Latent Representation of Input: A Defense Against Adversarial Examples in Deep Q Networks

Bibek Poudel [1]

Abstract

Similar to models in Computer Vision and Natural Language Processing, Deep Reinforcement Learning (DRL) policies are also known to be vulnerable to adversarial examples. This presents serious concerns about widespread adoption of such policies in safety critical applications. In this work, we adopt techniques from image classification namely, latent representation of inputs, more specifically, feature squeezing, to nullify the effects of the modifications in observations caused by a white-box adversary. We modify the input processing pipeline such that the agent is successfully able to choose correct actions even when the observations are perturbed. (See Appendix for videos).

1. Introduction

Reinforcement Learning (RL) has achieved superhuman performance in recent years in control and manipulation tasks that often require long-term planning, such as autonomous driving and Atari games. These advancements rely on RL policies capable of learning directly from observations to produce a distribution over actions to take. Such policies are often parameterized by neural networks (Riedmiller, 2005).

The threat of adversarial examples (Szegedy et al., 2013) to neural networks has been widely studied in the context of Machine Learning (ML) classifiers in images and language. Adversarial examples have been shown to degrade the test-time performance of these classifiers. More recently, they have also been demonstrated to degrade the test-time performance of trained Reinforcement Learning (RL) policies (Huang et al., 2017), i.e., an adversary can stealthily modify the observations and cause a trained policy to take incorrect actions. It is essential to address the threats that adversarial examples pose before the widespread adoption of RL policies in critical tasks.

Although there has been a significant amount of work in crafting adversarial attacks on RL, the number of defenses thus far is sparse. In this work, we apply the latent representation (feature squeezing by bit depth reduction) of input observations (Xu et al., 2017) and embed this as a function block in the input processing pipeline of a trained policy to defend the policy against adversarial attacks. This is an application of existing techniques from the image classification domain, where an adversarial example detector was built using this technique, to RL and more specifically to Deep Q Networks (DQN).

We successfully demonstrate that this technique can defend against an adversarial attack crafted using the Fast Gradient Sign Method (FGSM) on a DQN agent trained to play Pong in the Arcade Learning Environment (ALE) (Bellemare et al., 2013).

2. Related Work

In this section, we first introduce existing work on adversarial examples in RL, then we discuss the use of feature squeezing as a defense against adversarial examples in image classification.

2.1. Adversarial examples in Reinforcement Learning

The vulnerability of Reinforcement Learning to adversarial examples was first exposed by Huang et al.(Huang et al., 2017), where they discovered that adversarial examples can cause a significant drop in the performance of policies trained using state-of-the-art algorithms, such as Deep Q Networks (DQN), Trust Region Policy Optimization (TRPO), and Asynchronous Actor Critic (A3C). They performed their experiments in both white-box and black-box scenarios to further show that adversarial examples crafted for one algorithm can transfer to another algorithm as well. Other attack techniques such as the ”Strategically timed attack”(Lin et al., 2017), where adversarial examples are crafted only for certain time steps independent of all other time steps, have been developed since then. More recently, adversarial examples have also been crafted for automatic path planning tasks (Liu et al., 2017), (Bai et al., 2018).

Although defense techniques like Adversarial training and Defensive distillation have been widely used and validated for image classification, they have not been widely adopted

Reinforcement Learning class project, fall 2021 [1]University of Memphis. Correspondence to: <bpoudel@memphis.edu>.

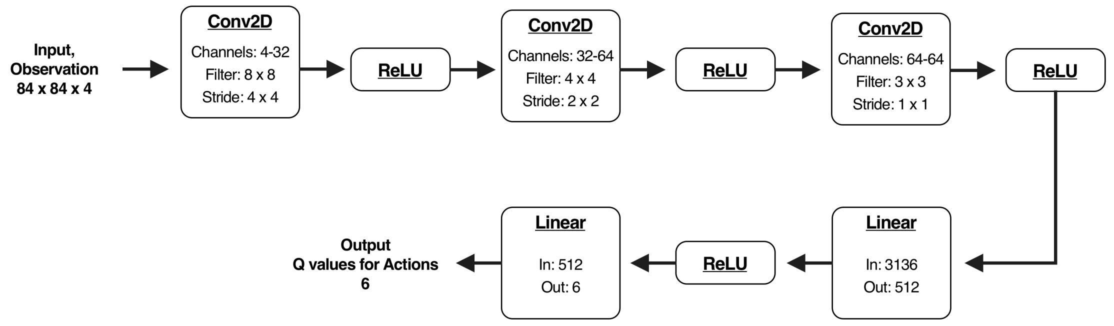

Figure 1. Architecture of the DQN. The input observation of four stacked 84×84 pixels is passed through Convolutional layers and Linear layers before mapping them to Q values for the six actions to be performed.

for adversarial examples in RL (Chen et al., 2019). Traditionally, there have been two techniques to defend against adversarial examples: one is to build a detector for adversarial examples, and the other is to use techniques to make it harder to construct adversarial examples (usually by modifying the objective function). In this work, we use neither of these techniques. Instead, we add a defensive functional block that can enable the trained agent to make correct decisions in the presence of an adversary.

2.2. Feature squeezing in image classification

Feature squeezing was proposed as a detector of adversarial examples by Xu et al. (Xu et al., 2017), where they compare the prediction of a DNN on a regular input as well as the feature squeezed input. The given input is detected as an adversarial example if the difference in the prediction vector, as measured by the L1 distance between the prediction on the regular input and the prediction on the feature squeezed input, is above a predefined threshold. The techniques used by the authors for feature squeezing are bit depth reduction and spatial smoothing, and they validated this to be an effective defense technique against adversarial examples crafted using seven different techniques. Although feature squeezing has been used in image classification to detect adversarial examples, it has not been used in RL to robustify a trained model.

3. Preliminaries

In this section, we briefly introduce the policy learning algorithm that was attacked, the concept of adversarial attacks, the algorithm used to generate adversarial examples, and the defense technique.

3.1. Deep Q Networks (DQN)

Introduced in (Mnih et al., 2013) as a variant of the Q learning algorithm, DQN is a generalized approximation of the Q function that achieved state-of-the-art performance in Atari

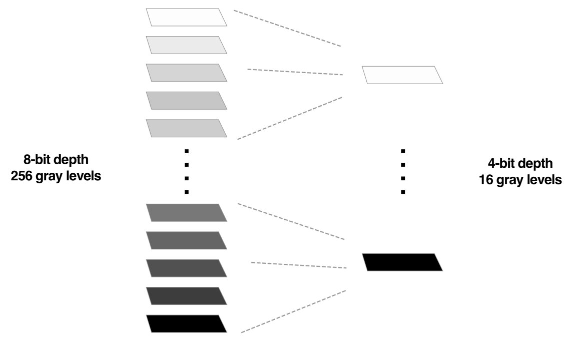

Figure 2. Feature squeezing by pixel bit depth reduction. The input observation typically represented by 8-bit pixels, i.e., 256 levels, is reduced to 4-bit pixels, i.e., 16 levels.

2600 games from the Arcade Learning Environment. During training, DQN learns to map the raw observations (input pixels) to the distribution of actions to take. Once trained, a policy is obtained that can select the optimal action given an observation. One of the games where it achieved superhuman performance was Pong.

The architecture of the DQN, which is trained to play Pong and craft adversarial examples in this work, is shown in Fig. 1. Here, the input is a stack of four 84×84 pixels which is passed through different layers (Convolutional, Activations, Linear) and finally mapped to the Q values of the six actions to take, namely: FIRE, NOOP, RIGHT, RIGHTFIRE, LEFT, and LEFTFIRE. Finally, the action taken is the one with the highest Q value, and the policy that is learned is obtained by greedily taking the action with the highest Q value for each observation.

3.2. Adversarial Attack

An adversarial attack on a deep neural network (f) is conducted through crafting an adversarial signal (x^*) by imposing minimal perturbation (δ_x) to the original signal (x). The perturbed signal (x^*) can be obtained via solving the

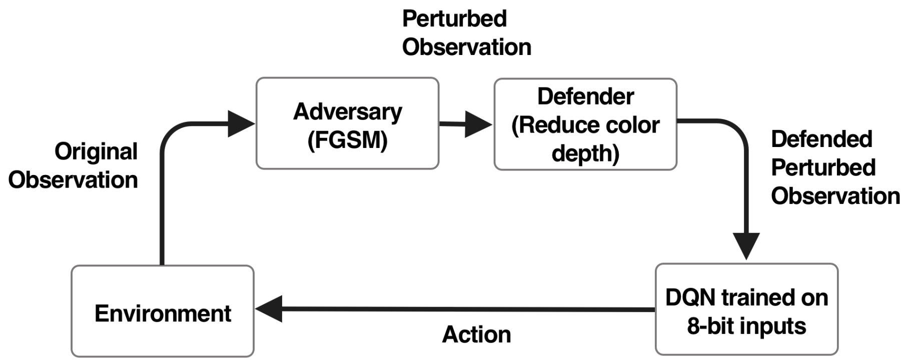
Perturbed Observation
Original Observation
Adversary (FGSM)
Defender (Reduce color depth)
Defended Perturbed Observation
Environment
Action
DQN trained on 8-bit inputs

