Attention-Privileged Reinforcement Learning

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

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

Abstract: Image-based Reinforcement Learning is known to suffer from poor sample efficiency and generalisation to unseen visuals such as distractors (task-independent aspects of the observation space). Visual domain randomisation encourages transfer by training over visual 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] .

Keywords: Robustness, Attention, Reinforcement Learning

1 Introduction

While image-based Deep Reinforcement Learning (RL) has recently provided significant successes in various high-data domains [1, 2, 3], its application to physical systems remains challenging due to expensive and slow data generation, challenges with respect to safety, and the need to be robust to unexpected changes in the environment.

When training visual models in simulation, we can obtain robustness either by adaptation to target domains [4, 5, 6], or by randomising system parameters with the aim of covering all possible environment parameter changes [7, 8, 9, 10, 11, 12]. Unfortunately, training under a distribution of randomised visuals [11, 12], can be substantially more difficult due to the increased variability. This often leads to a compromise in final performance [9, 7]. Furthermore, it is usually not possible to cover all potential environmental variations during training. Enabling agents to generalise to unseen visuals such as distractors (task-independent aspects of the observation space) is an important question in robotics where an agent's environment is often noisy (e.g. autonomous vehicles).

To increase robustness and reduce training time, we can make use of privileged information such as environment states, commonly accessible in simulators. By using lower-dimensional, more structured and informative representations directly as agent input, instead of noisy observations affected by visual randomisation, we can improve data efficiency and generalisation [13, 14].

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,

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

4th Conference on Robot Learning (CoRL 2020), Cambridge MA, USA.

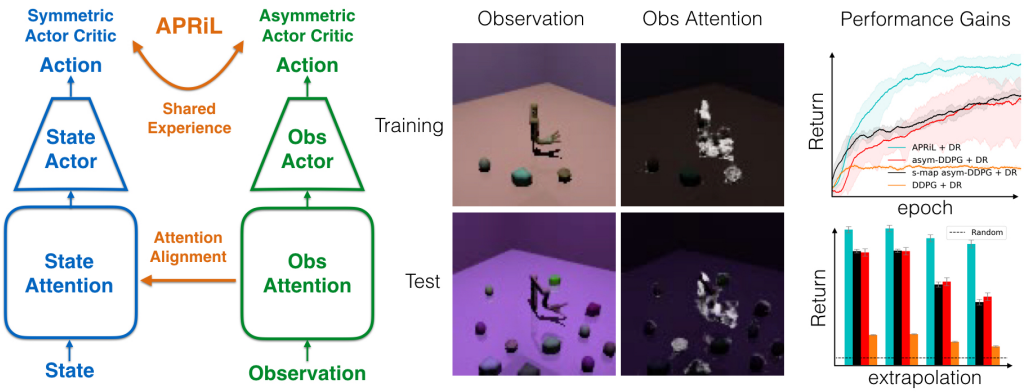

Figure 1: **Model diagram (left):** APRiL concurrently trains two attention augmented policies (one state-based, the other image-based). **Qualitative and quantitative results (middle & right):** By aligning the observation attention to that of the state, image-based attention quickly suppresses highly varying, task-irrelevant, information **(middle second column)**. This leads to increased learning rate **(top right)** and robustness to extrapolated domains with increasing levels of unseen additional distractors **(bottom right)**. For *JacoReach*, attention (**middle second column**; white and black signify high and low values) is paid only to the target object and jaco arm in training and extrapolated domains.

we can make use of information asymmetric actor-critic methods [10, 15] 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 readily and commonly available during training, such as simulator states and object segmentations [16, 17]), for increased robustness, sample efficiency, and generalisation to distractors. As a general extension to 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 whilst explicitly becoming invariant to task-irrelevant and noisy aspects of the environment (distractors). We demonstrate that this leads to faster image-based policy learning and increased robustness to task-irrelevant factors (both within and outside the training distribution). See Figure 1 for a visualisation of APRiL , its attention, and generalisation capabilities on one of our domains.

In addition, APRiL shares a replay buffer between both agents, which further accelerates training for the image-based policy. At test-time, the image-based policy can be deployed without 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 extrapolated and unseen settings with additional distractors.

2 Problem Formulation

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. [3] and Pinto et al. [10].

