Tactics of Adversarial Attack on Deep Reinforcement Learning Agents

Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, Min Sun

- Prakash K Naikade, prna00001@stud.uni-saarland.de

Paper's Contribution

- Empirical evidences shows studying the adversarial attack on deep RL agents is critical
- Strategically-Timed Attack
  - Attacking at critical moments in an episode to minimize the agent's reward
- Enchanting Attack
  - Luring the agent to a designated target/dangerous state

- Reinforcement Learning = RL
- Deep Neural Network = DNN

Why Adversarial Reinforcement Learning?

- DNNs are ideal function approximators for classical RL algorithms
- DNNs are vulnerable to the adversarial example attack
- Adversarial attack on deep RL agents is different from adversarial attack on classification system
  - Reducing rewards or luring agent to dangerous state Vs Reducing classification accuracy
- Real world adversarial examples and examples crafted by adversary
- Critical to understand these vulnerabilities to failproof the RL algorithms used in mission-critical tasks

Why New Tactics?

- Existing tactics
  - *Uniform Attack* [Huang et al., 2017]
  - Ignores the fact that observations are correlated
  - Attack at every time step
  - More prone to detection
- Goal should be
  - Attack at selective time steps
  - At a time step where it could be more effective
  - E.g. Ping Pong
    - When the ball is away from paddle - no need to attack
    - When the ball is close to the paddle - attacking could cause the dropping of the ball

Why new tactics?

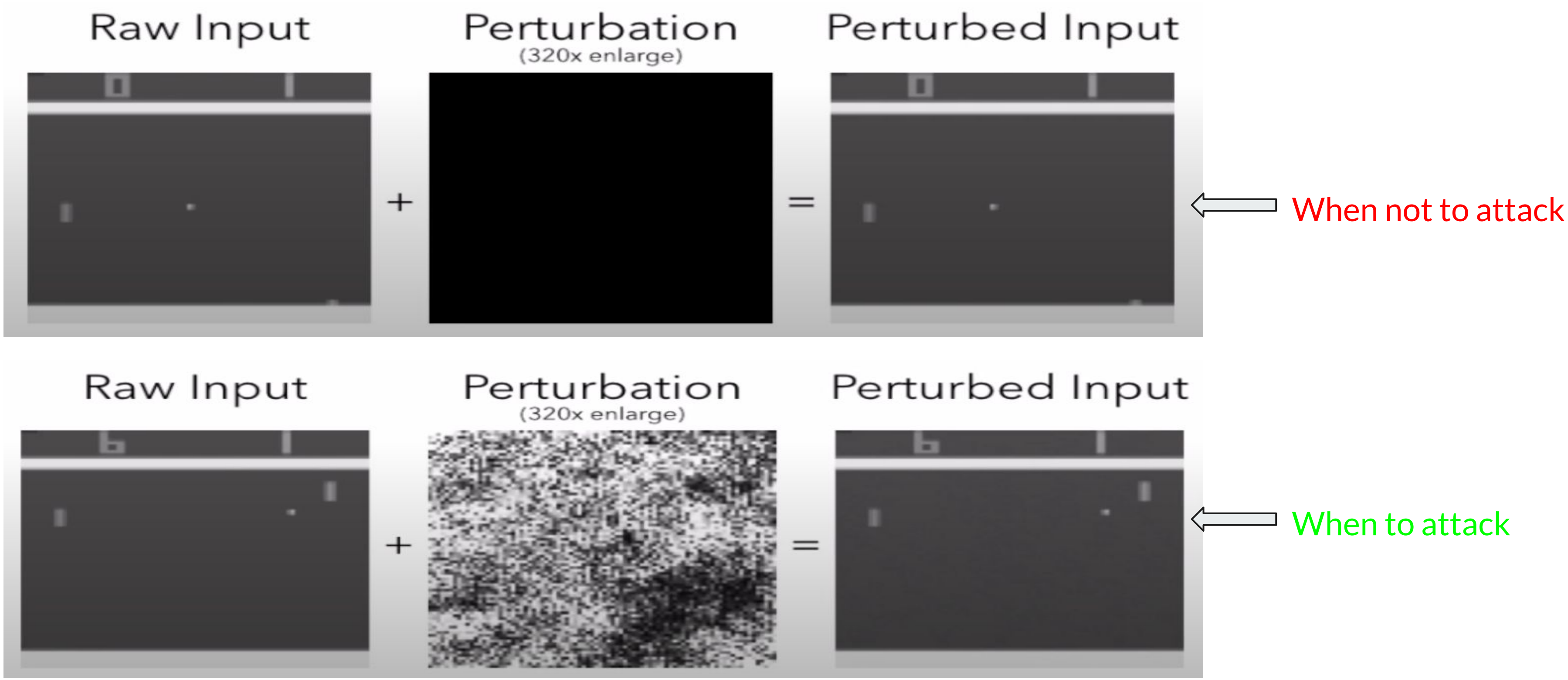


Fig 1 - When to attack

Two Novel Attacks

- Strategically-Timed Attack
  - Aims to reduce the reward with as fewer adversarial examples as possible
  - Adversarial example is only used when the attack is expected to be effective
  - Achieves same effect as the uniform attack by attacking four times less
- Enchanting Attack
  - Maliciously misguiding agent to a target state
  - Success rate > 70% in attacking agents

Reinforcement Learning

Let's structure RL problem as a Markov Decision Process (MDP),

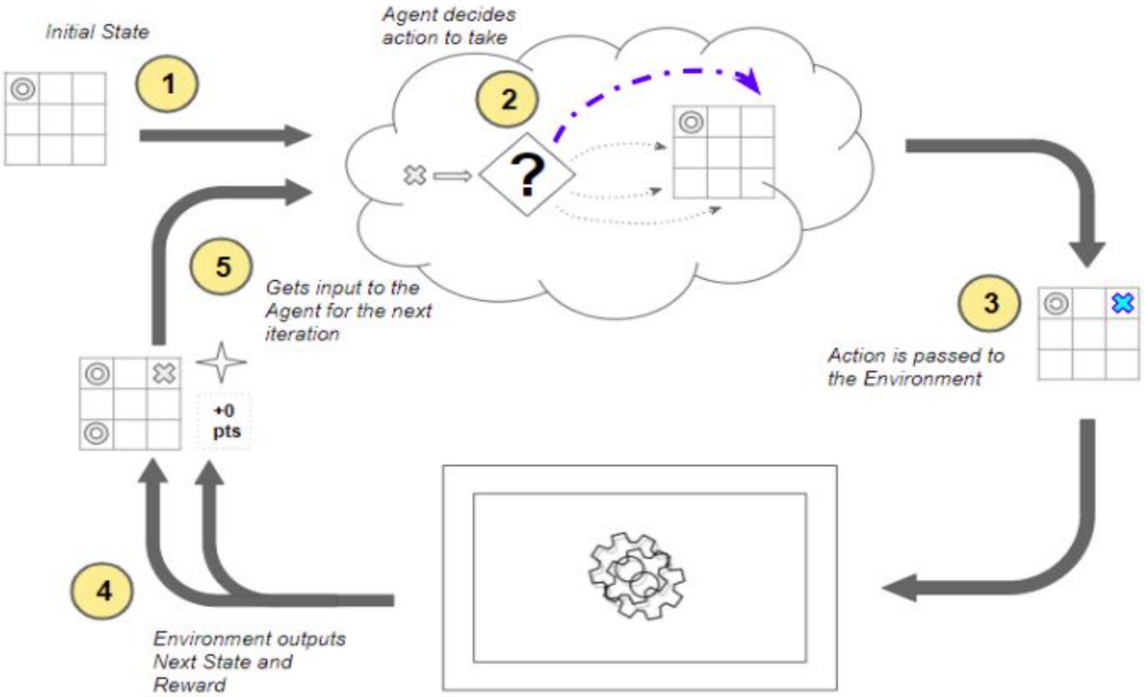


Fig 2 - How the MDP works (Image by Ketan Doshi)

