Tunnel Vision Attack on IMPALA - Questioning the Robustness of Reinforcement Learning Agents

Julian Bolick, Gino Brunner, Oliver Richter, Roger Wattenhofer
Department of Electrical Engineering
ETH Zurich
8092 Zurich, Switzerland
{jbolick,brunnegi,richtero,wattenhofer}@ethz.ch

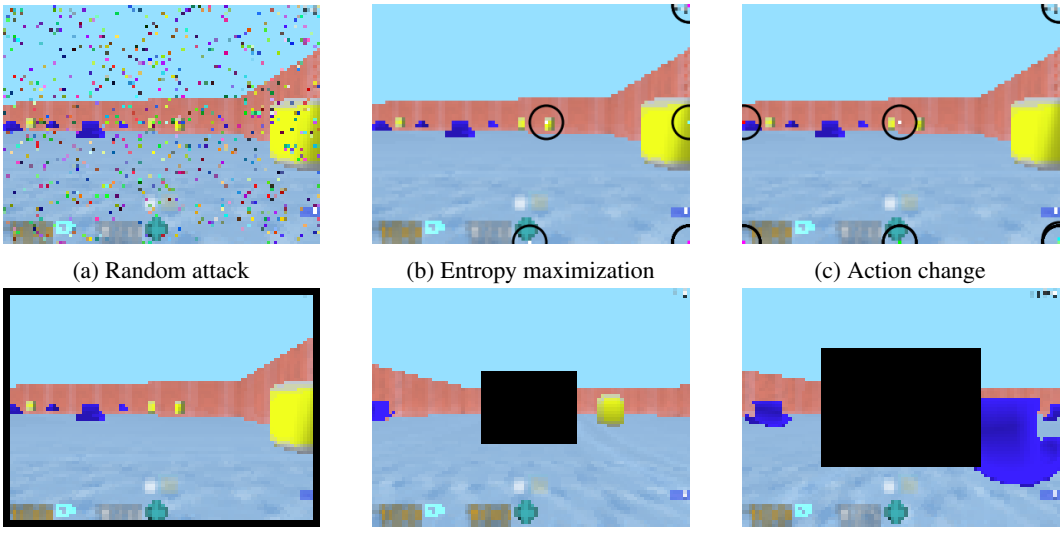
Abstract

We perform reward-agnostic attacks on an Importance Weighted Actor-Learner Architecture (IMPALA) based deep reinforcement learning agent by perturbing the input pixels in different ways. In one environment, where the agent needs to locate and pick up objects, we show that blacking out a two pixel wide border heavily impacts the agent's performance, while blacking out the center of the agent's visual field has minimal impact. This suggests that the agent is relying mainly on peripheral vision to solve this task. Interestingly, the same border-attack is less effective in a pure navigation task. We also find that an agent trained in a multi-task setup seems to be more resilient to attacks. Further, we report preliminary results of per-frame pixel attacks using a differential evolution algorithm and discuss directions for further research. Surprisingly, the most successful pixel perturbations found by differential evolution on the "collect good objects" task are concentrated at the edge of the frame, which agrees with the effectiveness of the attack that blacks out the border.

1 Introduction

Human eyes have evolved to provide the brain with a sharp and detailed view around the point of focus that gets less accurate in the peripheral vision. In contrast, reinforcement learning (RL) agents trained on pixel input can exploit their full visual field and thereby learn to rely on peripheral input in their policy. We exploit this difference to design an attack on RL agents whose change seems insignificant to humans but is devastating to the agent. Specifically, we show that simply removing pixels along the border from the input of a state of the art Importance Weighted Actor-Learner Architecture (IMPALA) agent [Espeholt et al., 2018] during testing can lead to a severe drop in performance, while the attack is almost imperceptible to humans. In contrast, we show that a trained IMPALA agent is robust during testing against random pixel changes and even targeted pixel-attacks to a degree that a human can easily perceive the attack before it hurts performance. However, we observe different behaviours for different environments, suggesting that each task requires specific attacks. This could also give further insights into semantic differences between tasks, e.g., peripheral vision seems important for collecting objects, but less so for finding the exit in a maze. With this work we highlight strengths and weaknesses of currently popular RL agents and hope to inspire future work in this area.

Workshop on Safety and Robustness in Decision Making at the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.