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, $P : S \times A \times S \rightarrow \{x \in \mathbb{R} | 0 \leq x \leq 1\}$ is the transition probability function, $r : S \times A \rightarrow \mathbb{R}$ 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 $\pi : S \rightarrow A$. 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) [10] 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 $\pi_\theta : O \rightarrow A$ (with parameters θ) used during training and test time, and a state-based Q-function (also known as critic) $Q^\pi_\phi : S \times A \rightarrow R$ (with parameters ϕ) which is only used during training.

To enable exploration, the method (like its symmetric version [18]) 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 [1]. Training examples sampled from the replay buffer are used to optimize the critic and actor. By minimizing 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:
$\mathcal{L}_{actor} = -E_{s,o\sim\pi_b(o)}[Q(s, \pi(o))]$.

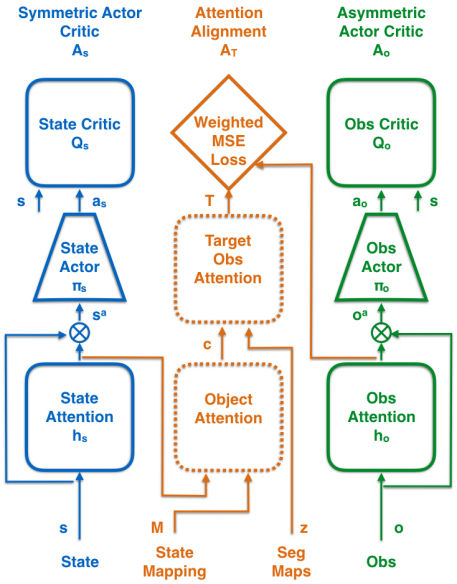

Figure 2: **APRiL 's architecture**. Blue, green and orange represent symmetric and asymmetric actor critic and attention alignment modules (A_s, A_o, A_T). The diamond represents the attention alignment loss. Dashed and solid blocks are non-trainable and trainable networks. The $\otimes$ operator signifies element-wise multiplication. Experiences are shared using a shared replay buffer.

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 [10] to train the observation based actor. Second, we additionally train a quicker learning state-based actor, while sharing replay buffers, and aligning attention mechanisms between both actors. We emphasise here that our approach can be applied to any asymmetric, off-policy, actor-critic method [19] with the expectation of similar performance benefits to those demonstrated in this paper. Specifically we choose to build off Asymmetric DDPG [10] due to its accessibility.

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 (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:

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

where β is a hyperparameter (set through grid-search, see Appendix C) to weigh the additional entropy objective, and π_b is the behaviour policy that obtained 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. We found that APRiL was not sensitive to the absolute value of β, only the magnitude, and was set low enough to not suppress task-relevant parts of the state-space.

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 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]:

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

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 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

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 ← FALSE
  while ¬ DONE do
    Render image observation o_t and segmentation maps z_t:
        o_t, z_t ← renderer(s_t)
    if episode mod 2 = 0 then
      Obtain action a_t using obs-behavioral policy and obs-attention network:
        a_t ← π_o(h_o(o_t) ⊙ o_t)
    else
      Obtain action a_t using state-behavioral policy and state-attention network:
        a_t ← π_s(h_s(s_t) ⊙ 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
```

critic, actor, and attention) with parameters $\{\phi_o, \theta_o, \psi_o\}$. The structure of this module is the same as

[1] Simulators (e.g., [16, 17]) commonly provide functionality to access these segmentations and semantic information for the environment state.

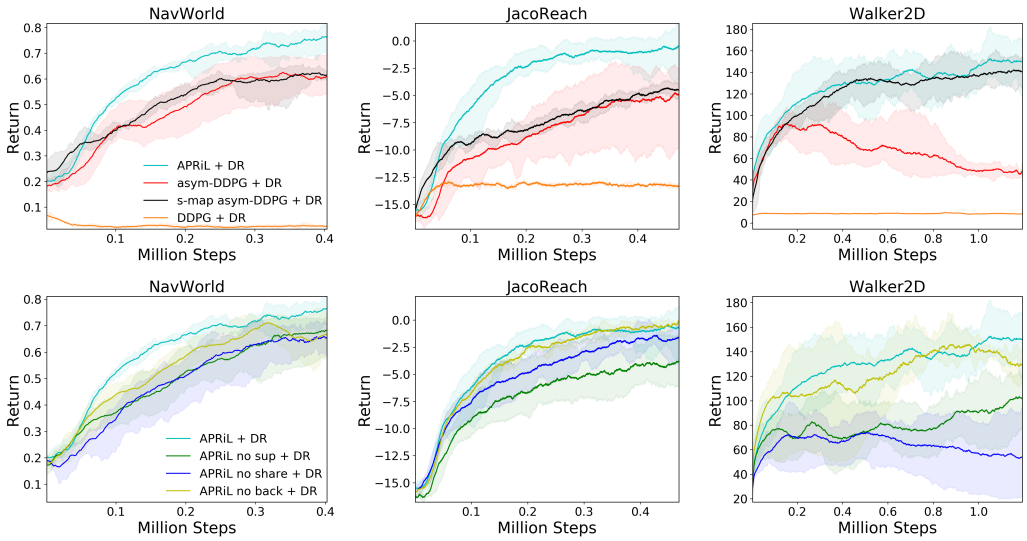

Figure 3: Learning curves for observation-based policies during Domain Randomisation (DR). **Top row**: comparison with baselines. **Bottom row**: comparison with ablations. **Solid line**: mean performance. **Shaded region**: covers minimum and maximum performances across 5 seeds. APRiL's attention and shared replay lead to stronger or commensurate performance.

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 in 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:

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

where weights w_{ij} denote the fraction of the image 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 (see Appendix C for exact value). 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. [15]). 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

We evaluate APRiL over the following environments (see Appendix A for more details): 1. *NavWorld*: the circular agent is sparsely rewarded for reaching the triangular target in the presence of distractors. 2. *JacoReach*: the Kinova arm is rewarded for reaching the diamond-shaped object in the presence of distractors. 3. *Walker2D*: this slightly modified (see Appendix A) Deepmind Control Suite environment [13] the agent is rewarded for walking forward whilst keeping its torso upright.

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

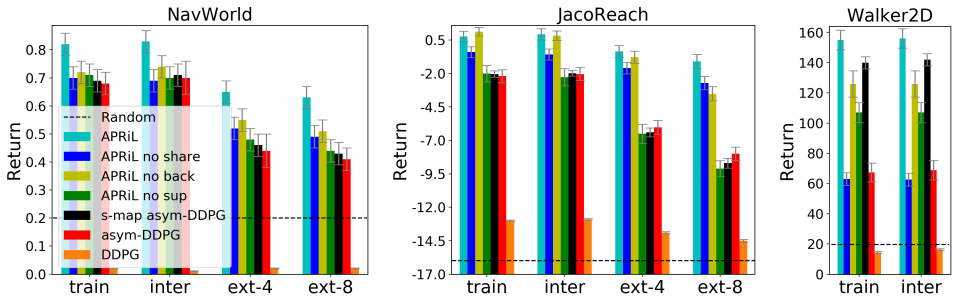

Figure 4: Comparing average return of the image-policy between training, interpolated and extrapolated domains (100 each). Plots reflect mean and 2 standard deviations for average return (5 seeds). APRiL generalises due to its attention and outperforms the baselines. We compare against a random agent to gauge the degree of degradation in policy performance between domains.

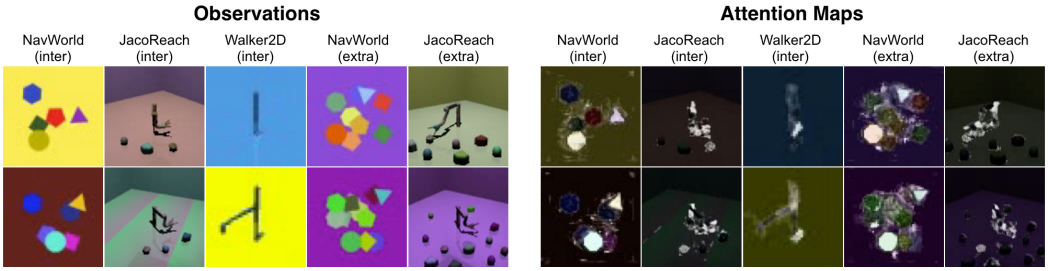

Figure 5: Held-out domains and APRiL attention maps. For the extrapolated domain columns (extra), top and bottom represent **ext-4** and **ext-8**. White/black signify high/low attention values. Attention suppresses the background and distractors and helps generalise.

During training, for APRiL , its ablations, and all baselines, we perform Domain Randomisation (DR) [7, 11], randomising the following environment parameters to enable generalisation with respect to them: camera position, orientation, textures, materials, colours, object locations, background (see Appendix B).

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 [10] 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 that further exploits state information in simulation [20] to learn informative representations that are robust to visual randomisation. This approach does not incorporate object-centric attention or leverage 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 also compare APRiL with DDPG to emphasise the difficulty of these DR tasks if privilege information is not leveraged.

We perform an ablation study to investigate which components of APRiL contribute to performance gains. The ablations consist of: 1. *APRiL no sup*: the full setup except without attention alignment. Here the observation attention module must learn without guidance from the state-agent. 2. *APRiL no share*: APRiL without a shared replay. 3. *APRiL no back*: uniform object attention values c are used to train the observation attention module, thereby only suppressing the background. Here we investigate the importance of object-suppression for generalisation.

We investigate the following to evaluate how well APRiL facilitates transfer across visually distinct domains: 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?

4.1 Performance On The Training Distribution

Figure 3 shows that APRiL outperforms the baselines for each environment (except *Walker2D* where it matches s-map asym-DDPG). The ablations in Figure 3 show that a shared replay buffer, background suppression, and attention-alignment each individually provide benefits but are most effective when combined together. Interestingly, background suppression is extremely effective for sample-efficiency, as for these domains the majority of the irrelevant, highly varying, aspects of the observation-space are occupied by the background. It is also surprising that 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 asym-DDPG is a competitive baseline and APRiL provides marginal gains.

4.2 Interpolation: Transfer To Domains From The Training Distribution

We evaluate performance on environments unseen during training but within the training distribution (see Appendix B). For *NavWorld* and *JacoReach*, the interpolated environments have the same number of distractors, sampled from the same object catalogue, as the training distribution. Figure 4 plots the return on these held-out domains. For all algorithms, we observe minimal degradation in performance between training and interpolated domains. However, as APRiL outperforms on the training distribution (apart from; *Walker2D* for s-map asym-DDPG, *JacoReach* for APRiL no back), its final performance on the interpolated domains is significantly better, emphasising the benefits of both privileged attention and a shared replay.

4.3 Extrapolation: Transfer To Domains Outside The Training Distribution

For *NavWorld* and *JacoReach*, we investigate how well each method generalises to extrapolated domains with additional distractor objects (specifically 4 or 8; referred as **ext-4** and **ext-8**). The textures and colours of these objects are sampled from a held-old out set not seen during training. The locations are sampled; randomly for *NavWorld*, from extrapolated arcs of two concentric circles of different radii for *JacoReach*. Shapes are sampled from the training catalogue of distractors. We do not extrapolate for *Walker2D*, as this domain does not contain distractors. However, we show (in the previous sections) that APRiL is still beneficial during DR for this domain and therefore demonstrate its application does not need to be restricted to environments with clutter. Please refer to Figure 5 for examples of the extrapolated domains.

