Neurocomputing 493 (2022) 143–165

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Analysing deep reinforcement learning agents trained with domain randomisation

Tianhong Dai [a,*,1], Kai Arulkumaran [a,b,*,1], Tamara Gerbert [a], Samyakh Tukra [a], Feryal Behbahani [a], Anil Anthony Bharath [a]

[a] BICI-Lab, Department of Bioengineering, Imperial College London, Exhibition Road, London SW7 2AZ, United Kingdom
[b] ARAYA Inc., 1-12-32 Akasaka, Minato-ku, Tokyo 107-6024, Japan

ARTICLE INFO

Article history:
Received 13 July 2021
Revised 22 December 2021
Accepted 3 April 2022
Available online 7 April 2022
Communicated by Zidong Wang

Keywords:
Deep reinforcement learning
Generalisation
Interpretability
Saliency

ABSTRACT

Deep reinforcement learning (DRL) has the potential to train robots to perform complex tasks in the real world without requiring accurate models of the robot or its environment. However, agents trained with these algorithms typically lack the explainability of more traditional control methods. In this work, we use a combination of out-of-distribution generalisation tests and post hoc interpretability methods in order to understand what strategies DRL-trained agents use to perform a reaching task. To do so, we train agents under different conditions, using comparison to better interpret both quantitative and qualitative results; this allows us to not only provide local explanations, but also broad categorisations of behaviour. A key aim of our work is to understand how agents trained with visual domain randomisation (DR)—a technique which allows agents to generalise from simulation-based-training to the real world—differ from agents trained without. Our results show that the primary outcome of DR is more robust, entangled representations, accompanied by greater spatial structure in convolutional filters. Furthermore, even with an improved saliency method introduced in this work, we show that qualitative studies may not always correspond with quantitative measures, necessitating the combination of inspection tools in order to provide sufficient insights into the behaviour of trained agents. We conclude with recommendations for applying interpretability methods to DRL agents.

1. Introduction

Deep reinforcement learning (DRL) is currently one of the most prominent subfields in AI, with applications to many domains [4]. One of the most enticing possibilities that DRL affords is the ability to train robots to perform complex tasks in the real world, all from raw sensory inputs. For instance, while robotics has traditionally relied on hand-crafted pipelines, each performing well-defined estimation tasks—such as ground-plane estimation, object detection, segmentation and classification, [40,50]—it is now possible to learn visual perception and control in an "end-to-end" fashion [24,45,104], without explicit specification and training of networks for specific sub-tasks.

A major advantage of using reinforcement learning (RL) versus the more traditional approach to robotic system design based on optimal control is that the latter requires a transition model for the task in order to solve for the optimal sequence of actions. While optimal control, when applicable, is more efficient, modelling certain classes of objects (e.g., deformable objects) can require expensive simulation steps, and often physical parameters (e.g., frictional coefficients) of real objects that are not known in detail. Instead, approaches that use RL can learn a direct mapping from observations to the optimal sequence of actions, purely through interacting with the environment. Through the powerful function approximation capabilities of neural networks (NNs), deep learning (DL) has allowed RL algorithms to scale to domains with significantly more complex input and action spaces than previously considered tractable. Such domains, which could involve inferring actions based on image inputs, make theoretical analysis of both the environment and agents that act within it practically infeasible.

In particular, NNs trained with DRL form a "black box" mapping from observations directly to actions, leaving us with the challenge of understanding the learned control policies. If we would like to deploy such policies—particularly on robots in the real world—we would also want to be able to *explain* them [3]. An interpretation

* Corresponding authors at: BICI-Lab, Department of Bioengineering, Imperial College London, Exhibition Road, London SW7 2AZ, United Kingdom.
E-mail addresses: tianhong.dai15@imperial.ac.uk (T. Dai), kai_arulkumaran@araya.org (K. Arulkumaran).
[1] Equal contributions.

https://doi.org/10.1016/j.neucom.2022.04.005
0925-2312/

of a model's "reasoning" can not only be used as a way to provide an explanation of the model's behaviour, but can also be used to characterise other properties, such as safety, fairness and reliability [13]. While there are now many methods available to interpret NNs [25], these methods are subjective to varying degrees. As such, we train DRL agents under a range of different settings, and use relative differences to better interpret their policies. In contrast to most prior work examining DRL policies [5,23,61,69,73,86,98], we not only provide explanations for how policies react to parts of states/individual states, but use our array of experiments (Section 3) to characterise their *global* strategies (Fig. 1). For instance, we examine the relative importance of different input modalities, and whether agents need their memory to complete tasks (Section 4).

Specifically, we train different robots with vision-based inputs to perform the task of reaching for a (red) target, and then use different techniques (Subsection 2.2) to understand what features of the environment the policies react to. Under certain training settings, the agent may localise the target by simply looking for anything red, and it is only under more challenging training conditions that the agent utilises both colour and shape. Clearly, the latter strategy better represents what we want the agent to learn, while the former is merely a "shortcut" [18].

1.1. Training Conditions

We vary training conditions over 3 different axes: 2 different robots with different morphologies and control schemes, using vision-only vs. vision + proprioception, and training with/without visual *domain randomisation* (DR). This results in 8 different configurations (Fig. 2), which we found sufficient to uncover a large compositional space of strategies (Fig. 1).

We chose these 3 axes for the purpose of studying *representation learning* in robotics tasks. The difference in the design of the robots, as well as a 2D vs. 3D target space (Subsection 3.1), results in one task being more difficult—in terms of both inferring the position of the target, as well as in the control. While task complexity is not a transitive property, the representations learned do reflect the difference between these tasks: for instance, localisation of the target in the latter case requires more complex image processing (Subsection 4.2).

For real-world deployment, one would want to use as many sensors as it is feasible to use. However, we are interested in how the absence of certain input modalities can affect learning, as this changes how the agent is able to extract information from the environment. While one might expect robots equipped with proprioceptive sensors to use these alone for pose estimation, agents can also learn to utilise vision if extracting pose from images is simple (Subsection 4.1).

Finally, analogously to how *data augmentation* [43,82] can be used to improve the generalisation of models in supervised learning settings, *domain randomisation* (Fig. 3) is a common training paradigm in "sim2real" approaches for learning real-world robotics control [1,34,66,75,89]. In DR, various properties of the simulation are varied, altering anything from the positions or dynamical properties of objects to their visual appearance, resulting in an expansion of the training domain. We therefore test how the agents trained with or without DR differ. Supporting previous results, we show that agents trained with DR are more robust to out-of-distribution (OoD) perturbations (Subsection 3.4). In our experiments, we use a standard visual DR setup, which is described in Tables 1,2 and visualised in Fig. 4.

DRL agents trained in standard simulators (due to the sample inefficiency of DRL algorithms) do not directly transfer to the real world. However, agents trained in simulators with DR can, which makes them of particular interest to study. It is common knowledge that while training in simulation is simpler, differences between the simulated and real worlds introduces a *reality gap* [33]. Of the several ways to address this gap—including fine-tuning a DRL agent on the real world [74], performing system identification to reduce the domain gap [10], or explicitly performing domain adaptation [91]—DR is unique in that it is essentially a form of data augmentation that only affects the input data, without changing the initial model or training objective. Thus, by comparing agents trained with DR against agents trained without, we expect to uncover what *strategies* they learn that would allow generalisation in the real world (as opposed to, adaptation to the real world). For example, some of the robustness of our DR agents comes through learning to perform some form of state estimation implicitly (Subsection 4.6).

1.2. Generalisation

The study of how DRL agents generalise has received considerable interest in the DRL community recently [11,36,62,96,100,102]. As RL agents are typically trained and evaluated on the same environment, generalisation requires changes to the traditional paradigm. In particular, works in this area have also focused on procedural content generation and OoD tests to evaluate generalisation. In this paper, we not only quantitatively evaluate generalisation, but also use a wide suite of interpretability methods to try and understand why our trained agents act the way they do. Furthermore, we are the first to perform an in-depth study focused on DR. One of our findings is that, depending on the training conditions, we can observe a failure of agents trained with DR to generalise to the much simpler default visuals of the simulator (Subsection 3.3), highlighting that conditions other than DR can have a significant effect on generalisation.

1.3. Interpretability

While our OoD tests (Subsection 3.4) provide a quantitative measure by which we can probe the performance of trained agents under various conditions, they treat the trained agents as black boxes. However, with full access to the internals of the trained models and even control over the training process, we can delve even further into the models. Using common interpretability tools such as saliency maps [56,80,87,99] and dimensionality reduction methods [47,51,65] for visualising NN activations [70], we can obtain information on why agents act the way they do. The results of these methods work in tandem with our OoD tests, as matching performance to the qualitative results allows us to have greater confidence in interpreting the latter; in fact, this process allowed us to debug and improve upon an existing saliency map method, as detailed in Subsection 2.2.1. Similarly to prior work[2] on interpretability in DRL agents [5,23,35,52,61,69,73,86,98] we use common methods: saliency maps (Subsection 2.2.1), activation maximisation (Subsection 2.2.2), unit ablations (Subsection 2.2.5) and dimensionality reduction (Subsection 2.2.8). In addition, we use layer re-initialisation (Subsection 2.2.6), recurrent ablations (Subsection 2.2.7), entanglement (Subsection 2.2.8), and a novel quantitative measure for spatial structure in convolutional filters (Subsection 2.2.4).

1.4. Summary of Findings

Under our set of experimental conditions, we show that our agents trained with DR:

[2] Based on interpreting standard DRL agents, as opposed to constructing more interpretable agents.

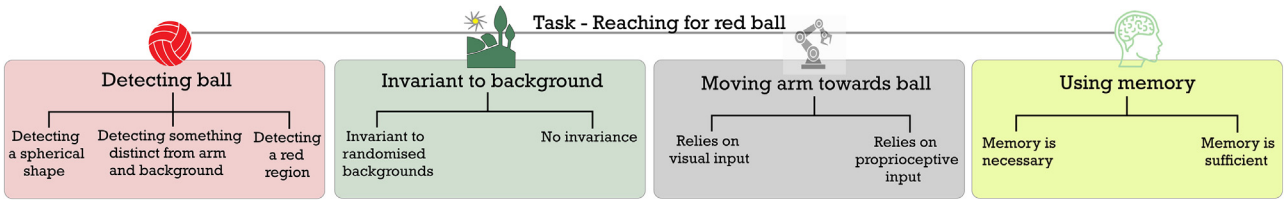


Fig. 1. The range of strategies an agent may learn to use when trained to reach a red target, which we split into three components. Firstly, the agent must visually localise the target—which can be accomplished by detecting red, spherical objects, or simply anything with a significant red component. Secondly, the agent can use vision and/or proprioception to guide its arm to the target. Finally, the agent may accomplish the task with varying levels of robustness to changes in the rest of the visual scene. Using a range of analyses, we show that agents trained to accomplish the same task learn different subsets of these strategies, depending on their training conditions.

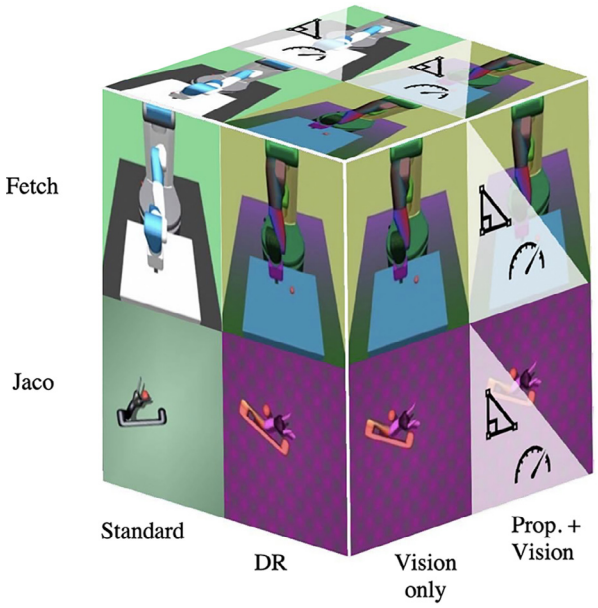


Fig. 2. The 8 different experimental conditions: 2 robots (Fetch vs. Jaco), 2 input modalities (vision vs. vision + proprioception), 2 visual conditions (DR vs. no DR).

- require more representational learning capacity (Subsection 4.5),
- are more robust to visual changes in the scene, exhibiting generalisation to unseen local/global perturbations (Subsection 3.4),
- use a smaller set of more reliable visual cues when not provided proprioceptive inputs (Subsection 4.1),
- more heavily rely on recurrent processing (Subsection 4.6),
- have filters that have higher norms or greater spatial structure (Subsection 4.3), which respond to more complex spatial patterns (Subsection 4.2),
- learn more robust (Subsection 4.4) and *entangled* [17] representations (Subsection 4.7),
- and can "overfit" to DR visuals (Subsection 3.3).

Furthermore, we characterise what strategies (Fig. 1) agents learn as a result of being trained under different conditions. In our discussion (Section 5), we revisit what this entails for future work on sim2real methods, and end with a set of recommendations for those interesting in applying interpretability methods to DRL agents.

1.5. Overview

The rest of the paper is organised as the follows: In Section 2, we introduce RL, and the neural network interpretability tech-

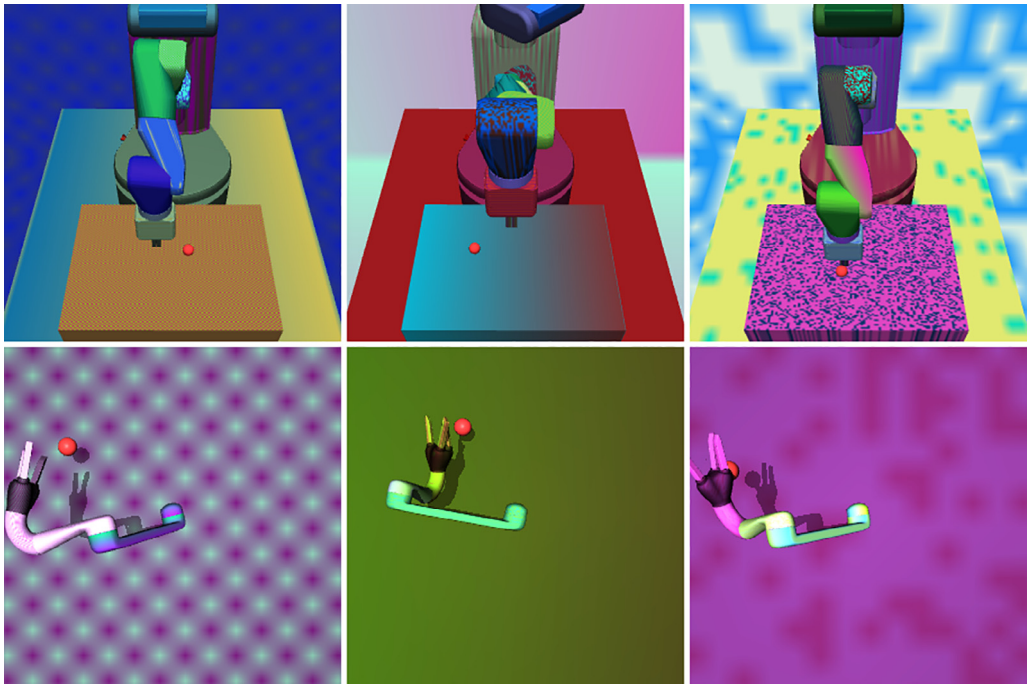


Fig. 3. Examples of visual domain randomisation in our Fetch (top) and Jaco (bottom) robotics experiments.

Table 1
DR textures used during training. At every environment timestep, for each component (e.g., robotic arm, skybox, table; the Jaco robot has 13 components, and the Fetch robot has 19 components), the appearance of the component is rendered using a sequential sampling process. In the first step of the sampling process, one of each of the 4 possible choices is made with the same probability, then the subsequent appearance of each component is determined by a second draw, described in the rightmost column of the table. Each RGB channel is drawn uniformly from $\{0 \dots 255\}$. The details of each option are described in Table 2.

No.	Option	Description
1	checkerboard	Render checkerboard pattern with two random RGB colours
2	gradient	Render gradient in vertical/horizontal direction with two random RGB colours
3	flat_rgb	Render single random RGB colour
4	noise	Render artificial noise with two random RGB colours

Table 2
Python pseudocode describing the DR texture generators. rgb1 and rgb2 are random RGB colours; the value of each RGB channel is drawn uniformly from $\{0 \dots 255\}$.

No.	Option	Expression
1	checkerboard	texture_mat[x, y] = rgb1 **if** (x + y) % 2 == 0 **else** rgb2
2	gradient	texture_mat[x, y] = (1 - p) · rgb1 + p · rgb2, vertical: p = y/ (height - 1), horizontal: p = x/ (width - 1)
3	flat_rgb	texture_mat[x, y] = rgb1
4	noise	texture_mat[x, y] = rgb1 **if** p > 0.9 **else** rgb2, p ∼ U(0, 1)

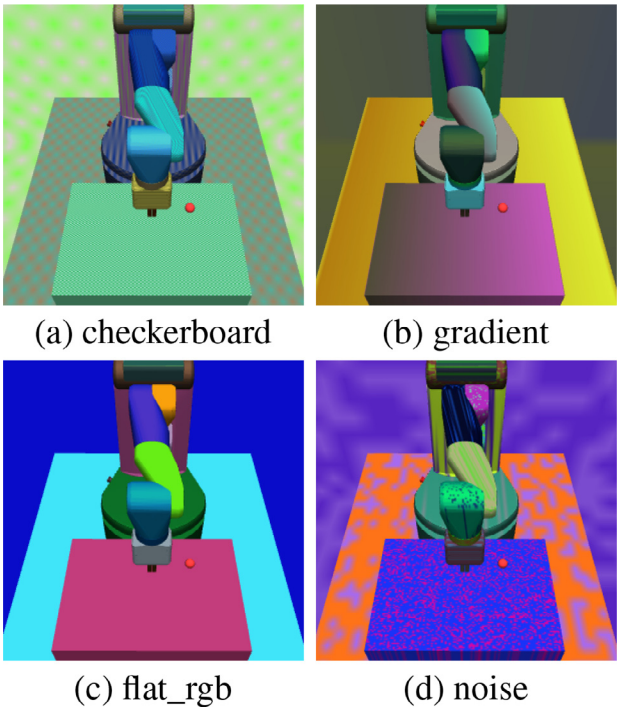


Fig. 4. Visualisation of different DR textures used during training: (a) checkerboard, (b) gradient, (c) flat_rgb, (d) noise.

niques used in this work. In Section 3, we detail our simulated environments, DRL agent network structure, training details, and test scenarios. In Section 4, we conduct an exhaustive analysis on the trained models using a broad suite of interpretability techniques. Finally, in Section 5, we discuss our findings, and provide recommendations for applying interpretability techniques to DRL agents. The code used in this paper is available at: https://github.com/TianhongDai/domain-rand-interp.

