ATTENTION PRIVILEGED REINFORCEMENT LEARNING FOR DOMAIN TRANSFER

Anonymous authors
Paper under double-blind review

ABSTRACT

Applying reinforcement learning (RL) to physical systems presents notable challenges, given requirements regarding sample efficiency, safety, and physical constraints compared to simulated environments. To enable transfer of policies trained in simulation, randomising simulation parameters leads to more robust policies, but also significantly extends training time. In this paper, we exploit access to privileged information (such as environment states) often available in simulation, in order to improve and accelerate learning over randomised environments. We introduce Attention Privileged Reinforcement Learning (APRiL), which equips the agent with an attention mechanism and makes use of state information in simulation, learning to align attention between state- and image-based policies while additionally sharing generated data. During deployment we can apply the image-based policy to remove the requirement of access to additional information. We experimentally demonstrate accelerated and more robust learning on a number of diverse domains, leading to improved final performance for environments both within and outside the training distribution. [1].

1 INTRODUCTION

Deep Reinforcement Learning (RL) has recently provided significant successes in a range of areas, including video games (Mnih et al., 2015), board games (Silver et al., 2017), simulated continuous control tasks (Lillicrap et al., 2015), and robotic manipulation (Haarnoja et al., 2018; Haarnoja, 2018; Riedmiller et al., 2018; OpenAI et al., 2018; Schwab et al., 2019; Andrychowicz et al., 2017). However, application to physical systems has proven to be challenging in general, due to expensive and slow data generation as well as safety challenges when running untrained policies. A common approach to circumvent these issues is to transfer models trained in simulation to the real world (Tobin et al., 2017; Rusu et al., 2016; Held et al., 2017). However, simulators only represent approximations of a physical system. Due to physical, visual, and behavioural discrepancies, naively transferring RL agents trained in simulation onto the real world can be challenging.

To bridge the gap between simulation and the real world, we can either aim to align both domains (Ganin et al., 2016; Bousmalis et al., 2016; Wulfmeier et al., 2017) or ensure that the real system is covered by the distribution of simulated training data (OpenAI et al., 2018; Tobin et al., 2017; Pinto et al., 2017; Sadeghi & Levine, 2016; Viereck et al., 2017). However, training under a distribution of randomised visual attributes of the simulator, such as textures and lighting (Sadeghi & Levine, 2016; Viereck et al., 2017), as well as physics (OpenAI et al., 2018), can be substantially more difficult and slower due to the increased variability of the learning domain (OpenAI et al., 2018; Tobin et al., 2017).

The more structured and informative the input representation is with respect to the task, the quicker the agent can be trained. A clear example of this effect can be found when an agent is trained with image inputs, versus training with access to the exact simulator states (Tassa et al., 2018; Pinto et al., 2017). However, visual perception is more general and access to more compressed representations can often be limited. When exact states are available during training but not deployment, we can make use of information asymmetric actor-critic methods (Pinto et al., 2017; Schwab et al., 2019) to train the critic faster via access to the state while providing only images for the actor.

[1] Videos comparing the policy behaviours of APRiL to the baseline can be found here

By introducing Attention Privileged Reinforcement Learning (APRiL), we aim to further leverage access to exact states. APRiL leverages states not only to train the critic, but indirectly also for an image-based actor. Extending asymmetric actor-critic methods, APRiL concurrently trains two actor-critic systems (one symmetric, state-based agent, and another asymmetric agent with image-dependent actor). Both actors utilise an attention mechanism to filter input data and by having access to the simulation rendering system, we can optimise image and state based attention masks to align.

By additionally sharing the replay buffer between both agents, we can accelerate the learning process of the image-based actor by training on better performing states that are more quickly discovered by the state-based actor due to its lower dimensional input that is invariant to visual randomisation.

The key benefits of APRiL lie in its application to domain transfer. When training with domain randomisation for transfer, bootstrapping via asymmetric information has displayed crucial benefits (Pinto et al., 2017). Visual randomisation substantially increases the complexity of the image-based actor's task. Under this setting, the attention network can support invariance with respect to the irrelevant, but highly varying, parts of the image. Furthermore, the convergence of the state-space actor remains unaffected by visual randomisation.

We experimentally demonstrate considerable improvements regarding learning convergence and more robust transfer on a set of continuous action domains including: 2D navigation, 2D locomotion and 3D robotic manipulation, and extend APRiL to the discrete action setting where we show significant increase in learning rate for the Atari Suite.

2 Problem Setup

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. (2017).

