Visual Diagnostics for Deep Reinforcement Learning Policy Development

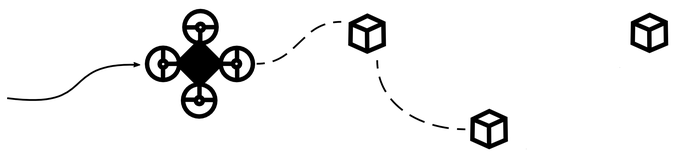
Jieliang Luo, Sam Green, Peter Feghali, George Legrady, and Çetin Kaya Koç
University of California, Santa Barbara
https://arxiv.org/abs/1809.06781, jieliang@ucsb.edu

UC SANTA BARBARA

K Ç
L A B

Background

Reinforcement learning (RL) is a family of methods aimed at training an **agent** to collect rewards from an environment through trial-and-error approaches. Since the deep Q-network (DQN) algorithm was introduced in 2013, there has been a surge of interest in using convolutional neural networks (CNN) in vision-based RL algorithms. In the context of cyber-physical systems, vision-based RL has exciting potential to provide high levels of autonomy in applications like robotics, self-driving cars, and infrastructure inspection. However, CNNs are known to be opaque to debugging and RL's emphasis on trial-and-error demands rigorous behavioral verification before they may be allowed control over safety-critical cyber-physical systems. This work adapts CNN visualization techniques to the domain of RL.



Vision-based RL algorithmic advancements are typically introduced in the context of simple games, e.g. benchmarking with the Arcade Learning Environment. In this work we modified Microsoft's AirSim, which is a photorealistic 3D simulator based on the Unreal Engine. We modified AirSim to support RL[1]. Our agent was a drone with three available actions: forward, left, and right. These actions had deterministic effects. At the beginning of each episode, the drone would be reset and cubes would be randomly distributed in front of it.

Reinforcement learning approach

The agent received a reward of +1 for each cube that it "collected". The goal for our agent was to collect as many rewards as possible each episode. The agent's policy was a CNN parameterized by weights θ, so the goal was to solve the following optimization problem:

$$\theta^\star = \arg\max_\theta \sum_{t=0}^{T-1} r(s_t, a_t),$$

where s_t and a_t are the states and actions at time t and $r(s_t, a_t)$ is the reward obtained from the environment for the given state and action. For this simple problem, the REINFORCE algorithm was adequate.

Visualization methods and results

CNN visualizations are useful for identifying strengths and weaknesses in a trained network. For example, the *class visualization* for GoogLeNet's "saxophone" class indeed extracts a saxophone shaped object from the network. That is, the method generates an image, which, when input into the trained GoogLeNet CNN, will maximize the output probability of the saxophone class. However, when looking at the generated image, one can clearly see that the outline of a man has also been extracted from the network! Thus, CNN visualizations provide insight into what a network has learned to pay attention to, and a human can then determine if modifications are required.

In the context of RL, existing CNN visualization techniques attempt to cluster inputs according to their resulting action, provide decision attribution, or visualize canonical actions. Techniques considered here include:

- t-SNE maps – Clusters similar inputs by the actions they trigger.
- Attribution visualization – Identifies image regions most responsible for an action decision.
- Action visualization – Generates inputs which trigger specified actions.

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a dimensionality reduction algorithm developed by [1]. It is well suited for visualizing high-dimensional datasets. The method positions each high-dimensional datapoint (e.g. image) in a two or three-dimensional map in a way that similar datapoints are nearby and dissimilar ones are distant. The most recent use of t-SNE is to use a trained convolutional neural network (CNN) to extract features from each image, feed the features to t-SNE to get the position of each image, and arrange the images on a 2D or 3D space based on the given positions.

The figures below are t-SNE visualizations of three policies: a high-performance, poor-performance, and right-and-forward-only policy. The tinting of each patch is based on the action taken by the policy, given the drone's observation: red indicates "forward", green indicates "left", and blue indicates "right". By studying the t-SNE outputs it is possible to identify inputs which cause correct or incorrect behavior.