2. Methods

2.1. Reinforcement Learning

In RL, the aim is to learn optimal behaviour in sequential decision problems [88], such as finding the best trajectory for a manipulation task. It can be described by a Markov decision process (MDP), whereby at every timestep t the agent receives the state of the environment $\mathbf{s}_t$, performs an action $\mathbf{a}_t$ sampled from its policy $\pi(\mathbf{a}_t|\mathbf{s}_t)$, and then receives the next state $\mathbf{s}_{t+1}$ along with a scalar reward r_{t+1}. Formally, a (discrete-time) MDP consists of:

- a set of (discrete/continuous) states, $\mathcal{S}$,
- a set of (discrete/continuous) actions, $\mathcal{A}$,
- (nonlinear) transition dynamics, $\mathbf{s}_{t+1} \sim \mathcal{T}(\mathbf{s}_t, \mathbf{a}_t)$,
- a reward function, $\mathrm{r}_{t+1} = \mathcal{R}(\mathbf{s}_t, \mathbf{a}_t)$,
- and a distribution over initial states, $p(\mathbf{s}_0)$.

The goal of RL is to find the optimal policy, π^*, which achieves the maximum expected return in the environment:

$$\mathbb{E}[R_{t=0}] = \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t r_{t+1}\right], \tag{1}$$

where in practice a discount value $\gamma \in [0, 1)$ is used to weight earlier rewards more heavily and reduce the variance of the return over an episode of interaction with the environment, ending at timestep T. Fig. 5 shows our experimental setup as an MDP, where π is optimised to control a robotic arm to reach a target.

Policy search methods are one way of finding the optimal policy. In particular, policy gradient methods that are commonly used with NNs perform gradient ascent on $\mathbb{E}_\pi[R]$ to optimise a parameterised policy $\pi(\cdot;\theta)$ [95]. Other RL methods rely on value functions, which represent the future expected return from following a policy from a given state: $V_\pi(\mathbf{s}_t) = \mathbb{E}_\pi[R_t]$. The combination of learned policy and value functions are known as actor-critic methods, and utilise the critic (value function) in order to reduce the variance of the training signal to the actor (policy) [7]. For example, instead of directly maximising the return R_t, the policy can be trained to maximise the advantage $A_t = R_t - V_t$. Here, the advantage is the difference between the empirical and predicted return, and represents the "advantage" of taking a specific action (resulting in R_t), over the average return following the policy π (given by V_t).

We note that in practice many problems are better described as partially-observed MDPs (POMDPs), where the observation $\mathbf{o}_t$ received by the agent does not contain full information about the state of the environment. Formally, a POMDP additionally consists of:

- a set of (discrete/continuous) observations, Ω,
- and a conditional distribution over observations, $\mathbf{o}_t \sim O(\mathbf{s}_t)$.

In visuomotor object manipulation, partial observation can occur as the end effector blocks the line of sight between the camera and the object, causing self-occlusion. A common solution to this is to utilise recurrent connections within the NN, allowing information about observations to propagate from the beginning of the episode to the current timestep [94]; implicitly, such a recurrent NN would learn an approximate "belief" over the current state of the underlying MDP. Given this approach, we henceforth use standard MDP notation to describe our RL setup.

2.1.1. Proximal Policy Optimisation

For our experiments, we train our agents using proximal policy optimisation (PPO) [79]. PPO is one of the most widely used DRL

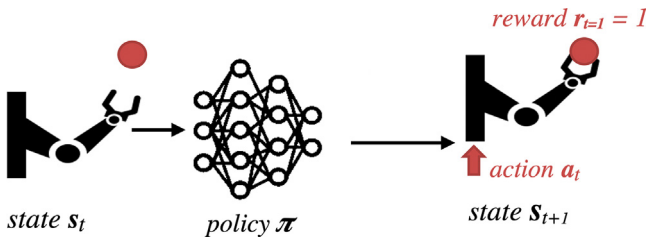


Fig. 5. A single state transition for a robotic reaching MDP. Upon observing state $\mathbf{s}_t$, the agent should choose an action $\mathbf{a}_t$ that brings it closer to the target. If the arm reaches the target in state $\mathbf{s}_{t+1}$, then it will receive a positive reward r_{t+1}.

algorithms [1,8,62]. It is also known to scale to challenging problems, including beating the world champions of the Dota 2 video game [8], and learning to perform a variety of complex manipulation tasks with a Shadow Dexterous Hand in the real world [1].

Rather than training the policy to maximise the advantage directly, PPO instead maximises the surrogate objective:

$$\mathscr{L}_{clip} = \mathbb{E}_t[\min(\rho_t(\theta)A_t, \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)A_t)], \tag{2}$$

with

$$\rho_t(\theta) = \frac{\pi(\mathbf{a}_t|\mathbf{s}_t;\theta)}{\pi_{old}(\mathbf{a}_t|\mathbf{s}_t;\theta_{old})}, \tag{3}$$

where $\rho_t(\theta)$ is the ratio between the current policy and the old policy, ϵ is the clip ratio which restricts the change in the policy distribution, and A_t is the advantage, which we choose to be the Generalised Advantage Estimate (GAE):

$$A_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \ldots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \tag{4}$$

that mixes Monte Carlo returns R_t and temporal difference errors $\delta_t = r_t + \gamma V_\pi(\mathbf{s}_{t+1}) - V_\pi(\mathbf{s}_t)$ with hyperparameter λ [78].

In practice, both the actor and the critic can be combined into a single NN with two output heads, parameterised by θ [54]. The full PPO objective involves maximising $\mathscr{L}_{clip}$, minimising the squared error between the learned value function and the empirical return:

$$\mathscr{L}_{value} = \mathbb{E}_t\left[\left(V_\pi(\mathbf{s}_t;\theta) - R_t\right)^2\right], \tag{5}$$

and maximising the (Shannon) entropy of the policy, which for discrete action sets of size $|\mathscr{A}|$, is defined as:

$$\mathscr{L}_{entropy} = \mathbb{E}_t\left[-\sum_{n=1}^{|\mathscr{A}|}\pi(a_n|\mathbf{s}_t;\theta)\log\left(\pi(a_n|\mathbf{s}_t;\theta)\right)\right]. \tag{6}$$

Entropy regularisation prevents the policy from prematurely collapsing to a deterministic solution and aids exploration [95].

Using a parallelised implementation of PPO, we are able to train our agents to strong performance on all training setups within a reasonable amount of time. Training details are described in Subsection 3.2.

2.2. Neural Network Interpretability

The recent success of machine learning (ML) methods has led to a renewed interest in trying to interpret trained models. In this work, we are primarily concerned with scientific understanding, but our considerations are grounded in other properties necessary for eventual real-world deployment, such as robustness to OoD inputs.

The challenge that we face is that, unlike other ML algorithms that are considered interpretable by design (such as decision trees or nearest neighbours [16]), standard NNs are generally considered black boxes. However, given decades of research into methods for interpreting NNs [12,56], we now have a range of techniques at our disposal [25]. Beyond simply looking at test performance (a measure of interpretability in its own right [13]), we will focus on a variety of techniques that will let us examine trained NNs both in the context of, and independently of, task performance. In particular, we discuss saliency maps (Subsection 2.2.1), activation maximisation (Subsection 2.2.2), weight visualisations (Subsection 2.2.3), statistical and structural weight characterisations (Subsection 2.2.4), unit ablations (Subsection 2.2.5), layer re-initialisation (Subsection 2.2.6), recurrent ablations (Subsection 2.2.7) and activation analysis (Subsection 2.2.8). By utilising a range of techniques we hope to cover various points along the trade-off between fidelity and interpretability [71].

2.2.1. Saliency Maps

Saliency maps are one of the most common techniques used for understanding the decisions made by NNs, and in particular, convolutional NNs (CNNs). In line with prior work on interpreting DRL agents [23], we use occlusion-based methods, in which parts of the image are masked to perform a sensitivity analysis with respect to the change in the network's outputs. The original method introduced by Zeiler et al. [99] proposes running a (grey, square) mask over the input and tracking how the network's outputs change in response. Greydanus et al. [23] applied this method to understanding actor-critic-based DRL agents, using the resulting saliency maps to examine strong and overfitting policies; they however noted that a grey square may be perceived as part of a grey object, and instead used a localised Gaussian blur to add "spatial uncertainty". The saliency value $S_{m,n}$ for each input location (m,n) is the Euclidean distance between the original output and the output given the input $\mathbf{x}^{occ}_{m,n}$ which has been occluded at location (m,n):

$$S_{m,n} = \|F(\mathbf{x}) - F\left(\mathbf{x}^{occ}_{m,n}\right)\|_2, \tag{7}$$

where $\|\cdot\|_p$ denotes the ℓ_p-norm.

However, we found that certain trained agents sometimes confused the blurred location with the target location—a failing of the attribution method against noise/distractors [38], and not necessarily the model itself. Atrey et al. [5] identified this issue of applying saliency methods to DRL agents as modifying observations in a manner that is incongruent with the true environment's state and generative process, and proposed intervening directly in the environment in order to employ counterfactual methods. However, in general this level of control over the environment is not possible. Motivated by the methods that compute interpretations against reference inputs [6,71,83,87], we replaced the Gaussian blur with a mask[3] derived from a baseline input, which roughly represents what the model would expect to see on average. Intuitively, this acts as a counterfactual, revealing what would happen if the specific part of the input was not there. For this we averaged over frames collected from our standard evaluation protocol (see Subsection 3.1 for details), creating an average input to be used as an improved mask for the occlusion-based method (Fig. 6). Unless specified otherwise, we use our average input baseline for all occlusion-based saliency maps. Contemporaneous work has examined the use of more advanced baselines for gradient-based saliency map methods [85]. Another recent work has introduced a more robust DRL-specific saliency method [69], but it is only applicable to agents which learn a state-action value function over discrete action spaces.

2.2.2. Activation Maximisation

Gradients can also be used to try and visualise what maximises the activation of a given neuron/channel. This can be for-

[3] Replacing a circular region of 5px radius around the (m,n) location.

mulated as an optimisation problem, using projected gradient ascent in the input space (where after every gradient step the input is clamped back to within $[0, 1]$) [15]. Although this would ideally show what a neuron/channel is selective for, unconstrained optimisation may end up in solutions far from the training manifold [49], and so a variety of regularisation techniques have been suggested for making qualitatively superior visualisations. We experimented with some of "weak regularisers" [60], and found that a combination of frequency penalisation (Gaussian blur) [58] and transformation robustness (random scaling and translation/jitter) [57] worked best, although they were not sufficient to completely rid the resulting visualisations of the high frequency patterns caused by strided convolutions [59]. We performed the optimisation procedure for activation maximisation for 20 iterations, applying the regularisation transformations and taking gradient steps in the ℓ_2-norm [48] with a step size of 0.1. Pseudocode for our method, applied to a trained network f, is detailed in Algorithm 1.

Algorithm 1: Activation maximisation procedure with transformation robustness, frequency penalisation and ℓ_2-norm gradient updates.

$f\prime \leftarrow$ network f truncated at intermediate layer
$i \leftarrow$ optimisation iterations
$n \leftarrow$ neuron/channel index
$\alpha \leftarrow$ step size
$x \sim U(0, 1)$ with dimensionality 3 $\times$ height $\times$ width
loop i steps
  $x \leftarrow$ *Random Scale*(x)
  $x \leftarrow$ *Random Jitter*(x)
  $x \leftarrow$ *Gaussian Blur*(x)
  $\mathscr{L} \leftarrow \text{mean}(f\prime(x)_n)$
  $x \leftarrow x + \alpha \frac{\nabla \mathscr{L}_x}{\|\nabla \mathscr{L}_x\|_2}$
  $x \leftarrow \min(\max(x, 0), 1)$
end loop
return x

2.2.3. Weight Visualisations

It is possible to visualise both convolutional filters and fully-connected weight matrices as images. Part of the initial excitement around DL was the observation that CNNs trained on object recognition would learn frequency-, orientation- and colour-selective filters [41], and more broadly might reflect the hierarchical feature extraction within the visual cortex [97]. However, as demonstrated by Such et al. [86], DRL agents can perform well with spatially unstructured filters, although they did find a positive correlation between spatial structure and performance for RL agents trained with gradients. Consistent with these findings, we find that spatial structure is correlated with OoD performance (Subsection 3.4). To support this, we developed a novel quantitative measure to compare filters, which we discuss below.

2.2.4. Statistical and Structural Weight Characterisations

Magnitude A traditional measure for the "importance" of individual neurons in a weight matrix is their magnitude, as exemplified by utilising weight decay as a regulariser [28]. Similarly, convolutional filters, considered as one unit, can be characterised by their ℓ_1-norms. Given that NN weights are typically randomly initialised with small but non-zero values [22,44], the presence of many zeros or large values indicate significant changes during training. We can compare these both across trained agents, and across the training process (although change in magnitude may not correspond with a change in task performance [101]).

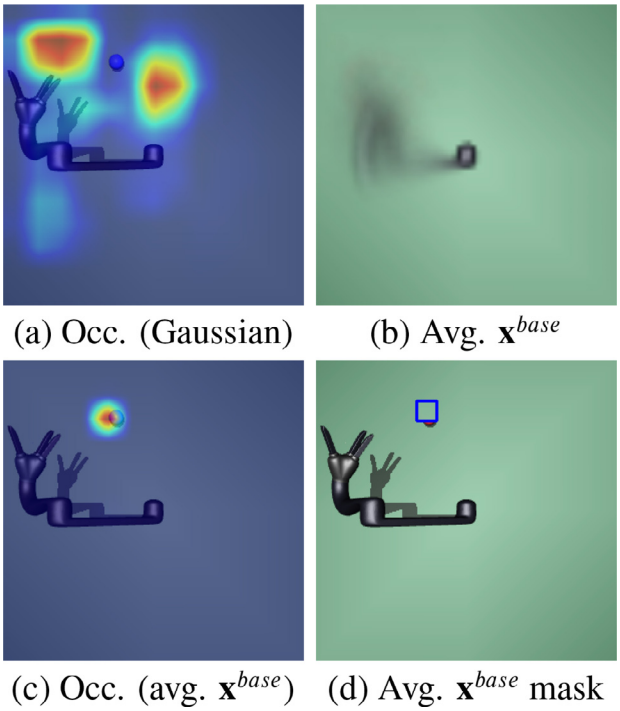


Fig. 6. Occlusion-based saliency map method (a) improved (c) by the use of an *average image* (b, d). Occlusion (occ.) with the *default Gaussian blur* (a) shows additional artifacts. By creating an average image (b) from a large set of trajectories, we can form an improved baseline for occlusion (c); the usage of the average image as the occlusion (with a blue outline added for emphasis) is pictured in (d).

Spectral Analysis Convolutional filters are typically initialised pseudo-randomly, so that there exists little or no spatial correlation within a single unit. We hence propose using the 2D discrete power spectral density (PSD) as a way of assessing the spatial organisation of convolutional filters, and the power spectral entropy (PSE) as a measure of their complexity. Given the mean-centred[4] 2D spatial-domain filter, $\mathbf{W}_{m,n}$, its corresponding spectral representation, $\widehat{\mathbf{W}}_{u,v}$, can be calculated via the 2D discrete Fourier transform of the original filter pattern ($j = \sqrt{-1}$):

$$\widehat{\mathbf{W}}_{u,v} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} W_{m,n} \exp\left[-\frac{j2\pi}{MN}(um + vn)\right], \tag{8}$$

and its PSD, $\mathbf{S}_{u,v}$, from the normalised squared amplitude of the spectrum:

$$\mathbf{S}_{u,v} = \frac{1}{UV}\left|\widehat{\mathbf{W}}_{u,v}\right|^2, \tag{9}$$

where (m, n) are spatial indices, (u, v) are frequency indices, (M, N) is the spatial extent of the filter, and (U, V) is the frequency extent of the filter.

When renormalised such that the sum of the PSD is 1, the PSD may be thought of as a probability mass function over a dictionary of components from a spatial Fourier transform. We can treat each location (u, v) in Fourier space as a symbol, and its corresponding value at $\mathbf{S}_{u,v}$ as the probability of that symbol appearing. The PSE is then simply the Shannon entropy of this distribution, which we use as a measure of spatial (dis)organisation. In our analysis (Subsection 4.3), we include statistics calculated over randomly initialised networks as a baseline. As the initial weights for units are typically drawn independently from a normal or uniform dis-

[4] Offsetting the mean exposes the *relative* spatial structure.

tribution, this leads to a fairly flat PSD with PSE close to $\log(MN)$—an upper-bound on PSE.