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) (Pinto et al., 2017) represents a type of actor-critic algorithm designed specifically for efficient learning of a deterministic, observation-based policy in simulation for sim-to-real transfer. 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 (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 D.1 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 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 $\mathcal{L}_{actor} = -E_{s,o\sim\pi_b(o)}[Q(s, \pi(o))]$.

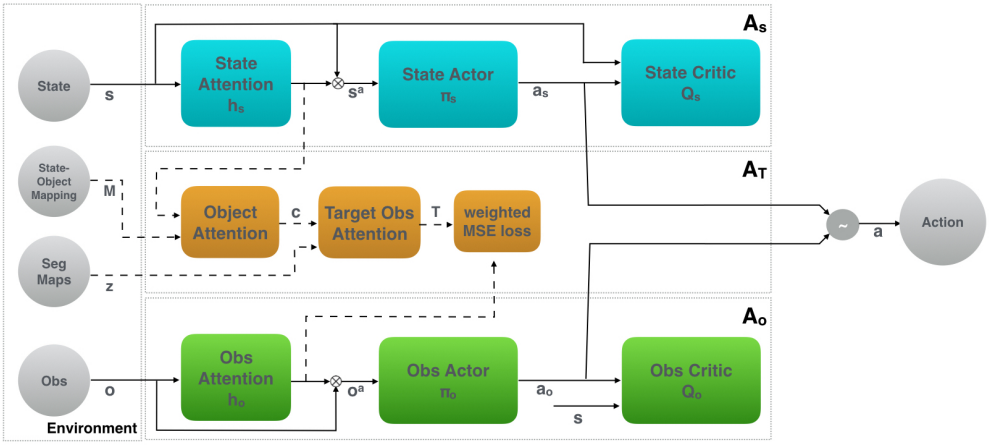

Figure 1: Attention Privileged Reinforcement Learning model structure. Dashed lines indicate attention alignment process. The $\sim$ operator signifies that experiences are evenly sampled from both agents. The $\otimes$ operator represents element-wise multiplication.

3 Attention Privileged Reinforcement Learning (APRiL)

APRiL proposes to improve the performance and sample efficiency of an observation-based agent by using a quicker learning actor that has access to exact environment states, sharing replay buffers, and aligning attention mechanisms between both actors. While we focus in the following sections on extending asymmetric DDPG (Pinto et al., 2017), these ideas are generally applicable to actor-critic methods (Konda & Tsitsiklis, 2000) and other off-policy reinforcement learning algorithms, as we shall see in the experimental analysis.

APRiL is comprised of three modules as displayed in Figure 1. The first two modules, A_s and A_o, are actor-critic algorithms with an attention network incorporated over the input to 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. 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 quicker and slower learners, A_s and A_o, respectively.

A_s consists of three networks: Q_s^π, 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, $s_t^a = h_s(s_t) \odot s_t$. To encourage a sparse masking function, we train this attention module on both the traditional DDPG loss as well as an entropy loss:

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

where β is a hyperparameter 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 respectively.

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-based agent in A_o. This is achieved in a two-step process. First, state-attention $h_s(s)$ is converted into the space of objects in the scene, c. Second, object-attention is converted to observation-space attention by performing a weighted sum over object-specific segmentation maps:

$$c = M \cdot h_s(s) \tag{2}$$

$$T = \sum_{i=0}^{N-1} c_i \cdot z_i \tag{3}$$

Here, $M \in \{0, 1\}^{N\times n_s}$ (where 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

then an attention vector over the N objects in the environment. $z_i \in \{0,1\}^{W \times H}$ is the segmentation mask of the ith object with same size as the image observation, and c_i corresponds to the ith object attention value.

The observation-based 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 input to the actor is the attention-filtered version of the observation, $o_t^a = h_o(o_t) \odot o_t$[2].The actor and critic networks π_o and Q_o are trained with the standard 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:

$$\mathcal{L}_{h_o} = E_{o,s\sim\pi_b}\left[\frac{1}{2}\sum_{ij}\frac{1}{w^{ij}}(h_o(o)-T)^{ij^2} - \nu Q_o(s,\pi_o(o^a))\right], \tag{4}$$

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 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. Combining the self-supervised attention loss and the RL loss leverages efficient state-space learning unaffected by visual randomisation.

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.

4 EXPERIMENTS

To demonstrate the performance and generality of our method, we apply APRiL to a range of environments, and compare with various competitive baselines and 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 D.1.

4.1 EVALUATION PROTOCOL

In order to investigate APRiL under varying conditions, we evaluate in scenarios of increasing complexity covering simple 2D navigation, 3D reaching and 2D dynamic locomotion.

We use the following continuous action-space environments (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 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.