(a) Random attack (b) Entropy maximization (c) Action change

(d) Tunnel vision (e) Peripheral vision - 638 pixels (f) Peripheral vision - 1728 pixels

Figure 1: Visualization of typical observations under different attack scenarios. In 1b and 1c, black circles mark the perturbation pixels. While the alterations in the entropy maximization and the action change attack seem minimal in the still images, they are easily detectable in a video.

2 Related Work

Safety-problems when looking at reinforcement learning (RL) algorithms have been categorized in past publications such as [Amodei et al., 2016]. While super-human performances in simulated or highly controlled environments such as games [Mnih et al., 2016] or robotic manipulation tasks [Levine et al., 2015] are now commonplace, safety-critical applications face additional challenges yet to be overcome. Besides problems like safe exploration, robustness to distributional shifts or under-representation of safety-critical situations in data, RL also falls victim to so called adversarial inputs. These are specific inputs to exploit weaknesses of the underlying neural network structure. Indeed, Thys et al. [2019] were able to fool surveillance cameras by printing and carrying adversarial patches, which lowered accurate person detection significantly. It is easy to imagine how such adversarial examples could be used maliciously.

The existence of adversarial inputs to deep neural network (DNN) image classifiers has been extensively researched. Starting with Szegedy et al. [2013], it was shown that imperceptibly small perturbations to input samples can lead to misclassification. Most classical computer vision methods will correctly assign samples the same class membership if they are close in pixel space. Having these smooth decision boundaries seems to be lost for DNN classifiers for the benefit of a better generalization ability in more abstract feature spaces. Notably, Su et al. [2019] showed that a range of data sets and network architectures are susceptible to *single-pixel attacks*, where only a single input pixel is perturbed. Moosavi-Dezfooli et al. [2017] showed the existence of minimal but universal adversarial perturbations - image-agnostic pixel configurations that lead to misclassification of natural images w.h.p given a DNN classifier. These results imply that this vulnerability persists even when including such adversarial examples during training. While it might reduce vulnerability to some degree, a new set of perturbations can usually be found. In RL, adversarial inputs have been generated with the fast gradient sign method [Goodfellow et al., 2014] and successful black- and white-box attacks were shown to exist [Huang et al., 2017]. To the best of our knowledge, however, we are the first to propose and investigate attacks independent of the task defining reward function.

3 Background and Methodology

In reinforcement learning [Sutton and Barto, 2018] an artificial agent is trained to optimize its trajectories within an environment by maximizing the expected cumulative (discounted) reward of its policy. More specifically, in every time step t the agent chooses an action $a_t \sim \pi$ according to its

(possibly stochastic) policy π and transitions to a new state $s_{t+1} \sim P(s_t, a_t)$ based on the current state s_t, action a_t and environment dynamics P. Every transition yields a reward $r_t(s_t, a_t, s_{t+1})$ which implicitly defines the task the agent should fulfill. In our setup the agent policy $\pi(o_{1:t})$ depends on a sequence of observations $o_{1:t}$, where each observation o_t reveals partial information about the underlying state s_t. Once an agent is trained, we record the score $S = \sum_t r_t$ it achieves within an episode, i.e., before termination or before the maximum episode length of T time steps. The average score over multiple test runs gives a measure of the agent's performance. We aim at investigating the robustness of a trained agent by devising input perturbation attacks that are completely agnostic to the task the agent tries to fulfill. That is, we aim to minimize the evaluation score S *without* access to the rewards r_t by perturbing the observations o_t slightly. Further, we limit ourselves to per-time-step attacks. That is, we perform attacks on each input image separately, without knowledge of the agent's history or prior attacks. Note that this is a very limited attack model, which most agents implicitly train against by having some randomness in the policy throughout training - a few actions deviating from the policy should not negatively affect the score. We detail below 5 attack models in this setup: a random pixel change attack, a maximum entropy attack, an action change attack, a tunnel vision attack and a peripheral vision attack. A visual example of each attack can be seen in Figure 1. Note that the attacks with tunnel and peripheral vision are static across all frames.

Random Pixel Change: Our first attack is a naïve baseline attack in which we simply replace d pixels from the input observation o_t with random pixel values. A single random pixel value is a 5-tuple consisting of image coordinates and RGB-channels. See Figure 1a.

