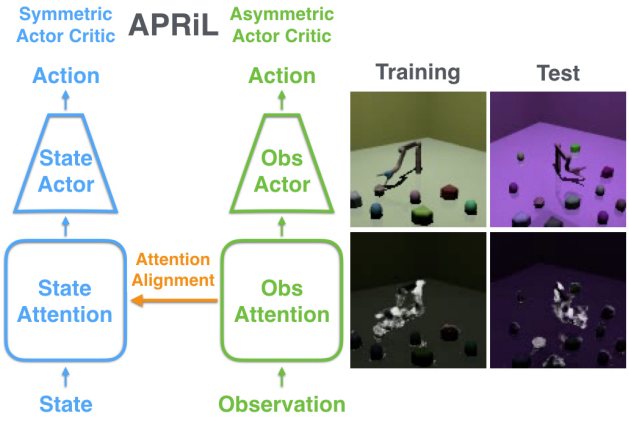

Figure 1: APRiL concurrently trains two attention augmented policies (one state-based, the other image-based). By aligning the observation attention to that of the state, image-based attention quickly suppresses highly varying task-irrelevant information. This leads to increased learning rate and robustness to additional distractors at test time not seen during training. For *JacoReach*, attention (second row; white and black signify high and low attention values) is paid only to the target object and jaco arm in training and extrapolated test environments.

1. Introduction

While Deep Reinforcement Learning (RL) has recently provided significant successes in various high-data domains, (Mnih et al., 2015; Silver et al., 2017; Lillicrap et al., 2015; OpenAI et al., 2018), its application to physical systems remains challenging. In particular, this is due to expensive and slow data generation, safety challenges during and after training, and challenges for adapting to unexpected changes in the environment.

When training models in simulation, we can obtain robustness by focusing on quick adaptation to target domains (Ganin et al., 2016; Bousmalis et al., 2016; Wulfmeier et al., 2017), or randomise system parameters aiming to cover all possible changes in environment parameters (Tobin et al., 2017; Rajeswaran et al., 2016; OpenAI et al., 2018; Pinto et al., 2018; Sadeghi & Levine, 2016; Viereck et al., 2017). While increasing the robustness of the method, training under a distribution of randomised visual (Sadeghi & Levine, 2016; Viereck et al., 2017) and physical (Peng et al., 2018) attributes, can be substantially more difficult and slower due to the increased variability of the learning domain. This often leads to a compromise in final performance both within and outside the training distribution (OpenAI et al., 2018; Tobin et al., 2017).

To increase robustness and reduce required training time,

[1]Applied AI Lab, University of Oxford, {sasha, ingmar}@robots.ox.ac.uk [2]Deepmind, London, {dushyantr, mwulfmeier, raia}@google.com.

[3]Videos comparing APRiL and asym-DDPG baseline: https://sites.google.com/view/april-domain-randomisation/home

we can make use of privileged information such as environment states which are commonly accessible in simulators. Using more structured and informative representations directly as agent input, we can improve data efficiency and generalisation (Tassa et al., 2018; Peng et al., 2018).

However, raw observations can be easier to obtain and dependence on privileged information during deployment can be restrictive. When exact states are available during training but not deployment, we can make use of information asymmetric actor-critic methods (Pinto et al., 2018; Schwab et al., 2019) to train the critic faster via access to the state while providing only images for the actor.

By introducing Attention-Privileged Reinforcement Learning (APRiL), we further leverage privileged information available during training, such as simulator states and object segmentations (Todorov et al., 2012; Dosovitskiy et al., 2017)), for increased robustness and sample efficiency. Extending asymmetric actor-critic methods, APRiL concurrently trains two actor-critic systems (one symmetric with a state-based agent, the other asymmetric with an image-dependent actor). Both actors utilise attention to filter their inputs, and we encourage alignment between both attention mechanisms. As state-space learning is unaffected by visual randomisation, the observation attention module efficiently attends to state- and task-dependent aspects of the image, leading to faster image-based policy learning and robustness to task-irrelevant factors. See Figure 1 for a visualisation of APRiL and its attention on one of our domains.

In addition, APRiL shares a replay buffer between both agents, which further accelerates training for the the image-based policy. At test-time, the image-based policy can be deployed without access to privileged information. We test our approach on a diverse set of simulated domains across robotic manipulation, locomotion, and navigation; and demonstrate considerable performance improvements compared to competitive baselines when evaluating on environments from the training distribution as well as in unseen settings with additional distractors.

The principal contributions of the paper are as follows:

1. Algorithmic improvements: We present a novel extension of asymmetric actor-critic methods which better exploit privileged information during training. We train two agents, one with access to environment states; and improve sample efficiency and generalisation of the other by employing a shared replay buffer and aligning the attention between both agents, via the use of privileged object segmentations.
2. Experimental improvements and generalisation: We demonstrate strong improvements for data efficiency and final performance on a set of continuous control domains including, 2D navigation, 2D locomotion, and 3D robotic manipulation. Furthermore, we investigate performance on held-out environments with both interpolated and extrapolated environment parameters to evaluate different types of generalisation.
3. Detailed ablations: We provide detailed ablations over different forms of transfer applied in APRiL and test against alternate privileged information methods.

2. Preliminaries

Before introducing Attention-Privileged Reinforcement Learning (APRiL), this section provides a background for the RL algorithms used. For a more in-depth introduction please refer to Lillicrap et al. (2015) and Pinto et al. (2018).

2.1. Reinforcement Learning

We describe an agent's environment as a Partially Observable Markov Decision Process which is represented as the tuple $(S, O, A, P, r, \gamma, s_0)$, where S denotes a set of continuous states, A denotes a set of either discrete or continuous actions, $P : S \times A \times S \to \{x \in \mathbb{R} | 0 \leq x \leq 1\}$ is the transition probability function, $r : S \times A \to \mathbb{R}$ is the reward function, γ is the discount factor, and s_0 is the initial state distribution. O is a set of continuous observations corresponding to continuous states in S. At every time-step t, the agent takes action $a_t = \pi(\cdot|s_t)$ according to its policy $\pi : S \to A$. The policy is optimised as to maximize the expected return $R_t = E_{s_0}[\sum_{i=t}^{\infty} \gamma^{i-t} r_i | s_0]$. The agent's Q-function is defined as $Q_\pi(s_t, a_t) = E[R_t | s_t, a_t]$.

2.2. Asymmetric Deep Deterministic Policy Gradients

Asymmetric Deep Deterministic Policy Gradients (asymmetric DDPG) (Pinto et al., 2018) represents a type of actor-critic algorithm designed specifically for efficient learning of a deterministic, observation-based policy in simulation. This is achieved by leveraging access to more compressed, informative environment states, available in simulation, to speed up and stabilise training of the critic.

The algorithm maintains two neural networks: an observation-based actor or policy $\pi_\theta : O \to A$ (with parameters θ) used during training and test time, and a state-based Q-function (also known as critic) $Q^\pi_\phi : S \times A \to R$ (with parameters ϕ) which is only used during training.