One weakness of spectral analysis is that these measures will fail to pick up strongly localised spatial features, as such filters would also result in a roughly uniform PSD. In practice, global structure is still useful to quantify, and matches well with human intuition (Fig. 7).

Entropy as an information-theoretic measure has been used in DL in many functions, from predicting neural network ensemble performance [27] to usage as a regulariser [37] or pruning criteria [46] when applied to activations. Spectral entropy has been used as an input feature for NNs [42,53,84,103], but, to the best of our knowledge, not for quantifying aspects of the network itself.

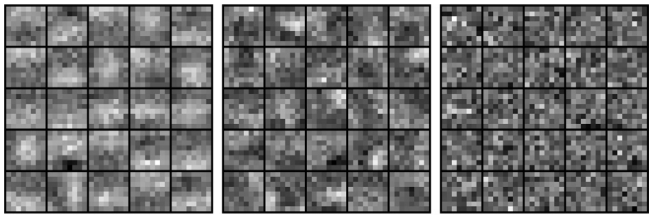
2.2.5. Unit Ablations

Another way to characterise the importance of a single neuron/convolutional filter is to remove it and observe how this affects the performance of the NN: a large drop indicates that a particular unit is by itself very important to the task at hand. More generally, rather than only looking at performance, one might look for a large change in the output. While more sophisticated unit ablations have been applied in DRL [52], this has only been in the context of a control task with a low-dimensional symbolic state space.

2.2.6. Layer Re-initialisation

One can extend the concept of ablations to entire layers, and use this to study the *re-initialisation robustness* of trained networks [101]. Typical neural network architectures, as used in our work, are compositions of multiple parameterised layers, with parameters $\{\theta_1, \theta_2, \ldots, \theta_L\}$, where L is the depth of the network. Using θ_l^t to denote the set of parameters of layer $l \in [1, L]$ at training epoch $t \in [1, T]$ over a maximum of T epochs, we can study the evolution of each layer's parameters over time—for example through the change in the ℓ_∞- or ℓ_2-norm of the set of parameters.

Zhang et al. [101] proposed re-initialisation robustness as a measure of how important a layer's parameters are with respect to task performance over the span of the optimisation procedure. After training, for a given layer l, re-initialisation robustness is measured by replacing the parameters θ_l^T with parameters checkpointed from a previous timepoint t, that is, setting $\theta_l^T \leftarrow \theta_l^t$, and then re-measuring task performance. They observed that for common CNN architectures trained for object classification, while the parameters of the latter layers of the networks tended to change a lot by the ℓ_∞- and ℓ_2-norms, the same layers were robust to re-initialisation at checkpoints early during the optimisation procedure, and even to the initialisation at $t = 0$. In the latter case, the parameters are independent of the training data, which means that the effective number of parameters is lower than the total number of parameters. Given that the effective number of parameters is a better measure for model complexity than total number, this potentially allows us to differentiate between models with the same architecture. Unlike Zhang et al. [101], we use re-initialisation robustness to study the effect of task complexity (training with and without DR, and with and without proprioceptive inputs), but with networks of similar capacity.



(a) PSE = 2.63 (b) PSE = 3.19 (c) PSE = 3.86

Fig. 7. Convolutional filters from models trained with DR, with varying power spectral entropies, ranked from the lowest (a) to highest (c). The PSE value refers to the centre filter.

2.2.7. Recurrent Ablation

When using recurrent units in the network architecture, we can test if non-trivial recurrent dynamics are being used by forcing the hidden state to be constant. If the performance of the agent degrades, then it is somehow using the recurrent dynamics to perform the task—although it is difficult to say what the exact "strategy" might be. However, if the performance drop is zero or minimal, then the recurrency is not being utilised. The constant values of the hidden states should be set to the empirical average of the values during normal operation, as naively setting all values to zero could cause a considerable shift in the distribution of expected inputs—as the hypothesis is that the network may have learned a constant offset, rather than completely ignoring the hidden state. While it is possible to investigate the hidden states of DRL agents over time, visualising and inspecting a high-dimensional time series can be difficult, even for domain experts [35].

2.2.8. Entanglement

Finally, we consider analysing the internal activations of trained networks. One of the primary methods for examining activations is to take the high-dimensional vectors and project them to a lower-dimensional space (commonly $\mathbb{R}^2$ for visualisation purposes) using dimensionality reduction methods that try and preserve the structure of the original data [70]. Common choices for visualising activations include both principal components analysis (PCA; a linear projection) [14,65] and t-distributed stochastic neighbor embedding (t-SNE; a nonlinear projection) [26,47,55]. Prior work has used t-SNE to explore the state visitations of DRL agents, clustering states which are associated with similar actions [55,98].

While these works qualitatively examine the projections of the activations for a single network, or compare them across trained networks, we additionally introduce a method to use the projections quantitatively. In supervised learning settings, one can examine class overlap in the projected space [70]. In our RL setting there is no native concept of a "class", but we can instead use activations taken under different OoD test scenarios (Subsection 3.4) to see (beyond the generalisation performance) how the internal representations of the trained networks vary under the different scenarios. Specifically, we measure *entanglement* ("how close pairs of representations from the same class are, relative to pairs of representations from different classes" [17]) using the soft nearest neighbour loss, $\mathscr{L}_{SNN}$ [76], defined over a batch of size B with samples $\mathbf{x}$ and classes y (where in our case $\mathbf{x}$ is a projected activation and y is a test scenario) with temperature T (and using $\delta_{i,j}$ as the Kronecker-delta):

$$\mathscr{L}_{SNN} = \frac{1}{B}\sum_{n=1}^{B}\left(\log\left[\sum_{b=1}^{B}(1-\delta_{b,n})\cdot e^{-\frac{\|\mathbf{x}_n-\mathbf{x}_b\|_2^2}{T}}\right] - \log\left[\sum_{a=1}^{B}(1-\delta_{a,n})\cdot\delta_{y_a,y_n}\cdot e^{-\frac{\|\mathbf{x}_n-\mathbf{x}_a\|_2^2}{T}}\right]\right) \tag{10}$$

In particular, if representations between different test scenarios are highly entangled, this indicates that the network is largely invariant to the factors of variation between the different scenarios. Considering DR as a form of data augmentation, this is what we might expect of networks trained with DR.

3. Experiments

3.1. Environments

We base our experiments on the common setup of performing target-reaching with visuomotor control. The tasks involve moving the end effector of a robot arm to reach a randomly positioned target during each episode, with visual (one RGB camera view) and sometimes proprioceptive (joint positions, angles and velocities) input provided to the agent. Unlike many DRL experiments where the position of the joints and the target are explicitly provided [68], in our setup the agent must infer the position of the target, and sometimes itself, purely through vision. Importantly, we use two robotic arms—the Fetch Mobile Manipulator and the KINOVA JACO Assistive robotic arm (pictured in Fig. 8; henceforth referred to as Fetch and Jaco, respectively)—which have different control schemes and different visual appearances. This leads to changes in the relative importance of the visual and proprioceptive inputs, which we explore in several of our experiments. These robot environments are commonly used within the DRL literature [2,24,74] and, therefore, adopting these in our experiments enables both comparison to prior work and applicability to the DRL field.

The Fetch has a 7 degrees-of-freedom (DoF) arm, not including the two-finger gripper. The original model and reaching task setup were modified from the `FetchReach` task in OpenAI Gym [9,68] in order to provide an additional camera feed for the agent (while also removing the coordinates of the target from the input). The target can appear anywhere on the 2D table surface. The agent has 3 sets of actions, corresponding to position control of the end effector ([-5, 5] cm in the x, y and z directions; gripper control is disabled).

The Jaco has been configured to be 6 DoF, with the 3 fingers disabled. The target can appear anywhere within a 3D area to one side of the robot's base. The agent has 6 sets of actions, corresponding to velocity control of the arm joints ([-0.6, +0.6] rad/s). Due to the difference in control schemes, 2D versus 3D target locations, and homogeneous appearance of the Jaco, reaching tasks with the Jaco are more challenging—particularly when proprioceptive input is not provided to the agent. A summary of the different settings for the Fetch and Jaco environments is provided in Table 3.

During training, target positions are sampled uniformly from within the set range, with episodes terminating once the target is reached (within 10 cm of the target centre), or otherwise timing out in 100 timesteps. The reward is sparse, with the only nonzero reward being +1 when the target is reached. During testing, a fixed set of target positions, covering a uniform grid over all possible target positions, are used; 80 positions in a 2D grid are used for Fetch, and 250 positions in a 3D grid are used for Jaco. By using a deterministic policy and averaging performance over the entire set of test target positions, we obtain an empirical estimate of the probability of task success. Test episodes are set to time out within 20 timesteps in order to minimise false positives from the policy accidentally reaching the target.

We only randomise initial positions (for all agents) and visuals (for some agents), but not dynamics, as this is still a sufficiently rich task setup to explore. Henceforth we refer to agents trained with visual randomisations as being under the DR condition, whereas agents trained without are the standard (baseline) condition. Apart from the target, we randomise the visuals of all other objects in the environment: the robots, the table, the floor and the skybox. At the start of every episode and at each timestep, we randomly alter the RGB colours, textures and colour gradients of all surfaces (Fig. 3).

Importantly, there are several aspects that are not altered, as we also want to test extrapolation to OoD scenarios (Subsection 3.4). For example, one of the tests that we apply to probe generalisation is to change a previously static property—surface reflectivity, which is completely disabled during training—and see how this affects the trained agents. All environments were constructed in MuJoCo [90], a fast and accurate physics simulator that is commonly used for DRL experiments.

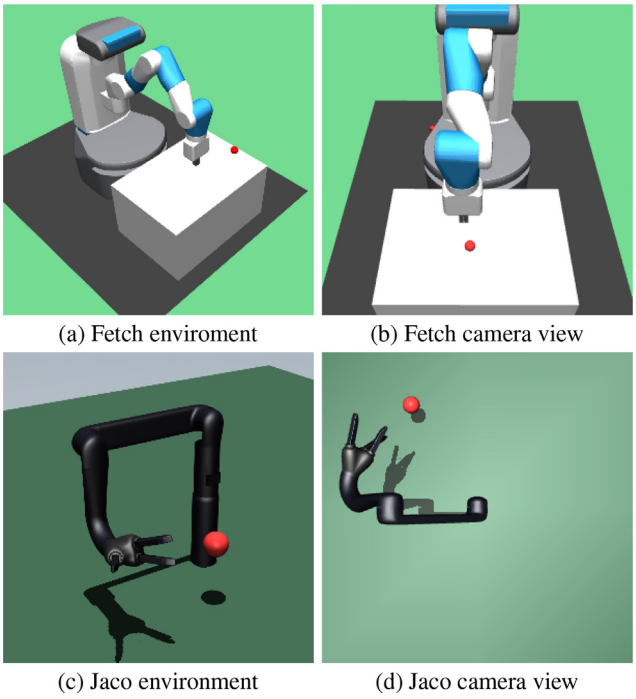


Fig. 8. Fetch (a) and Jaco (c) environments, with associated camera views (b, d) that are provided as input to the agents.

3.2. Networks and Training

We utilise the same basic actor-critic network architecture for each experiment, based on the recurrent architecture used by Rusu et al. [74] for their Jaco experiments. The architecture has 2 convolutional layers, a fully-connected layer, a long short-term memory (LSTM) layer [19,31], and a final fully-connected layer for the policy and value outputs; rectified linear units were used at the output of the convolutional layers and first fully-connected layer. Proprioceptive inputs, when provided, were concatenated with the outputs of the convolutional layers before being input into the first fully-connected-layer. The policy, $\pi(\cdot;\theta)$, is a product of independent categorical distributions, with one distribution per action dimension. Weights were initialised using orthogonal weight initialisation [32,77] and biases were set to zero. The specifics of the architecture are detailed in Fig. 9.

Table 3
Summary of Fetch and Jaco experimental setups.

Setting	Fetch	Jaco
Active (Total) DoF	7 (8)	6 (9)
Target Range	$21 \times 31\text{cm}^2$	$40 \times 40 \times 40\text{cm}^3$
Num. Test Targets	80	250
Vision Input	$3 \times 64 \times 64$	$3 \times 64 \times 64$
Proprioceptive Inputs	30	18
Control Type	Position	Velocity
Num. Actions	3	6
Action Discretisation	5	5
Control Frequency	6.67 Hz	6.67 Hz

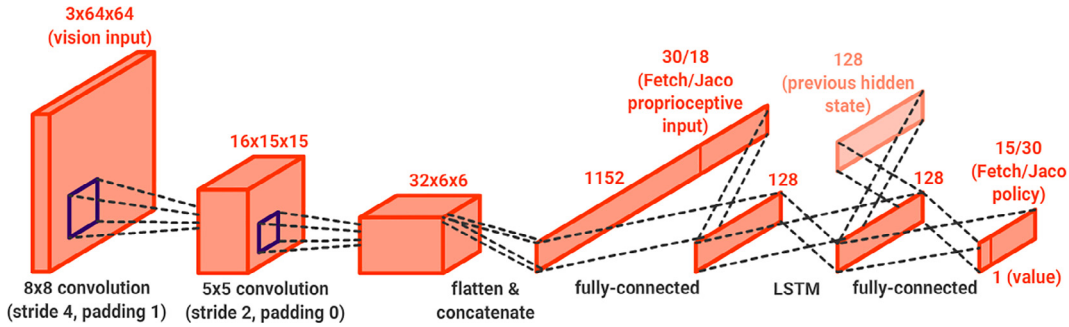


Fig. 9. Actor-critic network architecture.

During training, a stochastic policy $\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s};\theta)$ is used and trained with PPO with clip ratio $\epsilon = 0.1$, GAE trace decay $\lambda = 0.95$ and discount $\gamma = 0.99$. Each epoch of training consists of 32 worker processes collecting 128 timesteps worth of data each, then 4 PPO updates with a minibatch size of 1024. We train for up to 5×10^3 epochs, using the Adam optimiser [39] with learning rate $= 2.5 \times 10^{-4}$, $\beta s = \{0.9, 0.999\}$, and $\epsilon = 1 \times 10^{-5}$. $\mathscr{L}_{value}$ is weighted by 0.5 and $\mathscr{L}_{entropy}$ is weighted by 0.01. If the max ℓ_2-norm of the gradients exceeds 0.5 they are rescaled to have a max ℓ_2-norm of 0.5 [63]. During testing, the deterministic policy $\mathbf{a} = \text{argmax}_{\mathbf{a}}\pi(\mathbf{a}|\mathbf{s};\theta)$ is used. Our training was implemented using PyTorch [64]. Training each model (each random seed) for the full number of timesteps takes 1 day on a GTX 1080Ti; we trained models with 5 different seeds for each of the 8 conditions. The overall setup is detailed in Algorithm 2.

Algorithm 2: PPO + DR training.

Require environment E, actor network $\pi(a|s;\theta)$, critic network $V_\pi(s;\theta)$, number of training epochs N, number of timesteps to collect data T, number of PPO updates M, on-policy transition memory $\mathscr{B}$
for epoch $= 1, 2, \ldots, N$ **do**
  $\mathscr{B} \leftarrow \varnothing$
  for $t = 1, 2, \ldots, T$ **do**
    if E has terminated **then** Reset E and receive DR state s_0
    end if
    Sample an action $a_t \sim \pi(a|s_t;\theta)$
    Execute a_t in E and receive reward r_t, DR next state s_{t+1}, and *terminal*
    Store the transition $(s_t, a_t, r_t, s_{t+1}, terminal)$ in $\mathscr{B}$
  end for
  for PPO update $= 1, 2, \cdots, M$ **do**
    Update $\pi(a|s;\theta)$ using gradient ascent on Eqn. 2 + Eqn. 6 with data from $\mathscr{B}$
    Update $V_\pi(s;\theta)$ using gradient descent on Eqn. 5 with data from $\mathscr{B}$
  end for
end for

3.3. Domain Shift

Once agents are successfully trained on each of the different conditions (Fetch/Jaco, DR/no DR, proprioceptive/no proprioceptive inputs), we can perform further tests to see how they generalise. However, while the agents achieve practically perfect test performance on the conditions that they were trained under, the Jaco agents trained with DR but without proprioceptive inputs fare worse when tested under the simulator's standard visuals (Fig. 10), demonstrating a drop in performance under domain shift. It is both assumed and observed that domain shift occurs when transferring models trained with DR to the more complex and noisy visuals of the real world, but it is somewhat unexpected to see this happen when shifting to simpler visuals, which are expected to be a subset of DR visuals—this indicates that the agent is in some way overfitting to the DR visuals. Because of this, it is not completely straightforward to compare performance between different agents, but the change in performance of a single agent over differing test conditions is still highly meaningful.

We also trained agents with visual DR where the visuals were only randomised at the beginning of each episode, and kept fixed during. These agents exhibited the same gap in performance between the standard and randomised visuals, indicating that this is not an issue of temporal consistency in the DR setup. Therefore, while agents trained with visual DR may be invariant to the visual conditions observed during training, this invariance may have limited OoD generalisation with respect to backgrounds (Fig. 1).

3.4. Test Scenarios

In order to test how the agents generalise to different held-out conditions, we constructed a suite of tests for the trained agents (Fig. 11 for observations for Fetch under the different conditions[5], and Table 4 for the results):

Standard. This is the standard evaluation procedure with the default simulator visuals, where the deterministic policy is applied to all test target positions and the performance is averaged (1.0 means that all targets were reached within 20 timesteps).
Colour. This introduces a non-red sphere distractor object that is the same size and shape as the target. This tests the sensitivity of the policy to localising the target given another object of a different colour. We use yellow, blue, green, brown and purple; some of these colours have a red component (yellow, brown and purple), while others do not (blue and green).
Shape. This introduces a red distractor object that is the same width and colour as the target, but a different shape. We use a cube, ellipsoid, rectangle and diamond.
Illumination (Lvl.). This changes the diffuse colour of the main light. We use 5 illumination levels: 0.4 to 0.0 for Fetch, and 0.9 to 0.1 for Jaco.
Illumination (Dir.). This changes the location of the main light. We use 5 different directions for both robots.
Noise. This adds Gaussian noise $\sim N(0, 0.25)$ to the visual observations.
Reflection. This sets the table (for Fetch) or ground (for Jaco) to be reflective. This introduces reflections of the robot (and the target for Jaco) in the input.