Figure 4 shows that APRiL generalises and performs considerably better on the held-out domains than each baseline. Specifically, when comparing with the baselines that leverage privilege information, for *JacoReach* performance falls by 11%[2] for APRiL instead of 42% and 48% for asym-DDPG and s-map asym-DDPG respectively. The ablations demonstrate that effective distractor suppression is crucial for generalisation. This is particularly prominent for *JacoReach* where the performance drop for the methods that use attention-alignment (APRiL and APRiL no share) is 11% and 15%, which is far less than 27% and 51% (APRiL no back and APRiL no sup) for those that do not learn to effectively suppress distractors.

4.4 Attention Module Analysis

We visualise APRiL's attention maps (Figure 5, 8, 9 (in Appendix E) 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) and generalises well. For *JacoReach*, attention suppresses the distractors even on the extrapolated domains, achieving robustness with respect to them. Interestingly, as we encourage sparse attention, APRiL learns to only pay attention to every-other-link of the arm (as the state of an unobserved link can be inferred by observing those of the adjacent links). For *Walker2D*, dynamic object attention is learnt (different objects are attended based on the state of the system - see Figure 9). When upright, walking, and collapsing,

[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.

[2] Percentage decrease is taken with respect to additional return over a random agent on the training domain.

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 of 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 from a different domain [21, 22, 23]. 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 [4, 24, 5, 6]. These methods commonly require data from the target domain in order to transfer and adapt effectively.

In contrast, **domain randomisation** covers a distribution of environments by randomising visual [7] or dynamical parameters [14] during training in order to generalise [11, 8, 12, 25, 9, 26]. 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 this distribution during training [27] or trains a canonical invariant image representation [28]. However, while randomisation can enable us to learn robust policies, it significantly increases training time due to the increased environment variability [9], and can reduce asymptotic performance. Our work partially addresses this fact by training two agents, one of which is not affected by visual randomisations.