To enable exploration, the method (like its symmetric version (Silver et al., 2014)) relies on a noisy version of the policy (called behavioural policy), e.g. $\pi_b(o) = \pi(o) + z$ where $z \sim \mathcal{N}(0, 1)$ (see Appendix C for our particular instantiation). The transition tuples $(s_t, o_t, a_t, r_t, s_{t+1}, o_{t+1})$ encountered during training are stored in a replay buffer (Mnih et al., 2015). Training examples sampled from the replay buffer are used to optimize the critic and actor. By mini-

mizing the Bellman error loss $\mathcal{L}_{critic} = (Q(s_t, a_t) - y_t)^2$, where $y_t = r_t + \gamma Q(s_{t+1}, \pi(o_{t+1}))$, the critic is optimized to approximate the true Q values. The actor is optimized by minimizing the loss $\mathcal{L}_{actor} = -E_{s,o\sim\pi_b(o)}[Q(s, \pi(o))]$.

3. Attention-Privileged Reinforcement Learning (APRiL)

APRiL improves the robustness and sample efficiency of an observation-based agent by using multiple ways to benefit from privileged information. First, we use an asymmetric actor-critic setup to train the observation based actor. Second, we additionally train a quicker learning actor that has access to exact environment states, while sharing replay buffers, and aligning attention mechanisms between both actors. We focus in the following sections on extending asymmetric DDPG (Pinto et al., 2018). However, many of these ideas are generally applicable to off-policy actor-critic methods (Konda & Tsitsiklis, 2000).

APRiL is comprised of three modules as displayed in Figure 2. The first two modules, A_s and A_o, each represent a separate actor-critic with an attention network incorporated over the input for each actor. For the *state-based* module A_s we use standard symmetric DDPG, while the *observation-based* module A_o builds on asymmetric DDPG, with the critic having access to states. Finally, the third part A_T represents the alignment process between attention mechanisms of both actor-critic agents to more effectively transfer knowledge between the both learners respectively.

A_s consists of three networks: Q^π_s, π_s, h_s (respectively critic, actor, and attention) with parameters $\{\phi_s, \theta_s, \psi_s\}$. Given input state s_t, the attention network outputs a soft gating mask h_t of same dimensionality as the input, with values ranging between $[0, 1]$. The input to the actor is an attention-filtered version of the state, $s^a_t = h_s(s_t) \odot s_t$. To encourage a sparse masking function, we found that training this attention module on both the traditional DDPG loss as well as an entropy loss helped:

$$\mathcal{L}_{h_s} = -E_{s\sim\pi_b}[Q_s(s, \pi_s(s^a)) - \beta H(h_s(s))], \quad (1)$$

where β is a hyperparameter to weight the additional entropy objective, and π_b is the behaviour policy used to obtain experience (in this case from a shared replay buffer). The actor and critic networks π_s and Q_s are trained with the symmetric DDPG actor and Bellman error losses.

Within A_T, the state-attention obtained in A_s is converted to corresponding observation-attention T to act as a self-supervised target for the observation attention module in A_o. This is achieved in a two-step process. First, state-attention $h_s(s)$ is converted into object-attention c, which specifies how task-relevant each object in the scene is. The procedure uses information about which dimension of the environment

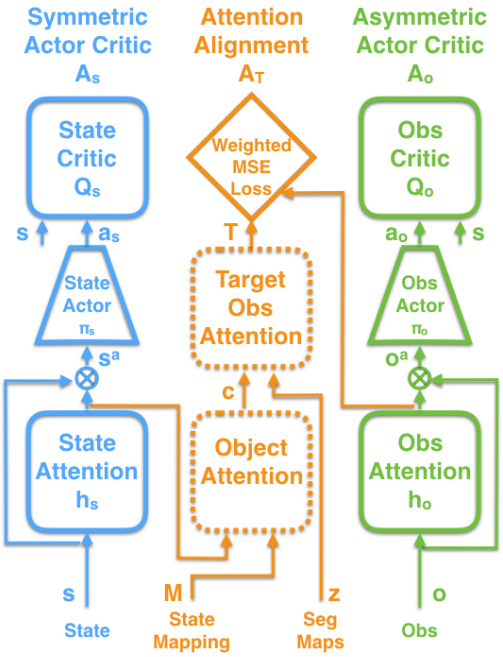

Figure 2: Attention-Privileged Reinforcement Learning model structure. Blue, green and orange colour coding represent symmetric actor critic, asymmetric actor critic and attention alignment modules (A_s, A_o, A_T) respectively. Dashed blocks represent predefined operators that are not learnt. The diamond block represents the attention alignment loss function. The remaining blocks represent trainable networks. The $\otimes$ operator represents element-wise multiplication. Each agent acts in a separate thread and experiences are shared using a shared replay buffer.

state relates to which object. Second, object-attention is converted to observation-space attention by performing a weighted sum over object-specific segmentation maps [1]:

$$c = M \cdot h_s(s), \qquad T = \sum_{i=0}^{N-1} c_i \cdot z_i \quad (2)$$

Here, $M \in \{0, 1\}^{N\times n_s}$ (n_s is the dimensionality of s) is an environment-specific, predefined adjacency matrix that maps the dimensions of s to each corresponding object, and $c \in [0, 1]^N$ is an attention vector over the N objects in the environment. c_i corresponds to the i^{th} object attention value. $z_i \in \{0, 1\}^{W\times H}$ is the binary segmentation map of the i^{th} object segmenting the object with the rest of the scene, and has the same dimensions as the image. z_i assigns values of 1 for pixels in the image occupied by the i^{th} object, and 0 elsewhere. $T \in [0, 1]^{W\times H}$ is the converted state-attention

[1] Simulators (e.g., (Todorov et al., 2012; Dosovitskiy et al., 2017)) commonly provide functionality to access these segmentations and semantic information for the environment state.

to observation-space attention to act as a target on which to train the observation-attention network h_o.

The observation module A_o also consists of three networks: Q_o^π, π_o, h_o (respectively critic, actor, and attention) with parameters $\{\phi_o, \theta_o, \psi_o\}$. The structure of this module is the same as A_s except the actor and critic now have asymmetric inputs. The actor's input is the attention-filtered version of the observation, $o_t^a = h_o(o_t) \odot o_t$ [1]. The actor and critic π_o and Q_o are trained with the asymmetric DDPG actor and Bellman error losses respectively defined in Section 2.2. The main difference between A_o and A_s is that the observation attention network h_o is trained on both the actor loss and an object-weighted mean squared error loss:

$$\mathcal{L}_{h_o} = E_{o,s\sim\pi_b}[\frac{1}{2}\sum_{ij}\frac{1}{w_{ij}}(h_o(o)-T)_{ij}^2 - \nu Q_o(s,\pi_o(o^a))] \tag{3}$$

where weights w_{ij} correspond to the fraction of the partial observation o that the object present in $o_{i,j,1:3}$ occupies, and ν represents a hyperparameter for the relative weighting of both loss components. The weight terms, w, ensure that the attention network becomes invariant to the size of objects during training and does not simply fit to the most predominant object in the scene.

During training, experiences are collected evenly from both state and observation based agents and stored in a shared replay buffer (similar to Schwab et al. (2019)). This is to ensure that: 1. Both state-based critic Q_s and observation-based critic Q_o observe states that would be visited by either of their respective policies. 2. The attention modules h_s and h_o are trained on the same data distribution to better facilitate alignment. 3. Efficient discovery of highly performing states from π_s are used to speed up learning of π_o.

Algorithm 1 shows the pseudocode for a single actor implementation of APRiL. In practice, in order to speed up data collection and gradient computation, we parallelise the agents and environments and ensure equal data is generated by state- and image-based agents.