Maximum Entropy Attack: Akin to the approach presented in Su et al. [2019], we use differential evolution (DE) [Storn and Price, 1997] for our second attack to generate adversarial inputs. That is, we optimize a population of perturbation pixels with DE to maximize the entropy of the agents policy in the current time step. This attack is specific to policy gradient RL algorithms that approximate the policy directly. The attack requires the ability to query the agent for the policy based on an observation and afterwards resetting the internal state of the agent. However, it does not require any gradient information from the agent. The aim of this attack is to bring the agents policy close to uniform in every time step, thereby reducing the agents performance towards that of a random agent. See Figure 1b.

Action Change Attack: This attack is equivalent to the maximum entropy attack with the only difference that instead of maximizing the entropy, we instead minimize the likelihood of the most likely action given the unperturbed image. If successful, this attack drives the agent, on average, towards choosing an action different from the one it would have chosen otherwise, ultimately leading to a worse policy. See Figure 1c.

Tunnel Vision: Supported by the results from the action change attack, we investigate the insight that humans have a focal vision that RL agents do not have, to design our tunnel vision attack. Here we obfuscate the outermost pixels, essentially framing the remaining center pixels with a 2 pixel wide frame. In contrast to the previous two attacks, this attack is a constant alteration that does not require any knowledge about the agent. See Figure 1d.

Peripheral Vision: In a similar fashion, we also look at the effect of obfuscating observations in the center of the field of view. See Figures 1e and 1f.

4 Evaluation

For a first comparison of the proposed attacks we trained an IMPALA agent based using the code released by the authors[1] on the Deepmind Lab [Beattie et al., 2016] level "Rooms - Collect Good Objects" [Higgins et al., 2017]. The goal in this environment is to collect all ten good objects in an open room, each rewarding one point, while avoiding any of the ten bad objects, each yielding a reward of -1. An episode ends after collecting ten objects of either kind or after $T = 3600$ environment steps. In each step t, the agent receives a visual input of 72×96 RGB pixels as observation o_t. In Table 1 we compare the five different attacks against the baseline performance of the agent. For the DE attacks we allow a maximum of 25 pixels for perturbation and run the evolutionary algorithm for 300 iterations or until the objective value (entropy or target action likelihood, respectively) converges to the third decimal place for all population members. We chose this small threshold of 25 pixels,

[1] `https://github.com/deepmind/scalable_agent`

Attack	Avg. score	Min.	Max.	Avg. frames	Pixels	Episodes
None	9.89	8	10	977	0	400
Random	8.87	6	10	2917	656	400
Entropy max.	10.0	10	10	958	≤ 25	3
Action change	9.5	9	10	2686	≤ 25	2
Tunnel vision	4.25	-2	9	3532	656	400
Peripheral S	9.51	6	10	2253	638	400
Peripheral L	6.57	2	10	1900	1728	400

Table 1: Average game scores, min. and max. score as well as average frames needed for completion of the level "Rooms - Collect Good Objects" under different attack scenarios.

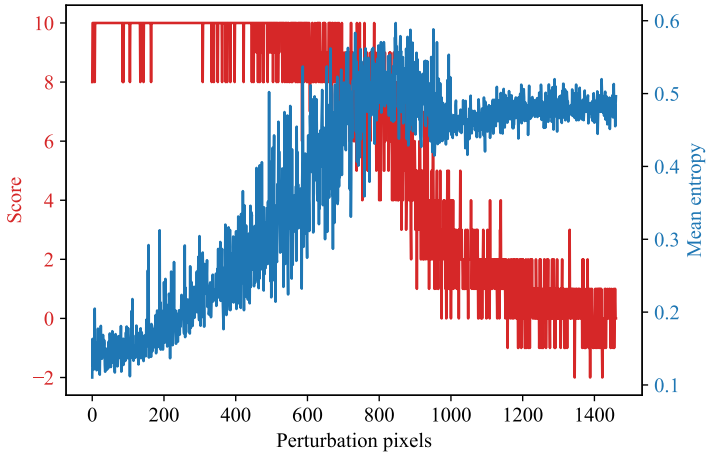


Figure 2: Game score and average entropy under random pixel changes

since already with this small amount of noise the attack is easily detectable by human investigation, as the altered pixel cause a flickering that humans can see. We refer to the videos provided to get an intuition[2]. In contrast, the frame given in the tunnel vision attack (see Figure 1d) seems natural, and a human observer unaware of the attack would most probably not suspect anything.