MDP has 5 components,

- Agent,
- Environment,
- State,
- Action,
- Reward

Reinforcement Learning

Let's structure RL problem as a Markov Decision Process (MDP),

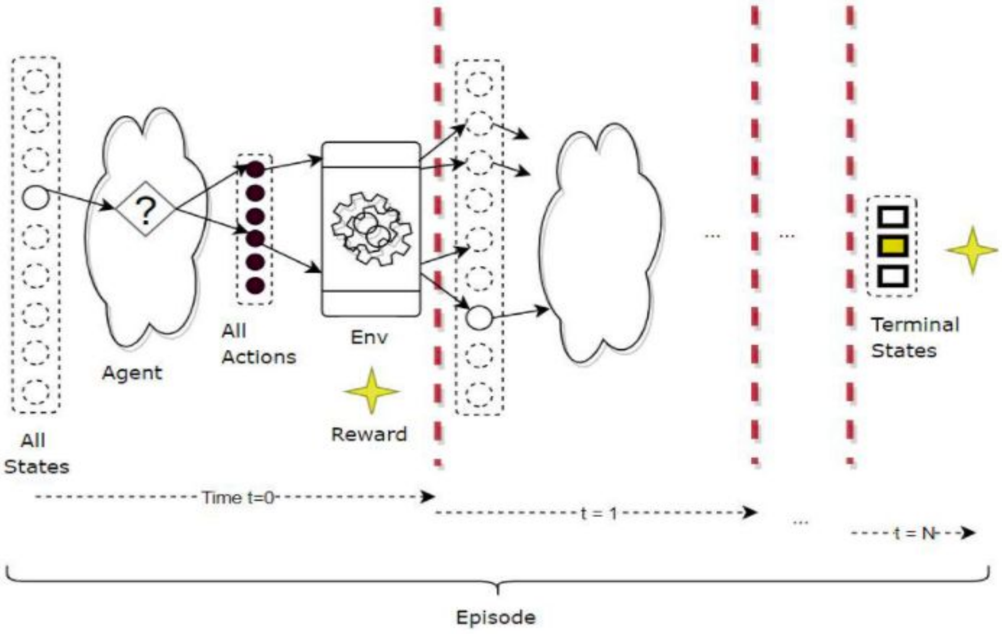


Fig 3 - An MDP iterates over a sequence of time steps (Image by Ketan Doshi)

- At each time step,
  - Agent performs an action based on the observation of the environment
  - For maximizing the accumulated future rewards
  - Action determination is through a policy π
- Goal of RL algorithm at time step t,
  - learn a policy that maximizes the accumulated future rewards,
    $$\mathcal{R}_t = r_t + r_{t+1} + \ldots + r_{\mathcal{L}}$$
  - policy π is learned through DNN
- Attack by,
  - Perturbing the observations by using crafted adversarial example
  - Sequence of adversarial example to lure agent to dangerous state

Strategically-Timed Attack

- Using our observation, we could minimize the expected return at first time step, by following optimization problem,

$$\begin{aligned} \min_{b_1,b_2,\ldots,b_L,\delta_1,\delta_2,\ldots,\delta_L} \quad & R_1(\bar{s}_1,\ldots,\bar{s}_L) \\ & \bar{s}_t = s_t + b_t\delta_t && \text{for all } t = 1,\ldots,L \\ & b_t \in \{0,1\}, && \text{for all } t = 1,\ldots,L \\ & \sum_t b_t \leq \Gamma && (1) \end{aligned}$$

Where,

$\{s_1, \ldots, s_L\}$ are sequence of observations or states in an episode,
$\{\delta_1, \ldots, \delta_L\}$ are sequence of perturbations,
$\mathcal{R}_1$ is the expected return at the first time step,
$b_1, \ldots, b_L$ denote when to attack.

- Mixed integer programming problem

- Problem size grows exponentially with L

- Divide the problem (1) into "When-to-Attack " $(b_1, \ldots, b_L)$ and "How-to-Attack" $(\{\delta_1, \ldots, \delta_L\})$

When-to-Attack

(Strategically-Timed Attack: Subproblem 1)

- Relative action preference function *c* to compute $b_1, \ldots, b_L$
  - For *policy gradient-based methods* (e.g. A3C algorithm),

$$c(s_t) = \max_{a_t} \pi(s_t, a_t) - \min_{a_t} \pi(s_t, a_t)$$

Where, π is policy network which maps a state–action pair (s_t, a_t) to a probability

