Effects of Domain Randomization on Simulation-to-Reality Transfer of Reinforcement Learning Policies for Industrial Robots

C. Scheiderer, N. Dorndorf, and T. Meisen

1 Introduction

In recent years, the paradigm of reinforcement learning (RL) has been employed successfully to solve a multitude of different control tasks from games [1, 2] to resource management [3] to robotics [4, 5]. The concept of using an autonomously exploring agent, which is able to leverage experience in order to deduce favorable strategies from observations and performance indicators promises potential to automate tasks where a solution strategy is hard to formulate.

The underlying principle of trial-and-error however limits the applicability of RL. Generating enough experience in order to derive sufficiently accurate control strategies may present a major challenge. This becomes especially apparent when using RL to solve real-world tasks in the domain of robotics, where the execution of actions is time-consuming and costly in comparison to tasks in the digital domain. In addition, the integral property of exploration implies the possibility of executing suboptimal actions, which in real-world scenarios may be disproportionately expensive or safety critical.

One possibility of reducing the need for task-specific experience is given by the application of transfer learning [6, 7]. Transfer learning is based on the idea of using knowledge acquired from a source task in order to reduce the training efforts required to learn a target task. One obvious choice of a source task, when considering real-world RL, is simulations.

C. Scheiderer (✉) · T. Meisen
Institute of Technologies and Management of the Digital Transformation, University of Wuppertal, Wuppertal, Germany
e-mail: scheiderer@uni-wuppertal.de

N. Dorndorf
RWTH Aachen University, Aachen, Germany

H. R. Arabnia et al. (eds.), *Advances in Artificial Intelligence and Applied Cognitive Computing*, Transactions on Computational Science and Computational Intelligence, https://doi.org/10.1007/978-3-030-70296-0_13

In this chapter, we focus on domain randomization [8] as a method for transferring RL agents from a simulation to the real world for robotic applications. In domain randomization, variations are purposefully induced into the simulation environment. The goal hereby is to increase the robustness of a trained RL agent with respect to deviations between simulation and the real world. We consider a robotic task inspired by the work of [9, 10], in which a RL agent is trained end-to-end on the wire-loop game. The task is designed to be based on real motion planning scenarios for industrial processes such as welding or gluing. The goal of the game is to guide a loop over a wire without touching it. Our agents are based on convolutional neural networks (CNNs) and are able to observe the environment via a camera. The agents directly control the movement of the robot holding the loop. In addition to the real-world setup, we use a corresponding simulation in which parameters, such as colors and textures of loop, wire and background, are adjustable. The agents are pretrained in the simulation with domain randomization and subsequently transferred to the real-world task. In our experiments we vary the parameters chosen for randomization and observe effects on the agents' ability to transfer. We finally use Grad-CAM [11] to visualize the attention placed by the agents on various parts of an observed camera image in order to assess the robustness of an agent after pretraining in the simulation as well as its data-efficiency when adapting to the real-world task after transfer.

2 Related Work

2.1 Reinforcement Learning

As RL is thoroughly discussed in various other works [9, 12, 13] only a short overview of the key aspects is provided. In general, RL describes the concept of iteratively training an agent in a trial-and-error fashion on solving a task in an environment $\mathcal{E}$. At each discrete time step t, the agent observes the environment's state $s_t \in \mathcal{S}$. Based on the observation, the agent then chooses an action a_t according to its policy. The policy is given by a probability distribution over all actions given an observed state $\pi(a|\, s)$. In deep RL, the policy is represented by artificial neural networks. The execution of action a_t yields the reward r_t and the next state s_{t+1}, completing one iteration of the learning process. The overall goal of an agent is to find a policy which maximizes the discounted future reward $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1}$, where the discount factor $\gamma \in [0, 1]$ adjusts the importance attributed to future rewards.