The results presented in Table 1 are quite compelling: Simply obfuscating the outermost pixels yields a much bigger performance drop than changing equally many pixels randomly all over the image. This suggests, that the agent heavily relies on its peripheral vision; a trait quite distinct from humans. This suspicion is strengthened by the results of the peripheral vision attack. With only the middle of the observation obfuscated, the agent performs better compared to tunnel vision and random attack. Further, the two attacks based on differential evolution (entropy maximization and action change) do not seem to have a significant effect on the score. However, the action change attack manages to prolong the episodes, which hints at a partial success.

We now look at each of the attacks in more depth, to get a better understanding of their inner workings.

4.1 Random Pixel Change Attack

To investigate further, how much of the input needs to be changed randomly, before the agent's performance declines, we varied the number of pixels changed and recorded the resulting score achieved as well as the entropy of the agent's policy in Figure 2.

The agent displays a steep decrease in score after 700 pixels, i.e., roughly 10% of the observation, have been randomized. At 20% (1400 pixels) the agent performs no better than a random agent. Meanwhile the entropy of the agents policy increases, but stays well below that of a uniform distribution which would be at $ln(9) \approx 2.2$ given the 9 possible actions of the environment. Since the IMPALA agent is

[2] `http://bit.ly/2kJXF0J`

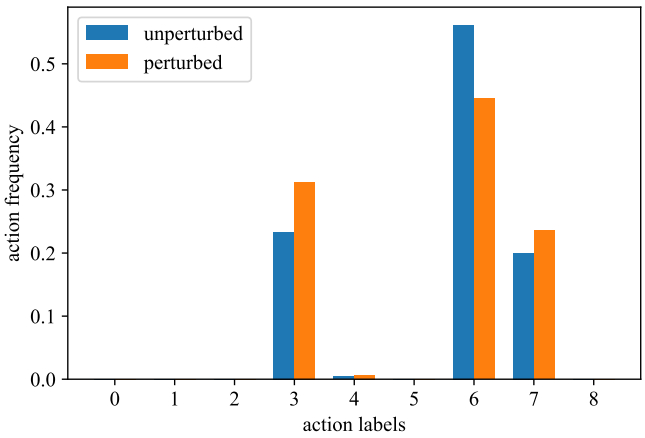


Figure 3: Comparison of how many times each action would have been performed over the 3 episodes under attack.

designed to be able to perform on all levels of DMLab-30 [Beattie et al., 2016], some of the actions are of little or no use in the instance "Rooms - Collect Good Objects", such as firing its weapon. It has therefore learned to completely ignore these here, independent of the input, which explains the lower entropy.

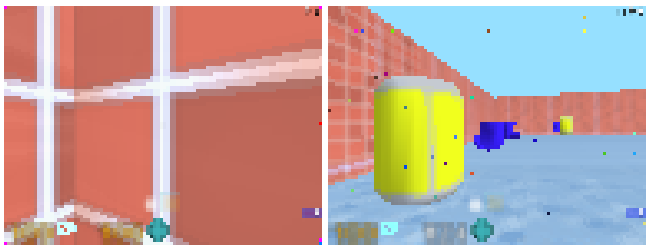
4.2 Entropy Maximization Attack

Next, we take a closer look at the entropy maximization attack, which seems to perform worse when compared to the other attacks. To get a better understanding of the attack's performance we plot the empirical action frequency of the perturbed and unperturbed policy in Figure 3. We note that the agent's policy does get less peaked, however, the attack is incapable of assigning probability mass towards actions that are never chosen by the agent: The agent has learned to robustly ignore these unuseful actions.

A similar conclusion - that the attack is working within its limits - can be drawn from the results in Table 2. The attack achieves an average entropy more than twice as high than the random attack with 25 pixels perturbed. But simply increasing entropy is not enough to mount a successful attack. Observations that lead to high entropy policies do occur naturally. This is also implied by the maximum values in Table 2. On the other hand, the attack is not able to manipulate all observations in a way that induce high entropy policies. This leads to a scenario, the agent is not unfamiliar with, where non-optimal actions can easily be amended in subsequent time steps, since the agent chooses its action stochastically from the policy. To further explain this notion, assume the attack achieves the worst result possible, i.e., a uniformly distributed policy. Depending on the size of the action space, there is still a high chance that the correct action will be chosen. Therefore it is unlikely, that the agent will get stuck in a loop of observations, that are susceptible to the attack. This is not the case for the action minimization attack, as we will explain in the following section.