For these domains we randomise visuals during training as 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.

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

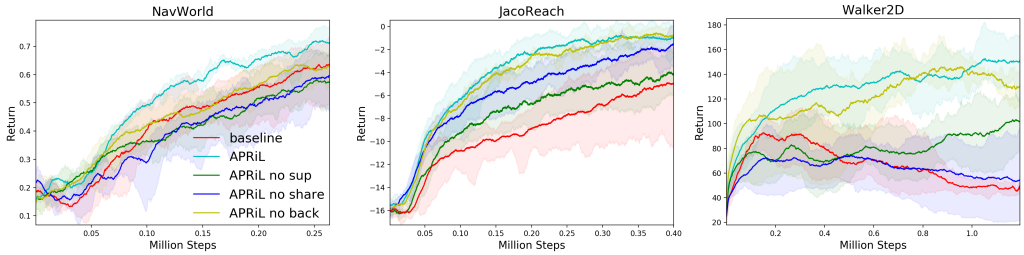

Figure 2: Learning curves during training. **Solid line**: mean performance. **Shaded region**: covers standard deviation range across 5 seeds.

We also test our approach on a rich environment with discrete action space: the *Atari Suite*. Previous work (Zhang et al., 2018) has shown that even such tasks can be highly sensitive to domain randomisation, failing to learn when background pixels are replaced with natural images. Inspired by this work, we evaluate our approach on a number of Atari games, using the canonical image as the state, and a randomised image as the observation. This allows us to evaluate the benefits of APRiL in learning policies that are robust to environments with randomised distractors, even when no clear lower-dimensional state space is available. In this domain, the state-based agent observes the canonical Atari image, while the observation-based agent operates with a "domain randomized" environment, where the background is replaced by video frame from the Kinetics dataset (Kay et al., 2017). Instead of a dual actor-critic setup, the agents are each a Double Deep Q-Network (DDQN) with dueling architecture (Wang et al., 2016). Given that the "privileged state information" is in this case a canonical image with the same dimensionality as the observation, the self-supervised loss can directly be applied between attention maps without object segmentation.

4.2 Key Research Questions

We investigate the following questions to evaluate how well APRiL accommodates for the transferring of policies 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 qualitatively analyse the 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 except without the self-supervision provided by the state agent to train the observation-based attention. 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 except each agent has access to its own replay buffer and does not share experiences. Under this setting, the observation agent will not be able to benefit from earlier visitation of lucrative states by the state agent.
3. *APRiL no background*: 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.

4.3 Performance on the Training Distribution

4.3.1 Continuous Control

We evaluate the performance on all domains during training and observe APRiL 's benefits. As seen in Figure 2, APRiL provides performance gains across all continuous action domains. APRiL not only helps learn useful representations quicker (improving learning rate) but also improves final policy performance (within the allotted training time).

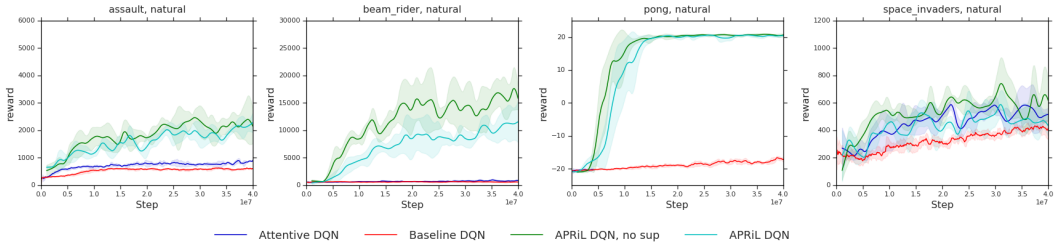

Figure 3: Performance on various Atari games (each column) with randomized natural image background, evaluated on environments sampled from the training distribution. **Solid line**: mean performance. **Shaded region**: covers minimum and maximum performances across 5 seeds.

The ablations demonstrate that self-supervision and shared replay both independently provide performance gains for *JacoReach* and *Walker2D*[3]. For *Walker2D*, shared replay is crucial as 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 speed up learning as 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 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.

4.3.2 Atari environments

Inspired by Zhang et al. (2018), we also evaluate APRiL in the Atari domains described in Section 4.1, measuring its ability to exploit attentive inductive biases and self-supervision using privileged state information (here in the form of canonical images) to encourage robustness to domain randomised backgrounds. As shown in Figure 3, we evaluate over a range of Atari levels (each column) for the different ablation cases outlined in Section 4.2 with an additional case incorporating just the attention mechanism into the baseline DQN. In each case, we evaluate on environments with randomisation drawn from the same distribution of natural images.

As reported by Zhang et al. (2018), we find that the baseline DQN struggles to solve each task in the presence of natural image randomisation. While Space Invaders benefits from attentive inductive biases alone, it becomes critical in the other cases to share replay between state and observation agents, with both APRiL and APRiL no sup outperforming the other baselines. Despite the fact that the privileged information ("state") in this setting is of the same dimensionality as the observation, APRiL is still able to exploit it via attention and shared experience. Crucially, in the continuous control domains evaluated, the low-dimensional state space yields an additional improvement in sample efficiency by guiding self-supervised attention.

4.4 Interpolation: transfer to domains from the training distribution

We evaluate the performance of all actor-critic algorithms on a hold out set of simulation parameters, unseen during training, from the training distribution. Table 1 displays final policy performance on these domains. For APRiL , we observe no degradation in policy performance between training and interpolated domains. We see a very similar trend for the baseline. However, as APRiL performs better on the training distribution, its final performance on the interpolated domains is significantly better. We therefore demonstrate that on these domains APRiL's attention mechanism does not hurt with respect to overfitting.

[3]We suspect that the simplicity of *NavWorld* means that APRiLdoesn't benefit from a slower learning observation attention module without self-supervision, no shared replay due to the sparsity of the domain, or simplistic background suppression.

4.5 EXTRAPOLATION: TRANSFER TO DOMAINS OUTSIDE THE TRAINING DISTRIBUTION

We investigate performances on simulation parameters outside the training distribution. Table 1 compares performances on extrapolated sets (except *Walker2D*) varying in difficulty. The extrapolated domains correspond to environments with additional distractor objects not seen during training. APRiL yields performance gains over the baseline on every extrapolated domain. For *JacoReach*, APRiL's generalisation is so effective that, for the hardest domain, its performance degrades by only 9%[4] opposed to 41% (baseline).

APRiL generalises favorably due to the attention module. Figure 4 shows that attention generalises and suppresses the additional distractors, thereby effectively converting the hold-out observations to those seen during training, which the image-policy can handle. The ablations in Table 1 confirms that in this setting, distractor suppression is crucial. This is seen when comparing the maximum degradation in policy performance of APRiL, APRiL no share, APRiL no back and APRiL no sup (9%, 16%, 27% and 47% respectively). 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.

Table 1: Ablation study comparing average return over various hold out sets (of 100 environments). Results reflect mean and standard deviation of average return over 5 seeds.

Domain	Baseline	APRiL no sup	APRiL no share	APRiL no back	APRiL
NavWorld (train)	0.72 ± 0.02	0.60 ± 0.02	0.61 ± 0.02	0.67 ± 0.02	$\mathbf{0.75 \pm 0.02}$
NavWorld (inter)	0.70 ± 0.03	0.60 ± 0.02	0.62 ± 0.02	0.69 ± 0.02	$\mathbf{0.80 \pm 0.02}$
NavWorld (ext-4)	0.45 ± 0.02	0.42 ± 0.02	0.43 ± 0.02	0.45 ± 0.02	$\mathbf{0.53 \pm 0.02}$
NavWorld (ext-8)	0.46 ± 0.02	0.42 ± 0.02	0.38 ± 0.02	0.47 ± 0.02	$\mathbf{0.50 \pm 0.02}$
JacoReach (train)	-2.38 ± 0.23	-3.03 ± 0.30	-0.10 ± 0.19	$\mathbf{1.10 \pm 0.16}$	0.03 ± 0.19
JacoReach (inter)	-2.19 ± 0.24	-3.38 ± 0.31	-0.29 ± 0.20	$\mathbf{0.81 \pm 0.18}$	0.13 ± 0.20
JacoReach (ext-4)	-6.07 ± 0.25	-6.81 ± 0.33	-1.26 ± 0.22	-0.79 ± 0.23	$\mathbf{-0.70 \pm 0.23}$
JacoReach (ext-8)	-8.02 ± 0.24	-9.26 ± 0.28	-2.69 ± 0.24	-3.53 ± 0.26	$\mathbf{-1.47 \pm 0.25}$
Walker2D (train)	67.4 ± 3.15	107 ± 3.31	63.0 ± 2.13	126 ± 4.33	$\mathbf{155 \pm 3.25}$
Walker2D (inter)	68.9 ± 3.25	107 ± 3.30	62.6 ± 2.11	126 ± 4.31	$\mathbf{156 \pm 3.20}$

4.6 ATTENTION MODULE ANALYSIS

To better comprehend the role of the attention, we visualise APRiL's attention maps (Figure 4 and 7) 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 of the Kinova arm. Interestingly, APRiL learnt that as the arm is a constrained system, the state of every other link can be indirectly inferred without explicit attention. The entropy loss over the state-attention module encourages this form of attention over minimal set of objects. Attention here generalise very well to the extrapolated domains. 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. 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 magnitude of the optimal action depends on the state of and as is largest for the lower links (due to stability), every link (coordination), and foot and upper body (large torque required), respectively.

