Blackbox Attacks on Reinforcement Learning Agents Using Approximated Temporal Information

A Preprint

Yiren Zhao*
University of Cambridge
Yiren.Zhao@cl.cam.ac.uk

Ilia Shumailov*
University of Cambridge
Ilia.Shumailov@cl.cam.ac.uk

Han Cui*
University of Bristol
hc13414@bristol.ac.uk

Xitong Gao
Shenzhen Institutes of Advanced Technology
xt.gao@siat.ac.cn

Robert Mullins
University of Cambridge
Robert.Mullins@cl.cam.ac.uk

Ross Anderson
University of Cambridge
Ross.Anderson@cl.cam.ac.uk

September 10, 2019

Abstract

Recent research on reinforcement learning has shown that trained agents are vulnerable to maliciously crafted adversarial samples. In this work, we show how adversarial samples against RL agents can be generalised from White-box and Grey-box attacks to a strong Black-box case, namely where the attacker has no knowledge of the agents and their training methods. We use sequence-to-sequence models to predict a single action or a sequence of future actions that a trained agent will make. Our approximation model, based on time-series information from the agent, successfully predicts agents’ future actions with consistently above 80% accuracy on a wide range of games and training methods. Second, we find that although such adversarial samples are transferable, they do not outperform random Gaussian noise as a means of reducing the game scores of trained RL agents. This highlights a serious methodological deficiency in previous work on such agents; random jamming should have been taken as the baseline for evaluation. Third, we do find a novel use for adversarial samples in this context: they can be used to trigger a trained agent to misbehave after a specific delay. This appears to be a genuinely new type of attack; it potentially enables an attacker to use devices controlled by RL agents as time bombs.

1 Introduction

Deep neural networks (DNNs) show outstanding performance on a wide spectrum of tasks, ranging from image classification [9] to language processing [27]. Recent advances in deep reinforcement learning (RL) demonstrate that DNNs have the ability to learn policies that solve complex problems by mapping raw environment inputs directly to an action space. Trained deep RL agents show human-level or even superhuman performance in playing Go [25] and Atari games [16]. Following their success in game playing, RL agents are starting to be exploited in safety-critical fields, such as robotics [11], as well as in recommendation systems [29] and trading [19].

Recently, researchers have found that DNNs are vulnerable to crafted adversarial perturbations. DNN-based image classifiers produce incorrect results when inputs are injected with small perturbations that are not perceptible by humans [5]. Attackers can thus create adversarial examples that cause DNN-based systems to misbehave, including systems for

*Equal Contribution
face recognition [10] and autonomous driving [3]. One feature of adversarial samples is their transferability. Crafted inputs that affect one model often affect others too [20, 28]. So adversarial samples could present a real threat. However, attacking RL agents is different from fooling image classification systems. First, there is no notion of supervised labels in an RL agent, as its performance is assessed purely on the rewards it earns in an episode of game playing. Second, an action that a well-trained RL agent performs depends on a sequence of observations containing historical data that an attacker cannot modify. These properties of reinforcement learning make it a challenging problem to create realistic Black-box adversarial attacks.

Many researchers have looked for effective attacks on RL agents [8, 13] in White-box or Grey-box setups by assuming the attackers have access to some of the agents’ internal states or to some of their training methods. In this work, we stick to a Black-box assumption that attackers have no knowledge of either the agent’s parameters or the training methods. RL agents learn across time using a sequence of observations, and produce actions that can also be treated as a temporal sequence. The question of approximating RL agents can be formed as a sequence-to-sequence (seq2seq) learning problem which is well-studied in language-translation tasks [26, 4]. Given a sequence of observations on the trained agent, we build a seq2seq model to predict a future action sequence. This seq2seq model only needs to watch how a trained agent performs, and requires no knowledge of the agent’s internals and training methods. We show empirically that the model predicts future sequences of actions with consistently higher than 80% accuracy on different games with various RL training algorithms. The generated action sequence prediction is called the approximated temporal information.

We then demonstrate how to use the seq2seq model to produce adversarial samples, which are transferred to the trained agents. In our experiments, we show how this Black-box attack can decrease the reward of a trained agent, though in general such attacks do not decrease the reward any more than can be achieved with random Gaussian noise. However, we find we can use the approximated temporal information in a novel attack. By an appropriate perturbation of the current input, we can influence a future action after a specific delay with a high probability of success. This gives us a time-bomb attack on RL agents.

The contributions of this paper are the following.

- We provide an open-source framework to perform Black-box attacks on RL agents.
- We show how to build sequence-to-sequence models to predict one or multiple future agent actions with above 80% accuracy on Cartpole and Atari games trained with different RL algorithms.
- We demonstrate that although adversarial attacks cause trained agents to perform unwanted actions more frequently, random Gaussian noise can still be as effective as adversarial attacks in reducing the game score of a trained agent. This strongly suggests that, when evaluating attacks on such agents, random noise jamming should be used as the baseline case.
- We show a novel time-series Black-box attack, named the time-bomb attack, that uses adversarial samples to flip actions after a specific delay. This attack opens up a new temporal dimension for adversarial reinforcement learning.

Figure 1: An illustration of the sequence-to-sequence network’s architecture, the network is multi-head: observation head ($S_{t-1}$), action head ($A_{t-1}$) and current observation head ($s_t$). The output of the seq2seq model is a sequence of predicted future actions ($A^f_t$). The details of designing each head is game dependent and shown in Table 2.
2 Related Work

Using the gradients of DNNs, Goodfellow et al. proposed the Fast Gradient Sign Method (FGSM) to produce adversarial samples. The samples contain small perturbations that are imperceptible to humans, yet DNNs produce high-confidence incorrect answers using these images as inputs. Later researchers showed how to apply scaled gradients iteratively to the original input image [10, 14]. Iterative methods such as the projected gradient descent (PGD) attack, proposed by Madry et al., show stronger performance in comparison to single step attacks (e.g., FGSM). The Carlini & Wagner attack (CW) shows how to generate adversarial samples by solving an optimization problem [2]. However, the large number of iterations required makes it expensive to execute in real time.

Huang et al. were among the first to apply adversarial attacks to RL agents. They evaluated the performance of FGSM attacks in both White and Black-box settings. They found that adversarial attacks are effective on RL agents and various RL training algorithms show different levels of robustness. However, their Black-box setups assume attackers having access to the agent’s training environments and DNN structures. Pattanaik et al. have further extended the approach produced by Huang et al. and constructed a loss function that can be used to attack RL systems more effectively. Furthermore, the authors introduced adversarial training in RL agents and evaluated the performance of attacks after agents received adversarial training. Alternative defense mechanisms are yet to be applied to RL [15, 23, 24].

Behzadan and Munir took a similar approach to Huang et al., they designed a replica of the original agent and conducted FGSM and JSMA attacks on the replica to create transferable adversarial samples. In their evaluation, they allowed attackers to periodically sample both the parameters and network architecture of the original agent [1]. They further assumed the attacker had full access to the training environment and training hyperparameters.

Lin et al. argue that the attack presented in Huang et al. contains an impractical assumption, namely that attacks are conducted at every step; they also noted that the attacks ignored the facts that the subsequent observations are correlated, and one should not use adversarial samples every step because of the detection risk. They reported a new timed-strategic attack: an adversary can attack a quarter as often, but achieve the same reward degradation as if the attack was performed at every step. They also explained that the attacks against RL are different from those on image classifiers: first, the goal is not to cause misclassification, but to reduce reward; and second, future observations depend on the current observation. In order to forecast the target agent’s future actions, they exploited a video sequence prediction model for sequence planing. However, the generation of adversarial samples still requires access to the target agents’ parameters.

3 Method

3.1 Preliminaries

At each time period $t$, the environment provides a state $s_t$ to an agent. This $s_t$ can be an image, or scalar inputs from sensors. The agent responds with an action $a_t$, and the environment feeds back a reward $r_t$. The interaction between the agent and the environment forms a Markov Decision Process, and an agent learns a policy $\pi$ that describes a probability distribution on the action space given the current state $s_t$ [16]. The policy $\pi$ is trained to maximize the expected discounted return $R_t$, where $R_t = \sum_{i=0}^{t-1} \lambda i t + 1$, $l$ is the total number of steps in an episode of game play and $\lambda$ is the discount factor. A trained agent typically takes states or state-action pairs as inputs to decide what action to take ($a_t$).