For tasks with continuous action spaces, actor–critic architectures have proven to be a suitable approach [14, 15]. They consist of two models, the actor and the critic, which are trained interdependently. On the one hand, the actor models the agent's policy and maps states onto actions. The critic on the other hand learns to evaluate and improve the actor's performance. Analogous to [9], we use deep deterministic

policy gradient (DDPG) [15] to train actor–critic architectures, whereby both actor and critic are represented by artificial neural networks.

2.2 Simulation-to-Reality Transfer Learning

The aim of transfer learning lies in reusing knowledge acquired during solving one source task in order to efficiently learn a new target task [7]. Especially in computer vision, the benefit of transfer learning has successfully been shown for tasks such as object detection [16], pose estimation [17], or image segmentation [18].

When applying transfer learning to RL setups, the question about suitable source tasks arises. One intuitive possibility for enriching data from a real-world process is by using simulations. This option is attractive especially in industrial contexts, as obtaining experience from suboptimal process situations (e.g., low-quality products or slower production cycles) or critical conditions (e.g., high wear and tear of a tool) can be realized in virtual environments without consequences to the real-world production line.

The major challenge to overcome when using simulation data for pretraining an agent is the reality gap [19]. The reality gap describes the fact that simulations do not model the real world perfectly. Deviations arise, for example, from inadequately modeled parameters or lower resolution of the simulation environment. One strategy for overcoming the reality gap lies in an effort to improve the quality of a simulation, making the simulation as realistic as possible [20, 21]. This approach however assumes a deep understanding of underlying physical principles in the real-world process in order to be able to model them. In addition, increasing the quality of a simulation often requires extensive computation, which limits the applicability of the simulation.

A different strategy is aimed at finding a mapping from the distribution of simulation data onto the distribution of real-world data using generative adversarial networks (GANs) [22]. This mapping in turn can be used to enhance the realism of the artificial training samples. Alternatively, mappings of both simulation and real data in a shared feature space on which the agent is trained can mitigate the reality gap [23]. Finding such mappings however relies on the availability of sufficient real-world data.

In contrast to the aforementioned approaches, domain randomization [8] circumvents the need for both perfect simulations and real-world experience by purposefully randomizing visual (e.g., colors, textures) or even physical (e.g., friction) properties. This way the agent is exposed to a manifold of environments, which forces the agent to disregard the parameter variations. Although the majority of environments are not necessarily realistic, it has been shown by various works that agents are able to generalize and find robust control policies applicable to the real-world target task [8, 24–28]. While the successful application of domain randomization has been demonstrated before, the effects of different randomizations

regarding the robustness and transferability of pretrained networks, to our knowledge, were not investigated to date.

2.3 Attention Maps

While artificial neural networks have proven to be very powerful tools, their lack of interpretability remains a key obstacle to the application in industrial environments. One possibility to gain insights into their decision-making process is to take a look at what areas of an input influence the final decision the most. This can be achieved by examining the gradient of an output with respect to the input [29] or the effects of perturbations to the input to the prediction [30, 31]. In this chapter we employ Grad-CAM [11], which was developed for CNNs. By calculating the gradient of an output with respect to the feature maps of convolutional layers, an importance weighting of the feature maps is determined. Using this weighting the feature maps are recombined to arrive at attention maps which highlight important regions in the input image and thusly promote transparency for the network's decision.

While previous works have covered the application of attention maps for RL [32], the application of attention maps for investigating the effects of domain randomization proposed in this chapter is, again to our knowledge, novel.

3 Experimental Setup

3.1 Learning Environment

Our use-case scenario is inspired by the work of Meyes et al. [9], in which a six-axis industrial robot is tasked with playing the wire-loop game. The loop is constructed out of a 3D-printed base to which two metal rods are mounted. By measuring the conductivity between the metal rods and the wire it is possible to determine whether the loop is touching the wire. The hardware setup is depicted in Fig. 1a.