[5] Simulation environment parameters of MuJoCo can be referenced from http://www.mujoco.org/book/XMLreference.html.

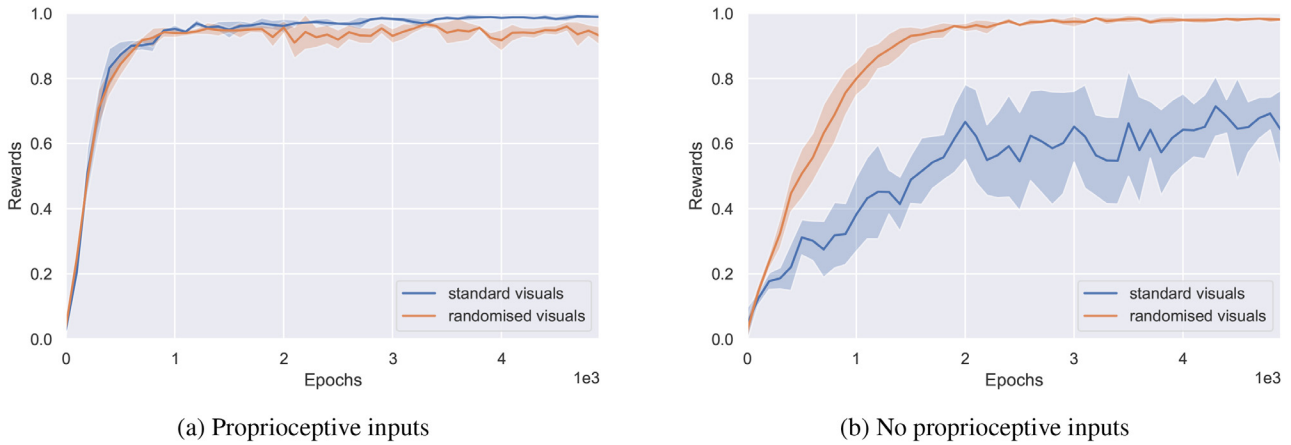


Fig. 10. Test performance of Jaco agents trained with DR and (a) with or (b) without proprioceptive inputs; the agents are tested against both standard and randomised visuals. Without proprioceptive inputs, the agents fail to fully deal with the domain gap between the randomised and standard visuals. Statistics (median and 95% confidence interval) are calculated over all models (seeds) and test target locations.

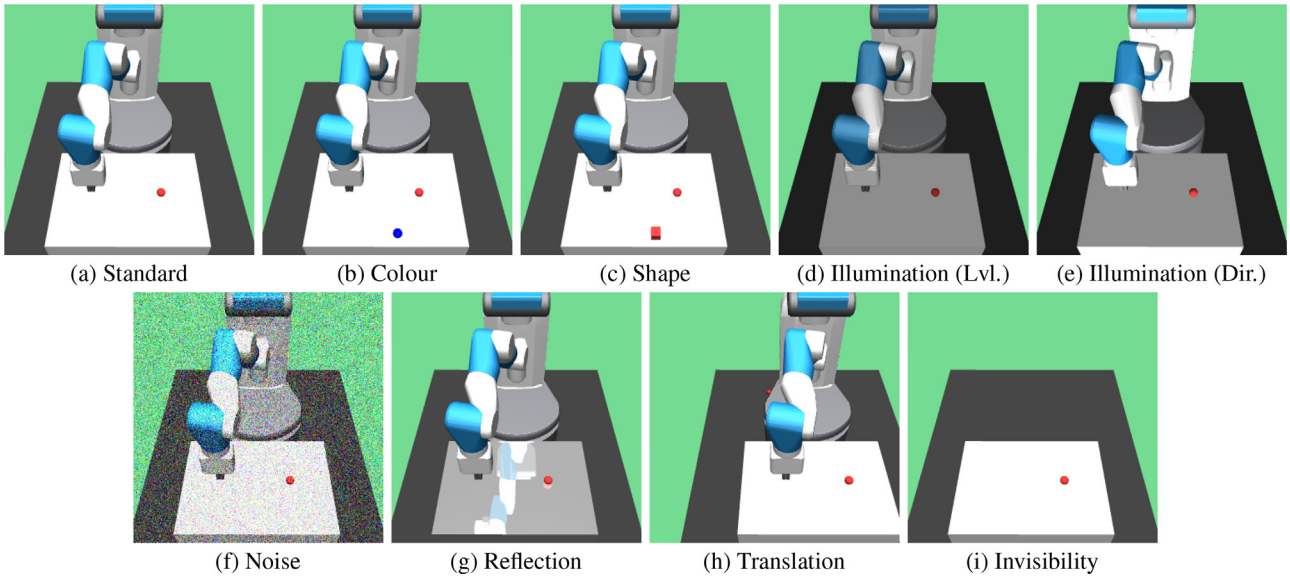


Fig. 11. Camera observations for Fetch under different test conditions.

Table 4
Test performance of all models with local visual changes (distractors), global visual changes, and invisibility (visual self-localisation test). Colour and shape results are averaged over 9 different distractor locations, as well as all colours and shapes, respectively. Checkmarks and crosses indicate enabling/disabling DR and proprioceptive inputs (Prop.), respectively. Statistics are calculated over all models (seeds) and test target locations.

Robot	DR	Prop.	Standard	Colours	Shapes	Illumination (Lvl.)	Illumination (Dir.)	Noise	Reflection	Translation	Invisibility
Fetch	×	×	1.000±0.000	0.993±0.004	0.827±0.056	0.694±0.049	0.539±0.069	0.980±0.006	0.447±0.039	0.270±0.079	0.000±0.000
Fetch	×	✓	1.000±0.000	0.937±0.027	0.524±0.087	0.593±0.064	0.491±0.066	0.988±0.004	0.570±0.078	0.231±0.078	0.000±0.000
Fetch	✓	×	0.983±0.004	0.974±0.004	0.939±0.018	0.957±0.008	0.740±0.058	0.985±0.007	0.972±0.011	0.480±0.064	0.000±0.000
Fetch	✓	✓	0.997±0.002	0.997±0.001	0.978±0.009	0.992±0.002	0.883±0.038	0.970±0.008	0.985±0.005	0.536±0.063	0.023±0.015
Jaco	×	×	0.995±0.003	0.772±0.058	0.367±0.069	0.966±0.012	0.838±0.035	0.635±0.028	0.734±0.032	0.556±0.063	0.000±0.000
Jaco	×	✓	0.995±0.001	0.808±0.044	0.382±0.064	0.876±0.032	0.911±0.014	0.478±0.059	0.618±0.061	0.688±0.048	0.001±0.001
Jaco	✓	×	0.650±0.056	0.652±0.022	0.593±0.031	0.643±0.027	0.549±0.032	0.575±0.040	0.429±0.060	0.310±0.041	0.007±0.002
Jaco	✓	✓	0.991±0.004	0.989±0.002	0.843±0.061	0.816±0.043	0.882±0.031	0.896±0.007	0.946±0.006	0.658±0.054	0.916±0.022

Translation. This offsets the RGB camera. We use 5 locations: −20 to 20 cm in the x direction for Jaco, and −20 to 20 cm in the y direction for Fetch.

Invisibility. This makes the robot transparent; this is not a realistic alteration, but is instead used to test the importance of the visual inputs for self-localisation.

3.4.1. Local Visual Changes

Noting that the baseline performance of the Jaco model trained with DR but without proprioception is lower under standard visuals, across both robots, DR confers robustness to both the colour and shape distractors (Table 4). However, there is not as consistent a pattern between agents trained without DR. Table 5 provides an

Table 5
Test performance of all models with local visual changes (distractors). Checkmarks and crosses indicate enabling/disabling DR and proprioceptive inputs (Prop.), respectively. Statistics are calculated over all models (seeds), test target locations, and 9 distractor locations.

Robot	DR	Prop.	Yellow	Blue	Green	Brown	Purple	Cube	Ellipsoid	Rectangle	Diamond
Fetch	×	×	0.993±0.007	1.000±0.000	1.000±0.000	0.978±0.017	0.995±0.004	0.775±0.085	0.993±0.007	0.960±0.036	0.580±0.142
Fetch	×	✓	0.875±0.088	1.000±0.000	1.000±0.000	0.825±0.069	0.983±0.006	0.243±0.064	0.980±0.006	0.783±0.079	0.090±0.048
Fetch	✓	×	0.970±0.011	0.980±0.006	0.975±0.006	0.973±0.010	0.973±0.011	0.913±0.042	0.975±0.009	0.965±0.013	0.905±0.048
Fetch	✓	✓	0.995±0.003	0.998±0.002	1.000±0.000	0.998±0.002	0.995±0.003	0.963±0.020	0.998±0.002	0.995±0.003	0.958±0.022
Jaco	×	×	0.281±0.067	0.871±0.036	0.994±0.003	0.730±0.095	0.984±0.008	0.274±0.077	0.407±0.098	0.759±0.066	0.027±0.006
Jaco	×	✓	0.451±0.040	0.941±0.032	0.993±0.002	0.706±0.064	0.950±0.028	0.258±0.044	0.440±0.058	0.784±0.044	0.045±0.009
Jaco	✓	×	0.640±0.046	0.655±0.055	0.651±0.055	0.668±0.046	0.648±0.046	0.636±0.040	0.642±0.035	0.666±0.047	0.426±0.054
Jaco	✓	✓	0.987±0.005	0.990±0.003	0.987±0.003	0.990±0.003	0.990±0.003	0.970±0.017	0.970±0.011	0.990±0.003	0.441±0.128

extended breakdown across the different colours and shapes, which provides important information on what strategies the agents have learned.

With Fetch, colour distractors have little effect on the agents, but the shape distractor diminishes the performance of the non-DR agent trained without proprioception somewhat, and the non-DR agent trained with proprioception significantly. Given this, it seems that the latter agent relies mainly on detecting *pure* red (plus some shape information) in order to locate the ball. As a result of self-localising based on visual input alone, the former agent develops more sophisticated vision, allowing the model to better distinguish both shapes and colours.

With Jaco, both non-DR agents suffer noticeable drops in performance in the presence of distractors with a red component, whilst both DR agents experience only a very small decrease in performance across all local distractors (bar the diamond, which looks the most similar to a sphere, especially at lower resolutions). While the non-DR agents also have reduced success with the blue sphere distractor, it is less pronounced, indicating that non-DR Jaco agents are primarily detecting large red components as the target object.

In order to test that the location of the distractor does not also influence the models' responses, we varied this and recorded the corresponding success rates. The low standard deviations shown in Table A1 indicate that the location only has a minimal impact on the results.

3.4.2. Global Visual Changes

Referring to Table 4, DR generally confers more robustness, although this time the DR agents do exhibit noticeable drops in performance across many of these tests.

Reducing the illumination levels drops the performance of all agents monotonically with respect to dimness (Table A2), although the Fetch agents trained with DR are the most robust. Intriguingly, the Jaco agents trained without proprioception are more robust with respect to this change, as compared to the agents trained with. Their need to self-localise visually necessitates a more complex visual system, whereas simpler visual processing may be thrown off by the reduction in contrast or even simply the change in the pixel values of the target. Given that the DR agents trained with proprioception tend to be the most robust across most of the test conditions, this motivates an additional consideration for training—when performing sensor fusion within a model, the combination of information should be more resilient to the loss or faulty functioning of any individual sensory input.

Changing the direction of the main illumination degrades the performance of all agents. As before, Fetch agents trained with DR are more robust than those trained without, but for Jaco agents the presence of proprioceptive inputs are more important than training with DR. This trend holds across different illumination directions (Table A3).

Additive Gaussian noise has very little effect on the Fetch agents, but reduces the performance of the Jaco agents—by over 30% for agents trained without DR, but only by about 10% for agents trained with DR. Considering other factors are consistent across training conditions, either the visual layout or difficulty of the task caused the Fetch agents to be more robust to noise than the Jaco agents.

Making the table surface reflective throws off the Fetch agents trained without DR, with an approximately 50% drop in performance, but with DR the agents are resilient to this change. The Jaco agents trained without DR also incur a significant, yet smaller drop in performance. A likely explanation for this difference is that the size of the robots relative to the image differs, and the reflection of the Jaco arm simply changes the input less. When given proprioceptive inputs, both the Fetch and the Jaco agent trained with DR display similar levels of resilience.

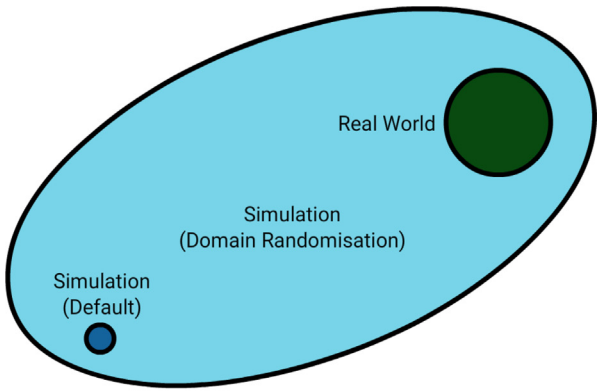
Translating the camera causes a dramatic drop in performance in all agents. DR confers some amount of resilience to this for the Fetch agents. However, all Jaco agents are similarly affected, DR or not, and their average success rates are higher than those of the Fetch agents. This suggests that all Jaco agents manage to learn a degree of translation invariance for their policies. One hypothesis for this is that the requirement to reach a target in 3D confers a more generalisable representation of space. Performance drops monotonically with deviation from the original location (Table A4). The drop is symmetric for the Fetch agents, but asymmetric for the Jaco agents—a relative shift in the target towards the centre of the image input is better than a shift away.

3.4.3. Visual Self-localisation

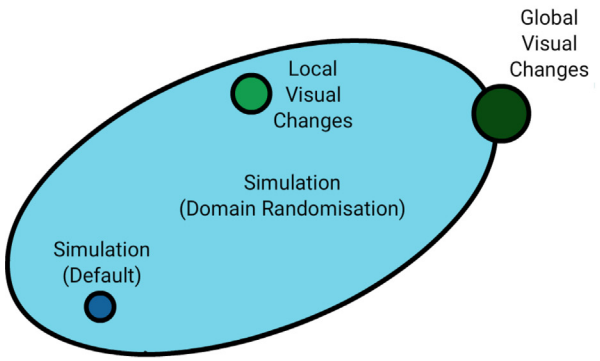
For nearly all agents, rendering the robot invisible drops performance to zero. There are four non-zero performance scores, but three of these are low enough to be attributable to chance. This test indicates that perhaps either directly or indirectly the position of the robot is inferred visually, although we cannot rule out that the drop in performance is due to the domain shift that results from rendering the arm invisible. The standout is the Jaco agent with proprioceptive inputs and DR training, which only incurs a small drop in performance—this agent is able to self-localise solely based on proprioceptive input.

3.4.4. Tests Summary

There is no single clear result from our evaluation of different setups with different types of tests, beyond the general importance of sensor fusion and DR to improve the ability for agents to generalise. The type of DR used during training—randomising colours and textures—allows generalisation to localised changes—distractor objects—but fails to reliably improve generalisation across the more global changes, such as illumination or translation (Fig. 12). This should not come as a surprise given that our DR never changed the position of the robot, nor the illumination of the target. The takeaway is that "generalisation" is more nuanced, and performing systematic tests can help probe what strategies networks might be using to operate. Finding failure cases for "weaker" agents can still be a useful exercise for evaluating more robust



(a) Idealised theory



(b) Our experiments for testing OoD generalisation

Fig. 12. Domain randomisation methodology. (a) In theory, the range of simulation parameters should be varied in such a way as to successfully encompass visual/physical properties that would be encountered in the real world. (b) In practice, we limited the scope of variation to test generalisation to OoD inputs within simulation, and observed different levels of success depending on the type of change; depending on the training settings we even observe some level of overfitting. The local and global visual changes in our tests are designed to evaluate generalisation in a way that is reflective of the real world.

agents, as it enables adversarial evaluation [92], and can inform us about the design of DR.

4. Model Analysis

The unit tests that we constructed can be used to evaluate the performance of an arbitrary black box policy under differing conditions, but we also have the ability to inspect the internals of our trained agents. Although we cannot obtain a complete explanation for the learned policies, we can still glean further information from both the learned parameters and the sets of activations in the networks.

4.1. Saliency Maps

One of the first tests usually conducted is to examine saliency maps to infer which aspects of the input influence the output of the agent. We use the occlusion-based technique with average baseline, and focus on distractors: we show saliency maps for both the standard test setup, and with either the different colour (yellow) or different shape (cube) distractors.

The saliency maps for the Fetch agents (Fig. 13) differ between all models. Apart from the model trained with DR and with proprioception (Fig. 13j-l), all agents seem to use the gripper to self-localise. Despite having access to clean proprioceptive inputs, the Fetch agent trained without DR still pays attention to its own body in the image (i.e., visual localisation strategy; Fig. 1)—so it is not necessarily the case that agents will even utilise the inputs that we may expect. The Fetch agents trained without DR show saliency on the distractors (Fig. 13a-f), while the agents trained with DR do not (with the exception of the model trained with DR and proprioception on the shape distractor, as seen in Fig. 13).