Here, we provide a relaxed notion to the inputs of the agent’s policy, since popular techniques such as frame stacking can provide an agent with a history of its inputs. Let us consider the sequence of states $S_t = (s_t, s_{t-1}, \ldots, s_{t-n})$ and the sequence of actions $A_t = (a_t, a_{t-1}, \ldots, a_{t-n})$, where $n$ is the length of the sequences and $0 \leq n \leq t$; we have the policy function:

$$a_t = \pi(S_t, A_{t-1}).$$

(1)

To mount an adversarial attack on a trained policy $\pi$, we construct an adversarial sample for the current state $s_t$ only, as all other states in $S_t$ are historical and cannot thus be modified. The objective is to compute a small perturbation $\delta_t$ for the input to the policy function, which now becomes $\tilde{S}_t = (s_t + \delta_t, s_{t-1}, \ldots, s_{t-n})$, such that the resulting action

$$\tilde{a}_t = \pi(\tilde{S}_t, A_{t-1})$$

(2)

differs from the intended action $a_t$, in such a way as to give a successful attack.

3.2 Threat Model

The threat model assumes that the attacker can modify the environment so that the agent receives a perturbed state $s_t$. We define the attack target as $s_t$; past states should not be modified. If frame stacking is conducted on the agent side, the attack should not change previously stacked states. Since the attack is Black-box, we hold assumptions that
We found it is necessary to increase the complexity of each head when training seq2seq models for different games. We trained a sequence-to-sequence neural network as illustrated in Figure 1 as
\[ f \]
which output predicts a sequence of future actions
These heads affect the quality of information extraction, so we tuned them manually; a full per-game architecture for the agent relies on a learned policy \( \pi \). Table 2: Black-box approximation network configurations and accuracies for different games. We measure how accurate the approximation network predicts the next 10 consecutive actions from time \( t \) (\( a_t \) to \( a_{t+9} \)), if it is not Seq, only a single action (\( a_t \)) is predicted. **Obs Head**, **Action Head** and **Current Obs Head** refers to different parts of the networks as illustrated in Figure 1. Input Seq shows the input sequence length for both actions and observations.

### 3.3 Temporal Information Approximation

The agent relies on a learned policy \( \pi \) to take an action, such a decision made by the agent not only depends on the agent’s observation \( s_t \) but also relates to its previous actions \( A_{t-1} \) and previous states \( S_{t-1} \). Moreover, an approximation model should predict a sequence of future actions instead of a single one. We will explain how this model opens up temporal opportunities to the attackers later. For convenience, we define the approximation model as:

\[ A^f_I = f(A_{t-1}, S_{t-1}, s_t), \]

which output predicts a sequence of future actions \( (a_t, a_{t+1}, a_{t+2}, \cdots, a_{t+m}) \).

Function \( f \) now is an approximator that takes sequence inputs \( A_{t-1} \) and \( S_{t-1} \) and current state \( s_t \). We use \( n \) and \( m \) to represent input time steps and output time steps respectively; this is equivalent to an input sequence having a length of \( n \) and an output sequence having a length of \( m \). The output sequence \( A^f_I \) contains \( m \) future actions. Following the above Black-box setup, we only need to observe how the agent is playing the game to build up a collection of \( ((A_{t-1}, S_{t-1}, s_t), A^f_I) \) and hold this collection as training data to train the approximator \( f \).

We trained a sequence-to-sequence neural network as illustrated in Figure 1 as \( f \). The proposed network is multi-head to consume three different inputs \( (A_{t-1}, S_{t-1}, s_t) \). Two of the inputs \( ((A_{t-1}, S_{t-1}) \) are sequence inputs and we utilised LSTMs in their relative paths to extract information as it is a popular building block in seq2seq learning [7, 26]. We found it is necessary to increase the complexity of each head when training seq2seq models for different games. These heads affect the quality of information extraction, so we tuned them manually; a full per-game architecture for each head is shown in Table 2. After information passes through each head, we duplicate the embedding \( m \) times to form an output sequence that has a temporal dimension of \( m \). These later blocks remain unchanged for different games.
Blackbox Attacks on Reinforcement Learning Agents
Using Approximated Temporal Information

A PREPRINT

since empirically we found changing the heads is enough to provide us good accuracy on predicting actions on different games.

The other problem is to determine an $n$ that is the number of input time steps, describing the length of the input sequence. It is challenging to derive an optimal $n$ formally before training an approximator $f$, so we just have to search for it. Luckily, the accuracy of a seq2seq model with various values of $n$ differs at an early stage of training. In practice, we set a search budget of $N_t = 0.01N$, where $N$ is the number of training epochs. Algorithm 1 describes the procedure of training a seq2seq model, where $[\ ]$ denotes an empty list and the addition of two lists joins them sequentially. First, we run a trained policy $\pi$ to produce action $a_t$; internally, $\pi$ might have stacked previous states or actions. The sequence $E$ consists of the historical states and actions in an episode of game playing. The set $D$ consists of multiple sequences $E$, and are used as the data for training ($D_{\text{train}}$) and evaluation ($D_{\text{eval}}$). Given a epoch count $N$, the selection of an input sequence length requires training of $n_{\text{max}}$ seq2seq models, each trained for $N_t = 0.01 \times N$ epochs. The train function requires an input sequence length, a training dataset and the number of epochs to execute training for the seq2seq model. The Split function randomly shuffles the collected data, marking 90% of the data as training data and 10% as evaluation. We found that the evaluation accuracy of the seq2seq model after training for a small number of epochs was enough to pick the optimal input sequence length ($n$). After picking $n$, we then train with $N$ epochs to get our fully-trained seq2seq prediction model. In our experiments, we used $N = 500$ and $n_{\text{max}} = 50$.

### Algorithm 1 Training Algorithm For Seq2Seq Model

1: $D \leftarrow \emptyset$
2: while $|D| < N$ do
3:     $E \leftarrow [\ ]$
4:     $s_t = \text{Init Env}$
5:     while (Game not done) do
6:         $a_t = \pi(s_t)$
7:         $s_{t+1} = \text{Env}(a_t)$
8:         $E \leftarrow E + [s_t, a_t, s_{t+1}]$
9:     end while
10: $D \leftarrow D \cup E$
11: end while
12: $n \leftarrow 0$
13: $\text{acc}_m \leftarrow 0$
14: $N_t \leftarrow 0.01N$
15: $D_{\text{train}}, D_{\text{eval}} \leftarrow \text{Split}(D)$
16: for $n_i \in \{0, 1, \ldots, n_{\text{max}}\}$ do
17:     $w \leftarrow \text{train}(n_i, D_{\text{train}}, N_t)$
18:     $\text{acc} \leftarrow \text{eval}(w; n_i, D_{\text{eval}})$
19:     if $\text{acc} > \text{acc}_m$ then
20:         $n \leftarrow n_i$
21:         $\text{acc}_m \leftarrow \text{acc}$
22: end if
23: end for
24: $w \leftarrow \text{train}(n, D_{\text{train}}, N)$

3.4 Transferring Adversarial Samples

Once we have a seq2seq model that approximates the RL agent, we consider how to attack the RL agents.

In this paper we have used a number of attacks with varying complexity. As the weakest attacker we assumed Random Gaussian Noise – the attack itself does not actually use any information from our model. It merely injects noise of a particular amplitude into the inputs of the agent, and thus serves as a baseline attack.

As a more sophisticated attacker, we used FGSM [5]. The idea behind FGSM is to use the gradient information from a particular classifier to attack it. As the most complex attacker we consider PGD [14], which can be seen as an iterative version of FGSM. PGD is considered to be more complex as it requires iterative gradient computation per sample, but it produces smaller adversarial perturbations.

Note that the list of attackers being tested in this paper is not exhaustive and many other complex attacks could be conducted. CW [2] is generally considered as the strongest attacker, and was used to attack the RL agents by Lin et al.. However, to get good misclassification rates and transferability, one needs to run CW for a very large number of iterations and with a large binary search factor. Unlike adversarial machine learning in image classification where
Blackbox Attacks on Reinforcement Learning Agents
Using Approximated Temporal Information

Figure 2: The attacker observes how a trained agent plays a game and can inject perturbations into the agent’s observation $s_t$. We collect a sequence of observations to feed into the seq2seq model to generate adversarial perturbations to attack the trained agent. The adversarial perturbation in this figure is from an FGSM attack with $\epsilon = 0.01$.

The evaluation can include only a thousand inputs, a single game episode in RL may contain thousands of decisions, making the evaluation of CW on RL agents infeasible. As we will highlight in later sections, a stronger attack in RL only implies a larger transferability in a Black-box setup, but does not necessarily translate to a direct decrease in agents’ accumulated rewards.

The detailed attack flow is illustrated in Figure 2. Our Black-box attack starts after $n$ time steps when the rollout FIFO is full. The rollout FIFO records playing histories so that inputs $(A_{t-1}, S_{t-1}, s_t)$ are prepared. In our attack, the attacker has no access to the agents’ internal states and parameters, or to the training environment. After preparing the input sequence, the trained seq2seq model generates a prediction which can be used to craft adversarial perturbations. The seq2seq model is multi-input, the adversarial perturbation is only generated for $s_t$, and the other two inputs $A_{t-1}$ and $S_{t-1}$ remain unchanged. The perturbation is then injected to $s_t$ to attack the agent. Although the seq2seq model requires an input sequence of length $n$ for its inputs $A_{t-1}$ and $S_{t-1}$, this does not mean we can only attack every $n$ steps. We need to wait for the initial $n$ steps to start the first attack, but the following attacks can be generated consecutively since the FIFO is always full thereafter.

4 Evaluation

4.1 Experiment Setup

We trained agents using three different RL algorithms: DQN [16], A2C [17] and Rainbow [6]. The targeting games are Cartpole, Space Invader and Pong. The Cartpole game takes only 4 input signals from the cart, while the other two are classic Atari games; we followed Mnih et al.’s method of cropping them to $84 \times 84$ image inputs. The RL algorithms and game setups are developed on Ray and RLlib [18][12].

As mentioned previously, we consider Random Gaussian Noise, FGSM [5] and PGD [14] attacks. The implementation of the attacks was adapted from the commonly used adversarial machine learning library Cleverhans [21]. Adaptations and extensions were made on the framework as its current form Cleverhans does not support multi-input model or sequential outputs.
4.2 Seq2seq Approximation

The seq2seq models vary in complexity when targeting various games and RL agents. As mentioned previously, we alter the multi-head component in the seq2seq model as illustrated in Figure 1 to adapt to various games. The building blocks of each approximator are shown in Table 2. The input sequence length of each approximator is determined using Algorithm 1. The approximators are then trained with Stochastic Gradient Descent (SGD) with a learning rate of 0.0001. We collect $N = 500$ episodes of game play from trained agents and use this collected data as a training dataset. The approximator, at every training time, takes bootstrapped training data from the 500 episodes of playing experience with a batch size of 32. The seq2seq models are then tested on the unseen evaluation data of the agent playing the game. As shown in Table 2, we achieve on average above 80% accuracy on all seq2seq models. We found Space Invaders a more challenging game to approximate using a seq2seq model, and hypothesize that this is because of interaction with a more complex environment. Another interesting observation is that Pong requires a small input sequence count, implying that the Pong agent effectively only requires short-term observations. But whether predicting a single action or a sequence of 10 actions in future time steps, our trained seq2seq model predicts them correctly with high accuracy.

4.3 Reward-focused Black-box Attack

In this section, we discuss how our proposed Black-box attack reduces an agent’s accumulated reward. This score is a direct measurement of the agent’s game-playing quality; an efficient attack should be able to decrease it quickly by injecting perturbations. Figure 3 shows the performance of attacking agents trained for the Cartpole game. The agents are trained with DQN, A2C and Rainbow. Similarly, we show how agents trained with these three different RL algorithms perform on Atari games (Space Invader and Pong) in Figure 4. In these plots, we present error bars that are generated from 20 distinct runs, and the mean values are extracted to plot the lines. In our attack, we start to inject adversarial perturbations into the agent after collecting $n$ samples so that the FIFO rollout is full (Figure 2). However, the reward keeps accumulating in the initial collection stage, so even a powerful attack cannot reduce the score to zero. Notice these attacks are untargeted, and the results generated are similar to Huang et al.'s, but Huang et al.
ran their attacks in White-box and Grey-box setups. The score decreases with increasing $l_2$ norms, meaning that agents are more vulnerable to larger perturbations. We define transferability as the rate between the number of adversarial samples that successfully caused misclassifications and the total number of adversarial samples generated. In terms of transferability, FGSM and PGD outperform random noise, but the surprising thing is that the reward reduction caused by random noise is almost as great. As illustrated in Figure 5, adversarial attacks (FGSM and PGD) with the same $l_2$ norm have greater transferability in the Cartpole game across all RL algorithms. This, however, does not translate to efficiency in reducing the rewards. Many published works do not mention how reward decreases with increasing random noise. In our experiments, we found random noise generates comparable results to popular adversarial techniques in reward-based attacks given the same noise budget, which should have served as a baseline for work of this kind. In particular, FGSM and PGD do not do significantly better than random jamming.