At every time step, an agent observes the shape of the wire immediately in front of the loop in the form of a down-sampled image taken by a camera mounted to the loop (Fig. 1b). The agent subsequently has to decide for a continuous action $a_t \in \mathbb{R}^3$, which describes the longitudinal, lateral, and rotational movement of the loop. The game is considered lost in case the loop touches the wire or if the agent is not progressing along the wire. Conversely, the game is won once the agent successfully arrives at the end of the wire without touching it. We choose our reward function analogous to Meyes et al. [9], whereby the reward function is continuous between $[-1, 1]$ and incentivizes progress along the wire.

Our simulation is based on the simulation software V-REP [33] and mirrors the real-world setup (Fig. 1c–d). In addition to controlling the robot and detecting con-

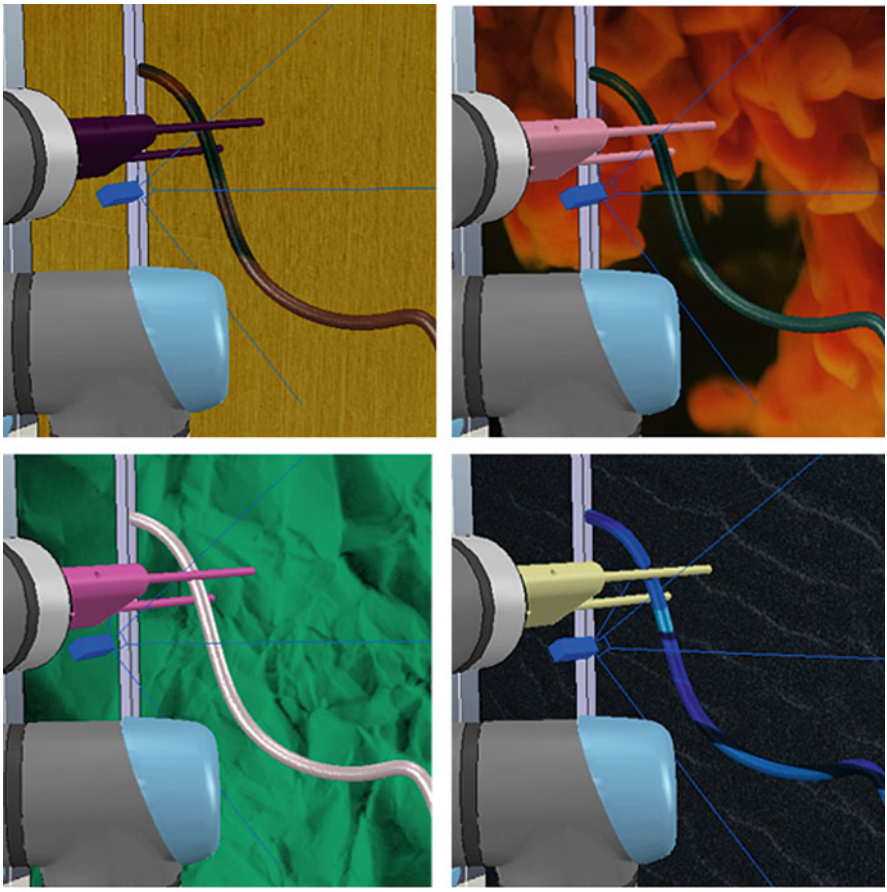

Fig. 1 Examples of randomized simulation parameters

tact between wire and loop, we can also dynamically change multiple environment parameters in order to conduct domain randomization:

- The *colors* of wire, background, and loop can be set to arbitrary RGB values
- In addition, 20 different *textures* can be chosen for the wire and the background
- The *camera pose* can be varied with a maximum displacement of 1.5 cm and a maximum rotation of 5° in every direction
- Random Gaussian noise the image taken by the camera, normalized between 0 and 1, can optionally be superimposed with random Gaussian noise with a standard deviation of 0.01.

As shown in Fig. 2, the variation of these environment parameters allows for the generation of highly diverse scenarios.