The saliency maps for the Jaco agents (Fig. 14) are more homogeneous, with a large amount of attention on the target, and little elsewhere. The saliency for the agent trained without DR and without proprioception clearly shows some attention around the base of the arm (Fig. 14a-c) that would indicate visual self-localisation. On an initial inspection, it may appear that there is no saliency around the arm for the agent trained with DR and without proprioception, although we know that in order to succeed it must be relying on visual self-localisation. Indeed, there is saliency present around the arm (Fig. 14g-i), but it is difficult to perceive. This example indicates the subjective nature of interpreting saliency maps, and hence why they should not be the sole tool for analysis.

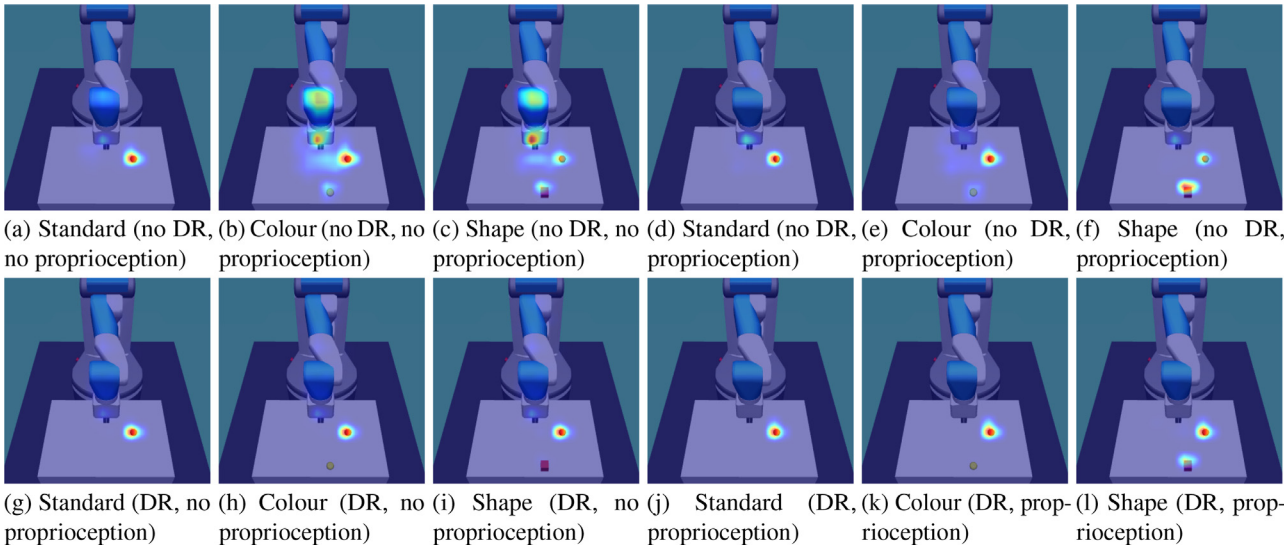
This recommendation is also borne out by the mismatch between the saliency maps and performance. For the Fetch agents trained with DR, the agent with proprioception shows saliency over the shape distractor (Fig. 13l) in contrast to without proprioception (Fig. 13i); conversely, the performance drop is greater in the latter than the former (Table 5). Similarly for the Jaco agents trained without DR, the agent without proprioception shows a large amount of saliency over the shape distractor (Fig. 14c), while the agent with proprioception demonstrates only a minimal amount of saliency (Fig. 14f); however, they both have a similar drop in performance (Table 5).

4.2. Activation Maximisation

In line with Such et al. [86], activation maximisation applied to the first convolutional layer results in edge detectors, with larger-scale spatial structure in the latter layers (Fig. 15 and Fig. 16). There are several trends that apply to both the Fetch and Jaco agents. Firstly, the agents trained without DR develop simpler, more colourful filters in both layers. In contrast, the agents trained with DR develop more edge-like detectors, with higher contrast, in their first convolutional layers. In their second convolutional layers, the feature detectors resemble the red target itself, surrounded by a complementary blue-green. This style of detector is consistent across both the DR-trained Fetch and Jaco agents, which suggests that it was not developed in response to the green floor in the Jaco environment. One noticeable difference between the second set of convolutional filters between the Fetch (Fig. 15g,h and Jaco (Fig. 16g,h) agents is that the latter also develop positional sensitivity, indicating that target localisation may be more difficult.

The Jaco agent trained without DR and without proprioceptive inputs has convolutional filters that respond maximally to yellow (Fig. 16b), and is also the most affected by the yellow distractor, with performance dropping to 28% (Table 5). Therefore this agent has learned a ball-localisation strategy that is not purely based on detecting red or spherical objects (Fig. 1).

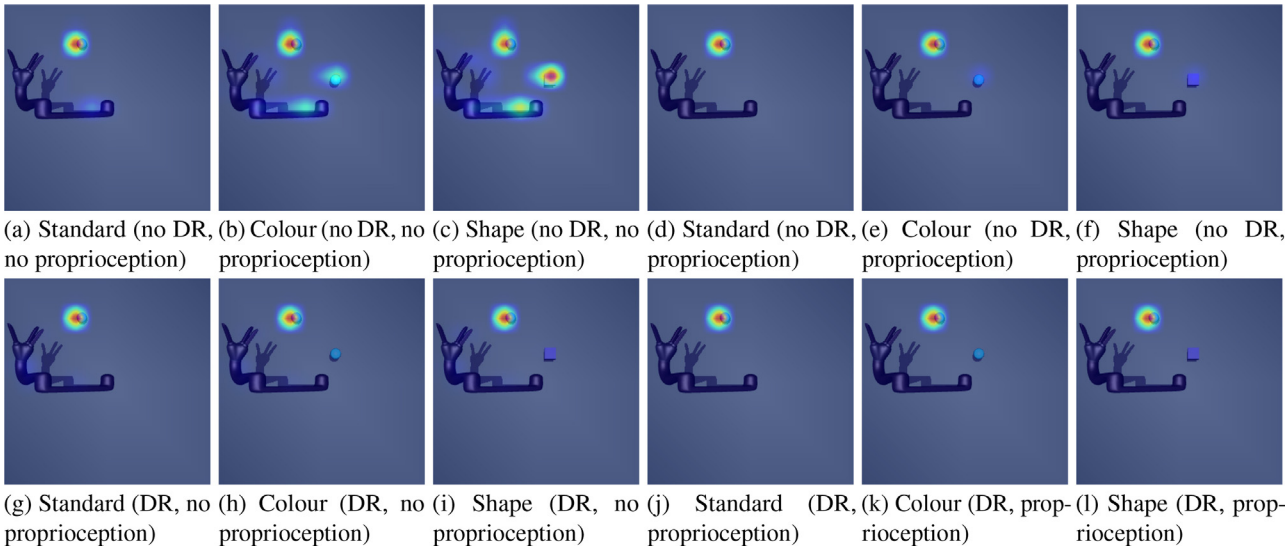
Finally, there is a more global, but largely uninterpretable structure when maximising the value function or policy outputs (choosing the unit that corresponds to the largest positive movement per action output). For Fetch agents without DR, the visualisations are dominated by red (the target colour), but with DR there is a wider spectrum of colours. This trend is the same for the Jaco agents, although without DR and without proprioceptive inputs the colours that maximise the value output are purple and green (a constant hue shift on the usual red and blue). The agents trained with DR but without proprioception have the most plain activation maximisation images for the policy, perhaps suggesting a more



(a) Standard (no DR, no proprioception) (b) Colour (no DR, no proprioception) (c) Shape (no DR, no proprioception) (d) Standard (no DR, proprioception) (e) Colour (no DR, proprioception) (f) Shape (no DR, proprioception)

(g) Standard (DR, no proprioception) (h) Colour (DR, no proprioception) (i) Shape (DR, no proprioception) (j) Standard (DR, proprioception) (k) Colour (DR, proprioception) (l) Shape (DR, proprioception)

Fig. 13. Occlusion-based saliency maps with Fetch models trained with (g-l) or without (a-f) DR and with (d-f, j-l) or without proprioception (a-c, g-i) in three different distractor conditions. The best Fetch model was used for each training condition.



(a) Standard (no DR, no proprioception) (b) Colour (no DR, no proprioception) (c) Shape (no DR, no proprioception) (d) Standard (no DR, proprioception) (e) Colour (no DR, proprioception) (f) Shape (no DR, proprioception)

(g) Standard (DR, no proprioception) (h) Colour (DR, no proprioception) (i) Shape (DR, no proprioception) (j) Standard (DR, proprioception) (k) Colour (DR, proprioception) (l) Shape (DR, proprioception)

Fig. 14. Occlusion-based saliency maps with Jaco models trained with (g-l) or without (a-f) DR and with (d-f, j-l) or without proprioception (a-c, g-i) in three different distractor conditions. The best Jaco model was used for each training condition.

factorised control scheme. For the Jaco agent (Fig. 16k), only the first and third actuators are activated by strong visual inputs (given zeroes as the proprioceptive inputs and hidden state), which correspond to the most important joints for accomplishing this reaching task (the rotating base and the elbow).

As a reminder we note that activation maximisation may not (and is practically unlikely to) converge to images within the training data manifold [49]—a disadvantage addressed by the complementary technique of finding image patches within the training data that maximally activate individual neurons [21]. Alternatively, one could train a generative model on the state distribution and query this model to generate novel states of interest [61,73].

4.3. Statistical and Structural Weight Characterisations

We calculated statistical and structural weight characteristics over all trained models (Fetch and Jaco, with/without proprioception, with/without DR, 5 seeds), which allows us to average over 40 conditions to examine the effects of DR. We analysed the norms (Subsection 2.2.4) of all of the weights of the trained agents, and could not find consistent trends across all layers. The most meaningful characterisations were the ℓ_1-norm and the power spectral entropy, PSE, (Subsection 2.2.4), applied to the convolutional filters.

Fig. 17 shows a KDE of the ℓ_1-norms and PSEs of all of the 2D filters within the first and second convolutional layers. For the ℓ_1-norm, in layer 2 the distribution is skewed towards higher values when the model is trained with DR. For the PSE, in both layers, but particularly layer 1, the distribution is skewed towards lower values when the model is trained with DR. Using the nonparametric Kolmogorov–Smirnov (K-S) two-sided test between the two distributions (DR versus non-DR), the p-value of the ℓ_1-norms is 0.014 (K-S statistic 0.072) for layer 1 and ~ 0 (K-S statistic 0.285) for layer 2, and the p-value of the PSEs is 5.71×10^{-27} (K-S statistic 0.251) for layer 1 and 3.32×10^{-9} (K-S statistic 0.044) for layer 2. Given the same weight initialisation distributions across all mod-

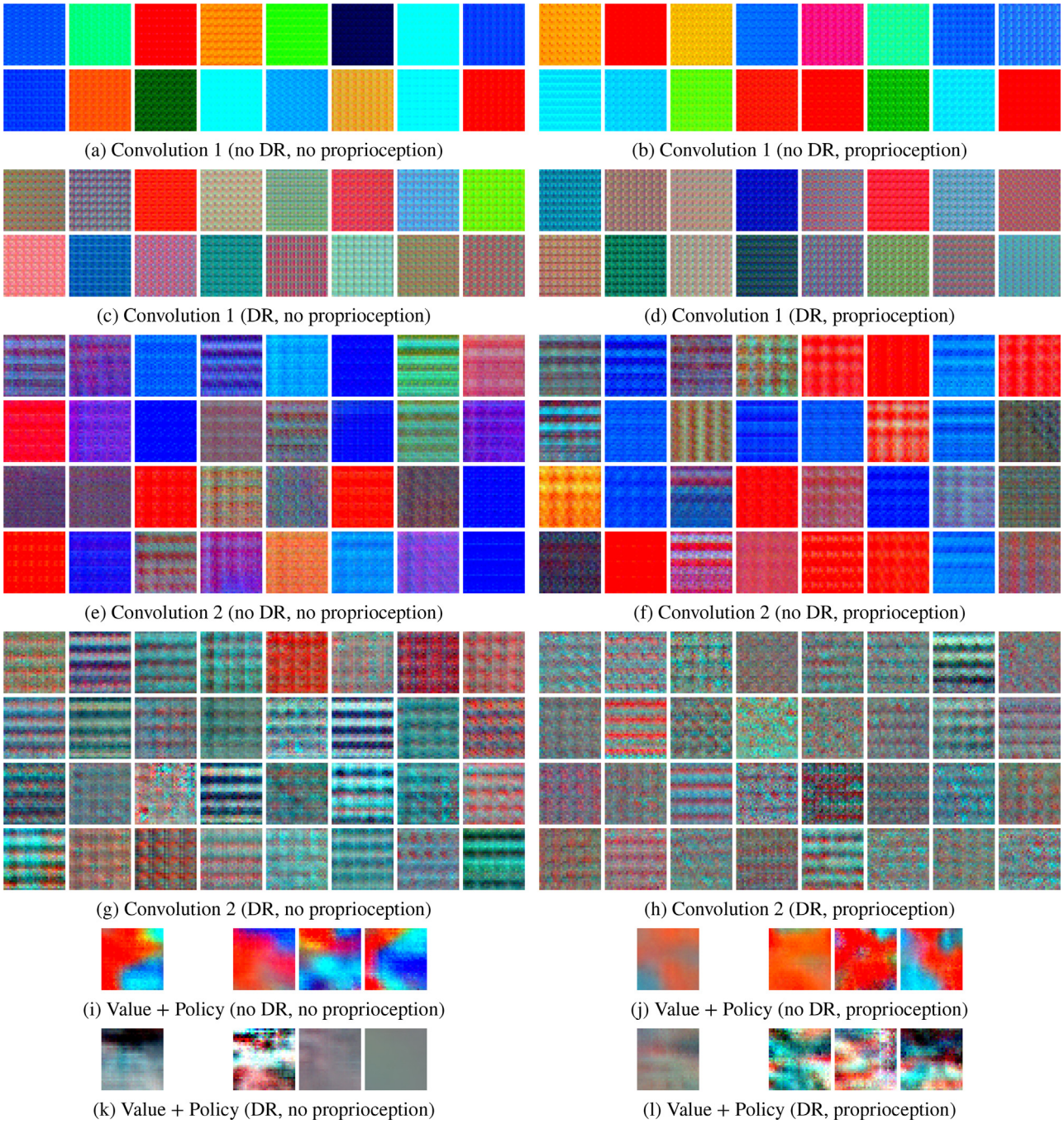


Fig. 15. Activation maximisation for trained Fetch agents: first convolutional layer (a-d); second convolutional layer (e-h); value and policy outputs (i-l). The best Fetch model was used for each training condition. Proprioceptive inputs and hidden state for value and policy visualisations are set to zero. Agents trained without DR have many red filters (the colour of the target) in the second layer (e, f), while agents trained with DR have more structured oriented red-blue filters (g, h). In comparison, the Jaco task induces more structured filters even without DR (see Fig. 16).

els, this difference indicates that DR causes a significant change in the final distribution of weights, with both larger weights and greater spatial structure.

4.4. Unit Ablations

Given access to the trained models, unit ablations allow us to perform a quantitative, white box analysis. To ablate units, we manually zero the activations of one of the output channels in either the first or second convolutional layers, iterating the process over every channel. We then re-evaluate the modified agents for each of the 8 training settings, using the agent with the best performance over all 5 seeds for each one (noting that the performance of the best Jaco agent trained with DR and without proprioception is significantly higher than the average, as reported in Table 4). These agents are tested on a single $x - y$ plane of the fixed test targets—the full 80 for Fetch, and 125 for Jaco—and both the standard visual and additive Gaussian noise test scenarios (see Subsection 3.4), as the latter is often used to mimic sensor noise in robotic learning tasks [33]. The results of the ablations are presented in Fig. 18.

We can make several observations from the plots in Fig. 18. Firstly, the Fetch agents are barely affected by unit ablations,

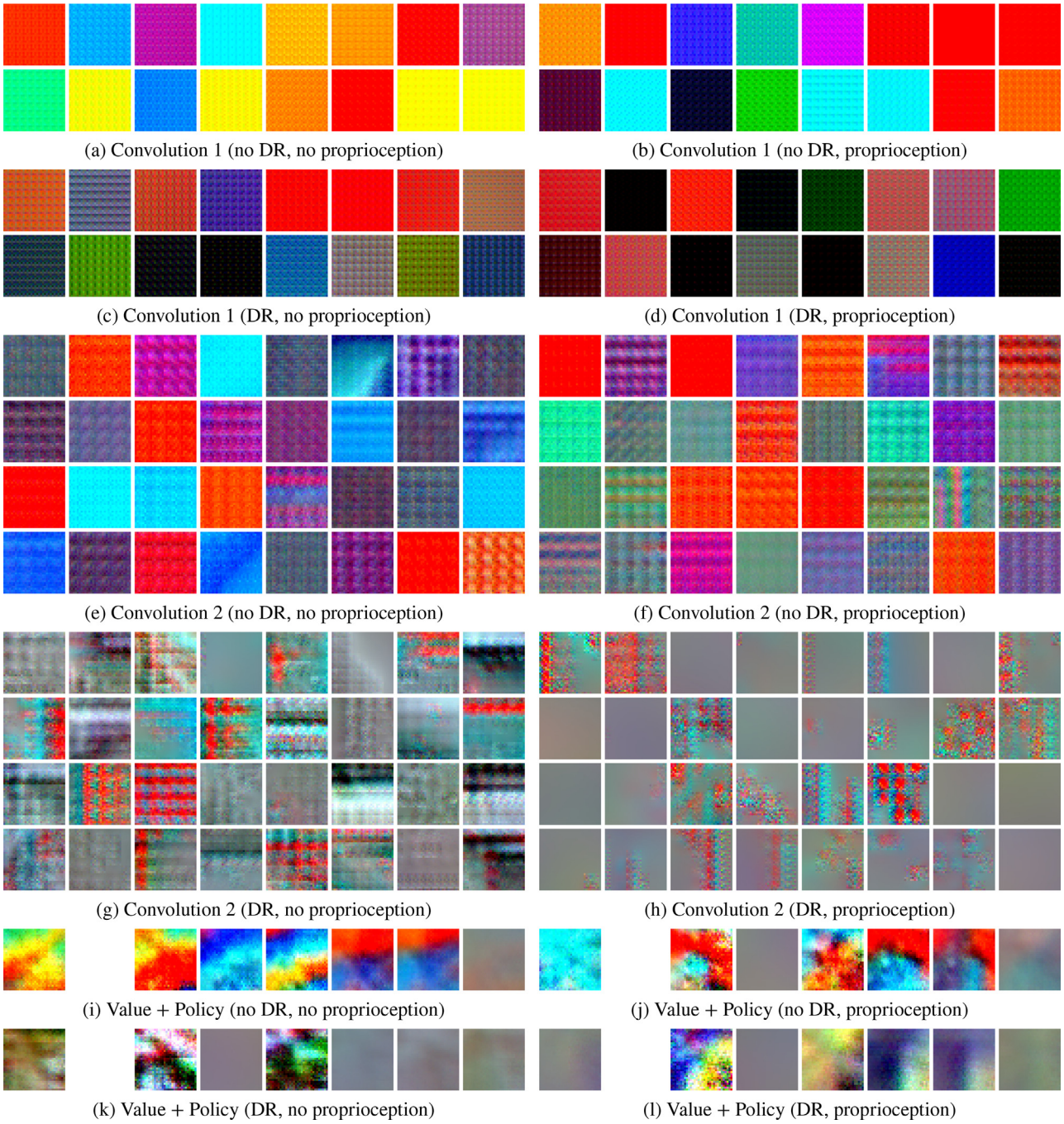


Fig. 16. Activation maximisation for trained Jaco agents: first convolutional layer (a-d); second convolutional layer (e-h); value and policy outputs (i-l). The best Jaco model was used for each training condition. Proprioceptive inputs and hidden state for value and policy visualisations are set to zero. All agents have colour-gradient filters in the second layer (e-h), indicating more visual complexity than needed for the Fetch task (Fig. 15).