[4]Percentage decrease is taken with respect to initial and final policy performance on training distribution

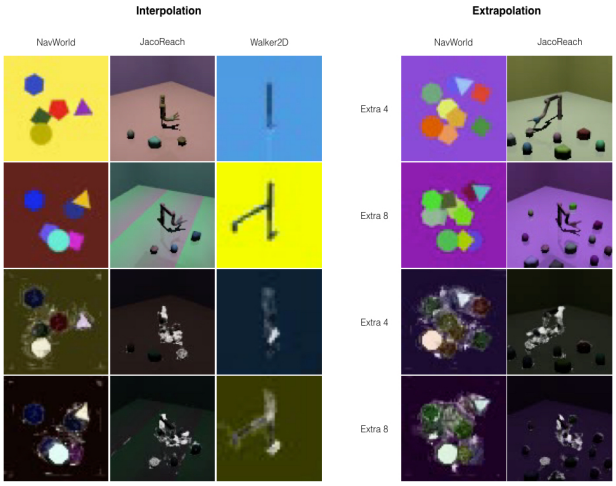

Figure 4: Examples of held-out domains (top) and corresponding APRiL attention maps (bottom).

5 Related Work

Domain Randomisation has been applied for reinforcement learning to facilitate transfer between domains (Tobin et al., 2017; Pinto et al., 2017; Sadeghi & Levine, 2016; Viereck et al., 2017; OpenAI et al., 2018; Held et al., 2017) and increase robustness of the learned policies (Rajeswaran et al., 2016). However, while domain randomisation enables us to generate more robust and transferable policies, it leads to a significant increase in required training time (OpenAI et al., 2018).

Existing comparisons in the literature demonstrate that, even without domain randomisation, the increased dimensionality and potential partial observability introduced by pixel-based environments leads to slower convergence and 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., 2017). 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.

6 Conclusion

We introduce Attention Privileged Reinforcement Learning (APRiL), an extension to asymmetric actor-critic algorithms that leverages access to privileged information like exact simulator states. The method benefits in two ways, via sharing a replay buffer as well as aligning attention masks between image- and state-space agents. By leveraging simulator ground-truth information about system states, we are able to learn efficiently in the image domain especially during domain randomisation where feature learning becomes increasingly difficult. Our evaluation on a diverse set of environments demonstrates significant improvements over the competitive baseline and reveals that APRiL learns to generalise favourably to environments not seen during training (both within and outside of the training distribution) in comparison to various strong baselines; emphasising the importance of attention and shared experience for robustness of the learnt policies.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pp. 265–283, 2016.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay, 2017.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems*, pp. 343–351, 2016.

Alexandre Campeau-Lecours, Hugo Lamontagne, Simon Latour, Philippe Fauteux, Véronique Maheu, François Boucher, Charles Deguire, and Louis-Joseph Caron L'Ecuyer. 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.

Jinyoung Choi, Beom-Jin Lee, and Byoung-Tak Zhang. Multi-focus attention network for efficient deep reinforcement learning. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

Oliver Groth, Fabian B. Fuchs, Ingmar Posner, and Andrea Vedaldi. 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.

T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine. Learning to Walk via Deep Reinforcement Learning. *arXiv e-prints*, December 2018.

Tuomas Haarnoja. *Acquiring Diverse Robot Skills via Maximum Entropy Deep Reinforcement Learning*. PhD thesis, UC Berkeley, 2018.

David Held, Zoe McCarthy, Michael Zhang, Fred Shentu, and Pieter Abbeel. Probabilistically safe policy transfer. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 5798–5805. IEEE, 2017.

Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

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

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

Yuan Chang Leong, Angela Radulescu, Reka Daniel, Vivian DeWoskin, and Yael 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.

Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Franois Goudou, and David Filliat. State representation learning for control: An overview. *Neural Networks*, 2018.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2015.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Alex Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, and Danilo J. Rezende. Towards interpretable reinforcement learning using attention augmented agents. *ArXiv*, abs/1906.02500, 2019.

OpenAI, :, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation, 2018.

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Volodymyr Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing - solving sparse reward tasks from scratch, 2018.