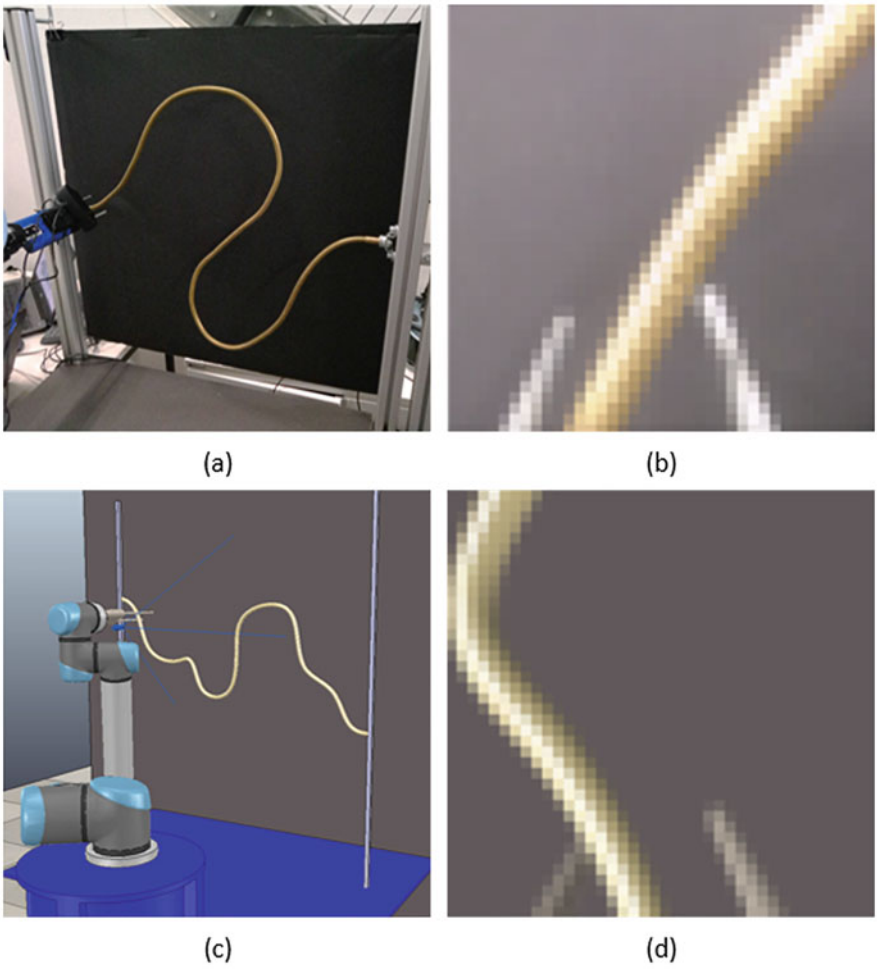

Fig. 2 The experimental setup consisting of a real (**a**) and a simulated (**c**) industrial robot. The environment is observed via real (**b**) and simulated (**d**) camera images

3.2 Agent Architecture

Analogous to [9], the agent comprises an actor and a critic network trained with DDPG and Adam as the optimizer. The actor network consists of three convolutional layers with 30, 15, and 10 filters and respective kernel sizes of 5×5, 5×5, and 3×3. After every convolutional layer, batch normalization is performed. MaxPooling is applied after the first two convolutional layers with a pooling size of 2×2. The CNN is followed by four densely connected layers with sizes of 400, 200, 100, and 3. ReLU is used as activation function after every layer, except for the

last one. Instead, the last layer has one of its three neurons activated with a sigmoid function and the remaining two with tanh activations. The three outputs constitute the longitudinal and lateral movement as well as the rotation of the loop. Using the sigmoid activation for the longitudinal action results in the prevention of backward movement.

Similarly, the critic network employs a CNN of the same topology as the actor network. The output of the CNN as well as the action input of the critic are each processed by one respective densely connected layer of size 200. The two resulting outputs are concatenated and further passed through two densely connected layers of sizes 200, 100, and 1.