4. Experiments

To demonstrate the performance and generality of our method, we apply APRiL to a range of environments, and compare with two competitive asymmetric actor-critic baselines and various ablations. We evaluate APRiL over different metrics to investigate how attention helps with robustness and generalisation to unseen environments and transfer scenarios. Further experimental details can be found in Appendix C and videos of policy rollouts together with attention maps can be found here.

[1] In practice, the output of $h_o(o_t)$ is tiled to match the number of channels that the image contains

Algorithm 1 Attention-Privileged Reinforcement Learning

Initialize the actor-critic modules A_s, A_o, attention alignment module A_T, replay buffer R
for episode$= 1$ **to** M **do**
  Initial state s_0
  Set DONE $\leftarrow$ FALSE
  while $\neg$ DONE **do**
    Render image observation o_t and segmentation maps z_t:
      $o_t, z_t \leftarrow$ renderer(s_t)
    if episode **mod** $2 = 0$ **then**
      Obtain action a_t using obs-behavioral policy and obs-attention network:
        $a_t \leftarrow \pi_o(h_o(o_t) \odot o_t)$
    else
      Obtain action a_t using state-behavioral policy and state-attention network:
        $a_t \leftarrow \pi_s(h_s(s_t) \odot s_t)$
    end if
    Execute action a_t, receive reward r_t, DONE flag, and transition to s_{t+1}
    Store $(s_t, o_t, z_t, a_t, r_t, s_{t+1}, o_{t+1})$ in R
  end while
  for $n = 1$ **to** N **do**
    Sample minibatch $\{s, o, z, a, r, s', o'\}_0^B$ from R
    Optimise state- critic, actor, and attention using $\{s, a, r, s'\}_0^B$ with A_s
    Convert state-attention to target observation-attention $\{T\}_0^B$ using $\{s, o, z\}_0^B$ with A_T
    Optimise observation- critic, actor, and attention using $\{s, o, T, a, r, s', o'\}_0^B$ with A_o
  end for
end for

4.1. Evaluation Protocol

In order to investigate APRiL under varying conditions, we evaluate in various environments of increasing complexity covering simple 2D navigation, 3D reaching, and 2D dynamic locomotion (see Appendix A for further details):

1. *NavWorld*: In this 2D environment, the goal is for the circular agent to reach the triangular target in the presence of distractors. The agent is sparsely rewarded if the target is reached.

2. *JacoReach*: In this 3D environment the goal of the Kinova arm (Campeau-Lecours et al., 2017) agent is to reach the diamond ShapeStacks object (Groth et al., 2018) in the presence of distractors. The agent is rewarded for approaching and reaching its goal.

3. *Walker2D*: In this slightly modified (see Appendix A) 2D Deepmind Control Suite environment (Tassa et al., 2018) the goal of the agent is to walk forward as far as possible within a time-limit. The agent receives a reward for moving forward as well as a reward for keeping its torso upright.

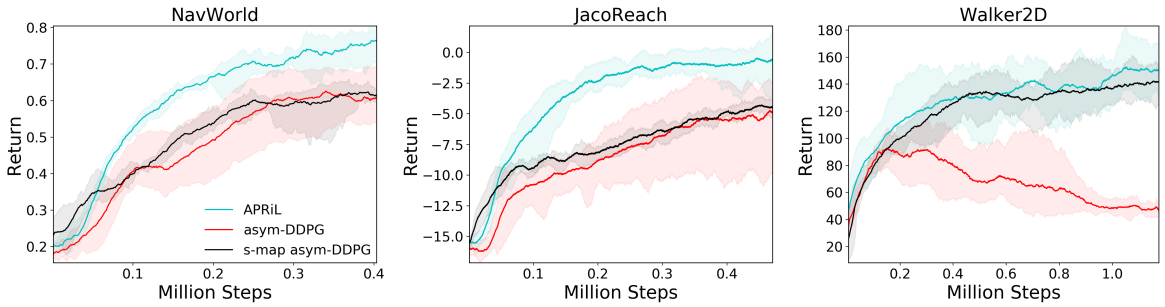

Figure 3: Learning curves for APRiL, the asymmetric DDPG baseline, and the state-mapping asymmetric DDPG baseline. **Solid line**: mean performance. **Shaded region**: covers minimum and maximum performances across 5 seeds. APRiL performs stronger or commensurate to the baselines on all domains.

4.2. Key research questions

For these domains we randomise visuals during training to enable generalisation to these variable aspects of the environment. We randomise a combination of: camera position and orientation, textures, materials, colours, object locations, background. Refer to Appendix B for more details.

We investigate the following questions to evaluate how well APRiL accommodates for the transfer across visually distinct environments: Does APRiL 1. Increase **sample-efficiency** during training? 2. Affect **interpolation** performance on unseen environments from the training distribution? 3. Affect **extrapolation** performance on environments outside the training distribution?

We start by comparing APRiL against two competitive baselines that also exploit privileged information during training. We compare against the *Asymmetric DDPG* (asym-DDPG) baseline (Pinto et al., 2018) to evaluate the importance of privileged attention and shared replay for learning and robustness to distractors. Our second baseline, *State-Mapping Asymmetric DDPG* (s-map asym-DDPG), introduces a bottleneck layer trained to predict the environment state using an L_2 loss. This is another intuitive approach to exploiting state information in simulation (Zhang et al., 2016), without incorporating object-centric attention and leveraging privileged object segmentations. We note that since this baseline learns state estimation it is not expected to extrapolate well to domains with additional distractor objects and varying state spaces (with respect to the training domain).

We continue by qualitatively analysing APRiL's learnt attention maps (both on interpolated and extrapolated domains). Finally, we perform an ablation study to investigate which parts of the APRiL contribute to performance gains. This ablation consists of the following models:

1. *APRiL no self-supervision* (APRiL no sup): APRiL except without the self-supervision provided by the state agent to train the observation-based attention. Both agents are still equipped with an attention module, but the observation attention must now learn without guidance from the state agent. Without bootstrapping from the state agent in this way we expect learning of informative observation-based attention to be hindered.

2. *APRiL no shared buffer* (APRiL no share): APRiL except each agent has its own replay buffer, instead of one shared buffer, and hence does not share experiences during training. Under this setting, the observation agent cannot benefit from earlier visitation of lucrative states by the state agent. Both agents have an attention module and attention alignment still occurs.

3. *APRiL no background* (APRiL no back): APRiL except the state agent's attention is no longer used to calculate object-space attention values c. Instead, all objects are given equal attention and we hence learn a background suppressor. This most competitive ablation investigates how important object suppression is for learning, robustness, and generalisation. Both agents still have attention and a shared replay buffer.

4.3. Performance on the training distribution

We evaluate the performance on all domains during training and observe APRiL's benefits. As seen in Figure 3, APRiL outperforms the asym-DDPG baseline across all domains. APRiL not only helps learn useful representations quicker (improving learning rate) but also improves final policy performance. Interestingly, the s-map asym-DDPG baseline, which learns to map to environment states, does not outperform asym-DDPG and does not match APRiL's performance for *NavWorld* and *JacoReach*. For these domains, predicting states (including those of distractors) is difficult [1] and prediction errors limit policy performance. For *Walker2D*, in the absence of distractor objects, s-map

[1] For *JacoReach* prediction errors and policy performance are sensitive to state-space. In Figure 3 we plot the best performing state-space. Refer to Appendix E for further details.