  - For *value-based methods* (e.g. DQN algorithm),

$$c(s_t) = \max_{a_t} \frac{e^{\frac{Q(s_t,a_t)}{T}}}{\sum_{a_k} e^{\frac{Q(s_t,a_k)}{T}}} - \min_{a_t} \frac{e^{\frac{Q(s_t,a_t)}{T}}}{\sum_{a_k} e^{\frac{Q(s_t,a_k)}{T}}}$$

- Represents criticality to perform specific action to increase return
- Performs attack :

$$b_t = 1 \text{ when } c(s_t) \geq \beta$$

β controls the number of attacks to be performed

How-to-Attack

(Strategically-Timed Attack: Subproblem 2)

- Change most preferred action to least preferred to reduce accumulated reward

- Cue: Output of a trained deep RL agent to craft effective adversarial example by adding perturbation

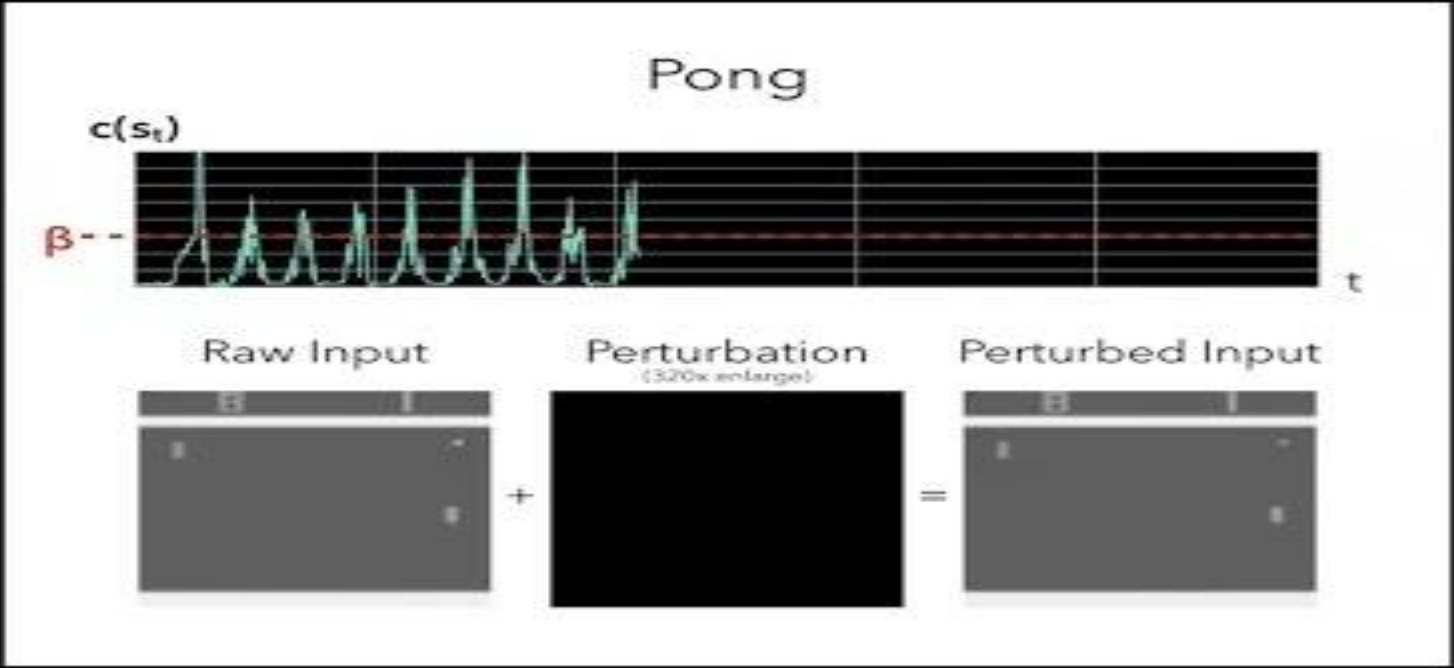
- Adversarial example crafting method [Carlini and Wagner, 2016]
    - Approximating following optimization problem,

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

where, x is an image,
f is DNN,
$\mathcal{D}_I$ is an image similarity metric which finds minimal perturbation δ

Strategically-Timed Attack

- Example: (http://yenchenlin.me/adversarial_attack_RL/)