3.3 Design of Experiments

The method of domain randomization is applied to improve the generalization of the agent by artificially increasing the variability of the training data in the simulation. In our case we uniformly randomize the environment parameters *colors*, *textures*, *camera pose*, and *camera noise* for each training run. In order to investigate how the individual parameters influence the training and the success on the real robot, we consider six different configurations which randomize different combinations of the parameters available (Table 1). Parameters which are not randomized are kept at the value of the standard setup shown in Fig. 1c–d. The agents are tested after every 50th training iteration on the standard setup. Once the wire is traversed successfully training is stopped.

After pretraining the agents in simulation, we transfer them to the real-world setup, whereby the actor and critic neural networks including their optimizer states as well as the experience replay buffer are saved and transferred to maintain continuous training. We reset the exploration rate back to full exploration, which enables the agent to collect experience in the real world with high diversity. This procedure was chosen experimentally, as it resulted in the best performance of the agents after transfer. Training is continued until the agent is able to complete the

Table 1 Experiment configurations

Configuration	Parameter			
	Colors	Textures	Camera pose	Camera noise
1				
2	✓		✓	
3	✓		✓	✓
4	✓	✓	✓	
5	✓	✓		✓
6	✓	✓	✓	✓

Overview of parameters chosen for randomization during the respective training of the 6 agents

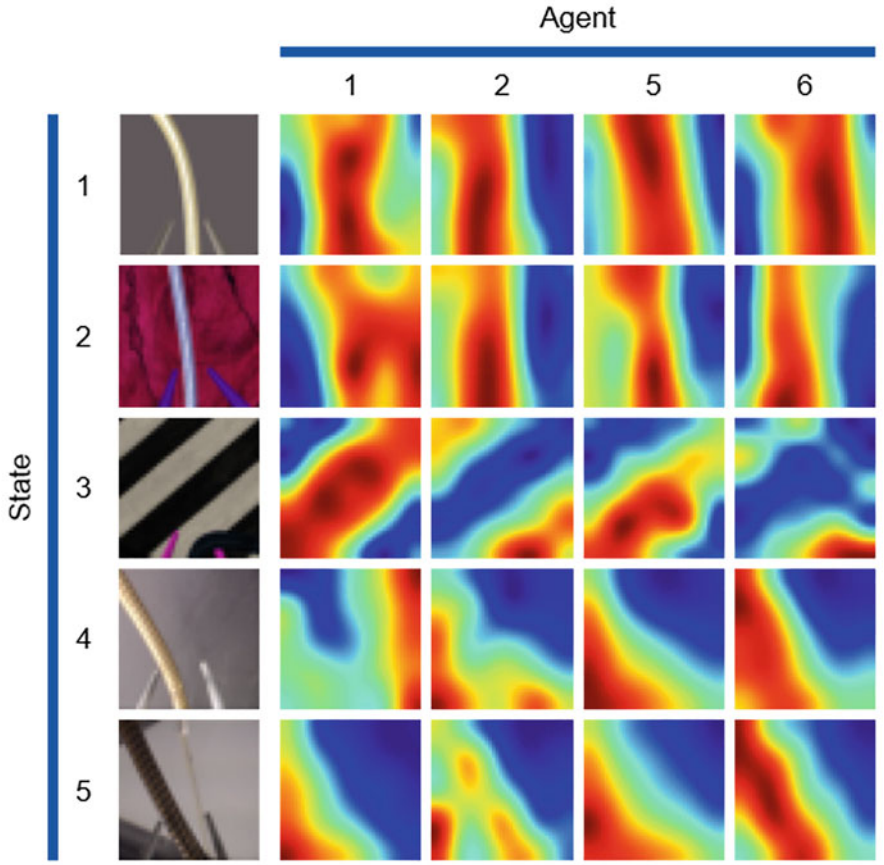

Fig. 3 In order to better understand an agent's behavior, attention maps are generated for 4 agents (1, 2, 5, 6) using different states from simulation (states 1–3) and the real world (4 & 5). The states vary in complexity with regard to colors and textures. A rainbow-coloring scheme is used, which spans from red (high attention) to blue (low attention)

real-world task as well. Analogous to the simulation, the performance is tested after every 50th training iteration.

Finally, we take the pretrained actor networks and generate attention maps for different input images using Grad-CAM. A variety of images are chosen from both the simulation and the real world (see Fig. 3, left column). In combination with the behavior observed during training the attention maps are qualitatively examined and conclusions regarding different randomization strategies are drawn.

Table 2 Required training steps until game finished

Agent	Simulation (pretraining)	Real-world (transfer)
1	*400*	600
2	1500	250
3	1900	250
4	2400	*0*
5	700	350
6	2850	*0*

4 Results

In the following we compare and discuss the training results using the above-mentioned experimental configurations. Table 2 lists the amounts of training steps required to learn the wire loop game in simulation using different randomization strategies, as well as the training steps required once the respective agents are transferred to and retrained on the real-world setup.

4.1 Training in Simulation

Intuitively, the training effort increases with a growing complexity of the observations associated with more randomization. The amount of training steps increases by a factor of 7 when applying full-domain randomization (agent 6) in comparison to no-domain randomization (agent 1). Especially changes in the camera position and orientation have a large influence on the learning process, as the number of training iterations increases only slightly with a fixed camera (agent 5) in contrast to the other experimental setups where it is randomized. Textures (agent 4) seem to have a stronger impact on the training than noise (agent 3). One possible explanation is that even with noise the detection of the wire by color is still fairly straightforward, whereas changing textures are more likely to confuse the neural network.

4.2 Transfer to Real World

The results of transferring the pretrained agents to the real world are also reported in Table 2. Two agents were able to complete the task immediately: the agent trained with full domain randomization (agent 6) and the agent trained with domain randomization but without noise (agent 4). This confirms the ability of domain randomization toward generating robust agents solely in simulation. Additionally, the previously observed minor impact of noise randomization is further substantiated.

All agents trained with some form of domain randomization in the simulation require less additional training in the real world than the agent trained without

domain randomization. We also observe a strong negative correlation [$r(4) = -.94$, $p = 0.005$] between the required training steps in simulation and the real world. This intuitively translates into a positive effect of pretraining in more complex environments, and thusly increasing robustness, for solving the target task.

The added Gaussian noise does not seem to play a major role in the transfer of policies to the real robot, as the agents trained without additional noise in the simulation perform as well in the real world as their counterparts trained without noise. Randomization of the camera pose and textures, however, have a big effect on the transferability. These findings are plausible, as the mechanical connections between camera and tool as well as tool and robot experience some mechanical play in the real-world setup. Additionally, a camera image taken in the real world will always exhibit some sort of texture compared to the perfectly simulated images.

4.3 Attention Maps

As mentioned above, we apply Grad-CAM to the policies (actor networks) of our agents after pretraining in simulation. A selection of resulting attention maps is shown in Fig. 3. The states 1–3 are taken from simulation, and states 4 and 5 from the real world. To obtain state 5 we removed the black cardboard background from the demonstrator to increase complexity of the input image.

Each agent is pretrained in simulation. Afterwards training is continued on the real robot. Each training is run until the agent is able to complete the game. The enumeration coincides with Table 1 (Configuration n ➔ Agent n).

In general, we expect a well-performing agent to focus on the wire and disregard the background, as identifying the wire is essential to solving the task. This behavior can be observed for state 1, for which the differences in attention maps for the agents are only marginal. This is plausible as the standard setup is part of the state space for all randomization configurations.

For state 2 we observe a significant degradation of the ability of agent 1 to locate the wire due to the addition of texture, which the agent did not need to account for during its training process. Interestingly enough, agent 2 correctly identifies the wire, even though it was not exposed to texture randomization. This indicates that the variation of camera poses reinforces the agent to learn a more complete concept of the wire.