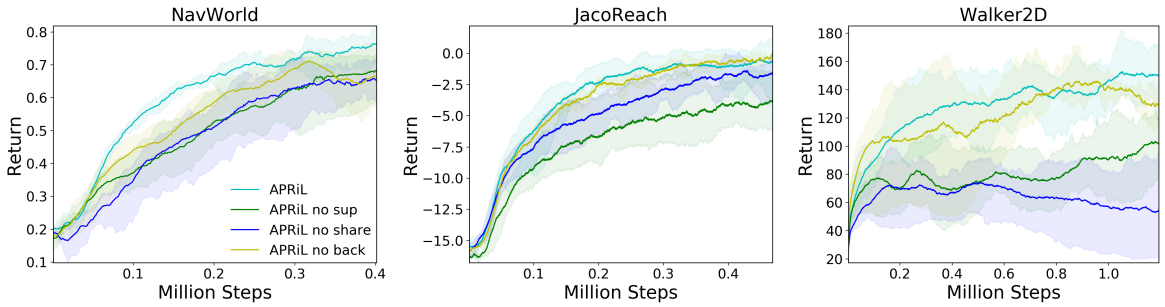

Figure 4: Learning curves for APRiL and its ablations. **Solid line**: mean performance. **Shaded region**: covers minimum and maximum performances across 5 seeds. APRiL's attention and shared replay lead to stronger or commensurate performance.

asym-DDPG is a competitive baseline and APRiL provides marginal gains. Refer to Appendix E for an in-depth analysis of the s-map asym-DDPG baseline's performance and limitations in particular with respect to explicit regression to representations of rotation.

The ablations demonstrate, in Figure 4, that self-supervision and shared replay both individually, to different extents, provide performance gains over the baselines and progress closer to APRiL's performance. For *Walker2D*, shared replay is crucial as it stabilises learning (observe APRiL, APRiL no back, APRiL no sup), due to constant visitation for highly performing states. Suppression of task-irrelevant, yet highly varying, information also improves learning rate as it simplifies the observation space. For this reason, APRiL no back proves to be a competitive ablation, approaching the performance of APRiL for *JacoReach* and *Walker2D*. For these domains, the background occupies the majority of the observation space and ignoring it already suppresses most of the irrelevant, highly varying, information. Minimal improvement can be achieved by suppressing additional irrelevant objects. None of the ablations, however, are able to outperform the full APRiL framework, demonstrating that the combination of a shared replay buffer and state-space-informed image-attention module cooperate constructively toward more efficient feature learning and effective policy and critic updates.

4.4. Interpolation: transfer to domains from the training distribution

We evaluate the performance of all actor-critic algorithms on a held-out set of simulation parameters, unseen during training, from the training distribution. For a detailed description of the training distribution for each domain please refer to Appendix B. For both *NavWorld* and *JacoReach*, the interpolated environments have the same number of distractors, sampled from the same object catalogue, as the training distribution. Figure 5 plots policy performance in terms of the additional return over an agent whose actions are random. We plot this performance metric to better gauge the significance of the performance drop on the held-out domains. For APRiL, we observe no degradation in policy performance between training and interpolated domains. We see a very similar trend for the baselines and ablations. However, as APRiL performs significantly better on the training distribution than the baselines (apart from *Walker2D* where performance gains are marginal over s-map asym-DDPG), its final performance on the interpolated domains is significantly better. APRiL also outperforms all its ablations (apart from APRiL no back for *JacoReach* where performance is comparable), to varying degrees, emphasising the benefits of both privileged attention and a shared replay.

4.5. Extrapolation: transfer to domains outside the training distribution

We investigate the performance on simulation parameters outside the training distribution. In particular, we investigate how well APRiL, its ablations, and baselines, generalise to increasingly cluttered environments with more distractor objects than seen during training. For *NavWorld* and *JacoReach*, we run two sets of increasingly extrapolated experiments with an additional 4 or 8 distractors (refered to as ext-4 and ext-8 in Figure 5). The textures and colours of these objects are sampled from a held-old out set of simulation parameters not seen during training. For *NavWorld*, the locations and orientations of the additional distractors are randomly sampled. For *JacoReach*, the locations are sampled from arcs of two concentric circles of different radii (extrapolated arcs and radii to those seen during training). For each domain, the shapes of the additional distractor objects are sampled from the training catalogue of distractor objects. Please refer to Figure 6 for examples of the extrapolated domains.

Figure 5 compares performances on the extrapolated sets (except *Walker2D*) varying in difficulty (ext-4 and ext-8). APRiL yields considerable performance gains over both baselines on each extrapolated domain thanks to its im-

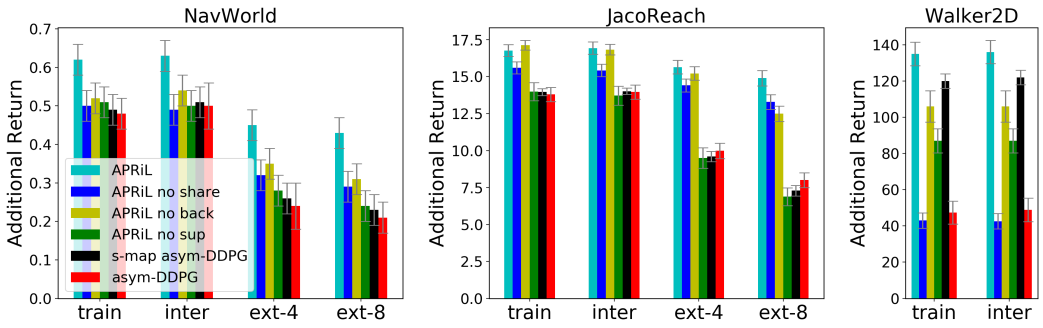

Figure 5: Comparison of additional average return (additional return achieved over a random agent) between training, interpolated and extrapolated environments (100 each). Results reflect the mean and 2 standard deviations for additional average return (across 5 seeds). APRiL generalises favorably thanks to its attention module (as seen when comparing performance drop between **train** and **ext-8** domains for APRiL, APRiL no share, and the remaining models), and hence outperforms the baselines and ablations on increasingly extrapolated domains (**ext-8**).

proved performance during training and generalisation capabilities. Figure 5 shows that each baseline experiences drastic policy degradation. Without attention, neither baseline is able to generalise favorably to observations substantially different from the training domain. APRiL's generalisation is so effective that, for the hardest domain with additional 8 distractors (ext-8), its performance degrades by only 14% opposed to 42% and 48% (for asym-DDPG and s-map asym-DDPG respectively) for *JacoReach* and 31% opposed to 56% and 53% for *NavWorld*[1].

APRiL generalises favorably due to the attention module. Figure 6 shows that attention generalises and suppresses the additional distractors, thereby effectively converting the held-out observations to those seen during training, which the image-policy can handle. The ablations in Figure 5 confirm that for this setting, distractor suppression is crucial. This is particularly prominent for *JacoReach* when we compare the maximum degradation in policy performance of APRiL, APRiL no share, APRiL no back and APRiL no sup (11%, 15%, 27% and 51% respectively). APRiL and APRiL no share both align attention between image and state agents during training, and therefore effectively suppress distractors (yielding a favourable decrease in policy performance of only 11% and 15%). APRiL no back learns a background suppressor, but does not suppress the distractors (leading to a larger degradation of 27%). APRiL no sup has an attention module trained only on the asymmetric actor-critic loss and yields the worst extrapolated performance (51% policy degradation). For these extrapolated domains, the successful suppression of the background **and** additional distractors (achieved only by the full APRiL framework), creates policy invariance with respect to them and helps generalise.