Figure 1. High-performance policy

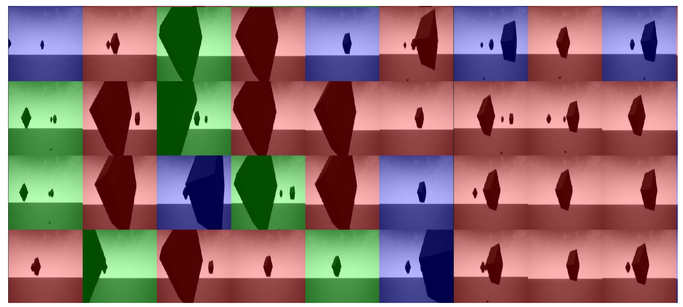


Figure 2. Poor-performance policy

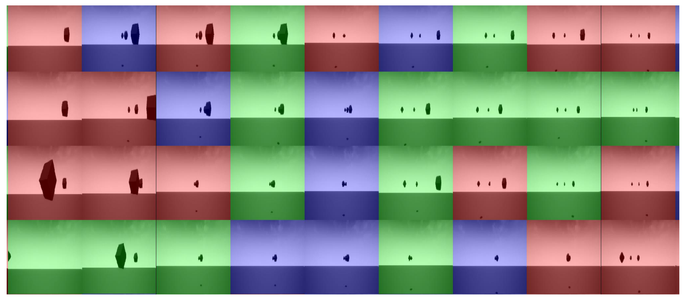
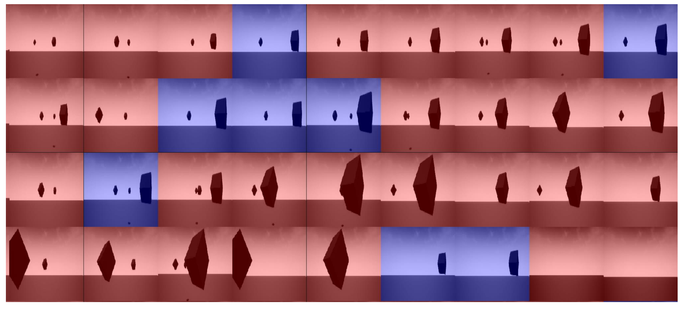


Figure 3. Forward-and-right-only policy



Attribution visualization techniques highlight regions in an input which are most responsible for a particular action in a CNN-based policy. We will use an attribution visualization technique called **Gradient-weighted Class Activation Mapping** (Grad-CAM) [2], which highlights regions in the input image most responsible for an action probability.

1. New versions of AirSim now have native RL support.

The high-performance policy's class visualization in Fig. 4 clearly explains what the policy is looking for, where the bias toward the "left", "forward", or "right" depends on the position of the cube. Similarly, Figs. 5 and 6 provide insights into the poor and forward-and-right-only policies.

Figure 4. High-performance policy

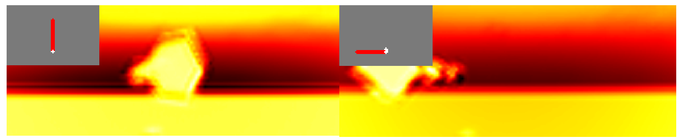


Figure 5. Poor-performance policy

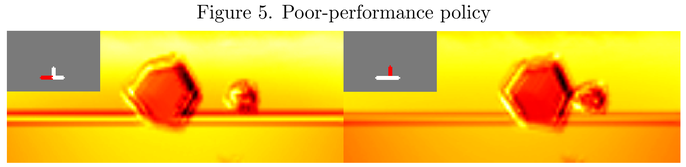
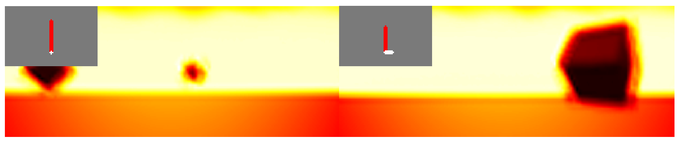


Figure 6. Forward-and-right-only policy



Action visualization methods generate visual inputs which activate a particular output in a *trained* neural network. This approach allows for a high-level of human comprehension about the behavior of a network, rather than treating the network like a black-box model. For our specific feature visualization approach, we use **Class Model Visualization** (CMV) [3].

Action visualizations for the forward-and-right-only policy Fig. 9 highlight one of the challenges in reinforcement learning. In this case, the drone experienced an early success by moving "right" and "forward", which resulted in the elimination of "left" action probabilities. The remedy for this was to lower the learning-rate of the policy updates.

Figure 7. High-performance policy

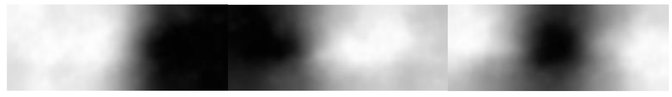


Figure 8. Poor-performance policy

Figure 9. Forward-and-right-only policy

[1] L. V. D. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.
[2] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradientbased localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 618–626
[3] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," arXiv preprint arXiv:1312.6034, 2013

Arxiv paper: