Domain Shifts in Reinforcement Learning: Identifying Disturbances in Environments

Tom Haider[1*†], **Felippe Schmoeller Roza**[1†], **Dirk Eilers**[1], **Karsten Roscher**[1] and **Stephan Günnemann**[2]

[1]Fraunhofer IKS

[2]Technical University of Munich

Abstract

A significant drawback of End-to-End Deep Reinforcement Learning (RL) systems is that they return an action no matter what situation they are confronted with. This is true even for situations that differ entirely from those an agent has been trained for. Although crucial in safety-critical applications, dealing with such situations is inherently difficult. Various approaches have been proposed in this direction, such as robustness, domain adaption, domain generalization, and out-of-distribution detection. In this work, we provide an overview of approaches towards the more general problem of dealing with disturbances to the environment of RL agents and show how they struggle to provide clear boundaries when mapped to safety-critical problems. To mitigate this, we propose to formalize the changes in the environment in terms of the Markov Decision Process (MDP), resulting in a more formal framework when dealing with such problems. We apply this framework to an example real-world scenario and show how it helps to isolate safety concerns.

1 Introduction

Deep Reinforcement Learning (RL) has been successfully applied to many different domains, achieving state-of-the-art performance on a wide range of high-dimensional control problems [Mnih *et al.*, 2013; Silver *et al.*, 2017; Levine *et al.*, 2016; Lillicrap *et al.*, 2015]. However, RL systems are still not commonly applied to safety-critical scenarios. A relevant factor for this is that RL agents are highly sensitive to disturbances in the environment they were trained in [Eysenbach and Levine, 2021]. Even slight changes in the specifications of the environment can already lead to severe malfunctions or failures of many RL-based systems.

Take as an example a humanoid robot that is trained to walk on a rigid body surface, such as concrete or wood. While state-of-the-art algorithms can be deployed to train policies that solve this task [Haarnoja *et al.*, 2018; Schulman *et al.*, 2017], it is still difficult to ensure that the robot will function properly if tested on a softer surface such as grass or sand. Even though the task essentially stays the same, most RL algorithms struggle with disturbances like these, leading to potential failures.

In the classic supervised or unsupervised learning paradigm, we could frame this problem as out-of-distribution (OOD). In that context, training samples are typically assumed to be Independent and Identically Distributed (IID) whereas for OOD samples, the 'identically' assumption is violated. Strictly applying this definition we could interpret soft and rigid surfaces to be sampled from non-identical distribuions and thus a normal operation of the robot can not be expected. However, solid and soft surfaces could theoretically come from the same distribution (the distribution of surfaces) but during training, only a small part of the distribution was experienced (the part of the distribution where surfaces are solid). In this case, this is not an OOD instance but rather a generalization problem. The robot should be able to 'generalize' to the entire distribution. In Control Theory and RL, this is also known as 'robustness to external influences'. Note that even this simple example can become ambiguous and one could think that OOD and robustness are dealing with the same phenomena.

In this work, we provide insights towards a better formalization of disturbances to the environment of RL agents. We propose a simple but powerful decomposition of any given disturbance into its aspects of the Markov Decision Process (MDP). We compare this decomposition to a set of commonly used terms, that also describe disturbances to the environment and, by that, aim to disambiguate any commonalities and differences. To illustrate our method, we apply it to a series of potential disturbances in an exemplary real-world scenario. We show that this method helps at identifying potential risks that can occur during the deployment of RL agents. We see this as a necessary step towards guaranteeing the safety of RL agents in non-idealistic problem settings. We do not attempt at building a unifying framework, that combines all existing work in this direction. Rather, we want to provide an alternative view on disturbances in the environments of RL agents, that should help when approaching this problem and allow to compare approaches from different related fields.

*Contact Author (firstname.lastname@iks.fraunhofer.de)

†equal contributions

2 Related Work

In this section we formalize the RL framework and present different concepts related to the problem of dealing with changes from the training domain to the testing domain.

2.1 Preliminaries

In RL, we consider an agent that sequentially interacts with an environment modeled as a Markov Decision Process (MDP) [Puterman, 2014]. An MDP is a tuple $\mathcal{M} := (S, A, R, P, \mu_0)$, where S is the set of states, A is the set of actions, $R : S \times A \times S \mapsto \mathbb{R}$ is the reward function, $P : S \times A \times S \mapsto [0, 1]$ is the transition probability function which describes the system dynamics, where $P(s_{t+1}|s_t, a_t)$ is the probability of transitioning to state s_{t+1}, given that the previous state was s_t and the agent took action a_t, and $\mu_0 : S \mapsto [0, 1]$ is the starting state distribution. At each timestep the agent observes the current state $s_t \in S$, takes an action $a_t \in A$, transitions to the next state s_{t+1} drawn from the distribution $P(s_t, a_t)$, and receives a reward $R(s_t, a_t, s_{t+1})$.

2.2 Distributional Shift and OOD

Distributional shift describes changes in data distributions. When only the input distribution changes while the conditional distribution of outputs remains unchanged, it is called covariate shift. When the testing distribution differs from the training distribution, machine learning systems may not only exhibit poor performance but also wrongly assume that their performance is good [Amodei *et al.*, 2016].

Out-of-distribution (OOD) defines the data that lay outside the training distribution. More formally, consider that P_X and Q_X are two distinct data distributions defined on the input space $\mathcal{X}$. If a model is trained on a dataset drawn from the distribution P_X, samples from this set will be defined as in-distribution while Q_X is part of the OOD domain [Liang *et al.*, 2017].

Although formally defined, mapping real in- and out-of-distributions into training and testing sets is challenging for high-dimensional feature spaces. Using datasets for testing that contain classes different from those composing the training set is an established practice in image classification [DeVries and Taylor, 2018]. Despite being an interesting approach, it can induce a bias towards a chosen OOD set and can, nevertheless, only be applied to classification problems.

In RL, there is not a consensus on how to frame OOD into practical examples yet. [Sedlmeier *et al.*, 2019] provide one of the first publications on this topic and define OOD as every state-action tuple not experienced during training. The difficulty in generating OOD scenarios due to a lack of controlled and reproducible datasets in RL is also mentioned. [Mendonca *et al.*, 2020] use a more relaxed interpretation, defining OOD as tasks never seen by the agent.

2.3 Transfer Learning, Domain Adaptation and Domain Randomization

The objective of transfer learning (TL) is to learn a task $T_{\mathcal{B}}$ that belongs to a target domain $\mathcal{B}$, by utilizing experience (prior knowledge) from some source task $T_{\mathcal{A}}$ from source domain $\mathcal{A}$, i.e. to transfer knowledge from $\mathcal{A}$ to $\mathcal{B}$ [Yosinski *et al.*, 2014].

Domain adaptation (DA) is a sub-field of transfer learning. DA can be defined as the capability to deploy a model trained in one or more source domains into a different target domain. While DA encompasses cases where the source and target domains have the same feature space, TL includes cases where the feature space of the target domain differs from the source domain [Redko *et al.*, 2019]. Typically, invariance is assumed, meaning that features that differ between two domains are irrelevant for the task. [Tzeng *et al.*, 2014] use this assumption to train a CNN architecture to learn a domain invariant feature representation that transfers well between two domains. [Ganin *et al.*, 2016] deploy domain adversarial training to learn features that are both discriminative for the main learning task on the source domain and indiscriminate with respect to the shift between the domains.

In domain randomization, the source domain is manipulated at random over a set of parameters in order to train a more generalizable model. [Sadeghi and Levine, 2016] and [Tobin *et al.*, 2017] show that randomizing the rendering settings for a simulator can be used to train policies that generalize to the real world without requiring the simulator to be extremely realistic. [Rajeswaran *et al.*, 2016] show that randomizing over a set of system dynamics can lead to more robust policies that can generalize to a broad range of possible target domains, including effects that are not modeled in the training distribution.

2.4 Novelty Detection and Intrinsic Motivation

Novelty is a known mechanism used by human beings to explore and learn new things. Many studies in the neuroscience field bring the converging evidence that novelty can activate the reward region of the brain in both humans and animals, playing an important role in their reinforcement learning process [Houillon *et al.*, 2013].