	No attack	Random attack (25 px)	Entropy maximization
Mean Entropy	0.11	0.15	0.38
Std deviation	0.03	0.02	0.37
Min	$8 * 10^{-8}$	$4.9 * 10^{-8}$	$9.5 * 10^{-8}$
Max	1.06	1.16	1.23

Table 2: Comparison of entropy over 3 episodes under random and entropy max. attack.



(a) convergence (b) no convergence

Figure 4: side by side comparison of (4a) a successful adversarial example and (4b) a perturbed observation, where no adversarial example was found.

4.3 Action Change Attack

As can be seen in Table 1 the action change attack results were also not successful in hurting agent performance overall. The scores of both episodes fall within the same range, as for the unperturbed scenario. We now investigate when and where the action change attack works, and where it fails. First we note, that for almost all natural/unperturbed observations, the agent policy chooses a single action with high ($> 99.9\%$) probability. With this attack, we try to find adversarial inputs that minimize the probability of these actions. We observe two kinds of perturbations depicted in Figure 4. One would expect the noisier observation (4b) with 25 distinct perturbations to be more detrimental for the agent's performance. Counter-intuitively, if DE converges, i.e., a locally minimal set of pixel perturbations is found, only a few (< 10) locations are targeted.

As previously mentioned, agent policies that assign a single action high probability are the most common. Keeping this in mind, minimizing action probability for a given action will likely lead to another action being assigned high probability. Those examples will lie close to the natural input distribution. Therefore the most successful adversarial examples induce policies that assign high probability to another action., i.e., no more pixels than necessary are changed in the observation.

Interestingly, in both runs evaluated, the agent performs well initially and collects 9 out of 10 objects within the first 900 frames. We attribute this towards the simplicity of the game and ambiguity of the most used actions "look right, move forward" and "look left, move forward". Both allow the agent to move towards objects while at most compromising the speed, at which it is able to collect the objects. During one episode, the agent manages to reach maximum score only after 1772 frames. In the other episode it gets stuck in a loop of left right movements in the corner of the room and does not manage to collect any more objects for the remaining 2700 frames. This suggests that while the attack did not manage to decrease the score significantly, it did manage to slow the agent at solving the task, hinting at a partial success. A more quantitative analysis of this attack would reveal if such loops also occur at the start of an episode. The reason why this was not done is also a big draw back of this attack: Computational complexity of DE scales super-linearly with input dimensions and convergence is highly dependent on its input parameters [Chen et al., 2015]. Furthermore, it also requires access to the agents' policy and network states. An attack in real-time is therefore not feasible. However, by investigating the adversarial examples generated during an action change attack, we find that over 90% of all perturbations are found along the outermost pixels of the observation. This supports our results of the tunnel vision attack, where the outermost pixels are obfuscated consistently.

4.4 Tunnel Vision

In contrast to the targeted differential evolution attacks, our simple tunnel vision attack does not require any access to the agent nor access to large computational resources. To investigate whether the attack is specific to the setup investigated above or whether it also works in different environments and with different training setups for the IMPALA agent, we trained two more agents: One specialized on the level "Explore goal locations small" and one trained in the multi-task setup on all DMLab-30 tasks (see [Espeholt et al., 2018]).

Table 3 shows the result of these experiments, where the first two rows correspond to the two specialized agents evaluated on their respective environment, while the last 6 rows correspond to the multi-task agent evaluated on the same 2 tasks and 4 further environments.

Attack:	None	Rnd. (656 px)	Rnd (25 px)	Tunnel vision
Rooms collect good objects	9.89(0.16)	8.87(0.90)	9.96(0.28)	4.25(2.15)
Explore goal locations small	214(46)	19.63(19.52)	94.4(51.74)	155(18)
Rooms collect good objects	7.66(1.65)	3.1(2.23)	7.48(1.89)	7.0(1.8)
Explore goal locations small	154.8(64.04)	50.8(40.9)	134.6(60.04)	142.4(63.26)
R. exploit deferred effects	9.6(1.2)	10.51(7.8)	9.38(1.42)	9.76(0.95)
R. select non-matching obj.	7.84(21.86)	2.82(12.49)	7.5(21.68)	7.32(22.68)
Psychlab continous recogn.	31.68(4.69)	30.97(2.26)	31.32(4.72)	32.83(0.97)
Natlab varying map random.	31.06(12.51)	30.15(11.96)	23.0(11.24)	34.16(9.06)