[1]Percentage decrease is taken with respect to additional return on the training domain.

4.6. Attention Module Analysis

To qualitatively comprehend the role of the attention, we visualise APRiL's attention maps (Figure 6, 8, 9 and these videos) on both interpolated and extrapolated domains. For *NavWorld*, attention is correctly paid to all relevant aspects (agent and target; circle and triangle respectively). Attention generalises reasonably well to the extrapolated environments. For *JacoReach*, attention looks at the target, diamond-shaped, object as well as every other link (alternating links) of the Kinova arm. Interestingly, APRiL learns that as the arm is a constrained system, the state of every other link can be indirectly inferred without explicit attention. The state of the unobserved link can be inferred by observing the links either side of it. The entropy loss over the state-attention module encourages this form of attention over minimal set of objects. Attention here generalises very well to the extrapolated domains and correctly suppreses the additional distractors, leading to policy robustness with respect to them. For *Walker2D*, we observe attention that is dynamic in object space. The attention module attends different subsets of links depending on the state of the system (see Figure 9). When the walker is upright, walking, and collapsing, APRiL pays attention to the lower limbs, every other link, and foot and upper body, respectively. We suspect that in these scenarios, the optimal action depends most on the state for the lower links (due to stability), every link (coordination), and foot and upper body (large torque required), respectively.

5. Related Work

A large body of work investigates the problem of learning robust policies that generalise well outside of the training distribution. Work on **transfer learning** leverages representations from one domain to efficiently solve a problem

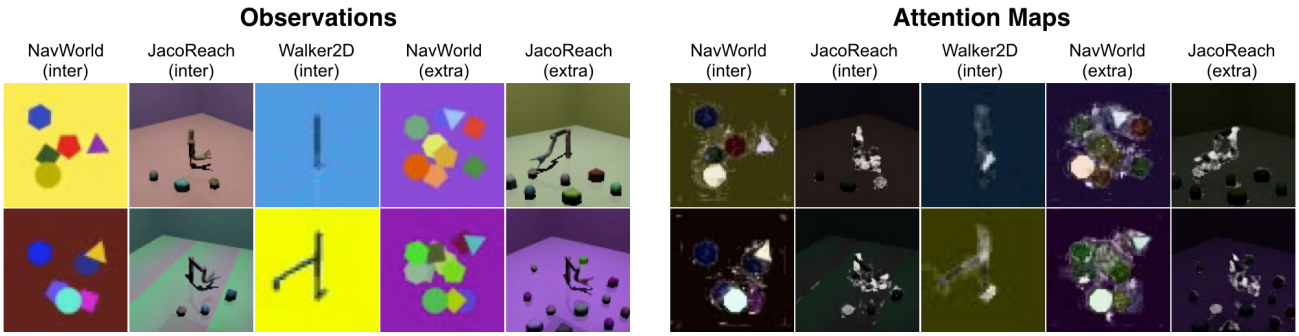

Figure 6: Example held-out domains with corresponding APRiL attention maps. For the extrapolated domain columns (extra), the first and second rows correspond to **ext-4** and **ext-8** respectively. White and black signify high and low attention values. Attention correctly suppresses background and distractors for all domains and helps generalise favourably.

from a different domain (Donahue et al., 2014; Oquab et al., 2014; Rusu et al., 2016). In particular, **domain adaptation** techniques aim to adapt a learned model to a specific target domain, often optimising models such that representations are invariant to the shift in the target domain (Ganin et al., 2016; Long et al., 2015; Bousmalis et al., 2016; Wulfmeier et al., 2017). These methods commonly require data from the target domain in order to transfer and adapt effectively.

In contrast, **domain randomisation** aims at covering a distribution of environments by randomising visual (Tobin et al., 2017) or dynamical parameters (Peng et al., 2018) during training in order to generalise (Sadeghi & Levine, 2016; Rajeswaran et al., 2016; Viereck et al., 2017; Held et al., 2017; OpenAI et al., 2018). In doing so, such methods shift the focus from adaptation to specific environments to **generalisation and robustness** by covering a wide range of variations. Recent work automatically varies the distribution of domains during training (Akkaya et al., 2019) or trains a canonical invariant image representation (James et al., 2019). However, while domain randomisation can enable us to learn robust policies, it leads to a significant increase in training time due to the increased variability in the environment (OpenAI et al., 2018), and can reduce asymptotic performance. Our work partially addresses this fact by training two agents, one of which is not affected by visual randomisations.

Existing comparisons in the literature demonstrate that, even without domain randomisation, the increased dimensionality and potential partial observability complicates learning for RL agents (Tassa et al., 2018; Schwab et al., 2019; Watter et al., 2015; Lesort et al., 2018). In this context, accelerated training has been achieved by using **access to privileged information** such as environment states to asymmetrically train the critic in actor-critic RL (Schwab et al., 2019; Pinto et al., 2018; Foerster et al., 2018). In addition to using additional information to train the critic, Schwab et al. (2019) use a **shared replay buffer** for data generated by image- and state-based actors to further accelerate training for the image-based agent. Our method extends these approaches by sharing information about relevant objects by aligning agent-integrated attention mechanisms between an image- and state-based actors.

Recent experiments have demonstrated the strong dependency and bidirectional interaction between attention and learning in human subjects (Leong et al., 2017). In the context of machine learning, **attention mechanisms** have been integrated into RL agents to increase robustness and enable interpretability of an agent's behaviour (Sorokin et al., 2015; Choi et al., 2017; Mott et al., 2019). In comparison to these works, we focus on utilising the attention mechanism as an interface to transfer information between two agents to enable faster training and better generalisation.

6. Conclusion

We introduce Attention-Privileged Reinforcement Learning (APRiL), an extension to asymmetric actor-critic algorithms that leverages attention mechanisms and access to privileged information such as simulator environment states. The method benefits in two ways in addition to asymmetry between actor and critic: via aligning attention masks between image- and state-space agents, and by sharing a replay buffer. Since environment states are not affected by visual randomisation, we are able to learn efficiently in the image domain especially during domain randomisation where feature learning becomes increasingly difficult. Evaluation on a diverse set of environments demonstrates significant improvements over competitive baselines including asym-DDPG and s-map asym-DDPG; and show that APRiL learns to generalise favourably to environments not seen during training (both within and outside of the training distribution). Finally, we investigate the relative importance of the different components of APRiL in an extensive ablation.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pp. 265–283, 2016.

Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., and Erhan, D. Domain separation networks. In *Advances in Neural Information Processing Systems*, pp. 343–351, 2016.

Campeau-Lecours, A., Lamontagne, H., Latour, S., Fauteux, P., Maheu, V., Boucher, F., Deguire, C., and L'Ecuyer, L.-J. C. Kinova modular robot arms for service robotics applications. *Int. J. Robot. Appl. Technol.*, 5(2):49–71, July 2017. ISSN 2166-7195. doi: 10.4018/IJRAT.2017070104. URL https://doi.org/10.4018/IJRAT.2017070104.

Choi, J., Lee, B.-J., and Zhang, B.-T. Multi-focus attention network for efficient deep reinforcement learning. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pp. 647–655, 2014.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017.

Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

Groth, O., Fuchs, F. B., Posner, I., and Vedaldi, A. Shapestacks: Learning vision-based physical intuition for generalised object stacking. In *ECCV (1)*, volume 11205 of *Lecture Notes in Computer Science*, pp. 724–739. Springer, 2018.

Held, D., McCarthy, Z., Zhang, M., Shentu, F., and Abbeel, P. Probabilistically safe policy transfer. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 5798–5805. IEEE, 2017.

James, S., Wohlhart, P., Kalakrishnan, M., Kalashnikov, D., Irpan, A., Ibarz, J., Levine, S., Hadsell, R., and Bousmalis, K. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 12627–12637, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Konda, V. R. and Tsitsiklis, J. N. Actor-critic algorithms. In *Advances in neural information processing systems*, pp. 1008–1014, 2000.

Leong, Y. C., Radulescu, A., Daniel, R., DeWoskin, V., and Niv, Y. Dynamic interaction between reinforcement learning and attention in multidimensional environments. *Neuron*, 93(2):451 – 463, 2017. ISSN 0896-6273. doi: https://doi.org/10.1016/j.neuron.2016.12.040. URL http://www.sciencedirect.com/science/article/pii/S089662731631039X.

Lesort, T., Díaz-Rodríguez, N., Goudou, J.-F., and Filliat, D. State representation learning for control: An overview. *Neural Networks*, 2018.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning, 2015.

Long, M., Cao, Y., Wang, J., and Jordan, M. I. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Mott, A., Zoran, D., Chrzanowski, M., Wierstra, D., and Rezende, D. J. Towards interpretable reinforcement learning using attention augmented agents. *ArXiv*, abs/1906.02500, 2019.

OpenAI, :, Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W. Learning dexterous in-hand manipulation, 2018.

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1717–1724, 2014.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 1–8. IEEE, 2018.

Pinto, L., Andrychowicz, M., Welinder, P., Zaremba, W., and Abbeel, P. Asymmetric actor critic for image-based robot learning. *Robotics: Science and Systems*, 2018.

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., and Andrychowicz, M. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

Rajeswaran, A., Ghotra, S., Ravindran, B., and Levine, S. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

Romero, A., Ballas, N., Kahou, S., Chassang, A., Gatta, C., and Bengio, Y. Imagenet classification with deep convolutional neural networks. In *International Conference on Learning Representations*, 2015.

Rusu, A. A., Vecerik, M., Rothörl, T., Heess, N., Pascanu, R., and Hadsell, R. Sim-to-real robot learning from pixels with progressive nets. *arXiv preprint arXiv:1610.04286*, 2016.

Sadeghi, F. and Levine, S. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

Schwab, D., Springenberg, T., Martins, M. F., Lampe, T., Neunert, M., Abdolmaleki, A., Herkweck, T., Hafner, R., Nori, F., and Riedmiller, M. Simultaneously learning vision and feature-based control policies for real-world ball-in-a-cup. *arXiv preprint arXiv:1902.04706*, 2019.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *ICML*, 2014.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Sorokin, I., Seleznev, A., Pavlov, M., Fedorov, A., and Ignateva, A. Deep attention recurrent q-network. *arXiv preprint arXiv:1512.01693*, 2015.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pp. 23–30. IEEE, 2017.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.

Viereck, U., Pas, A. t., Saenko, K., and Platt, R. Learning a visuomotor controller for real world robotic grasping using simulated depth images. *arXiv preprint arXiv:1706.04652*, 2017.

Watter, M., Springenberg, J., Boedecker, J., and Riedmiller, M. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pp. 2746–2754, 2015.

Wulfmeier, M., Posner, I., and Abbeel, P. Mutual alignment transfer learning. *arXiv preprint arXiv:1707.07907*, 2017.

Zhang, F., Leitner, J., Upcroft, B., and Corke, P. Vision-based reaching using modular deep networks: from simulation to the real world. *arXiv preprint arXiv:1610.06781*, 2016.

Zhou, Y., Barnes, C., Lu, J., Yang, J., and Li, H. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5745–5753, 2019.

Appendix

A. Environments

1. *NavWorld*: In this sparse reward, 2D environment, the goal is for the circular agent to reach the triangular target in the presence of distractor objects. Distractor objects have 4 or more sides and apart from changing the visual appearance of the environment cannot affect the agent. The state space consists of the $[x, y]$ locations of all objects. The observation space comprises RGB images of dimension $(60 \times 60 \times 3)$. The action space corresponds to the velocity of the agent. The agent only obtains a sparse reward of $+1$ if the particle is within ϵ of the target, after which the episode is terminated prematurely. The maximum episodic length is 20 steps, and all object locations are randomised between episodes.

2. *JacoReach*: In this 3D environment the goal of the agent is to move the Kinova arm (Campeau-Lecours et al., 2017) such that the distance between its hand and the diamond ShapeStacks object (Groth et al., 2018) is minimised. The state space consists of the quaternion position and velocity of each joint as well as the Cartesian positions of each ShapeStacks object. The observation space comprises RGB images and is of dimension $(100 \times 100 \times 3)$. The action space consists of the desired relative quaternion positions of each joint (excluding the digits) with respect to their current positions. Mujoco uses a PD controller to execute 20 steps that minimises the error between each joint's actual and target positions. The agent's reward is the negative squared Euclidean distance between the Kinova hand and diamond object plus an additional discrete reward of $+5$ if it is within ϵ of the target. The episode is terminated early if the target is reached. All objects are out of reach of the arm and equally far from its base. Between episodes the locations of the objects are randomised along an arc of fixed radius with respect to the base of the Kinova arm. The maximum episodic length is 20 agent steps.

3. *Walker2D*: In this 2D modified Deepmind Control Suite environment (Tassa et al., 2018) with a continuous action-space the goal of the agent is to walk forward as far as possible within 300 steps. We introduce a limit to episodic length as we found that in practice this helped stabilise learning across all tested algorithms. The observation space comprises of 2 stacked RGB images and is of dimension $(40 \times 40 \times 6)$. Images are stacked so that velocity of the walker can be inferred. The state space consists of quaternion position and velocities of all joints. The absolute positions of the walker along the x-axis is omitted such that the walker learns to become invariant to this. The action space is setup in the same way as for the *JacoReach* environment. The reward is the same as defined in (Tassa et al., 2018) and consists of two multiplicative terms: one encouraging moving forward beyond a given speed, the other encouraging the torso of the walker to remain as upright as possible. The episode is terminated early if the walker's torso falls beyond either $[-1, 1]$ radians with the vertex or $[0.8, 2.0]$m along the z axis.

B. Randomisation Procedure

In this section we outline the randomisation procedure taken for each environment during training.

1. *NavWorld*: Randomisation occurs at the start of every episode. We randomise the location, orientation and colour of every object as well as the colour of the background. We therefore hope that our agent can become invariant to these aspects of the environment.