Enchanting Attack

- To lure agent from current state s_t , at time step t , to a specified target state s_g , after H steps
- Need to craft a series of adversarial example $s_{t+1} + \delta_{t+1}, \ldots, s_{t+H} + \delta_{t+H}$
- Problem divided into two subtasks
- First Subtask:
    - Assume that adversary has full control of the agent to take arbitrary actions at each step
    - Task reduced to planning a sequence of actions for reaching the target state
    - By using online planning algorithm
- Second Subtask:
    - Craft example $s_t + \delta_t$
    - To force an agent to take the first action of planned action sequence
    - Gradually create adversarial examples in order to persuade an agent to take planned action sequence $s_{t+1} + \delta_{t+1}, \ldots, s_{t+H} + \delta_{t+H}$ [Carlini and Wagner, 2016]

Online Planning Algorithm

- “Future state prediction” + “Sampling-based action planning”

- *Future state prediction and evaluation*:

    - Next video-frame prediction model M to predict a future state given a sequence of actions ,

      $$s_{t+H}^{M} = M(s_t, A_{t:t+H})$$ [Oh et al., 2015]

      Where,

      $A_{t:t+H} = \{a_t, \ldots, a_{t+H}\}$ is the given sequence of H future actions , beginning at step t,

      s_t is the current state,

      s_{t+H}^{M} is the predicted future state

    - Success of the attack based on distance between s_g and s_{t+H}^{M} , which is given by $D(s_g, s_{t+H}^{M})$ ….. (L_2- norm)

Online planning algorithm

- Future state prediction + Sampling-based action planning

- *Sampling-based action planning:*
  - Sampling-based cross-entropy method to compute a sequence of actions [Rubinstein and Kroese, 2013]
  - Sample N action sequences of length H : $\{A^n_{t:t+H}\}^N_{n=1}$
  - Rank each of them based on the distance between the final state obtained after performing the action sequence and the target state s_g
  - Keep the best K action sequences and refit our categorical distributions to them