Figure 3. Systematic diagram of the attack and defense. An adversarial example is generated from the observation obtained from the environment, which is then defended before it is fed as input to the trained DQN.

following optimization problem (Szegedy et al., 2013):

$$\begin{aligned} \text{minimize} \quad & D(\mathbf{x}, \mathbf{x} + \delta_{\mathbf{x}}) \\ \text{s.t.} \quad & f(\mathbf{x}) \neq f(\mathbf{x}^*), \end{aligned} \tag{1}$$

where D is a distance metric such as L_0, L_2, or L_∞. Multiple methods can be used to solve Eq. 1, one of them is the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014). FGSM is designed to produce an adversarial signal in a fast, non-iterative manner. It computes the gradient of the cost function J with respect to input $\mathbf{x}$ and scales the sign of the gradient by an L_∞ constraint to generate $\delta_{\mathbf{x}}$:

$$\mathbf{x}^* = \mathbf{x} + \varepsilon \cdot \text{sign} \left(\nabla_x J \left(\theta, \mathbf{x} \text{```} latex(2) \right. \right.$$

where θ represents the parameters of f; y is the target; and ε is the L_∞ constraint parameter controlling the attack magnitude. Increasing the value of ε will increase the attack effectiveness but also make the adversarial example more distinguishable (i.e., having a large $\delta_{\mathbf{x}}$).

3.3. Feature Squeezing using bit depth reduction

Although neural networks (differentiable models) assume the inputs to them to be continuous, typically, images are quantized to 8-bit representations in digital form, i.e., pixels in each channel have 256 levels leading to 24-bit color in the RGB scheme and 8-bit color in the Grayscale scheme. Xu et. al. (Xu et al., 2017) first hypothesized that reducing the color depth of the input representations can reduce the opportunity for an adversary to cause harm without affecting the classifier.

As shown in Fig. 2, feature squeezing coalesces a number of similar inputs into a single one, reducing the high dimensional space of input to its latent representation. Since adversarial examples work by adding fine details to a given input, in this work, the reduction in pixel depth is expected to eliminate its effects.

4. Methodology

The defense is implemented against a non-adaptive white-box adversary, i.e., one who knows everything about the model but not the defense technique that is being implemented. The goal of the adversary is to modify the observation to cause a trained model to produce an incorrect action, whereas the goal of the defense is to eliminate the effects of changes produced by the adversary, i.e., to make the agent take the correct action in the presence of the adversary.

4.1. Training DQN

The reward in the game of Pong is defined as $+1$ if a round of the game is won and -1 if a round of the game is lost. Each episode is composed of 21 rounds, i.e., any player can get a maximum reward of $+21$ and a minimum reward of -21. The DQN is trained with experience replay following the algorithm given in (Mnih et al., 2013). As a pre-processing step, to reduce the computational load, the input observation obtained from the ALE, which contains RGB frames of size 210×160, is grayscaled, cropped to the relevant area of the frame, and resized to 84×84.

Other training hyperparameters are: discount factor $= 0.99$, learning rate $= 10^{-4}$, batch size $= 32$, replay buffer size $= 10000$ transactions, optimizer = Adam (Kingma & Ba, 2014), loss = Mean Squared Error (MSE). Further, the agent starts training with a 100% probability of exploration, which gradually decays to a minimum of 2% probability of exploration as it achieves better performance. The training is terminated when the mean reward of the last 21 games is $+19$.

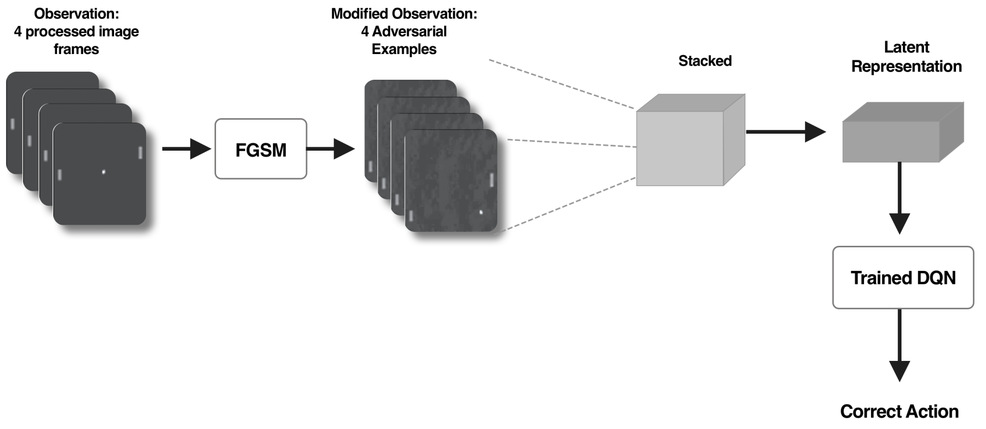

Figure 4. Defense mechanism embedded in the input processing pipeline. The 'Latent Representation' functional block performs the feature squeezing step on observations that were perturbed by the adversary.