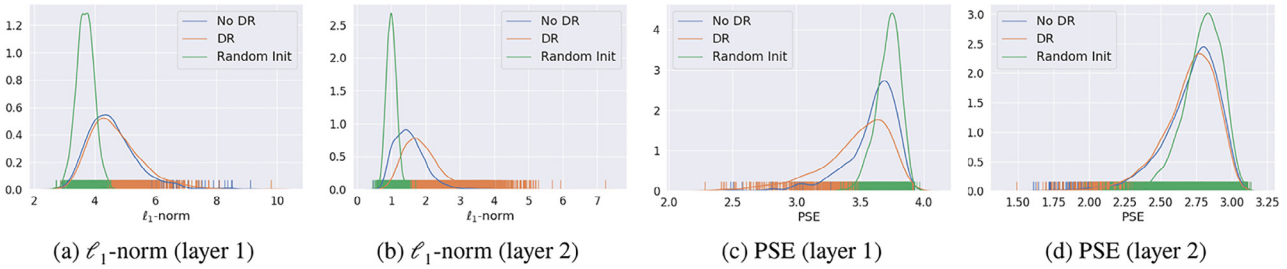


Fig. 17. Effect of DR on statistical and structural characterisations of convolutional filters, using all filters from all models, along with models with randomly initialised weights. This effect is layer-dependent, with a large change in ℓ_1-norm for layer 2, but not layer 1, and a relatively larger change in PSE for layer 1 as compared to layer 2.

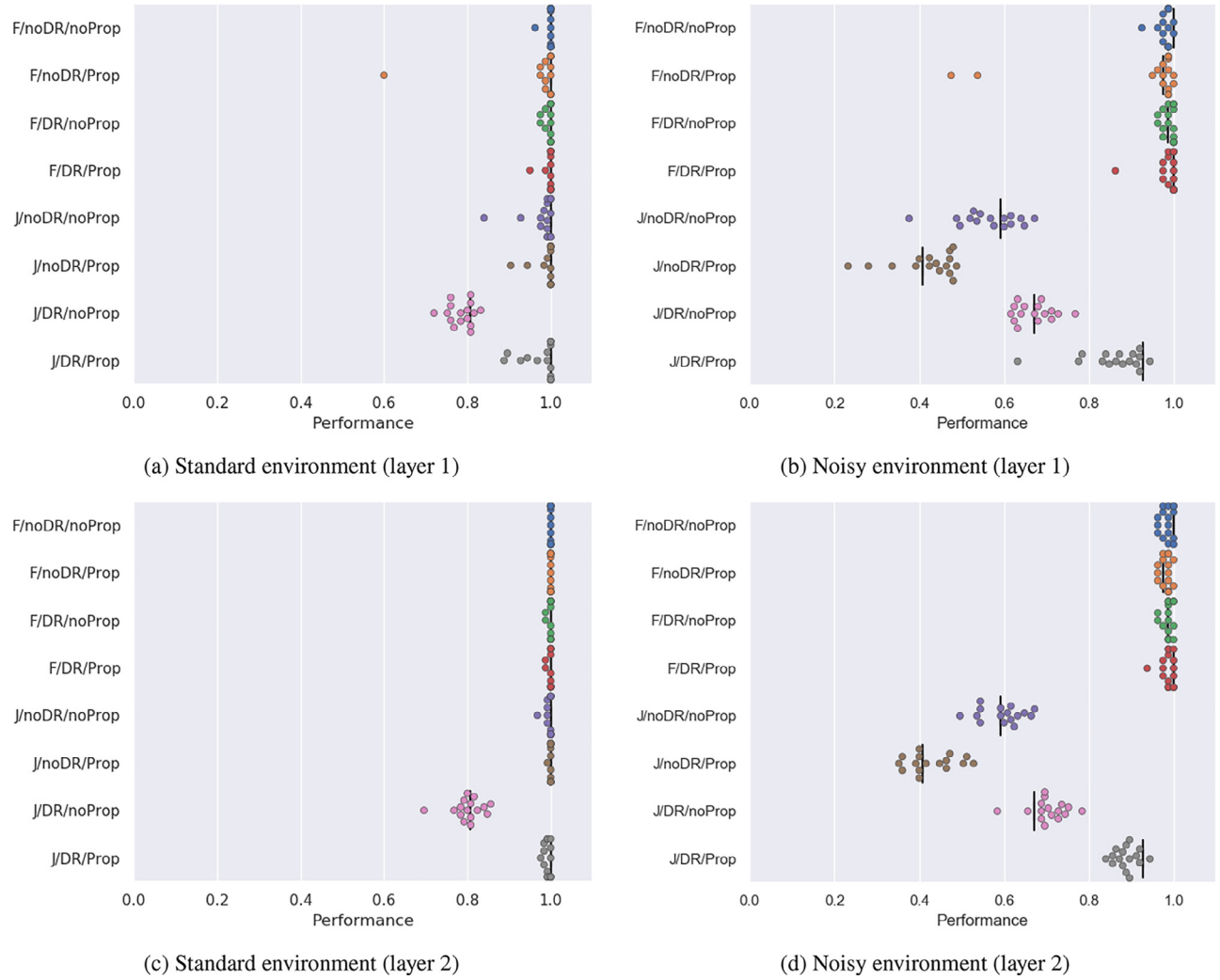


Fig. 18. Unit-wise ablation tests in two different visual test environments. Each point corresponds to one unit in layer 1 (a, b) or layer 2 (c, d), with the vertical bars representing baseline performance in the test environment. The training settings correspond to the Fetch (F) and Jaco (J) robots, whether additional proprioceptive inputs are available (Prop), and if DR was used. The best model was used for each training condition. Note that the Jaco agent trained with DR but without proprioception already has a lower base performance on the standard visuals than the other models (Table 4).

whereas they have varying effects on the Jaco agents. The higher variability for Jaco agents could be due to the increased complexity of the Jaco task (both in terms of extracting relevant information from the sensory inputs, and the difficulty of the actuation).

Secondly, there is a greater spread of values in layer 1 ablations (Fig. 18a,b) versus layer 2 (Fig. 18c,d). In particular, there appear to be a few highly important units in layer 1, resulting in highly skewed distributions. We believe this supports what we observe in the activation maximisation plots (Fig. 15 and Fig. 16), where there is a greater diversity in the layer 1 filters.

We observe a greater variability in the noisy environment (Fig. 18b,d). Intriguingly, ablations can improve performance beyond the baseline results, even for agents trained with DR—perhaps indicating a sensitivity to high-frequency noise. While a thorough discussion is beyond the scope of this work, we note that research on corruption and adversarial robustness in supervised learning settings could provide further insights on properties such as these [20,29].

One of our original hypotheses was that DR might force the learned representations to become more redundant—as quantified by reduced variability under unit ablation—but the results do not support this. Instead, the only clear outcome is that the baseline performance of the agents trained with DR is simply higher than that of the agents trained without DR in the noisy environment.

4.5. Layer Re-initialisation

Moving on from unit ablations, we now show the re-initialisation robustness, as well as the change in ℓ_∞- and ℓ_2-norms of the parameters of my trained Fetch and Jaco agents in Figs. 19 and 20, respectively. We use re-initialisation robustness to study the effect of task complexity (training with and without DR, and with and without proprioceptive inputs), but with networks of similar capacity. Our results are mostly in line with Zhang et al. [101]—despite continual changes in the weights during training (as measured by weight norms), the latter layers of the network are robust to re-initialisation after a few epochs of training, and in the case of the Fetch agents, the policy layer is robust to re-initialisation to the original set of weights. The agents trained with DR are less robust to re-initialisation during early-to-intermediate stages of training, implying that meaningful changes in the learned representations occur for longer periods within the entirety of training.

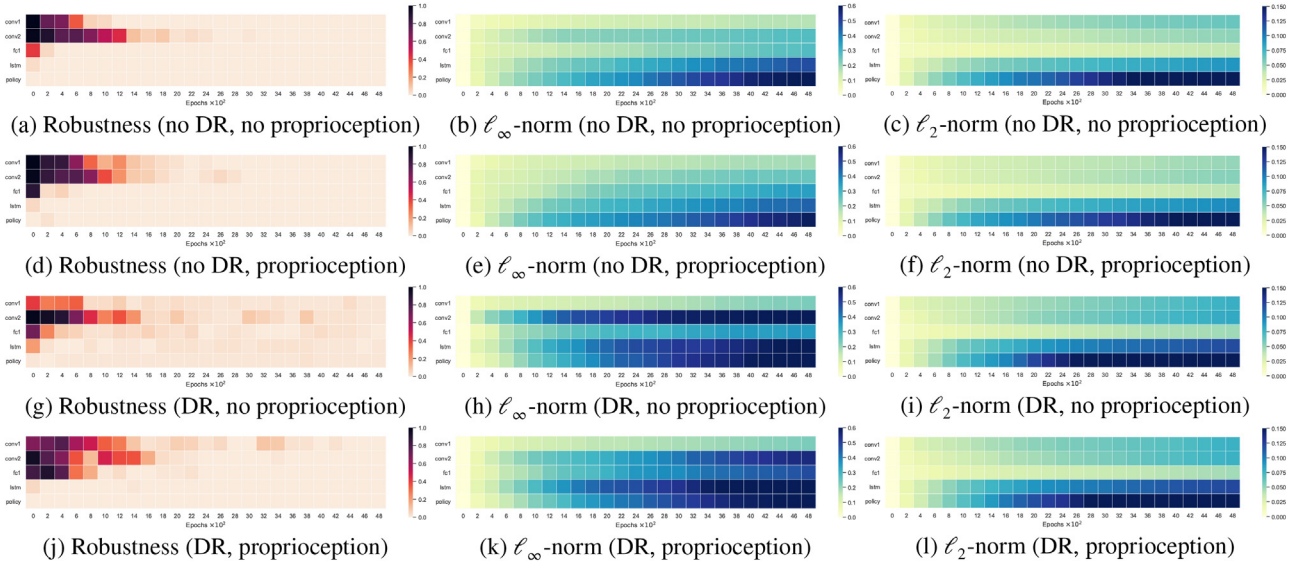


Fig. 19. Re-initialisation robustness (1 for complete failure, and 0 for complete success), and change in ℓ_∞- and ℓ_2-norm of parameters of Fetch agents trained with (g-l) and without (a-f) DR, and with (d-f, j-l) and without (a-c, g-i) proprioceptive inputs. Plots truncated to show detail during initial epochs. The best Fetch model was chosen for each training condition.

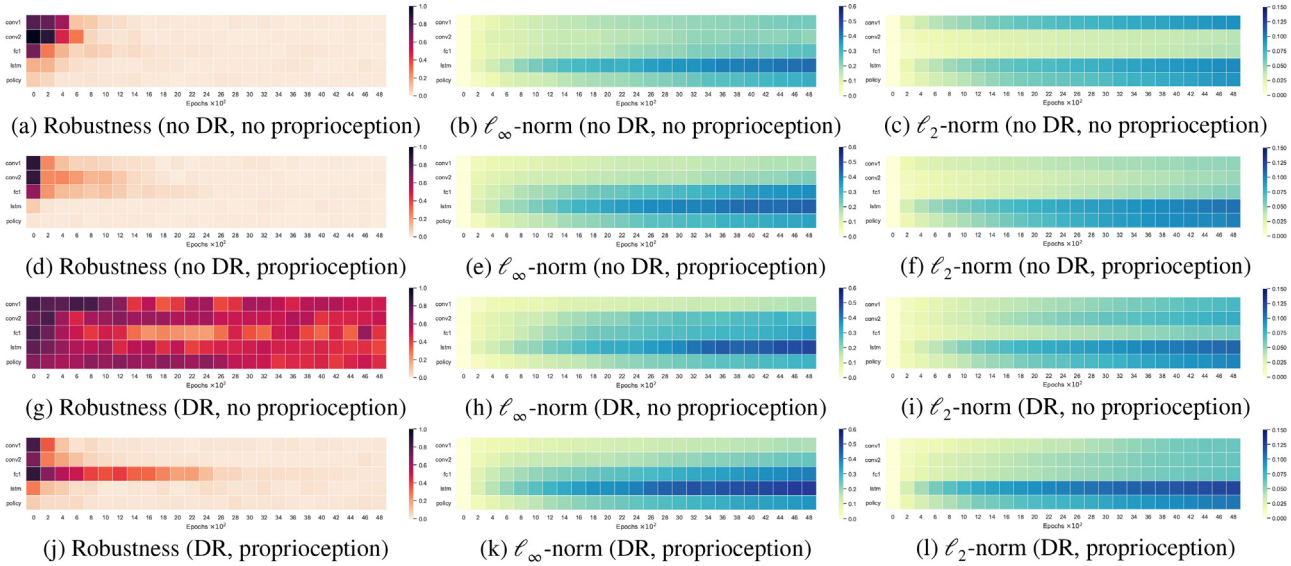


Fig. 20. Re-initialisation robustness (1 indicates complete failure, and 0 indicates complete success), and change in ℓ_∞- and ℓ_2-norm of parameters of Jaco agents trained with (g-l) and without (a-f) DR, and with (d-f, j-l) and without (a-c, g-i) proprioceptive inputs. Plots were truncated to show detail during initial epochs. The best Jaco model was chosen for each training condition. Note that the final failure rate of the best Jaco agent trained with DR and without proprioception on the standard environment is around 20%. The re-initialisation robustness plot for this condition (g) indicates that all layers are necessary and that training continues to improve performance in the epochs depicted and beyond.

All agents require the second convolutional layer—where more sophisticated target location occurs—to be trained for longer than the first layer. Additionally, all agents with DR require the first fully-connected layer (fc1) to be trained for longer than their corresponding non-DR counterparts. This is most noticeable for the Jaco agent trained with DR and proprioception (Fig. 20j)—which is the only agent that can self-localise in the absence of visual inputs (Subsection 3.4.3).

For nearly all agents, the recurrent layer is quite robust to re-initialisation to the original set of weights (despite noticeable changes in the weights as measured by both the ℓ_∞- ℓ_2-norms)—while this does not necessarily indicate that the agents do not utilise information over time, it does imply that training the recurrent connections is largely unnecessary for these tasks—a hypothesis we test further in Subsection 4.6.

4.6. Recurrent Ablation

To test how useful the LSTM is, we set the hidden and cell states to constant values and re-evaluated all models. Rather than naively zeroing the hidden states, which may not be representative of the values during rollouts, we instead use the empirical average values, as calculated over the normal execution of the models in testing. Table 6 shows the results of this ablation—there is a slight effect for agents trained without DR, but a significant effect for agents trained with DR. This indicates that recurrent processing may not

Table 6
Test performance of all models with standard operation versus constant (empirical average) hidden states. Checkmarks and crosses indicate enabling/disabling DR and proprioceptive inputs (Prop.), respectively. Statistics are calculated over all models (seeds) and test target locations.

Robot	DR	Prop.	Standard	Constant Hidden
Fetch	×	×	1.000±0.000	0.988±0.016
Fetch	×	✓	1.000±0.000	0.990±0.012
Fetch	✓	×	0.983±0.004	0.658±0.106
Fetch	✓	✓	0.997±0.002	0.838±0.051
Jaco	×	×	0.995±0.003	0.919±0.032
Jaco	×	✓	0.995±0.001	0.943±0.022
Jaco	✓	×	0.650±0.056	0.422±0.083
Jaco	✓	✓	0.991±0.004	0.746±0.060

be necessary for solving either robotic task without DR, but it is useful when DR is active. In terms of the strategies *learned* by the agents (Fig. 1), memory is sufficient for the agents trained without DR, but necessary for the agents trained with DR.