In machine learning, novelty describes data that belong to an unknown pattern [Hajer *et al.*, 2020]. Novelty detection is oftentimes used to identify outliers and faulty operation states. In RL, however, identifying undesired operations is more closely related to distributional shift and OOD detection while novelty is commonly utilized as a mechanism for exploration purposes. As shown by [Burda *et al.*, 2018], this mechanism is also referred to as intrinsic motivation or curiosity-driven exploration. [Lehman and Stanley, 2008] show how exploration based on novelty can be an efficient way to avoid getting stuck at local optima regions, a recurrent issue in problems where climbing the stepping stones that ultimately lead to the goal is not rewarded by the objective function.

Despite its importance, seeking novelty brings a paradoxical consequence: exploring the unknown may lead to a completely unexpected outcome, including a bad one. Although learning from bad experiences is valuable, in safety-critical applications some states are dangerous and must not be visited. In such cases, safe exploration methods are paramount to avoid violating the safety constraints.

2.5 Robust RL

Robustness is an important topic in ML since trained models are expected to work despite small deviations in the input features. Recently, discussions regarding adversarial attacks

have shown that neural networks are vulnerable to specifically designed small changes in the input which result in drastic changes in their predictions [Madry *et al.*, 2017]. The very existence of adversarial attacks may suggest an inherent weakness of deep learning models, reigniting the interest of researchers in the search for more robust models. This same problem can affect RL models, as shown by [Pinto *et al.*, 2017]. [Nilim and El Ghaoui, 2003] argue that in RL, robustness is usually related to uncertainties in the transition matrix of the MDP while [Pinto *et al.*, 2017] shows that RL models should also be robust to model initializations and modeling errors.

Robustness regarding RL can also be viewed by the optics of control theory since RL can be described as a dynamical closed-loop system. In control theory, robustness is a widely studied field that focuses on the control of systems subject to uncertainty, noise, and disturbances, consisting of the synthesis (i.e., designing robust controllers) and analysis (i.e., evaluating the robustness of a controller) problems [Zhou and Doyle, 1998; Bhattacharyya and Keel, 1995]. Robust control can be used to increase operational safety and performance on safety-critical applications, such as the control of nuclear power plants [Jin *et al.*, 2010]. [Calafiore and Campi, 2006] argue that robust control has received criticism for being too rigid in describing uncertainty, resulting in overly conservative controllers and opening the path to alternative approaches. However, formalizing robustness in RL is not a trivial task and robust RL algorithms are not yet widely accepted as a replacement to classical robust control methods.

2.6 Operational Design Domain

Operational Design Domain (ODD) is defined by [NHTSA, 2017], in the context of automated driving systems. ODD is used to describe the specific conditions on which the system is intended to function safely. Information like roadway types, geographic area, speed range, among others, should be included in the ODD and documented accordingly to describe the desired operation domain. How the system should react when getting outside of its defined ODD is also a relevant related topic.

[Koopman and Fratrik, 2019] show how to use ODD in ML problems and extend ODD by including object and event detection and response, vehicle maneuvers, and fault management. A list including examples relevant for the automated driving scenario, which fall in each of these categories, is then provided as a starting point when designing systems based on ML.

Despite being originally described for automated driving applications, ODD provides an explicit framework to specify the intended deployment domain regarding safety-relevant aspects and is a powerful tool that can be helpful when designing RL systems and defining their operation domain.

3 How can these concepts help in RL problems?

The concepts presented in the previous section are of relevance when dealing with uncertainties and changes in the environment. However, the distinction between them is subtle, and bringing these concepts down to RL problems is not straightforward. One approach that could help in that endeavor is shown in Figures 1 and 2.

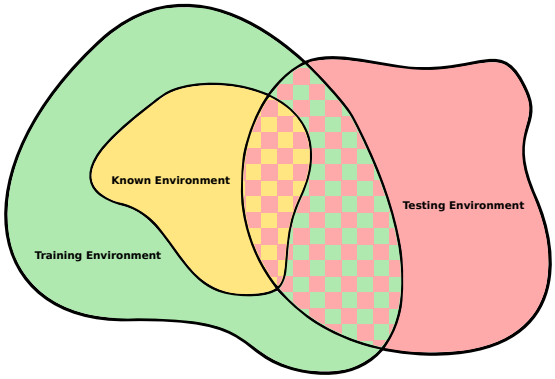

Figure 1: Visualizing the problem: distinction between training, known, and testing domains. The chess-like patterns describe the intersections with the testing region.

Figure 1 shows the whole problem domain, i.e., the distribution over environments reachable by the agent during training and testing. The *Training Environment* considers all the situations the agent can encounter during training. It can alternatively be described as the in-distribution domain. Designing this region using the ODD framework is helpful when defining safe boundaries that the agent must respect. The region corresponding to the already explored portion of the training domain is hereby defined as the *Known Environment*. The last domain is the *Testing Environment*, which encompasses the whole region the agent can come across after deployment. There can be overlaps between the different domains, as depicted in the figure. The optimal scenario happens when the whole testing domain is included in the training region and the agent is able to explore the domain completely during training, which is often not feasible for large problems.

When dealing with the problem of domain shifts in RL, mapping the regions of Figure 1 into the concepts described in the previous section can help with its characterization, as depicted in Figure 2. This mapping is not unique and is not intended to reflect the whole theory behind domain shifts. It is rather, in our perspective, the most appropriate for RL problems subject to changes in the environment. Analyzing the *Known Domain* is not the focus of this paper and is therefore ignored here. However, we would like to point out that unsafe states can also occur within the known environment of a system. The region inside the training domain that is still unknown to the agent is mapped as the *Novelty Region*, i.e., the region that can be explored during training. After deployment, the agent should be robust to novel states, which belong to the same distribution of the training examples, here defined as the *Robustness Region*. The last region is defined as the OOD region, composed of samples from a different distribution to the training distribution.

A drawback of this approach is that it is difficult to draw specific boundaries to differentiate these regions, especially

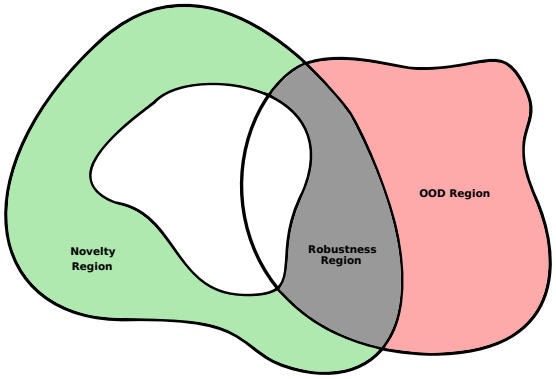

Figure 2: A possible mapping for the domain regions from Fig. 1 into ML approaches that deal with domain shifts. For RL problems, the regions of interest can be related to novelty, robustness, and OOD problems.

because it is not trivial to list the changes to the environment that can lead to situations outside of the training domain. To make it even worse, a consensus on how to apply some of the concepts in RL is still missing. Therefore, different interpretations may arise based on how the ideas are ported from more general ML problems into the RL framework. For instance, if we follow the definition from [Sedlmeier *et al.*, 2019], everything outside the known environment is considered OOD. On the other hand, based on the interpretation from [Mendonca *et al.*, 2020], OOD is composed of tasks semantically different from the training ones. However, formally defining semantic differences in the tasks is difficult.

4 Formalizing Domain differences as Aspects of the MDP

The methods described in section 2 all share a common characteristic: they deal with problems in which the train/source task differs from the test/target task. In other words, the test/target task is disturbed. The difference in these methods arises from how this disturbance manifests, i.e., how training and testing tasks relate to each other. As a first step towards safe RL, we believe it is necessary to gain a better understanding of this relationship. For this, we borrow an idea that is commonly used in the Transfer Learning literature [Taylor and Stone, 2009; Zhu *et al.*, 2021], which we coin MDP decomposition. That is, any disturbance is sub-divided into the individual components that build up the MDP. This decomposition should come naturally since MDPs are used to describe RL problems in the first place. We want to point out that Safe RL has more many more dimensions, such as safety during training or robustness against adversarial attacks. Safety concerns that arise from differences between training and deployment environment are only one aspect, which is the aspect tackled by the hereby proposed approach.

