Defending against adversarial attacks on deep reinforcement learning agents

Himani Arora
Columbia University
ha2434@columbia.edu

Rajath Kumar
Columbia University
rm3497@columbia.edu

Jason Krone
Columbia University
jpk2151@columbia.edu

October 16, 2017

Abstract

We introduce a method for defending against strategically timed adversarial attacks on reinforcement learning agents trained by deep reinforcement learning. Strategically timed attacks aim to minimize the agents reward by attacking the agent with an adversarial example a small fraction of the time. They do this by perturbing the state in such a way as to increase the agents probability of taking an action which is estimated to have low return. To detect this attack we predict what we expect the state to be at a given time step and look for large discrepancies in the action distribution for the predicted state and the "actual" state. We then evaluate a variety of action selection strategies to combat the detected attack.

Keywords: Adversarial Examples; Deep Reinforcement Learning; Generative Adversarial Networks.

1 Related Work

Atari 2600 games provide challenging environments for Reinforcement Learning(RL) because of high-dimensional visual observations, partial observability, and delayed rewards. Approaches that combine deep learning and RL have made significant advances [Mnih et al. (2013, 2015); Guo et al. (2014)]. Specifically, DQN [Mnih et al. (2013)] combined Q-learning [Watkins and Dayan (1992)] with a convolutional neural network (CNN) and achieved state-of-the-art performance on many Atari games.

Although there has been several attempts to modeling videos, building a generative model that predicts the future frame is still a very challenging problem because it often involves high-dimensional natural-scene data with complex temporal dynamics. Thus, recent studies have mostly focused on modeling simple video data, such as bouncing balls or small patches, where the next frame is highly-predictable given the previous frames [Michalski et al. (2014); Lotter et al. (2016)]. In many applications, however, future frames depend not only on previous frames but also on control or action variable. Guo et al. (2014) used the Arcade Learning Environment (ALE) emulator for making action-conditional predictions with a Monte-Carlo tree search method [Kocsis and Szepesvári (2006)], to generate training data for a fast-acting CNN, which outperformed DQN on several domains. Our approach is mainly motivated by the idea of building a predictive model for vision-based RL problems introduced by Schmidhuber and Huber (1991). They proposed a neural network that predicts the attention region given the previous frame and an attention-guiding action. Building on the above, Lenz et al. (2015) proposed a recurrent neural network with multiplicative interactions that predicts the physical coordinate of a robot. In our problem statement, as Atari games, can involve tens of objects with one or more objects being controlled by the actions directly and many other objects being influenced indirectly, can involve entry and departure of objects, Oh et al. (2015) introduces two neural network variants, which solves the above issues. Our frame prediction model is based on Oh et al. (2015).

Adversarial attacks on DNN's has been explored largely in the last few years, following Szegedy et al. (2013) several methods for attacking DNN's have been proposed. Moosavi-Dezfooli et al. (2016) estimated linear decision boundaries between classes of a DNN in the image space and iteratively shifted an image toward the closest of these boundaries for crafting an adversarial example. Majority of the proposed methods generated an adversarial example via seeking a minimal perturbation of an image that can confuse the classifier [Goodfellow et al. (2014a); Kurakin et al. (2016)].

Adversarial attacks on deep RL agents has been recently established, Huang et al. (2017) proposes uniform attack, which attacks a deep RL agent with adversarial examples at every time step in an episode for reducing the reward of the agent. Lin et al. (2017) introduces strategically timed attack which is similar to the uniform attack proposed in Huang et al. (2017) but achieves same effect by attacking four times less often on average. Lin et al. (2017) also introduces enchanting attack which claims to be the first planning based adversarial attack to misguide the agent toward a target state. Since RL agents can be seen as being deployed to carry out certain high risk tasks such as autonomous driving, it is important to design algorithms that are able to handle adversarial attacks in order to reduce the chances

of mishaps. With our project, we aim to explore one such strategy towards achieving this goal.

2 Approach

2.1 Preliminaries

In this work we assume that the adversarial attack against the agent is constructed using a strategically timed attack as put forward by Szegedy et al. (2013). This attack modifies the image x input to a neural network f by solving this optimization problem:

$$\min_{\delta} D_I(x, x+\delta)$$
$$\text{subject to} f(x) \neq f(x+\delta)$$

where D_I is an image similarity metric. In our experiments, x is a sequence of four Atari game video frames and f is a Q network or policy network. The attack is carried out when the output $f(x)$ indicates that the agent has a large preference for taking a particular action $a^\star$ because this indicates that $a^\star$ will likely lead to large return and taking any other action will lead to low return. The attacker can exploit this fact by perturbing x by a small amount δ such $f(x) \neq f(x+\delta)$ i.e. such that the agent no longer takes action $a^\star$.