2. *JacoReach*: Randomisation occurs at the start of every episode. We randomise the textures and materials of every ShapeStacks object, Kinova arm and background. We randomise the locations of each object along an arc of fixed radius with respect to the base of the Kinova arm. Materials vary in reflectance, specularity, shininess and repeated textures. Textures vary between the following: noisy (where RGB noise of a given colour is superimposed on top of another base colour), gradient (where the colour varies linearly between two predefined colours), uniform (only one colour). Camera location and orientation are also randomised. The camera is randomised along a spherical sector of a sphere of varying radius whilst always facing the Kinova arm. We hope that our agent can become invariant to these randomised aspects of the environment.

3. *Walker2D*: Randomisation occurs at the start of every episode as well as after every 50 agent steps. We introduce additional randomisation between episodes due to their increased duration. Due to the MDP setup, intra-episodic randomisation is not an issue. Materials, textures, camera location and orientation, are randomised in the same procedure as for *JacoReach*. The camera is setup to always face the upper torso of the walker.

C. Implementation details

In this section we provide more details on our training setup. Refer to table 1 for the model architecture for each component of APRiL and the asymmetric DDPG baseline. *Obs Actor* and *Obs Critic* setup are the same for both APRiL and the asymmetric DDPG baseline. *Obs Actor* model structure

comprises of the convolutional layers (without padding) defined in table 1 followed by one fully connected layer with 256 hidden units (FC([256])). The state-mapping asymmetric DDPG baseline has almost the same architecure as *Obs Actor*, except there is one additional fully connected layer, directly after the convolutional layers that has the same dimensions as the environment state space. When training this intermediate layer on the L_2 state regressor loss, the state targets are normalised using a running mean and standard deviation, similar to DDPG, to ensure each dimension is evenly weighted and to stabilise targets. All layers use ReLU (Romero et al., 2015) activations and layer normalisation (Ba et al., 2016) unless otherwise stated. Each actor network is followed by a tanh activation and rescaled to match the limits of the environment's action space.

Table 1: Model architecture. FC() and Conv() represent a fully connected and convolutional network. The arguments of FC() and Conv() take the form [nodes] and [channels, square kernel size, stride] for each hidden layer respectively.

Domain	NavWorld and JacoReach	Walker2D
State Actor Obs Actor	FC([256]) Conv([[18, 7, 1], [32, 5, 1], [32, 3, 1]])	FC([256]) Conv([[18, 8, 2], [32, 5, 1], [16, 3, 1], [4, 3, 1]])
State Critic Obs Critic	FC([64, 64]) FC([64, 64])	FC([400, 300]) FC([400, 300])
State Attention Obs Attention	FC([256]) Conv([[32, 8, 1], [32, 5, 1], [64, 3, 1]])	FC([256]) Conv([[32, 8, 1], [32, 5, 1], [64, 3, 1]])
Replay Size	10^4	2×10^5

The *State Attention* module includes the fully connected layer defined in table 1 followed by a Softmax operation. The *Obs Attention* module has the convolutional layers (with padding to ensure constant dimensionality) outlined in table 1 followed by a fully connected convolutional layer (Conv($[1, 1, 1]$)) with a Sigmoid activation to ensure the outputs vary between 0 and 1. The output of this module is tiled in order to match the dimensionality of the observation space.

During each iteration of APRiL (for both A_o and A_s) we perform 50 optimization steps on minibatches of size 64 from the replay buffer. The target actor and critic networks are updated with a Polyak averaging of 0.999. We use Adam (Kingma & Ba, 2014) optimizer with learning rate of 10^{-3}, 10^{-4} and 10^{-4} for critic, actor and attention networks. We use default TensorFlow (Abadi et al., 2016) values for the other hyperparameters. The discount factor, entropy weighting and self-supervised learning hyperparameters are $\gamma = 0.99$, $\beta = 0.0008$ and $\nu = 1$. To stabilize learning, all input states are normalized by running averages of the means and standard deviations of encountered states. Both actors employ adaptive parameter noise (Plappert et al., 2017) exploration strategy with initial std of 0.1, desired action std of 0.1 and adoption coefficient of 1.01. The settings for the baseline are kept the same as for APRiL where appropriate.

D. Attention Visualisation

Figures (8, 9) show APRiL's attention maps for policy rollouts on each environment and held-out domain. Attention attends to the task-relevant objects and generalises well.

E. State Mapping Asymmetric DDPG Ablation Study

We found that for *JacoReach*, the choice of state-space to regress to drastically affected the performance of the s-map asym-DDPG baseline. In particular, we observed that if we kept the regressor state as quaternions (for Jaco arm links; this is our default state-space setup), that the performance was considerably worse than regressing to cartesian positions and rotations, and significantly worse than simply regressing to cartesian positions (see Figure 7). Figure 10 demonstrates that it is the inability to accurately regress to quaternions and cartesian rotations that leads to inferior policy performance for these two s-map asym-DDPG ablations. Zhou et al. (2019) similarly observed that quaternions are hard for neural networks to regress and showed that it was due to their representations being discontinuous. It is for this reason why regressing **only** to cartesian positions performed best.

However, even with a representation which is better suited for learning, the agent's performance is still significantly below APRiL (see Figure 7). Given that the state-space agent used under the APRiL framework learns efficiently for this domain, this suggests that the remainder of the s-map asymmetric DDPG policy (layers dependent on the state-space predictor) is rather sensitive to inaccuracies in the regressor. Different methods for using privileged information, as given by APRiL's attention mechanism, provide more robust performance.

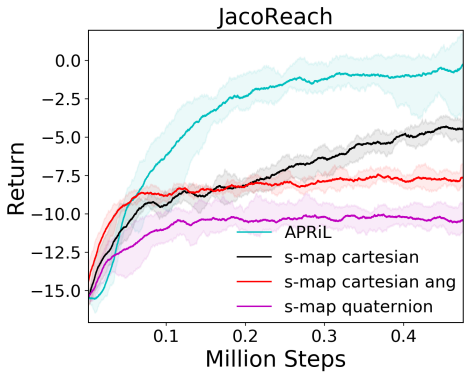

Figure 7: We compare learning of APRiL with variants of s-map asym-DDPG. For **s-map cartesian**, **s-map cartesian ang** and **s-map quaternion**, regressed states are cartesian position, cartesian position and rotation, and quaternions respectively (for Jaco arm - distractors are always cartesian).