Flow of Enchanting Attack

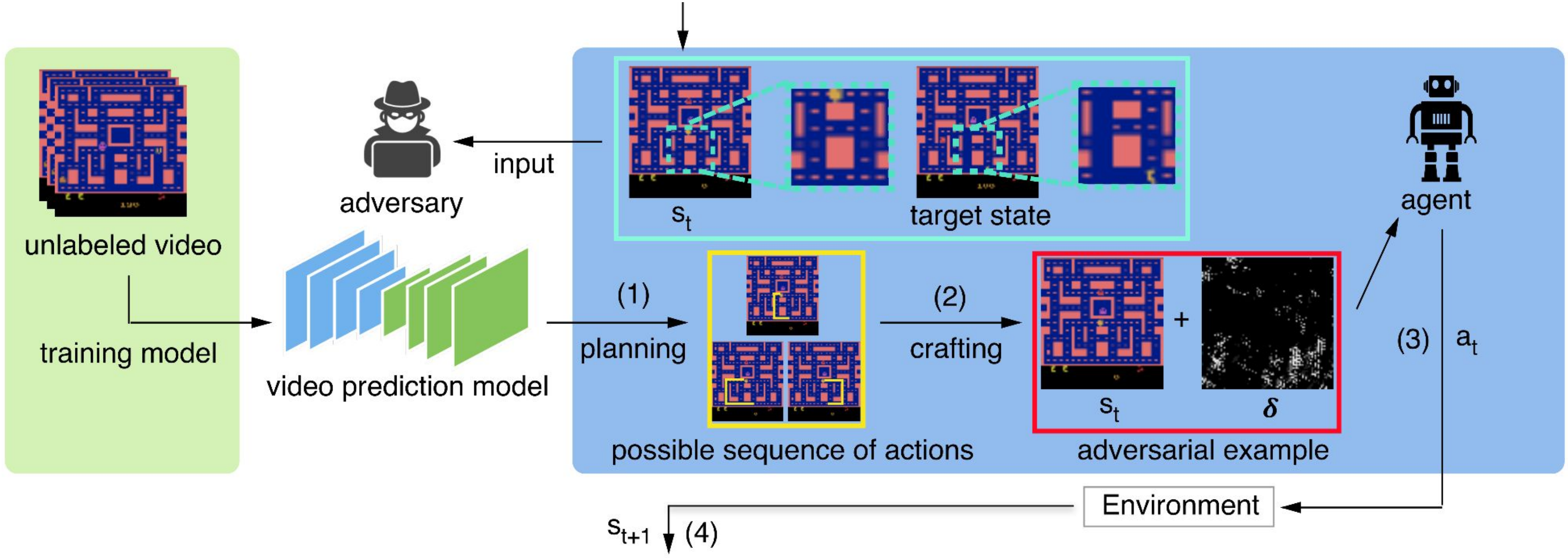
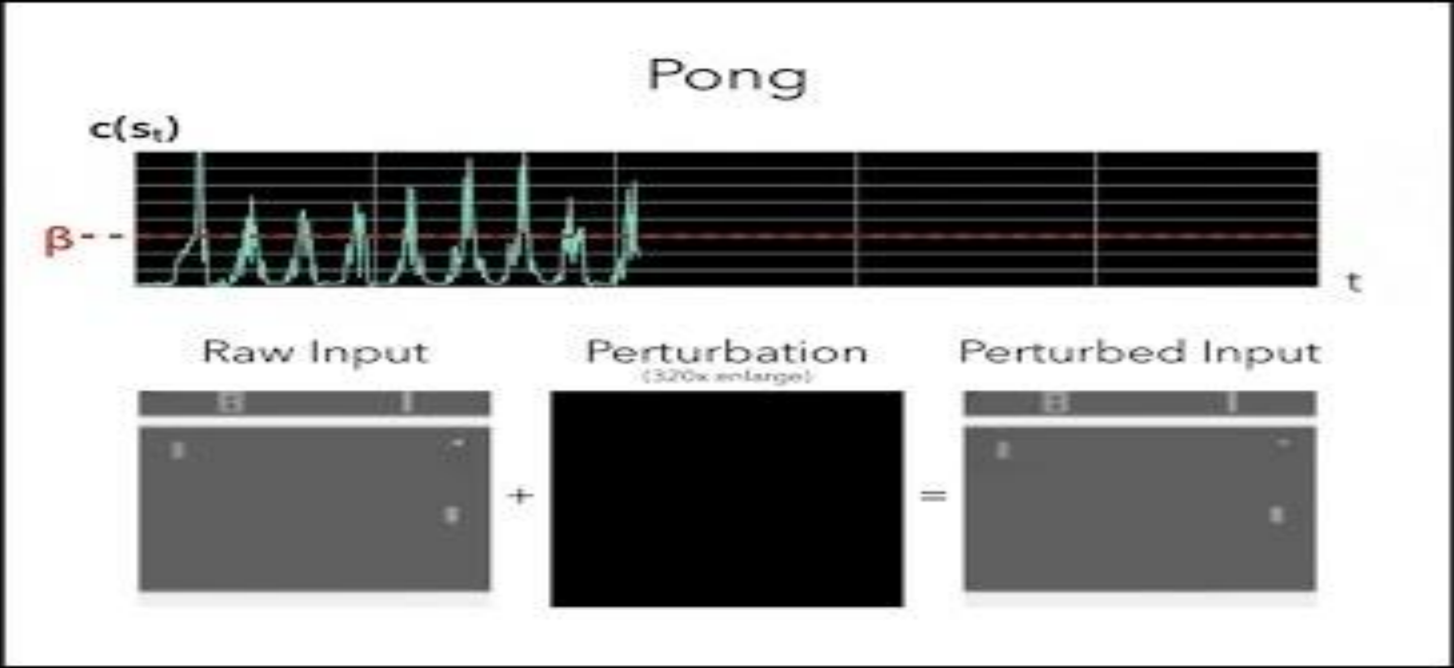


Fig 4 - Illustration of Enchanting Attack on Ms.Pacman.

