Project: Adversarial Examples in Reinforcement Learning and Imitation Learning

PI: Pieter Abbeel
Student: Sandy Huang

Background
Recent advances in deep learning and deep reinforcement learning (RL) have made it possible to learn end-to-end neural network policies that map directly from raw inputs (e.g., images) to a distribution over actions to take.

However, neural networks are vulnerable to adversarial-example attacks. An attacker constructs an *adversarial example* by adding a small worst-case perturbation to an existing input, to cause the targeted model to produce an incorrect output for the new input. Prior work primarily focuses on adversarial examples for neural network classifiers trained with supervised learning. For instance, image classification networks are easily fooled by adversarial examples, even when the original inputs and corresponding adversarial examples look essentially indistinguishable to humans. Additionally, these examples are *transferable*: an adversarial example designed to be misclassified by one model is often misclassified by other models trained to solve the same task.

Real-world adversaries often must operate in the black-box setting, with no access to the targeted model’s parameters. Transferability of adversarial examples enables an adversary to launch successful black-box attacks by training its own model for the same task, and attacking the targeted model with adversarial examples generated against its own model.

Progress
We have found that certain types of adversarial examples are transferable across policies trained with deep RL. In fact, this even holds when the two policies are trained with different deep RL algorithms. This enables adversaries to attack deep RL agents even when they have no access to the agent’s policy parameters (Figure 1), which is likely in real-world situations.

We also observe the presence of memory in recurrent network policies creates opportunities for novel attacks with a delayed effect on the targeted policy. We show it is possible to efficiently construct adversarial examples of this type, which we call *dormant* adversarial examples (Figure 2). Such attacks are particularly powerful because the adversary does not need access to the model’s input at the time that the adversary wishes for the agent to perform an adverse action; it is sufficient for the adversary to access the model’s input at an earlier time. This delay also makes dormant attacks harder to detect and defend against.

Videos at [http://rll.berkeley.edu/adversarial/](http://rll.berkeley.edu/adversarial/)
Figure 1: The adversary does not know which deep RL algorithm was used to train the targeted policy, so the adversary trains its own policy (with A3C) for Chopper Command, and attacks the targeted policy (trained with DQN) with adversarial examples computed against its own policy. This adversarial example is computed using the fast gradient method (FGM), with an \( l_2 \)-norm constraint on the adversarial perturbation. The perturbation is interpretable: a pixel is changed in the miniature global map at the bottom of the game screen, to obscure the enemy’s position. The green arrows indicate the direction the enemy is traveling in.

Figure 2: Dormant adversarial examples generated for a policy trained to navigate an I-Maze. It is possible to introduce a dormant adversarial example after the agent has seen and memorized the (green or yellow) marker, to fool the agent into believing it saw the other marker instead. In these examples, a single dormant adversarial example is introduced when the agent is in the gray square (marked in the top-down views on the left). Note that the green, yellow, red, and blue squares in the top-down view correspond to colored floor tiles in the maze environment (the gray squares do not).