Detecting the wire in state 3 in the lower right corner is a hard task even for humans, as the dark wire disappears in front of the dark texture stripes. As before, agent 1 clearly struggles with the texture. In addition, agent 5 is also completely distracted by the dark stripe. While agent 2 pays attention to the wire, it also falsely focuses on the upper left corner. The superior attention of agent 2 compared to agent 5 is unexpected considering the training configurations, but again displays that agent 2 seems to have learned a robust representation of the wire due to the variation of the camera pose. Agent 6 exhibits the best performance for state 3, highlighting the benefit of combining parameter randomizations to increase robustness.

Even though state 4 is very similar to state 1 from a human perspective, agent 1 is distracted by a gradient induced by the lighting. While agents 4 and 5 pay attention in the general area of the wire, they seem to be drawn more to the lower left corner instead of the wire itself. Agent 6 again is the only agent which clearly identifies the wire.

Similar observations can be found for state 5, where agents 1, 2, and 5 focus mostly on the sharp background structure in the lower left corner. Although very weakly, agent 2 at least pays some attention along the wire, again showing the positive effect of camera-pose randomization. As before, agent 6 demonstrates its ability to robustly localize the wire despite complex conditions.

Revisiting the agents' abilities to transfer to the real world, a correlation between the transfer success and the interpretation of attention maps becomes evident. When ranking the agents' abilities to focus on the wire under the discussed conditions, full randomization (agent 6) clearly resulted in the most robust policy. It is also apparent that without any randomizations (agent 1) the worst performance is achieved. Randomizing the camera pose (agent 2) arguably has a higher benefit than randomizing textures (agent 5), which becomes especially clear for states 3 and 5. As observed before, applying random Gaussian noise to the camera image does not have any significant effect.

We thus draw the conclusion that in addition to allowing interpretation of the decision-making process of artificial neural networks, attention maps are a valuable tool to assess the prospects of success of simulation-to-reality transfer learning. For vision-based tasks in particular our results suggest that randomization of the camera pose results in the highest amount of robustness, whereas the addition of random Gaussian noise has no effect on the agent's ability to focus its attention.

4.4 Summary and Outlook

In this chapter we examined domain randomization for pretraining RL agents in a simulation in order to decrease training effort in the real world. We investigated the effects of choosing different parameters for randomization on the transferability of the respective agents. In our experiments we found that domain randomization in any case improves the training efficiency in the real world. We also discovered a strong negative correlation between training effort in simulation and the real world under varying randomization configurations. We attribute this finding to a higher complexity in simulation leading to more robust agents leading to faster training in the real world. In addition, we used Grad-CAM to generate attention maps for pretrained agents and discovered a correlation between the qualitative interpretation of such attention maps and the exhibited performance of the agents after transferring them to the real world.

Going forward, the applicability of attention maps as means to predict the success of transfer learning requires further investigation. Especially for scenarios where

exploration in the target domain is very costly, attention maps may help to guide refinement of agent robustness prior to transfer.

Although simulations run significantly faster than the respective real-world process, the training time of an agent still increases with an increasing number of parameters chosen for randomization. In addition to optimizing the actual transfer, attention maps might also prove useful to decrease training time in simulation. As demonstrated, attention maps hold information regarding which kinds of scenarios the agent has already learned and more importantly, which ones it is still struggling with. It is thus conceivable to use this information to purposefully select parameter variations which will advance the agent's robustness, instead of the current random selection. Inspired by GANs, this could potentially be achieved and automated using an adversarial agent which is rewarded for identifying scenarios beneficial for training the former agent.