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

Andrei A Rusu, Mel Vecerik, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. *arXiv preprint arXiv:1610.04286*, 2016.

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

Devin Schwab, Tobias Springenberg, Murilo F Martins, Thomas Lampe, Michael Neunert, Abbas Abdolmaleki, Tim Herkweck, Roland Hafner, Francesco Nori, and Martin Riedmiller. Simultaneously learning vision and feature-based control policies for real-world ball-in-a-cup. *arXiv preprint arXiv:1902.04706*, 2019.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Ivan Sorokin, Alexey Seleznev, Mikhail Pavlov, Aleksandr Fedorov, and Anastasiia Ignateva. Deep attention recurrent q-network. *arXiv preprint arXiv:1512.01693*, 2015.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter 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*, pp. 23–30. IEEE, 2017.

Ulrich Viereck, Andreas ten Pas, Kate Saenko, and Robert Platt. Learning a visuomotor controller for real world robotic grasping using simulated depth images. *arXiv preprint arXiv:1706.04652*, 2017.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1995–2003, 2016.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. 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.

Markus Wulfmeier, Ingmar Posner, and Pieter Abbeel. Mutual alignment transfer learning. *arXiv preprint arXiv:1707.07907*, 2017.

Amy Zhang, Yuxin Wu, and Joelle Pineau. Natural environment benchmarks for reinforcement learning. *arXiv preprint arXiv:1811.06032*, 2018.

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.

4. *Atari*: In this domain, we utilise randomization as suggested by Zhang et al. (2018). In addition to the canonical image case ("none"), we also perform analysis with two other scenarios: "noise", where uniform noise is added to background pixels; and "natural", where background pixels are replaced by natural images from the Kinetics dataset (Kay et al., 2017). In the latter two cases, background pixels are defined as those taking on the most unique common pixel value (i.e. the mode) in the image.

C ADDITIONAL ANALYSIS

In addition to the "natural" case in the experimental section, we also evaluate APRiL on the aforementioned "noise" and "none" cases. As in Figure 5, we find that both the baseline and the attentive baseline are able to reach strong performance in the absence of any noise. In contrast the baseline DQN struggles to solve each task in the presence of randomisation. In some of the "noise" cases, incorporating attention is sufficient to alleviate this and significantly improve learning. Given this is a stationary source of noise, it is less challenging than the more sophisticated natural environments benchmark case, and easier to suppress distracting noise.

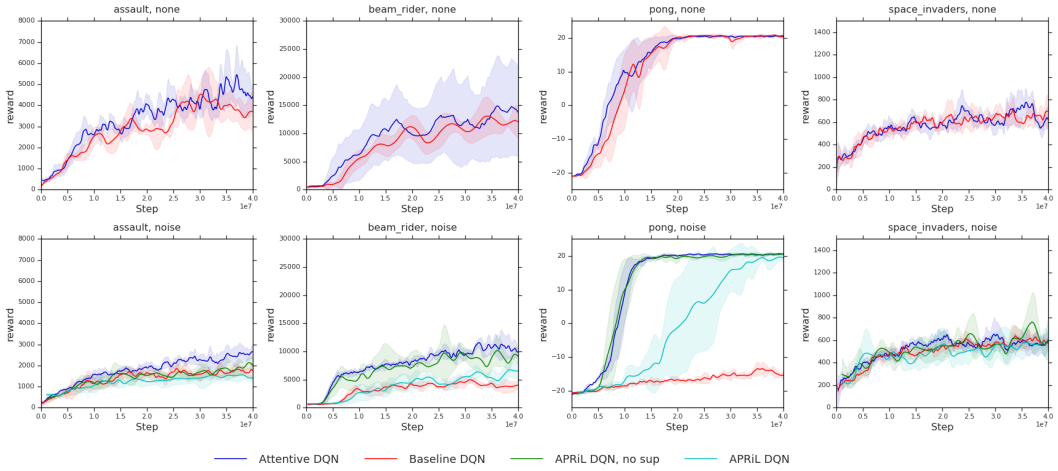

Figure 5: Performance on various Atari games (each column) with different forms of randomization (each row), evaluated on environments sampled from the training distribution. **Solid line**: mean performance. **Shaded region**: covers minimum and maximum performances across 5 seeds.

D IMPLEMENTATION DETAILS

D.1 CONTROL DOMAINS

Table 2: Model architecture. FC() represents a (multi-layered) fully connected network with the number of nodes per layer stated as argument. Conv() represents a (multi-layered) convolutional network whose arguments take the form [channels, square kernel size, stride] for each hidden layer.