4.7. Entanglement

Finally, we consider the quantitative analysis of activations from different trained agents under the different training conditions. Table 7 contains the entanglement scores [17] of the different trained agents, calculated across the first 4 layers (not including the policy/value outputs); as with the original work, we use a 2D t-SNE [47] embedding for the activations. There are two noticeable trends. Firstly, the entanglement scores increase deeper into the network; this supports the notion that the different testing conditions can result in very different visual observations, but the difference between them diminishes as they are further processed by the networks. Secondly, the agents trained with DR have noticeably higher entanglement scores for each layer as compared to their equivalents trained without DR. This provides quantitative support for the hypothesis that DR makes agents largely invariant to nuisance visual factors (as opposed to the agents finding different strategies to cope with different visual conditions).

We can also qualitatively support these findings by visualising the same activations in 2D (Fig. 21). We use three common embedding techniques in order to show different aspects of the data. Firstly, we use PCA [65], which linearly embeds the data into dimensions which explain the most variance in the original data; as a result, linearly separable clusters have very different global characteristics. Secondly, we use t-SNE [47], which attempts to retain local structure in the data by calculating pairwise similarities between datapoints and creating a constrained graph layout in which distances in the original high-dimensional and the low-dimensional projection are preserved as much as possible. Thirdly, we use uniform manifold approximation and projection (UMAP) [51], which operates similarly to t-SNE at a high level, but better preserves global structure. Although it is possible to tune t-SNE [93], by default, UMAP better shows relevant global structure.

Table 7
Entanglement scores of different agents, for the first and second convolutional (conv.), fully-connected (FC) and LSTM layer, calculated over different testing conditions as classes (with $T = 0$). Checkmarks and crosses indicate enabling/disabling DR and proprioceptive inputs (Prop.), respectively.

Robot	DR	Prop.	1st Conv.	2nd Conv.	FC	LSTM
Fetch	×	×	0.11	0.30	0.56	0.68
Fetch	×	✓	0.12	0.30	0.45	0.45
Fetch	✓	×	0.23	0.38	0.62	0.92
Fetch	✓	✓	0.24	0.41	0.58	1.15
Jaco	×	×	0.14	0.29	0.52	0.68
Jaco	×	✓	0.11	0.08	0.43	0.66
Jaco	✓	×	0.41	0.37	0.55	0.73
Jaco	✓	✓	0.65	0.56	1.21	1.37

5. Discussion

The main goal of this study was to understand the representations and strategies (Fig. 1) learned by DRL agents in a simulated robotics task. To do so, we examined 8 training configurations simultaneously, resulting in novel insights on how the setup can influence what agents learn and how well they generalise.

One of the main axes of variation was the presence of DR. In line with prior work, DR improves performance across a wider distribution of testing conditions. In particular, our implementation of DR, which varied colours and textures, allowed generalisation to scenarios with "local" perturbations, but was more variable when more global changes were made to the setup; overall, agents trained with DR were nearly always more robust than agents trained without (Subsection 3.4). Adding DR to a task makes it more challenging to solve, in terms of sample complexity, although under the current experimental setup the models do not appear to require additional architectural depth, as all agents[6] are robust to re-initialisation of the final (policy) layer (Subsection 4.5). The application of entanglement [17], with respect to visual perturbations, shows that throughout the network the representations that are learned appear to be more invariant to these changes in the visuals, as the embeddings of representations from the different conditions have higher overlap (Subsection 4.7).

At the lower levels of the networks, DR results in significant changes in the ℓ_1-norms of the convolutional filters (Subsection 4.3), with more sophisticated feature detectors (Subsection 4.2). Supporting this, visualising the saliency maps of the agents shows that DR agents have more focused attention on task-specific features, such as the arm or ball (Subsection 4.1). Counter to initial expectations, we did not find that DR reduced the variability of performance under convolutional filter ablations—the agents merely have better baseline performance (Subsection 4.4). Deeper within the networks, we found that DR caused the agents to utilise the recurrent dynamics of the LSTM, whilst the agents trained without DR were hardly impacted by keeping their recurrent state constant (Subsection 4.6).

While we observe these general trends, it is notable that some of the results are not *a priori* as obvious. For example, even when provided with proprioceptive inputs, the Fetch agent trained without DR still uses its visual inputs for self-localisation (Subsection 4.1), although the addition of DR removes this observed effect. We believe that the relative simplicity of the Fetch reaching task—including both sensing and actuation—leads to less pronounced effects with DR (Subsection 3.4). The most unexpected finding was that the performance of the Jaco agent trained with DR and without proprioception dropped when shifting from DR visuals to the standard simulator visuals, demonstrating that DR can overfit (Subsection 3.3). With proprioception the gap disappears, which supports the idea that the form of input can have a significant effect on generalisation in agents [30]—meriting further investigation. While we focused on investigating 8 configurations in detail, orthogonal factors of variation, such as in the RL algorithm, or network architecture, could lead to other insights. Furthermore, examining the states encountered/policies learned during training can be more informative than simply looking at agents post-training [81].

This work has focused on understanding the effects of DR, but also has a dual purpose, which is to inform research in an opposite

[6] Except for the Jaco agent trained with DR and without proprioception, which has lower final performance under standard visuals.

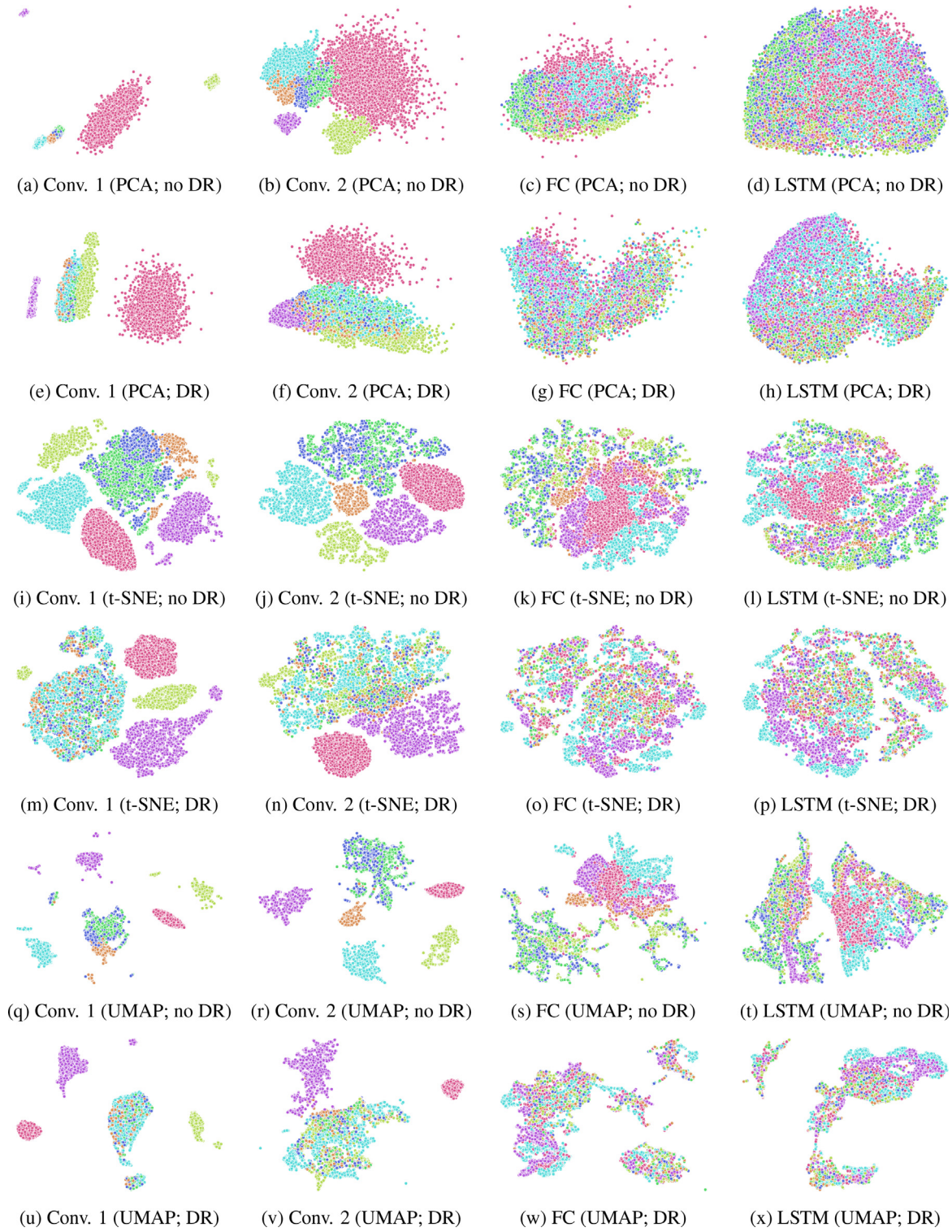


Fig. 21. Embeddings for trained Jaco agents with proprioceptive inputs with and without DR. Test conditions that are entangled with the normal observations (orange) typically include changing the colour (dark blue) or shape (green) of the target, shifting the camera (light blue), and, for DR, adding reflections (yellow). Global changes—adding Gaussian noise (red) or changing the global lighting (purple)—are the least entangled with the normal observations.

sense: in situations where DR is expensive or even infeasible, what approaches can we take to improve generalisation in sim2real transfer? If certain characteristics are positively correlated with DR training, explicitly enforcing them—without requiring the DR pipeline—may also lead to improved generalisation. For instance, Cobbe et al. [11] showed that standard regularisation techniques improve generalisation to a limited extent. Similarly, Pinto et al. [67] showed that adversarial training could improve the robustness of DRL policies. In line with this, enforcing greater spatial structure in the convolutional filters (Subsection 2.2.4), which is higher in agents trained with DR (Subsection 4.3), could be used as a novel regularisation objective.

A broader goal of these experiments was to assess the suitability of interpretability methods within the context of DRL. Beyond noticing limitations as discussed in previous works [38,49], there is a larger positive outcome from using a wide suite of interpretability techniques. Firstly, when used together they can cross-check the validity of each other's results. For example, supposedly "dead" units in the Jaco model with DR and proprioceptive inputs do in fact worsen performance when ablated (Subsection 4.2). Additionally, although the LSTM layer within DR agents are robust to re-initialisation at early stages of training (Subsection 4.5), the recurrent ablations show that the agents depend heavily on recurrent processing (Subsection 4.6). Secondly, the complementary answers these techniques provide lead to a better understanding of the model as a whole. For instance, unit ablations (Subsection 4.4) can be related to diversity in activation maximisation (Subsection 4.2), and entanglement (Subsection 4.7) can explain the generalisation of agents trained with DR (Subsection 3.4).

To conclude, we provide some recommendations for any practitioner aiming to study DRL agents:

- Use agents trained with test and control conditions. While it may be possible to interpret results absolutely, it is more reliable to interpret results relatively. An example is inferring what convolutional filters, as visualised using activation maximisation, are selective for.
- Do not assume that results generalise. While the Jaco agent trained with DR and without proprioception obeys some trends, it remains an outlier in many other aspects.
- Do not expect clear results. While some methods, such as entanglement, uncovered clear trends, others, such as unit ablations, did not. This does not mean that unit ablations are generally useless—one can imagine that if several units were highly important without DR and no units are individually important with DR, we would have found a significant trend using this method.
- If possible, use a range of interpretability methods. Uncertainty about results from one method may be resolved by results from another method.
- No single interpretability method is more useful than another—each type of method reveals different, complementary pieces of information.
- Finally, use interpretability methods before making claims about the strategies used by agents. While one may assume that an agent that performs well at a task on average is using an intelligent strategy, in all likelihood it may be using heuristics that fail to generalise [72].

CRediT authorship contribution statement

Tianhong Dai: Methodology, Software, Writing - original draft, Writing - review & editing. **Kai Arulkumaran:** Conceptualization, Software, Writing - original draft, Writing - review & editing. **Tamara Gerbert:** Software, Writing - review & editing. **Samyakh Tukra:** Software, Writing - review & editing. **Feryal Behbahani:** Writing - review & editing. **Anil Anthony Bharath:** Methodology, Supervision, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Detailed Test Scenarios

Table A1
Test performance of a single model with distractors locations varying over 9 different on the ground plane (Jaco) and table (Fetch). Checkmarks and crosses indicate enabling/disabling DR and proprioceptive inputs (Prop.), respectively. Statistics are calculated for the best model (seed), over all test target locations and all distractor locations.

Robot	DR	Prop.	Colours	Shapes
Fetch	×	×	0.993±0.004	0.567±0.047
Fetch	×	✓	0.937±0.027	0.507±0.047
Fetch	✓	×	0.974±0.004	0.639±0.051
Fetch	✓	✓	0.997±0.001	0.622±0.050
Jaco	×	×	0.772±0.058	0.567±0.047
Jaco	×	✓	0.808±0.044	0.507±0.047
Jaco	✓	×	0.652±0.022	0.639±0.051
Jaco	✓	✓	0.989±0.002	0.622±0.050

Table A2
Test performance of all models with different global illumination intensities; illumination is specificied as Fetch/Jaco. Checkmarks and crosses indicate enabling/disabling DR and proprioceptive inputs (Prop.), respectively. Statistics are calculated over all models (seeds) and test target locations.

Robot	DR	Prop.	0.0/0.1	0.1/0.3	0.2/0.5	0.3/0.7	0.4/0.9
Fetch	×	×	0.468±0.670	0.513±0.081	0.643±0.076	0.860±0.047	0.988±0.009
Fetch	×	✓	0.325±0.115	0.383±0.112	0.508±0.110	0.775±0.069	0.973±0.013
Fetch	✓	×	0.893±0.013	0.948±0.018	0.975±0.004	0.985±0.004	0.983±0.004
Fetch	✓	✓	0.983±0.006	0.990±0.002	0.995±0.003	0.995±0.003	0.995±0.003
Jaco	×	×	0.874±0.034	0.972±0.008	0.990±0.001	0.996±0.001	0.996±0.001
Jaco	×	✓	0.587±0.043	0.846±0.022	0.966±0.007	0.989±0.002	0.994±0.001
Jaco	✓	×	0.473±0.049	0.626±0.042	0.707±0.039	0.711±0.046	0.698±0.041
Jaco	✓	✓	0.442±0.018	0.713±0.013	0.946±0.012	0.986±0.002	0.994±0.002

Table A3
Test performance of all models with different main illumination directions; direction is specified as Fetch/Jaco. Checkmarks and crosses indicate enabling/disabling DR and proprioceptive inputs (Prop.), respectively. Statistics are calculated over all models (seeds) and test target locations.

Robot	DR	Prop.	0.0,0.0,-1.0/ 0.0,0.0,-1.3	−1.0,0.0,0.0/ −1.3,0.0,0.0	−1.0,0.0,-1.0/ −1.3,0.0,-1.3	0.0,-1.0,0.0/ 0.0,-1.3,-1.3	0.0,1.0,1.0/ −1.3,-1.3,-1.3
Fetch	×	×	1.000±0.000	0.020±0.004	0.593±0.095	0.518±0.091	0.563±0.066
Fetch	×	✓	1.000±0.000	0.068±0.041	0.493±0.074	0.453±0.079	0.440±0.066
Fetch	✓	×	0.983±0.004	0.338±0.117	0.850±0.087	0.763±0.097	0.765±0.075
Fetch	✓	✓	0.998±0.002	0.540±0.076	0.948±0.025	0.985±0.009	0.945±0.021
Jaco	×	×	0.995±0.003	0.818±0.022	0.650±0.121	0.805±0.044	0.922±0.015
Jaco	×	✓	0.995±0.001	0.852±0.019	0.930±0.029	0.855±0.023	0.922±0.011
Jaco	✓	×	0.650±0.056	0.608±0.044	0.455±0.070	0.506±0.068	0.529±0.074
Jaco	✓	✓	0.991±0.004	0.586±0.017	0.932±0.016	0.968±0.010	0.934±0.016

Table A4
Test performance of all models with different levels of camera translation. Checkmarks and crosses indicate enabling/disabling DR and proprioceptive inputs (Prop.), respectively. Statistics are calculated over all models (seeds) and test target locations.

Robot	DR	Prop.	−20 cm	−10 cm	0 cm	10 cm	20 cm
Fetch	×	×	0.013±0.004	0.168±0.101	1.000±0.000	0.163±0.091	0.008±0.004
Fetch	×	✓	0.043±0.027	0.075±0.039	1.000±0.000	0.035±0.010	0.000±0.000
Fetch	✓	×	0.268±0.042	0.625±0.056	0.983±0.004	0.433±0.056	0.093±0.040
Fetch	✓	✓	0.358±0.051	0.685±0.033	0.998±0.002	0.488±0.094	0.153±0.055
Jaco	×	×	0.107±0.030	0.570±0.065	0.995±0.003	0.714±0.022	0.394±0.055
Jaco	×	✓	0.464±0.062	0.826±0.026	0.995±0.001	0.757±0.022	0.399±0.040
Jaco	✓	×	0.175±0.025	0.265±0.049	0.650±0.056	0.321±0.030	0.141±0.034
Jaco	✓	✓	0.338±0.034	0.853±0.014	0.991±0.004	0.751±0.021	0.356±0.029

References

[1] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al., Learning dexterous in-hand manipulation, The International Journal of Robotics Research 39 (2018) 3–20.

[2] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, W. Zaremba, Hindsight experience replay, Advances in Neural Information Processing Systems (2017) 5055–5065.

[3] A.B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al., Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, Information Fusion 58 (2020) 82–115.

[4] K. Arulkumaran, M.P. Deisenroth, M. Brundage, A.A. Bharath, Deep reinforcement learning: A brief survey, IEEE Signal Processing Magazine 34 (2017) 26–38.

[5] A. Atrey, K. Clary, D. Jensen, Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning, in: International Conference on Learning Representations, 2020.

[6] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.R. Müller, W. Samek, On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation, PloS one 10 (2015) e0130140.