References

1. D. Silver et al., Mastering the game of go without human knowledge. Nature **550**, 354–359 (2017)
2. O. Vinyals et al., StarCraft II: A new challenge for reinforcement learning. ArXiv, abs/1708.04782 (2017)
3. H. Mao, M. Alizadeh, I. Menache, S. Kandula, Resource management with deep reinforcement learning, in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, (2016), pp. 50–56
4. S. Gu, E. Holly, T. Lillicrap, S. Levine, Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, (2017), pp. 3389–3396
5. A.S. Polydoros, L. Nalpantidis, Survey of model-based reinforcement learning: Applications on robotics. J. Intell. Robot. Syst. **86**(2), 153–173 (2017)
6. J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, G. Zhang, Transfer learning using computational intelligence: A survey. Knowl.-Based Syst. **80**, 14–23 (2015). https://doi.org/10.1016/j.knosys.2015.01.010
7. K. Weiss, T.M. Khoshgoftaar, D. Wang, A survey of transfer learning. J. Big Data **3**(1), 9 (2016). https://doi.org/10.1186/s40537-016-0043-6
8. J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zarembe, P. 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)*, (2017), pp. 23–30
9. R. Meyes, C. Scheiderer, T. Meisen, Continuous motion planning for industrial robots based on direct sensory input. Procedia CIRP **72**, 291–296 (2018)
10. C. Scheiderer, T. Thun, T. Meisen, Bézier curve based continuous and smooth motion planning for self-learning industrial robots. Procedia Manuf. **38**, 423–430 (2019)
11. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, D. Batra, Grad-CAM: Why did you say that? Visual explanations from deep networks via gradient-based localization. ArXiv, abs/1610.02391 (2016)
12. R.S. Sutton, A.G. Barto, *Reinforcement learning: An introduction* (MIT Press, 2018)
13. V. Mnih et al., Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015)
14. T. Haarnoja et al., Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905 (2018)

15. T. Lillicrap et al., Continuous control with deep reinforcement learning. CoRR, abs/1509.02971 (2015)
16. R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2014), pp. 580–587
17. J. Carreira, P. Agrawal, K. Fragkiadaki, J. Malik, Human pose estimation with iterative error feedback, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2016), pp. 4733–4742
18. J. Dai, K. He, J. Sun, Instance-aware semantic segmentation via multi-task network cascades, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2016), pp. 3150–3158
19. N. Jakobi, P. Husbands, I. Harvey, Noise and the reality gap: The use of simulation in evolutionary robotics, in *ECAL*, (1995) This file was created with Citavi 6.2.0.12
20. S. James, E. Johns, 3D simulation for robot arm control with deep Q-learning. ArXiv, abs/1609.03759 (2016)
21. B. Planche et al., DepthSynth: Real-time realistic synthetic data generation from CAD models for 2.5D recognition, in *2017 International Conference on 3D Vision (3DV)*, (2017), pp. 1–10
22. K. Bousmalis et al., Using simulation and domain adaptation to improve efficiency of deep robotic grasping, in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, (2017), pp. 4243–4250
23. A. Gupta, C. Devin, Y. Liu, P. Abbeel, S. Levine, Learning invariant feature spaces to transfer skills with reinforcement learning. arXiv preprint arXiv:1703.02949 (2017)
24. O.M. Andrychowicz et al., Learning dexterous in-hand manipulation. Int. J. Robot. Res. **39**(1), 3–20 (2020)
25. S. James et al., Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. ArXiv, abs/1812.07252 (2018)
26. X. Peng, M. Andrychowicz, W. Zaremba, P. Abbeel, Sim-to-real transfer of robotic control with dynamics randomization, in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, (2017), pp. 1–8
27. J. Tremblay et al., Training deep networks with synthetic data: Bridging the reality gap by domain randomization, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, (2018), pp. 969–977
28. S. James, A.J. Davison, E. Johns, Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. arXiv preprint arXiv:1707.02267 (2017)
29. K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: Visualising image classification models and saliency maps. CoRR, abs/1312.6034 (2013)
30. M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in *European Conference on Computer Vision*, (2014), pp. 818–833
31. R.C. Fong, A. Vedaldi, Interpretable explanations of black boxes by meaningful perturbation, in *Proceedings of the IEEE International Conference on Computer Vision*, (2017), pp. 3429–3437
32. L. Weitkamp, E. van der Pol, Z. Akata, Visual rationalizations in deep reinforcement learning for atari games, in *Benelux Conference on Artificial Intelligence*, (2018), pp. 151–165
33. E. Rohmer, S.P.N. Singh, M. Freese, CoppeliaSim (formerly V-REP): A versatile and scalable robot simulation framework, in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, (2013)