Table 3: Evaluation score on 6 different DMLab-30 levels. The numbers represent the average score over 100 episodes. The value in parentheses represents the standard deviation. The first 2 levels are evaluated with an agent specifically trained for these levels, while the remaining 6 rows are evaluations of a multi-task agent trained on all DMLab-30 tasks. The columns labled "Rnd." give the results of random pixel attacks for comparison. "R." in the 5th and 6th line abbreviates "Rooms"

As can be seen in the table, our tunnel vision attack also leads to a performance drop for the agent specialized for the "Explore goal locations small" task. However, we observe that the agent for the level "Rooms collect good objects" proves to be the most vulnerable to the tunnel vision attack, while being highly robust to random pixel attacks in comparison.

We further see that the agent trained in the multi-task setup seems to be more resilient to attacks in general. This hints that the multi-task setup helps the agent to learn more robust feature representations. Note however that the baseline performance is less for this agent. This training therefore seems to provide a trade-off between robustness and peak performance.

5 Conclusion

We perform a range of pixel-attacks to evaluate the robustness of a state-of-the-art RL agent based on IMPALA [Espeholt et al., 2018]. We find that a simple *tunnel vision* attack heavily confuses the agent, whereas a human would not even notice it. This indicates that the agent did not develop a deep scene-understanding, but instead overfitted to a few pixels on the edge of its field of view. The *peripheral vision* attack provides more evidence for this, since the agent's performance is much less affected by removing a large chunk of input pixels from the center of the input. Interestingly, a human would struggle more with this latter attack.

The same *tunnel vision* attack affects the agent's performance much less in an exploration task. First, this tells that static attacks do not necessarily affect the agent in the same way on different tasks. Second, this also gives us insights into the nature of the tasks and how the agent learns to solve them. For example, peripheral vision seems to be important to detect collectable objects as they enter the field of vision, whereas it is less critical when navigating through a maze. Interestingly, while the random attack did not significantly affect the agent on the collection task, it was devastating on the exploration task. Investigating performance under random attacks as well as with tunnel vision, we note, that the multi-task agent is more robust to these attacks for all 6 tasks we evaluated. We believe this result is explained by the more robust multi-task representations learned. However, while performance in a single task setting is on par with human performance, the performance of a multi-task agent are well below human level. Therefore, gaining robustness against our attacks this way, comes at the cost of loosing performance overall.

We also investigated a non-static attack that perturbs different pixels in every frame using differential evolution. We found that fewer pixels have to be perturbed than with a random attack to see an effect. However, more experiments are necessary to get a more complete evaluation of this type of attack. One interesting direction for future research could be to use differential evolution to find pixel-perturbation patterns that generalize across frames, and maybe even across tasks. This would allow to run the expensive algorithm once, and then use the found adversarial inputs as static patterns to attack an agent in real-time.

Finally, getting a better understanding of the vulnerabilities of state of the art RL agents can help us to develop new training procedures that will hopefully result in more robust agents.

References

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety, 2016.

Charles Beattie, Joel Z. Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, Julian Schrittwieser, Keith Anderson, Sarah York, Max Cant, Adam Cain, Adrian Bolton, Stephen Gaffney, Helen King, Demis Hassabis, Shane Legg, and Stig Petersen. Deepmind lab, 2016.

Stephen Chen, James Montgomery, and Antonio Bolufé-Röhler. Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution. *Applied Intelligence*, 42, 04 2015. doi: 10.1007/s10489-014-0613-2.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1480–1490. JMLR. org, 2017.

Sandy H. Huang, Nicolas Papernot, Ian J. Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *ArXiv*, abs/1702.02284, 2017.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies, 2015.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. doi: 10.1109/cvpr.2017.17. URL `http://dx.doi.org/10.1109/CVPR.2017.17`.

Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, 11(4):341–359, December 1997. ISSN 0925-5001. doi: 10.1023/A:1008202821328. URL `https://doi.org/10.1023/A:1008202821328`.

Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, page 1–1, 2019. ISSN 1941-0026. doi: 10.1109/tevc.2019.2890858. URL `http://dx.doi.org/10.1109/tevc.2019.2890858`.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL `http://incompleteideas.net/book/the-book-2nd.html`.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2013.

Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: adversarial patches to attack person detection, 2019.