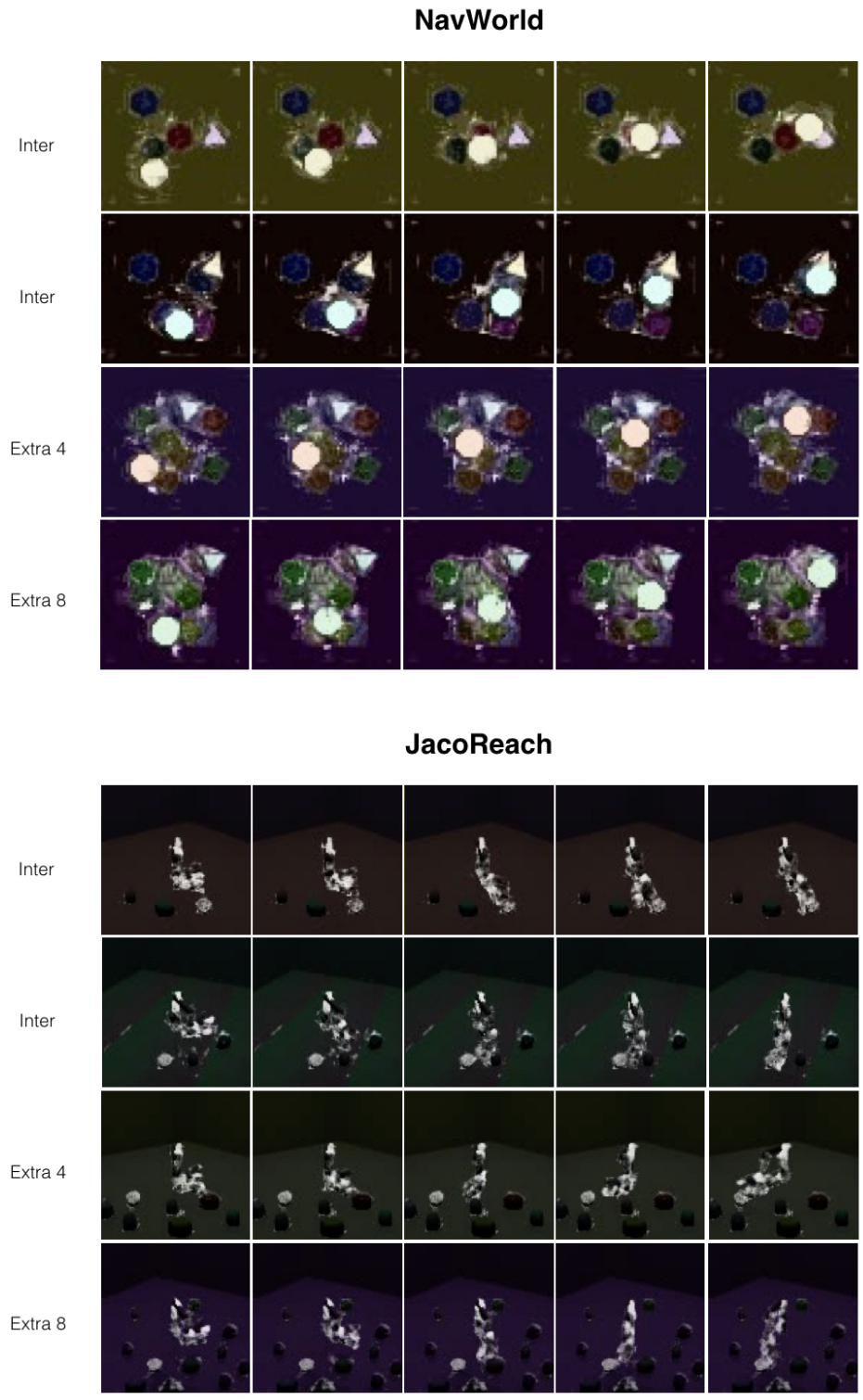

Figure 8: APRiL attention maps for policy rollouts on NavWorld and Jaco domains. White and black signify high and low attention values respectively. For NavWorld and JacoReach, attention is correctly paid only to the relevant objects (and Jaco links), even for the extrapolated domains. Refer to section 4.6 for more details.

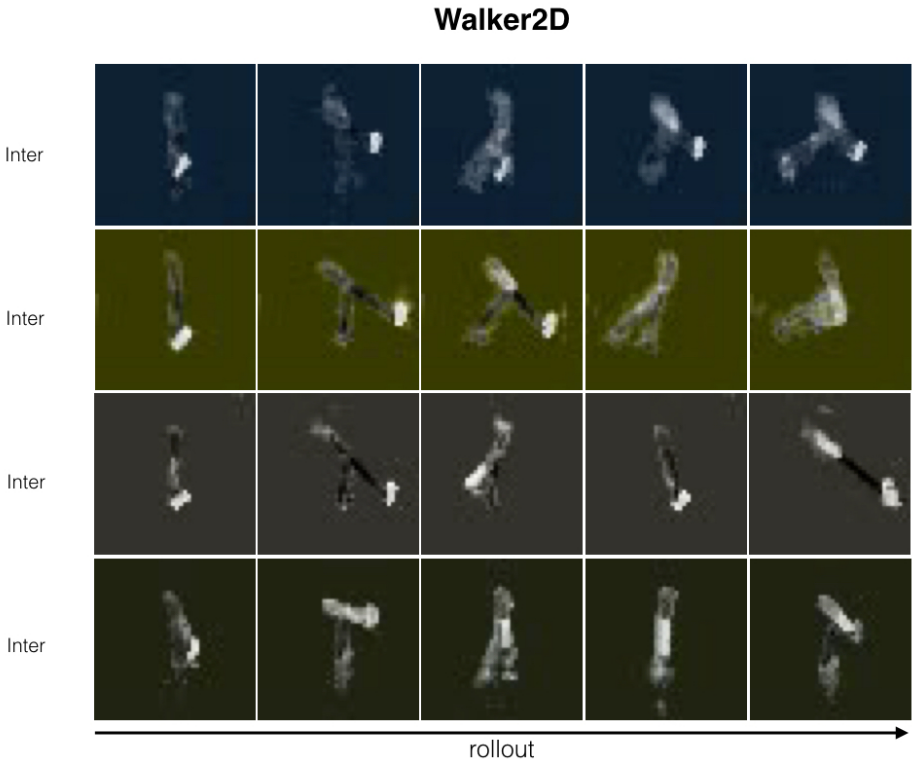

Figure 9: APRiL attention maps for policy rollouts on Walker domain. White and black signify high and low attention values respectively. Attention varies based on the state of the walker. When the walker is upright, high attention is paid to lower limbs. When walking, even attention is paid to every other limb. When about to collapse, high attention is paid to the foot and upper torso. Refer to section 4.6 for more details.

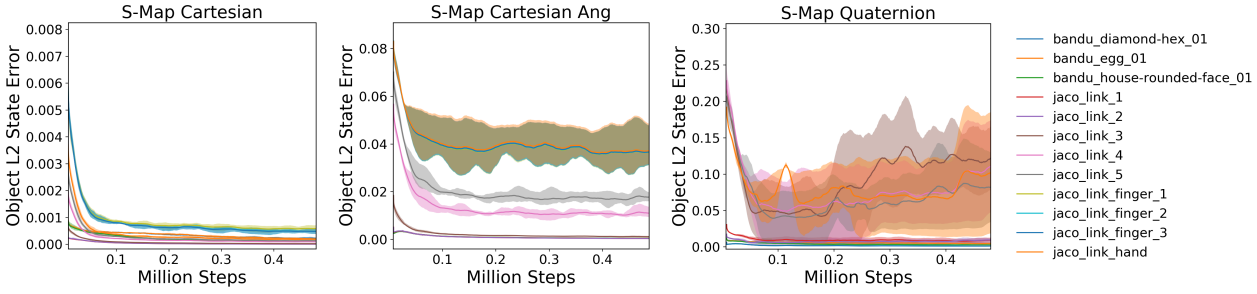

Figure 10: S-Map Asym-DDPG normalised state prediction errors. We compare individual object L_2 regressor losses (mean loss over states corresponding to a given object) between **s-map cartesian**, **s-map cartesian ang** and **s-map quaternion**. The object keys are on the right. **S-map quaternion** and **s-map cartesian ang** struggle to regress to quaternions and cartesian rotations and hence policy performance is restricted.