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 Buffer Size	10^4	2×10^5

In this section we provide more details on our training setup. Refer to table 2 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 baseline. *Obs Actor* model structure comprises of the convolutional layers (without padding) defined in table 2 followed by one fully connected layer with 256 hidden units (FC($[256]$)). 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.

The *State Attention* module includes the fully connected layer defined in table 2 followed by a Softmax operation. The *Obs Attention* module has the convolutional layers (with padding to ensure constant dimensionality) outlined in table 2 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 shared replay buffer. The target actor and critic networks are updated every iteration with a Polyak averaging of 0.999. We use Adam (Kingma & Ba, 2014) optimization with a learning rate of 10^{-3}, 10^{-4} and 10^{-4} for critic, actor and attention networks respectively. 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 $\nu = 1$ respectively. 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.2 ATARI ENVIRONMENTS

In the Atari environments, each agent is a Dueling Double DQN (Wang et al., 2016), implemented as a single learner and 64 decentralised actors gathering experience for the shared replay buffer. The learner is implemented on a single Tesla V-100 GPU, while the actors each use a single CPU.

During each iteration of APRiL (for both A_o and A_s) we perform 1 optimization step on minibatches of size 512 from the shared replay buffer. The target actor and critic networks are updated every iteration with a Polyak averaging of 0.999. We use Adam (Kingma & Ba, 2014) optimization with a learning rate of 10^{-4} for the actor and attention networks. We use default TensorFlow (Abadi et al., 2016) values for the other hyperparameters. The discount factor, ϵ value for the ϵ-greedy policy, and self-supervised learning hyperparameters are $\gamma = 0.99$, $\epsilon = 0.01$, and $\nu = 1$ respectively.

E PSEUDOCODE APRIL ALGORITHM

Algorithm 1 shows 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 data collection from state- and image- based agents is even.

Algorithm 1 Attention Privileged Reinforcement Learning

```
Initialize actor-critic algorithm A_s
Initialize asymmetric actor-critic algorithm A_o
Initialize attention alignment algorithm A_T
Initialize replay buffer R
for episode= 1 to M do
  Initial state s_0
  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 observation-behavioral policy and attention network:
        o_t^a ← h_o(o_t) ⊙ o_t
        a_t ← π_o(o_t^a)
    else
      Obtain action a_t using state-behavioral policy:
        s_t^a ← h_s(s_t) ⊙ o_t
        a_t ← π_s(s_t^a)
    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 and state-actor using {s, a, r, s'}_0^B with A_s
    Optimise attention network using {s, o, z}_0^B with A_T
    Optimise observation-critic and observation-actor using {s, o, a, r, s', o'}_0^B with A_o
  end for
end for
```