4.2. Generating Adversarial Examples for trained DQN

The white-box adversary has access to the input processing pipeline as shown in Fig. 3. Adversarial examples are generated using FGSM for each preprocessed frame obtained from the environment. The perceptibility value is set to 0.003 and the loss function maximized by the adversary is MSE. When perturbations are added to the observation, to a human observer, the two images are indistinguishable.

4.3. Using Feature squeezing as a defense

Feature squeezing by color bit depth reduction is applied to image frames perturbed by the adversary. The defense mechanism is represented as the 'Latent Representation' block, which is embedded in the input processing pipeline of the DQN as shown in Fig. 4. The latent representation essentially works as a filter that prevents the tiny perturbations from being processed at the input.

5. Experiments

In this section, we first describe the experiment setup, then present and discuss the experiment results. All experiments are conducted using an Intel(R) Core(TM) i7-10700 CPU, an Nvidia RTX 2080 SUPER GPU, and 32G RAM. PyTorch (Paszke et al., 2017) is used to implement the DQN and Advertorch (Ding et al., 2019) is used to generate adversarial examples.

After training the DQN until it met the convergence criteria defined in Section 4, it was run for a total of 20 episodes. The average reward collected by the agent is given in Table 1. The agent can take correct actions for almost all the rounds of games in every episode.

Following the performance on standard observations, adversarial perturbations were added to the input observation frames to generate modified observations, as shown in Fig. 4. Table 2 shows the performance of the trained DQN on modified (adversarial) observations. The adversary can significantly degrade the performance of the trained agent, causing it to lose every round of the game in every episode.

The modified observations from the adversary are then passed to a functional block where the input features are squeezed to 4-bit pixel depth reduction before being passed as observations to the DQN, which was trained on 8-bit observations. Table 3 shows the performance of the DQN on these defended inputs. Despite the presence of an adversary, the agent can perform well enough to win almost all rounds of the game in every episode.

Number of Episodes	Average Reward
20	+19.75

Table 1. Performance of a trained DQN on standard observations. As seen from the reward, the agent wins almost all rounds of the game in every episode.

Number of Episodes	Average Reward
20	-21

Table 2. Performance of a trained DQN on adversarial observations. The agent loses all rounds in every episode of the game it plays.

Number of Episodes	Average Reward
20	+21

Table 3. Performance of a trained DQN in the presence of a 'Latent representation' functional block. The average reward obtained per episode is similar to that with standard inputs.

6. Conclusion and Future Work

In this work, we embedded a functional block into the input processing pipeline of a DQN. This block performs the latent representation of the input and acts as a filter to nullify the effects of a white-box adversary. Please refer to Section A for videos demonstrating the performance. In the future, we plan to use latent representation techniques other than feature squeezing, such as Autoencoders, to address the same problem. Additionally, we plan to explore other problems as well.

References

Bai, X., Niu, W., Liu, J., Gao, X., Xiang, Y., and Liu, J. Adversarial examples construction towards white-box q table variation in dqn pathfinding training. In *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pp. 781–787. IEEE, 2018.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Chen, T., Liu, J., Xiang, Y., Niu, W., Tong, E., and Han, Z. Adversarial attack and defense in reinforcement learning-from ai security view. *Cybersecurity*, 2(1):1–22, 2019.

Ding, G. W., Wang, L., and Jin, X. AdverTorch v0.1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., and Abbeel, P. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lin, Y.-C., Hong, Z.-W., Liao, Y.-H., Shih, M.-L., Liu, M.-Y., and Sun, M. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748*, 2017.

Liu, J., Niu, W., Liu, J., Zhao, J., Chen, T., Yang, Y., Xiang, Y., and Han, L. A method to effectively detect vulnerabilities on path planning of vin. In *International Conference on Information and Communications Security*, pp. 374–384. Springer, 2017.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Riedmiller, M. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pp. 317–328. Springer, 2005.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Xu, W., Evans, D., and Qi, Y. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

A. Appendix A

Click on the items below for videos at various stages of input and the DQN performance:

- Standard and pre-processed inputs
- Performance of trained DQN on standard inputs
- Performance of trained DQN on inputs perturbed by adversary
- Performance of trained DQN on defended inputs