Two tasks $\mathcal{M}_{\mathcal{A}}$ and $\mathcal{M}_{\mathcal{B}}$, i.e., the training scenario $\mathcal{A}$ and the testing scenario $\mathcal{B}$, can thus be different only in the following aspects (or a combination thereof):

- S (State-space): Enlargements, reductions or changes to the set of possible states;
- A (Action-space): Additional or fewer actions, changed ranges of continuous actions;
- P (Transition Dynamics): Different transition probabilities, given the same state-action pairs;
- R (Reward Function): Modified reward function that reinforces different behaviors;
- μ_0 (Initial states): Different initial state distributions.

Although this may seem like a relatively trivial matter at first, the benefits of this simple method quickly become apparent in practice. Training an RL agent to solve tasks while reliably meeting necessary safety constraints is an intricate problem. While many safety-critical issues can be addressed by accurate modeling in the training environment, via domain randomization or hand-crafted safety controllers, accounting for every possible safety-critical situation is intractable given the complexity of the real world. We, therefore, have to make assumptions on the relationship between training and testing environments, otherwise guaranteeing safety becomes impossible. To formulate this relationship we can turn to the traditional problem formulations mentioned in 2. We find, however, that framing even simple examples into definitions like distributional shift, novelty detection, out-of-distribution or robustness is not only complicated but provides limited insights towards an appropriate safety argumentation. MDP decomposition on the other hand is straightforward and allows us to locate the problem in the individual components of the MDP.

It is important to mention that every element that composes an MDP is formally defined. Therefore, it is possible to outline operational limits, boundaries, or distributions over each element. As an example, one could train a robust agent where, due to degradation on the actuators that might occur with time, the maximum action can be reduced by up to 20% of the nominal value. Another example is an agent able to deal with some sort of noisy sensor, that will change the observation of the states. In that sense, it becomes clear how to design test cases that cover the range of operation, solely by changing the MDP element that is affected by these changes. This naturally helps with separating concerns, a crucial component when arguing about safety. Moreover, it is an initial step towards better comparability in the regime of Safe RL as it allows us to compare approaches that currently run under different terms and formulations on a more uniform scale.

Mapping which aspect of the MDP is affected by a change in the system relies on a good comprehension of the process. The state-space can be altered due to novel elements inserted in the surroundings of the agent. Also, changes in the sensors (noise, wear and tear, replacement, preprocessing changes) can lead to a different perception of the state-space. Novel elements can also have dynamics unseen by the agent, affecting the transition function. The underlying behavior of novel elements will also affect the transition function, as they can have a collaborative or adversarial behavior for instance. Changes in the action-space are usually related to failures or degradation of the actuators. Upgrading the system (e.g., the attachment of a new actuator in a robot) can also lead to a different action-space. Mapping changes to the reward function is a particular case since the function is designed to

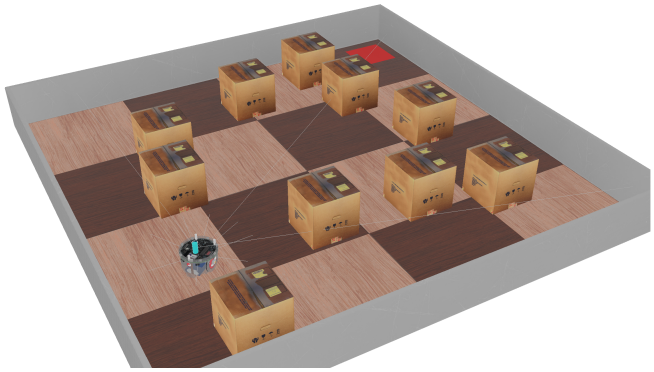

Figure 3: Environment used as example to apply the proposed formulation over different concepts. The AGV is a two-wheeled e-puck robot, the obstacles are represented as boxes, the goal is to reach the red tile in the upper corner. Note: disturbances are absent during training (e.g. no workers interacting with the AGV, no malfunctions, etc.).

translate the task into an objective function that allows the agent to learn. Therefore, changes in the task itself will cause changes in the reward function. Also, the reward function can be changed to guide the agent while learning (reward shaping). Isolating initial state changes is more straightforward and self-explanatory.

5 Example Scenario

To demonstrate the expressiveness of the decomposition described above, we apply it to an exemplary real-world scenario, depicted in Figure 3.

Consider an Automated Guided Vehicle (AGV) navigating in a warehouse. To determine its location and to identify its surroundings, it is equipped with an array of sensors (e.g., camera, lidar, accelerometer). The primary task goal for the AGV is to reach the destination position. The safety constraint is to avoid any collisions with its surroundings. In a realistic setting, there are several challenges the AGV might face, such as workers or other robots interacting with the AGV (collaborative and non-collaborative), multiple goals, malfunctions of AGV (flat tire, low battery, etc.), or noisy sensors. For the purpose of this example, we consider all of these hazards absent during training, i.e., they represent disturbances to the environment during deployment.

Table 1 shows that each of these hazards can be traced back to only a single or at most two components of the MDP. This mapping is straightforward and directly helps to isolate safety-relevant issues. Once isolated, these issues can be detected and handled more easily.

Conversely, applying the existing formulations covered in section 2 provides limited insights, as we show in the following. Apart from noisy sensors, all problems can be framed as an instance of domain shift, OOD, or novelty, depending on the definition, as described in 1. Therefore, domain shift/OOD/novelty as a proxy for safety-critical situations are not particularly helpful. If the severity of disturbances is not addressed explicitly, minor disturbances, which are essentially irrelevant to the safety of the system, are already deemed as a safety concern.

	S	A	P	R	μ_0
Workers interacting with the AGV	✓		✓		
Other robots interacting with the AGV	✓		✓		
Changed warehouse layout	✓				
Multiple goals				✓	
Unusual starting position					✓
Malfunctions of the AGV		(✓)	✓		
Noisy sensors	✓				

Table 1: Decompostion of potential hazards in the warehouse domain into components of the MDP.

Robustness is typically used when dealing with problems such as noisy sensors, slight malfunctions of the robot, or minor external disturbances. However, as soon as the disturbances are more severe, e.g., workers interacting with the environment, achieving robustness is usually infeasible.

ODD can help to define the boundaries of a system more clearly but it essentially requires anticipating all potential safety threads during the design of the system. MDP decomposition can help in this process, by giving deeper insights into potential causes of safety threads.

Transfer Learning/Domain adaptation as described in section 2 actually tackles another problem. They assume the changes from source to target domain to be static, and not ad hoc like sudden malfunctions of the robot or interactions with humans. If we can assume, however, that disturbances are static, e.g., changes to the layout of the warehouse, transfer learning approaches might be the tool of choice.

6 Conclusion

Handling disturbances in the environment is an essential step towards safe RL systems. We showed that previous work approaches this problem from various different angles, though often lacking a clear problem formalization within the RL domain. Thus, applying existing approaches from other areas to RL problems is not straightforward. To disambiguate how changes in the environment of an RL agent can manifest in a formal task definition, we proposed to decompose complex problems into the aspects that build up the MDP. This simple trick allows us to isolate different concerns and treat each of them separately. We applied this method to a simple obstacle avoidance task, where a wheeled robot has to navigate in a warehouse. By listing potential disturbances and analyzing how they affect the MDP, we laid ground for this framework to help when dealing with complex real-world scenarios.

We also see some clear limitations to this approach. Even if potential disturbances can be located in a single part of the MDP, it is still a problem of its own to properly handle them. For the above scenario, consider for example that we introduce novel obstacles that look the same as the cardboard boxes but which are also able to move. This would manifest as a change to the transition function only, since individual states stay the same and only sequences of states are different from before. Although posing a severe thread to safety, de-

tecting such a change is complicated, even when decomposed into MDP components.

Future work has to detail further how the MDP decomposition and structuring of the problem can be utilized for adding robustness and ensuring safety under environmental disturbances. We intend to use this formalization and analyze how we can both detect and handle disturbances in individual components of the MDP. We believe that this change of perspective results in a more accurate characterization of safety critical applications targeted for RL systems and naturally helps with formalizing safety objectives, bringing us closer to enable an appropriate validation of RL systems.

References

[Amodei *et al.*, 2016] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[Bhattacharyya and Keel, 1995] Shankar P Bhattacharyya and Lee H Keel. Robust control: the parametric approach. In *Advances in control education 1994*, pages 49–52. Elsevier, 1995.