Other works explicitly encourage representations **invariant** to observation space variations [29, 30, 28]. Contrastive techniques [29, 30] use a clear separation between positive (and negative) examples, predefined by the engineer, to encourage **invariance**. Unlike APRiL, these **invariances** are over abstract spaces and are not designed to exploit **privileged information**; shown to be beneficial by APRiL's ablations. Furthermore, APRiL's **invariance** is task-driven via attention. Approaches like [28, 20], learn invariances in a supervised manner, mapping from observation to a predefined space. Unlike APRiL, these methods are unable to discover task-independent aspects of the mapping-space, limiting robustness and generalisation. Finally, unlike APRiL as a model-free RL approach, some model-based works use forward or inverse models [31, 32, 33] to achieve invariance.

Existing comparisons in the literature demonstrate that, even without domain randomisation, the increased dimensionality and potential partial observability complicates learning for RL agents [13, 15]. In this context, accelerated training has also been achieved by using **access to privileged information** such as environment states to asymmetrically train the critic in actor-critic RL [15, 10, 34]. In addition to using additional information to train the critic, [15] 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 interaction between attention and learning in human subjects [35]. 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 [36, 37]. In comparison, 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

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

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

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

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

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

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

[7] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. 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*, pages 23–30. IEEE, 2017.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[26] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020.

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

[28] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis. 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*, pages 12627–12637, 2019.

[29] R. B. Slaoui, W. R. Clements, J. N. Foerster, and S. Toth. Robust domain randomization for reinforcement learning. *arXiv preprint arXiv:1910.10537*, 2019.

[30] A. Srinivas, M. Laskin, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020.

[31] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, volume 2017, 2017.

[32] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

[33] K. Schmeckpeper, A. Xie, O. Rybkin, S. Tian, K. Daniilidis, S. Levine, and C. Finn. Learning predictive models from observation and interaction. In *European Conference on Computer Vision*. Springer, 2020.

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

[35] Y. C. Leong, A. Radulescu, R. Daniel, V. DeWoskin, and Y. Niv. 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.

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

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

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

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