2.2 Detection

To detect a strategically timed attack we predict what we expect the state $s = x$ to be at a given time step and look for large discrepancies in the action distribution for the predicted state and the "actual" state. We predict the state $\hat{s}_t$ we expect to be input to f at time step t using the video prediction network g put forward by Lin et al. (2017) conditioned on the history $h_t = \{s_{t-1}, a_{t-2}, \ldots, a_0, s_0\}$ of previous frames and actions. Then to determine if there is an attack at time step t, we compare the difference in action distributions produced by $f(s_t)$ and $f(\hat{s}t_t)$ using the KL divergence. In summary, our procedure is as follows for each time step t:

1. predict the expected state $\hat{s}_t = g(h_t)$
2. compute policies for predicted state and current state $\hat{\pi}(\cdot|\hat{s}_t) = f(\hat{s}_t)$, $\pi(\cdot|s_t) = f(s_t)$
3. determine that there is an attack if KL $(\hat{\pi}(\cdot|\hat{s}_t) \parallel \pi(\cdot|s_t)) > c$

where c is a tuned threshold value.

We will evaluate the success of a number of methods for selecting action a_t based on $\hat{\pi}()$ and $\pi()$. Further details are given in the experiments section.

3 Experiments

We will evaluate our proposed method on the Atari games such as MsPacman, Freeway or Pong. We will train our agents on our adversary-resilient versions of A3C and DQN algorithms. We intend to use the fast gradient sign method [Goodfellow et al. (2014b)] for generating adversarial examples. However, if possible, we will also test against the method proposed by Carlini and Wagner (2017), which has been shown to resist several anti-adversarial attack methods. Since our focus in this paper is to mainly handle strategically-timed attacks, we will implement the method suggested by Lin et al. (2017) to decide when to attack the agent. Following the pattern from previous papers on adversarial attacks on policy gradients Huang et al. (2017); Lin et al. (2017), we will use average return as the metric for comparing the effect of the attack on the agent. Here, average return is the average cumulative reward across ten rollouts of the target policy. An agent is resistant to the adversary if its average reward remains relatively unchanged even in the presence of the attack.

We will run experiments and compare the average return both in the presence and absence of strategically-timed attacks on an agent trained using the proposed algorithm under the following scenarios:

- Compare performance for each of the following cases when an action is predicted using:
  - Both the input and predicted state at every time-step.
  - Only the predicted state when we believe an adversary may attack while using the input state all other times.
  - Using both the input and predicted state when we believe an adversary may attack while using the input state all other times.

  By doing this, not only can we judge if our proposed algorithm is effective in dealing with adversarial attacks but can also evaluate which setting is most resistant. In addition to this, since the video generation pipeline is expected to increase the computational load, therefore, by modifying the frequency at which we use its input we can make trade-offs between computation and average rewards.
- Evaluate performance against the average portion of time steps in an episode that an adversary attacks the agent. This is referred to as the attack rate which can be tuned by changing the threshold value for strategically-timed attack. A lower threshold means a higher attack rate.
- Measure performance for different values of ϵ in FGSM adversarial examples. A small value of ϵ corresponds to a small amount of perturbation to the input. This will help us understand the limit of our method since the greater the perturbation, the more we must rely on our video prediction model for sampling actions.
- A simple baseline model can be designed by using adversarial examples while training the agent which acts as a regularizer as done by Kos and Song (2017). An important

experiment will therefore be to compare the performance of our proposed method with this baseline under similar attack conditions.

- Compare which of the two modified algorithms DQN and A3C are more resistant to adversarial inputs and under which conditions.
- If time permits, we will also test our method for multiple games as it will be interesting to see for which game our proposed method works the best. We hypothesize that in games such as Seaquest in which there are many random and new objects, the predicted frames will be less reliable and hence may show comparatively poor performance.

4 Milestone Deliverables

We expect to achieve the following items by the project milestone date of Nov. 20.

1. Video prediction network tested and producing output frames
2. Vanilla A3C running on selected Atari games
3. Vanilla DQN running on selected Atari games
4. Fast gradient sign method implemented
5. Results from Lin et al. (2017) reproduced

References

Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE.

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

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

Guo, X., Singh, S., Lee, H., Lewis, R. L., and Wang, X. (2014). Deep learning for real-time atari game play using offline monte-carlo tree search planning. In *Advances in neural information processing systems*, pages 3338–3346.

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

Kocsis, L. and Szepesvári, C. (2006). Bandit based monte-carlo planning. In *ECML*, volume 6, pages 282–293. Springer.

Kos, J. and Song, D. (2017). Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452*.

Kurakin, A., Goodfellow, I., and Bengio, S. (2016). Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*.

Lenz, I., Knepper, R. A., and Saxena, A. (2015). Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*.

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

Lotter, W., Kreiman, G., and Cox, D. (2016). Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*.

Michalski, V., Memisevic, R., and Konda, K. (2014). Modeling deep temporal dependencies with recurrent grammar cells"". In *Advances in neural information processing systems*, pages 1925–1933.

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

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582.

Oh, J., Guo, X., Lee, H., Lewis, R. L., and Singh, S. (2015). Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems*, pages 2863–2871.

Schmidhuber, J. and Huber, R. (1991). Learning to generate artificial fovea trajectories for target detection. *International Journal of Neural Systems*, 2(01n02):125–134.

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

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.