[Burda *et al.*, 2018] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

[Calafiore and Campi, 2006] Giuseppe C Calafiore and Marco C Campi. The scenario approach to robust control design. *IEEE Transactions on automatic control*, 51(5):742–753, 2006.

[DeVries and Taylor, 2018] Terrance DeVries and Graham W Taylor. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*, 2018.

[Eysenbach and Levine, 2021] Benjamin Eysenbach and Sergey Levine. Maximum entropy rl (provably) solves some robust rl problems. *arXiv preprint arXiv:2103.06257*, 2021.

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

[Haarnoja *et al.*, 2018] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. 2018. Milestone Algorithm; Stochastic Actor-Critic.

[Hajer *et al.*, 2020] Jan Hajer, Ying-Ying Li, Tao Liu, and He Wang. Novelty detection meets collider physics. *Physical Review D*, 101(7):076015, 2020.

[Houillon *et al.*, 2013] A Houillon, RC Lorenz, W Boehmer, MA Rapp, A Heinz, J Gallinat, and K Obermayer. The effect of novelty on reinforcement learning. *Progress in brain research*, 202:415–439, 2013.

[Jin *et al.*, 2010] Xin Jin, Asok Ray, and Robert M Edwards. Integrated robust and resilient control of nuclear power plants for operational safety and high performance. *IEEE Transactions on Nuclear Science*, 57(2):807–817, 2010.

[Koopman and Fratrik, 2019] Philip Koopman and Frank Fratrik. How many operational design domains, objects, and events? *SafeAI@ AAAI*, 4, 2019.

[Lehman and Stanley, 2008] Joel Lehman and Kenneth O Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pages 329–336. Citeseer, 2008.

[Levine *et al.*, 2016] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. 17(1):1334–1373, 2016. Publisher: JMLR. org.

[Liang *et al.*, 2017] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

[Lillicrap *et al.*, 2015] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. 2015. continuous Q learning with actor network for approximate maximizatio.

[Madry *et al.*, 2017] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[Mendonca *et al.*, 2020] Russell Mendonca, Xinyang Geng, Chelsea Finn, and Sergey Levine. Meta-reinforcement learning robust to distributional shift via model identification and experience relabeling. *arXiv preprint arXiv:2006.07178*, 2020.

[Mnih *et al.*, 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[NHTSA, 2017] NHTSA. Automated driving systems 2.0: A vision for safety. *Washington, DC: US Department of Transportation, DOT HS*, 812:442, 2017.

[Nilim and El Ghaoui, 2003] Arnab Nilim and Laurent El Ghaoui. Robustness in markov decision problems with uncertain transition matrices. In *NIPS*, pages 839–846. Citeseer, 2003.

[Pinto *et al.*, 2017] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826. PMLR, 2017.

[Puterman, 2014] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[Rajeswaran *et al.*, 2016] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt:

Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

[Redko *et al.*, 2019] Ievgen Redko, Emilie Morvant, Amaury Habrard, Marc Sebban, and Younes Bennani. *Advances in domain adaptation theory*. Elsevier, 2019.

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

[Schulman *et al.*, 2017] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. 2017. deep RL with natural policy gradient and adaptive step size.

[Sedlmeier *et al.*, 2019] Andreas Sedlmeier, Thomas Gabor, Thomy Phan, Lenz Belzner, and Claudia Linnhoff-Popien. Uncertainty-Based Out-of-Distribution Classification in Deep Reinforcement Learning. December 2019.

[Silver *et al.*, 2017] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[Taylor and Stone, 2009] Matthew E Taylor and Peter Stone. Transfer Learning for Reinforcement Learning Domains: A Survey. page 53, 2009.

[Tobin *et al.*, 2017] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017.

[Tzeng *et al.*, 2014] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[Yosinski *et al.*, 2014] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*, 2014.

[Zhou and Doyle, 1998] Kemin Zhou and John Comstock Doyle. *Essentials of robust control*, volume 104. Prentice hall Upper Saddle River, NJ, 1998.

[Zhu *et al.*, 2021] Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. Transfer Learning in Deep Reinforcement Learning: A Survey. *arXiv:2009.07888 [cs, stat]*, March 2021. arXiv: 2009.07888.