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 such that the distance between its hand and the diamond-shaped object is minimised. The state space consists of the quaternion position and velocity of each joint as well as the Cartesian positions of each 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 [13] 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 [13] 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 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. The DDPG baseline has the same policy architecture as the other baselines except now the critic is image-based and has the same structure as the actor. All layers use ReLU activations and layer normalisation 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	FC($[256]$)	FC($[256]$)
Obs Actor	Conv($[[18, 7, 1], [32, 5, 1], [32, 3, 1]]$)	Conv($[[18, 8, 2], [32, 5, 1], [16, 3, 1], [4, 3, 1]]$)
State Critic	FC($[64, 64]$)	FC($[400, 300]$)
Obs Critic	FC($[64, 64]$)	FC($[400, 300]$)
State Attention	FC($[256]$)	FC($[256]$)
Obs Attention	Conv($[[32, 8, 1], [32, 5, 1], [64, 3, 1]]$)	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 optimizer with learning rate of 10^{-3}, 10^{-4} and 10^{-4} for critic, actor and attention networks. We use default TensorFlow 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 [38] 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 roll-outs 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 6). Figure 7 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. [39] 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.

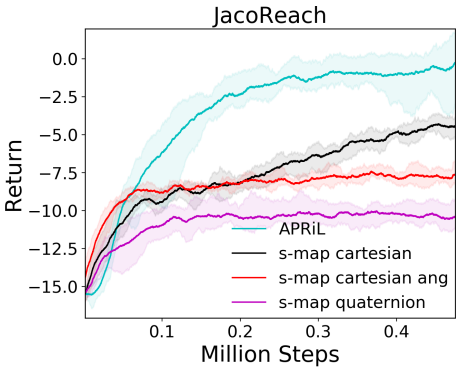

Figure 6: We compare learning of APRiL with variants of s-map asym-DDPG. For **s-map cartesian**, **s-map cartesian ang** and **s-map quaternion**, regressed states are cartesian position, cartesian position and rotation, and quaternions respectively (for Jaco arm - distractors are always cartesian).

However, even with a representation which is better suited for learning, the agent's performance is still significantly below APRiL (see Figure 6). 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.

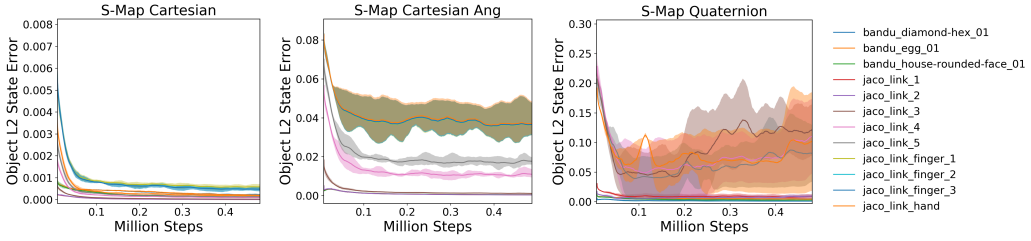

Figure 7: S-Map Asym-DDPG normalised state prediction errors. We compare individual object L_2 regressor losses (mean loss over states corresponding to a given object) between **s-map cartesian**, **s-map cartesian ang** and **s-map quaternion**. The object keys are on the right. **S-map quaternion** and **s-map cartesian ang** struggle to regress to quaternions and cartesian rotations and hence policy performance is restricted.

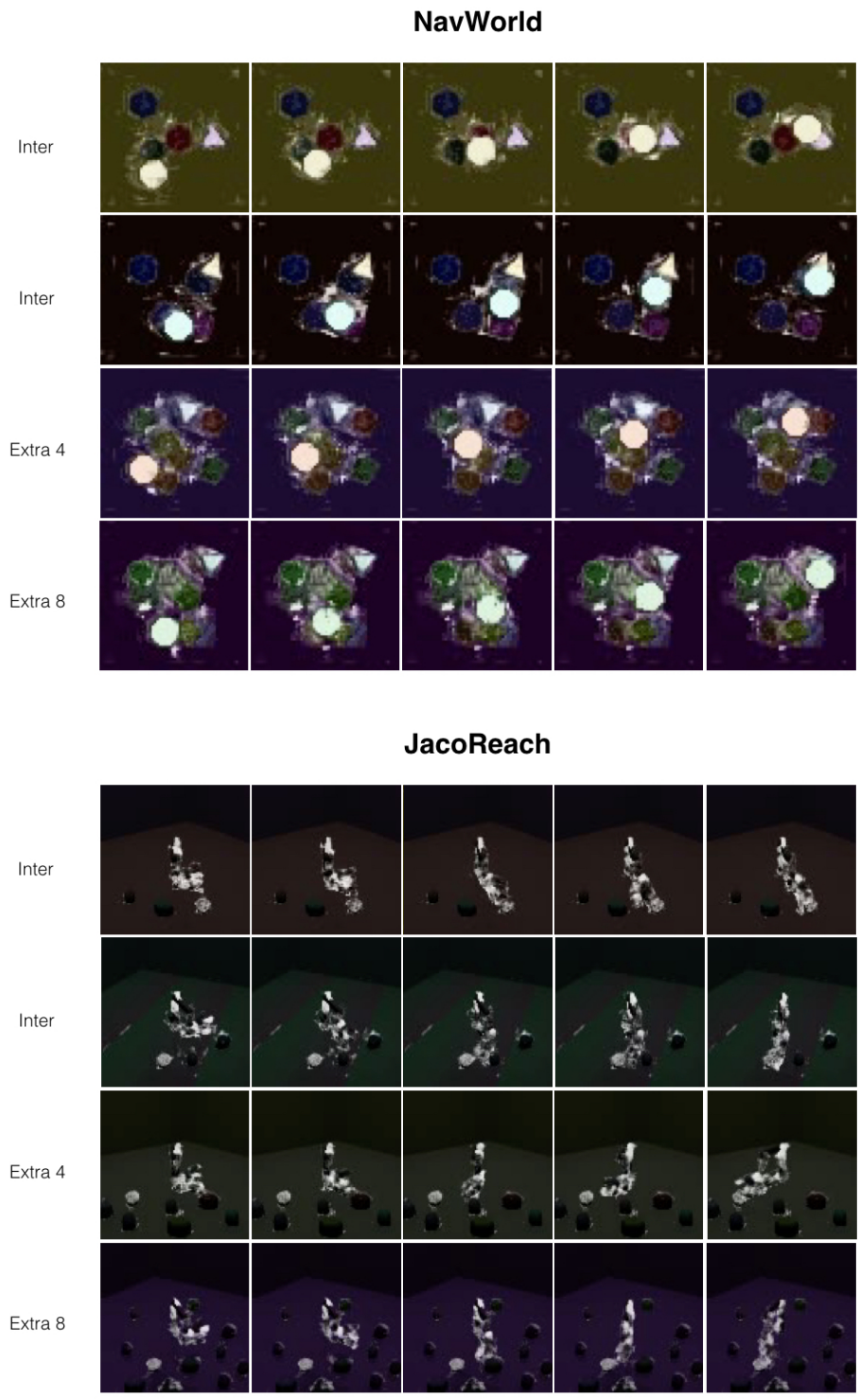

Figure 8: APRiL attention maps for policy rollouts on NavWorld and Jaco domains. White and black signify high and low attention values respectively. For NavWorld and JacoReach, attention is correctly paid only to the relevant objects (and Jaco links), even for the extrapolated domains. Refer to section 4.4 for more details.

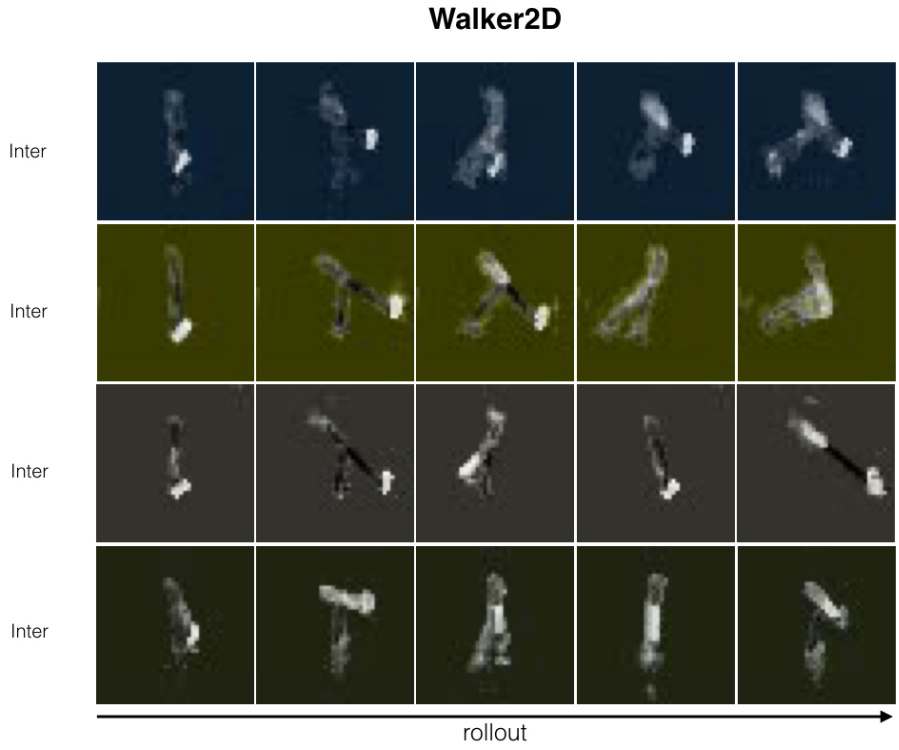

Figure 9: APRiL attention maps for policy rollouts on Walker domain. White and black signify high and low attention values respectively. Attention varies based on the state of the walker. When the walker is upright, high attention is paid to lower limbs. When walking, even attention is paid to every other limb. When about to collapse, high attention is paid to the foot and upper torso. Refer to section 4.4 for more details.