F ATTENTION VISUALISATION

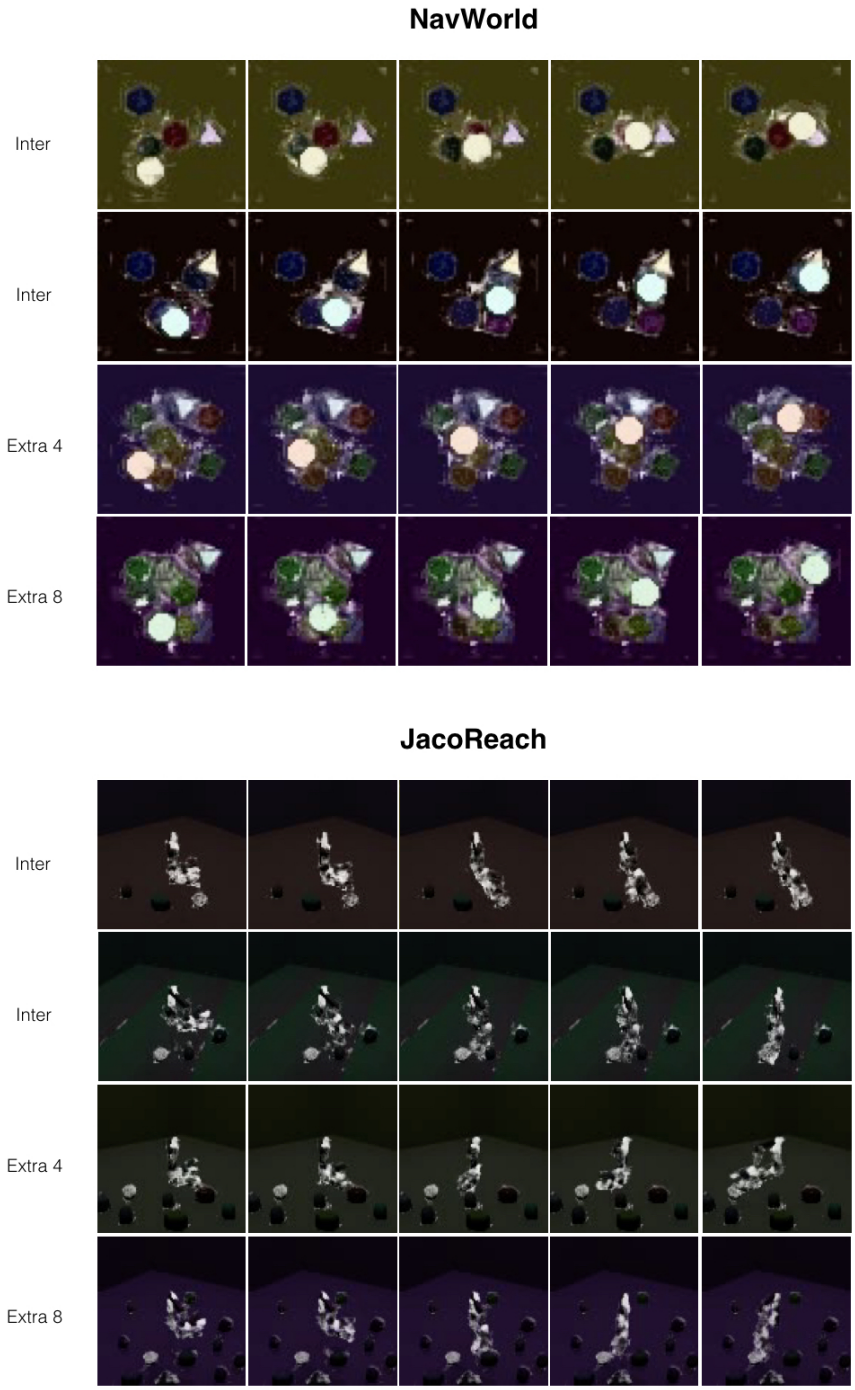

Figure 6: APRiL attention maps for policy rollouts on NavWorld and Jaco domains. White and black signify high and low attention values respectively. For NavWorld and JacoReach, attention is correctly paid only to the relevant objects (and Jaco links), even for the extrapolated domains. Refer to section 4.6 for more details.

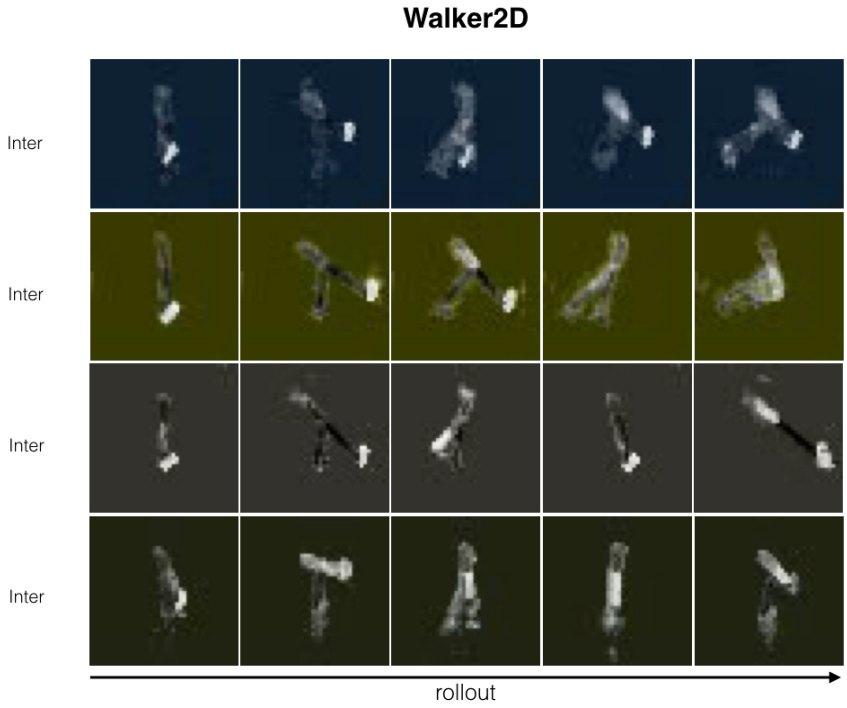

Figure 7: APRiL attention maps for policy rollouts on Walker domain. White and black signify high and low attention values respectively. Attention varies based on the state of the walker. When the walker is upright, high attention is paid to lower limbs. When walking, even attention is paid to every other limb. When about to collapse, high attention is paid to the foot and upper torso. Refer to section 4.6 for more details.