[7] A.G. Barto, R.S. Sutton, C.W. Anderson, Neuronlike adaptive elements that can solve difficult learning control problems, IEEE Transactions on Systems, Man, and Cybernetics (1983) 834–846.

[8] Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al., 2019. DotA 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680.

[9] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W., 2016. OpenAI Gym. arXiv preprint arXiv:1606.01540.

[10] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, D. Fox, Closing the sim-to-real loop: Adapting simulation randomization with real world experience, in: International Conference on Robotics and Automation, 2019, pp. 8973–8979.

[11] K. Cobbe, O. Klimov, C. Hesse, T. Kim, J. Schulman, Quantifying generalization in reinforcement learning, International Conference on Machine Learning (2019) 1282–1289.

[12] M. Craven, J.W. Shavlik, Extracting tree-structured representations of trained networks, Advances in Neural Information Processing Systems (1996) 24–30.

[13] Doshi-Velez, F., Kim, B., 2017. Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608.

[14] Elman, J.L., 1989. Representation and structure in connectionist models. Technical Report. Univ. of California at San Diego, La Jolla Center For Research In Language.

[15] Erhan, D., Bengio, Y., Courville, A., Vincent, P., 2009. Visualizing higher-layer features of a deep network. Technical Report 1341. University of Montreal.

[16] A.A. Freitas, Comprehensible classification models: a position paper, Explorations Newsletter 15 (2014) 1–10.

[17] N. Frosst, N. Papernot, G. Hinton, Analyzing and improving representations with the soft nearest neighbor loss, in: International Conference on Machine Learning, 2019, pp. 2012–2020.

[18] R. Geirhos, J.H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, F.A. Wichmann, Shortcut learning in deep neural networks, Nature Machine Intelligence 2 (2020) 665–673.

[19] F.A. Gers, J. Schmidhuber, F. Cummins, Learning to forget: Continual prediction with lstm, Neural Computation 12 (2000) 2451–2471.

[20] Gilmer, J., Ford, N., Carlini, N., Cubuk, E., 2019. Adversarial examples are a natural consequence of test error in noise, in: International Conference on Machine Learning, pp. 2280–2289.

[21] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587.

[22] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.

[23] S. Greydanus, A. Koul, J. Dodge, A. Fern, Visualizing and understanding atari agents, International Conference on, Machine Learning (2018) 1787–1796.

[24] S. Gu, E. Holly, T. Lillicrap, S. Levine, Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, in: International Conference on Robotics and Automation, 2017, pp. 3389–3396.

[25] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, D. Pedreschi, A survey of methods for explaining black box models, ACM Computing Surveys 51 (2018) 93.

[26] P. Hamel, D. Eck, Learning features from music audio with deep belief networks, in: ISMIR, 2010, pp. 339–344.

[27] L.K. Hansen, P. Salamon, Neural network ensembles, IEEE Transactions on Pattern Analysis & Machine Intelligence (1990) 993–1001.

[28] S.J. Hanson, L.Y. Pratt, Comparing biases for minimal network construction with back-propagation, Advances in Neural Information Processing Systems (1989) 177–185.

[29] D. Hendrycks, T. Dietterich, Benchmarking neural network robustness to common corruptions and perturbations, in: International Conference on Learning Representations, 2019.

[30] Hill, F., Lampinen, A., Schneider, R., Clark, S., Botvinick, M., McClelland, J.L., Santoro, A., 2019. Emergent systematic generalization in a situated agent. arXiv preprint arXiv:1910.00571.

[31] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (1997) 1735–1780.

[32] Ilyas, A., Engstrom, L., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., Madry, A., 2018. Are deep policy gradient algorithms truly policy gradient algorithms? arXiv preprint arXiv:1811.02553.

[33] N. Jakobi, P. Husbands, I. Harvey, Noise and the reality gap: The use of simulation in evolutionary robotics, in: European Conference on Artificial Life, 1995, pp. 704–720.

[34] S. James, A.J. Davison, E. Johns, Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task, in: Conference on Robot Learning, 2017, pp. 334–343.

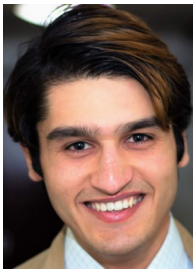
[35] Jaunet, T., Vuillemot, R., Wolf, C., 2020. Drlviz: Understanding decisions and memory in deep reinforcement learning, in: Eurographics Conference on Visualization.
[36] Justesen, N., Torrado, R.R., Bontrager, P., Khalifa, A., Togelius, J., Risi, S., 2018. Procedural level generation improves generality of deep reinforcement learning. arXiv preprint arXiv:1806.10729.
[37] M.A. Khabou, P.D. Gader, H. Shi, Entropy optimized morphological shared-weight neural networks, Optical Engineering 38 (1999) 263–274.
[38] Kindermans, P.J., Schütt, K., Müller, K.R., Dähne, S., 2016. Investigating the influence of noise and distractors on the interpretation of neural networks, in: NeurIPS Interpretable Machine Learning in Complex Systems Workshop..
[39] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: International Conference on Learning Representations, 2015.
[40] D. Kragic, M. Vincze, Vision for robotics, Foundations and Trends in Robotics 1 (2009) 1–78.
[41] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in Neural Information Processing Systems (2012) 1097–1105.
[42] Krkic, M., Roberts, S.J., Rezek, I., Jordan, C., 1996. Eeg-based assessment of anaesthetic depth using neural networks.
[43] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (1998) 2278–2324.
[44] Y. LeCun, L. Bottou, G.B. Orr, K.R. Müller, Efficient backprop, Neural Networks: Tricks of the Trade (1998) 9–50.
[45] S. Levine, C. Finn, T. Darrell, P. Abbeel, End-to-end training of deep visuomotor policies, Journal of Machine Learning Research 17 (2016) 1334–1373.
[46] Luo, J.H., Wu, J., 2017. An entropy-based pruning method for CNN compression. arXiv preprint arXiv:1706.05791.
[47] Maaten, L.v.d., Hinton, G., 2008. Visualizing data using t-sne. Journal of Machine Learning Research 9, 2579–2605.
[48] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: International Conference on Learning Representations, 2018.
[49] A. Mahendran, A. Vedaldi, Understanding deep image representations by inverting them, in: Conference on Computer Vision and Pattern Recognition, 2015, pp. 5188–5196.
[50] J. Martinez-Gomez, A. Fernandez-Caballero, I. Garcia-Varea, L. Rodriguez, C. Romero-Gonzalez, A taxonomy of vision systems for ground mobile robots, International Journal of Advanced Robotic Systems 11 (2014) 111.
[51] McInnes, L., Healy, J., Melville, J., 2018. Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426.
[52] Meyes, R., Schneider, M., Meisen, T., 2020. How do you act? an empirical study to understand behavior of deep reinforcement learning agents. arXiv preprint arXiv:2004.03237.
[53] Misra, H., Ikbal, S., Bourlard, H., Hermansky, H., 2004. Spectral entropy based feature for robust asr, in: International Conference on Acoustics, Speech, and Signal Processing, pp. I–193.
[54] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: International Conference on Machine Learning, 2016, pp. 1928–1937.
[55] 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 (2015) 529.
[56] N. Morch, U. Kjems, L.K. Hansen, C. Svarer, I. Law, B. Lautrup, S. Strother, K. Rehm, Visualization of neural networks using saliency maps, in: International Conference on Neural Networks, 1995, pp. 2085–2090.
[57] Mordvintsev, A., Olah, C., Tyka, M., 2015. Inceptionism: Going deeper into neural networks.
[58] A. Nguyen, J. Yosinski, J. Clune, Deep neural networks are easily fooled: High confidence predictions for unrecognizable images, in: Conference on Computer Vision and Pattern Recognition, 2015, pp. 427–436.
[59] A. Odena, V. Dumoulin, C. Olah, Deconvolution and checkerboard artifacts, Distill. (2016).
[60] Olah, C., Mordvintsev, A., Schubert, L., 2017. Feature visualization. Distill..
[61] M.L. Olson, R. Khanna, L. Neal, F. Li, W.K. Wong, Counterfactual state explanations for reinforcement learning agents via generative deep learning, Artificial Intelligence 295 (2021) 103455.
[62] Packer, C., Gao, K., Kos, J., Krähenbühl, P., Koltun, V., Song, D., 2018. Assessing generalization in deep reinforcement learning. arXiv preprint arXiv:1810.12282.
[63] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, in: International Conference on Machine Learning, 2013, pp. 1310–1318.
[64] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., PyTorch: An imperative style, high-performance deep learning library, Advances in Neural Information Processing Systems (2019) 8026–8037.
[65] K. Pearson, Liii. on lines and planes of closest fit to systems of points in space, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 2 (1901) 559–572.
[66] X.B. Peng, M. Andrychowicz, W. Zaremba, P. Abbeel, Sim-to-real transfer of robotic control with dynamics randomization, in: International Conference on Robotics and Automation, 2018, pp. 1–8.
[67] L. Pinto, J. Davidson, R. Sukthankar, A. Gupta, Robust adversarial reinforcement learning, International Conference on, Machine Learning (2017) 2817–2826.
[68] Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al., 2018. Multi-goal reinforcement learning: Challenging robotics environments and request for research. arXiv preprint arXiv:1802.09464.
[69] N. Puri, S. Verma, P. Gupta, D. Kayastha, S. Deshmukh, B. Krishnamurthy, S. Singh, Explain your move: Understanding agent actions using specific and relevant feature attribution, in: International Conference on Learning Representations, 2020.
[70] P.E. Rauber, S.G. Fadel, A.X. Falcao, A.C. Telea, Visualizing the hidden activity of artificial neural networks, IEEE Transactions on Visualization and Computer Graphics 23 (2017) 101–110.
[71] M.T. Ribeiro, S. Singh, C. Guestrin, Why should i trust you?: Explaining the predictions of any classifier, in: International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1135–1144.
[72] A. Ruderman, R. Everett, B. Sikder, H. Soyer, J. Uesato, A. Kumar, C. Beattie, P. Kohli, Uncovering surprising behaviors in reinforcement learning via worst-case analysis, in: International Conference on Learning Representations, 2019.
[73] C. Rupprecht, C. Ibrahim, C.J. Pal, Finding and visualizing weaknesses of deep reinforcement learning agents, in: International Conference on Learning Representations, 2020.
[74] A.A. Rusu, M. Večerík, T. Rothörl, N. Heess, R. Pascanu, R. Hadsell, Sim-to-real robot learning from pixels with progressive nets, in: Conference on Robot Learning, 2017, pp. 262–270.
[75] F. Sadeghi, S. Levine, Cad2rl: Real single-image flight without a single real image, in: Robotics: Science and Systems, 2017.
[76] R. Salakhutdinov, G. Hinton, Learning a nonlinear embedding by preserving class neighbourhood structure, Artificial Intelligence and Statistics (2007) 412–419.
[77] A.M. Saxe, J.L. McClelland, S. Ganguli, Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, in: International Conference on Learning Representations, 2014.
[78] Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P., 2015. High-dimensional continuous control using generalized advantage estimation. arXiv preprint arXiv:1506.02438.
[79] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.
[80] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: Visual explanations from deep networks via gradient-based localization, in: International Conference on Computer Vision, 2017, pp. 618–626.
[81] P. Sequeira, M. Gervasio, Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations, Artificial Intelligence 288 (2020) 103367.
[82] C. Shorten, T.M. Khoshgoftaar, A survey on image data augmentation for deep learning, Journal of Big Data 6 (2019) 60.
[83] A. Shrikumar, P. Greenside, A. Kundaje, Learning important features through propagating activation differences, in: International Conference on Machine Learning, 2017, pp. 3145–3153.
[84] V. Srinivasan, C. Eswaran, N. Sriraam, Artificial neural network based epileptic detection using time-domain and frequency-domain features, Journal of Medical Systems 29 (2005) 647–660.
[85] P. Sturmfels, S. Lundberg, S.I. Lee, Visualizing the impact of feature attribution baselines, Distill 5 (2020) e21.
[86] Such, F., Madhavan, V., Liu, R., Wang, R., Castro, P., Li, Y., Schubert, L., Bellemare, M.G., Clune, J., Lehman, J., 2018. An atari model zoo for analyzing, visualizing, and comparing deep reinforcement learning agents, in: NeurIPS Deep RL Workshop.
[87] M. Sundararajan, A. Taly, Q. Yan, Axiomatic attribution for deep networks, International Conference on, Machine Learning (2017) 3319–3328.
[88] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, MIT press, 2018.
[89] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, P. Abbeel, Domain randomization for transferring deep neural networks from simulation to the real world, in: International Conference on Intelligent Robots and Systems, 2017, pp. 23–30.
[90] E. Todorov, T. Erez, Y. Tassa, Mujoco: A physics engine for model-based control, in: International Conference on Intelligent Robots and Systems, 2012, pp. 5026–5033.
[91] Tzeng, E., Devin, C., Hoffman, J., Finn, C., Peng, X., Levine, S., Saenko, K., Darrell, T., 2015. Towards adapting deep visuomotor representations from simulated to real environments. arXiv preprint arXiv:1511.07111 2.
[92] J. Uesato, A. Kumar, C. Szepesvari, T. Erez, A. Ruderman, K. Anderson, N. Heess, P. Kohli, et al., Rigorous agent evaluation: An adversarial approach to uncover catastrophic failures, in: International Conference on Learning Representations, 2019.
[93] M. Wattenberg, F. Viégas, I. Johnson, How to use t-sne effectively, Distill 1 (2016) e2.
[94] D. Wierstra, A. Foerster, J. Peters, J. Schmidhuber, Solving deep memory pomdps with recurrent policy gradients, in: International Conference on Artificial Neural Networks, 2007, pp. 697–706.
[95] R.J. Williams, J. Peng, Function optimization using connectionist reinforcement learning algorithms, Connection Science 3 (1991) 241–268.

[96] Witty, S., Lee, J.K., Tosch, E., Atrey, A., Littman, M., Jensen, D., 2018. Measuring and characterizing generalization in deep reinforcement learning. arXiv preprint arXiv:1812.02868.
[97] D.L. Yamins, J.J. DiCarlo, Using goal-driven deep learning models to understand sensory cortex, Nature Neuroscience 19 (2016) 356.
[98] T. Zahavy, N. Ben-Zrihem, S. Mannor, Graying the black box: Understanding dqns, International Conference on, Machine Learning (2016) 1899–1908.
[99] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: European Conference on Computer Vision, 2014, pp. 818–833.
[100] Zhang, A., Ballas, N., Pineau, J., 2018. A dissection of overfitting and generalization in continuous reinforcement learning. arXiv preprint arXiv:1806.07937.
[101] Zhang, C., Bengio, S., Singer, Y., 2019. Are all layers created equal?, in: ICML Deep Phenomena Workshop.
[102] Zhao, C., Siguad, O., Stulp, F., Hospedales, T.M., 2019. Investigating generalisation in continuous deep reinforcement learning. arXiv preprint arXiv:1902.07015.
[103] B. Zheng, W. Qian, L.P. Clarke, Digital mammography: mixed feature neural network with spectral entropy decision for detection of microcalcifications, IEEE Transactions on Medical Imaging 15 (1996) 589–597.
[104] Y. Zhu, R. Mottaghi, E. Kolve, J.J. Lim, A. Gupta, L. Fei-Fei, A. Farhadi, Target-driven visual navigation in indoor scenes using deep reinforcement learning, in: International Conference on Robotics and Automation, 2017, pp. 3357–3364.

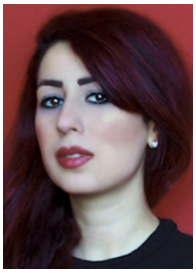
Tianhong Dai is a Ph.D. candidate in the Department of Bioengineering at Imperial College London. He received a B.Eng. (Hons.) in Electronic and Communication Engineering from the University of Liverpool in 2015 and an M.Sc. in Communication and Signal Processing from Imperial College London in 2016. He has been a Research Intern in Tencent AI Lab and Huawei Noah's Ark Lab. His research interests include deep reinforcement learning and computer vision.

Kai Arulkumaran is a Research Team Lead at Araya, Inc. He received a B.A. in Computer Science at the University of Cambridge in 2012, and a Ph.D. in Bioengineering at Imperial College London in 2020. He has been a Research Intern in Microsoft Research, Twitter, Facebook AI Research, DeepMind and NNAISENSE. His research interests are in deep learning and reinforcement learning.

Tamara Gerbert is a Machine Intelligence Engineer at Cambrium. She received a B.Sc. in Neuroscience from Kings College London in 2019, and a M.Res. in Neurotechnology from Imperial College London in 2020. Her research interests lie in the symbiotic relationship of neuroscience and deep learning.



Samyakh Tukra is a Ph.D. candidate in the Hamlyn Centre for Robotic Surgery at Imperial College London. He received an M.Eng. in Medical Engineering at Cardiff University in 2017, and an M.Sc. in Human and Biological Robotics at Imperial College London in 2018. His research focus is in computer vision, deep learning and mixed reality.



Feryal Behbahani is a Senior Research Scientist at DeepMind. Previously, she was a Lead Research Scientist at Latent Logic (now part of Waymo) where she worked on reinforcement learning for learning human-like behaviour for robotics, autonomous driving and games. She received her MSc in Artificial Intelligence from Imperial College London followed by a PhD focused on algorithms employed by the human brain for object representation, inference and control.

Anthony Bharath is a Professor in the Department of Bioengineering at Imperial College London and a Fellow of the Institution of Engineering and Technology. He received a B.Eng. in Electronic and Electrical Engineering from University College London in 1988, and a Ph.D. in Signal Processing from Imperial College London in 1993. He was an academic visitor in the Signal Processing Group at the University of Cambridge in 2006. He is a co-founder of Cortexica Vision Systems. His research interests are in deep architectures for visual inference.