Figure 5: Transferability of Black-box reward-focused attacks on DQN, A2C and Rainbow trained on Cartpole. The x-axis shows the size of the transferability of the adversarial samples, namely the ratio between the number of samples that fooled the agents and the total number of samples generated from the seq2seq models. The y-axis presents the $l_2$ norm.

Figure 6: Black-box temporal-focused attacks on A2C and Rainbow trained on Space Invader and Pong. The seq2seq model is trained against the DQN algorithm. The seq2seq model is then used to generate adversarial samples on A2C and Rainbow trained games to access transferability across RL algorithms. An adversarial image is sent only at $s_t$, but aims to perturb an action $n$ steps away $a_{t+n}$. The x-axis shows the targeted future step $n$, and the y-axis the perturbation rate averaged across 20 runs.

Adversarial attacks do not do better than random noise jamming at reducing agents’ rewards, but are efficient in action-targeted attacks. This section demonstrates a new temporal attack, which we called the time-bomb attack, and which uses the seq2seq model to produce an output sequence. With its ability to forecast future actions, the Black-box attack now can target a specific time delay. Suppose the attacker’s goal is not to decrease the game score of a deployed agent but to trigger adversarial actions after a delay. For example, the attacker might want to cause an autonomous truck containing explosive fuel to crash five seconds in the future, so they have time to get out of the way. In Figure 6, the x-axis shows the time delay. For instance, 1 means we aim to perturb action $a_{t+1}$, and 2 means $a_{t+2}$. We only send adversarial noise at $s_t$, and all future observations made by the agents are clean. The y-axis shows the success rate of such an action. This temporal-focused attack shows that the success rate of perturbing a particular future action relates to the games and the agents’ training algorithms. We used DQN to train a seq2seq model and directly transfer the adversarial samples from the trained seq2seq model to both A2C and Rainbow agents. The results in Figure 6 show that this time bomb attack works better on A2C trained agents than the Rainbow-trained agents, since Rainbow is a more complex algorithm. In addition, the performance in Space Invaders is better than in Pong, which suggests that
Pong is harder to sabotage in this way. (This makes intuitive sense since the game physics is much simpler.) We limited the attacks in Figure 6 to have $\epsilon = 0.3$ which implies the $l_\infty$ norm is bounded by 0.3. This attack budget is picked to demonstrate the differences across games and RL algorithms. With a larger attack budget (with $\epsilon$ bigger than 0.7), we do observe all agents on all games got attacked with success rates consistently higher than 0.7. We believe the time dimension of RL algorithms is not yet fully explored in the field of adversarial reinforcement learning. Our time-bomb attack shows that an attacker can in principle craft Black-box attacks that perturb only the current time frame, but cause a well-trained agent to misbehave after a specific delay. Intuitively, this might not be too surprising, since there are many games where one player can set a trap for another. The breakthrough here is to demonstrate it as a general attack on game-playing and similar RL agents.

5 Conclusion

This paper offers three things: an improvement in the state of the art, a critique of the research methodology used thus far, and a new research challenge.

We explored how attackers can craft Black-box attacks against reinforcement learning agents. The Black-box attack assumes that attackers have no access to any internal states or the training details of a RL agent. To our knowledge, this is the first fully Black-box attack against RL agents. We discovered three things.

First, we can use seq2seq models to predict a sequence of future actions that an agent will perform, and use these to generate highly transferable adversarial samples. This improves the state of the art, as previous attacks were White-box or Grey-box.

Second, although these adversarial samples are transferable, they do not outperform random Gaussian noise as a means of reducing the game scores of trained RL agents. This highlights a serious methodological deficiency in previous work on game-playing agents; random noise jamming should have been used as a baseline.

Our adversarial attacks do however have one advantage over random jamming: they can be used to trigger a trained agent to misbehave at a specific time in the future. This is our third discovery, and it appears to be a genuinely new type of attack; it potentially enables an attacker to use devices controlled by RL agents as time bombs. Given that many games allow or even encourage players to set traps for other players, this might perhaps have been expected. But in applications other than games, the stakes can be higher. Where RL techniques are used to control safety-critical equipment, it will be necessary to find ways to ensure that such traps cannot be discovered and exploited by adversaries. How that might be done is the new research question posed by this paper.

Acknowledgements

*Partially supported with funds from Bosch-Forschungsstiftung im Stifterverband.*
References


Blackbox Attacks on Reinforcement Learning Agents
Using Approximated Temporal Information