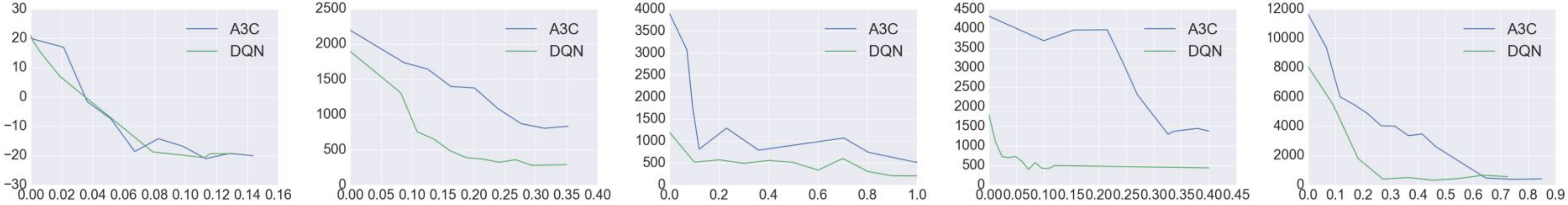
- Instead of directly crafting the next adversarial example with target action a_{t+1}, start another enchanting attack at state s_{t+1} for robustness

Enchanting Attack

- Example: (http://yenchenlin.me/adversarial_attack_RL/)

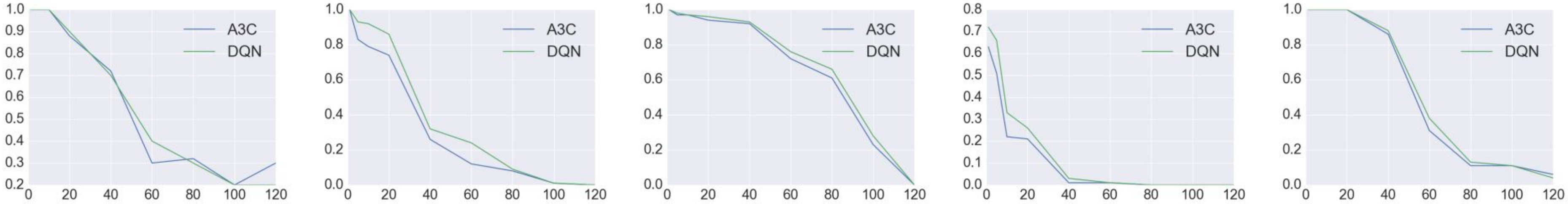


Experimental Results



(a) Pong (b) Seaquest (c) MsPacman (d) ChopperCommand (e) Qbert

Fig 5 - Accumulated reward (y-axis) v.s. Portions of time steps the agent is attacked (x-axis) of Strategically-timed Attack



(a) Pong (b) Seaquest (c) MsPacman (d) ChopperCommand (e) Qbert

Fig 6 - Success rate (y-axis) v.s.H steps in the future (x-axis) for Enchanting Attack

Experimental Results

- Strategically-timed attack can achieve same effect as the uniform attack by attacking just 25% of the time steps in an episode
- Enchanting attack forces agent toward maliciously defined target state in 40 steps with more than 70% success rate in 3 out of 5 games
- Enchanting attack was less effective on Seaquest and ChopperCommand
- Both games include multiple random enemies and therfore trained video prediction models were less accurate
- An agent trained using the DQN algorithm was more vulnerable than an agent trained with the A3C algorithm in most games except Pong
- Stronger deep RL agent may be more robust to the adversarial attack

Strengths

- In comparison to Huang et al., 2017, this is an innovative use of a "classic test time" adversarial attacks
- The overall experimental design is sound, and it covers all potential scenarios
- The research topic is interesting, and the contributions are substantial

Weaknesses

- The strategically-timed attack technique only considered the attack effect on one time step, ignoring the impact on the following states and actions, i.e., not considering the final end goal
- In an enchanting attack, accurate prediction and enforcement of future states and actions is difficult, and thus this approach suffers from a low attack success rate
- The experimental evaluation in general and in-terms of performance comparison with Huang et al., 2017 is weak and vague
- The paper's flaw is that the problem isn't well-motivated, and the authors jumped right into the content without explaining the publications they utilized to develop these adversarial tactics

Possible Improvements

- This paper requires a more thorough overview of the relevant work on which it is based
- More fluid narrative and structure

Possible Extensions

- Developing defenses against these kind of adversarial attacks
- Develop a more sophisticated strategically-timed attack strategy
- enhancing the generative model’s video prediction accuracy in order to increase the success rate of enchantment attacks in more complex environments

Thank You !

Questions Please !