Blackbox Attacks on Reinforcement Learning Agents Using Approximated Temporal Information

Yiren Zhao*†, Ilia Shumailov*†, Han Cui*‡, Xitong Gao§, Robert Mullins† and Ross Anderson†
† University of Cambridge
‡ University of Bristol
§ Shenzhen Institutes of Advanced Technology

***Abstract*—Recent research on reinforcement learning (RL) has suggested that trained agents are vulnerable to maliciously-crafted adversarial samples. In this work, we show how such samples can be generalised from White-box and Grey-box attacks to a strong Black-box case, where the attacker has no knowledge of the agents, their training parameters or 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. First, we show that our approximation model, based on time-series information from the agent, consistently predicts RL agents' future actions with high accuracy in a Black-box setup on a wide range of games and RL algorithms. Second, we find that although adversarial samples are transferable from the sequence-to-sequence model to our RL agents, they often outperform Random Gaussian Noise only marginally. Third, we propose a novel use for adversarial samples in Black-box attacks of RL agents: they can be used to trigger a trained agent to misbehave after a specific time delay. This potentially enables an attacker to use devices controlled by RL agents as time bombs.**

***Index Terms*—Reinforcement Learning, Adversarial Machine Learning**

I. Introduction

Deep neural networks (DNNs) have good performance on a wide spectrum of tasks, including image classification [1], object detection [2], emotion recognition [3] and language processing [4]. Recent advances in reinforcement learning (RL) demonstrate that DNNs can learn policies that solve complex problems by mapping raw environmental inputs directly to an action space. Trained deep RL agents show human-level or even superhuman performance in playing Go [5] and Atari games [6]. RL agents are now starting to be exploited in safety-critical fields, such as robotics [7], as well as in recommender systems [8] and trading [9].

Researchers have also found that DNNs are vulnerable to crafted adversarial perturbations. DNN-based image classifiers can produce incorrect results when inputs are subjected to small perturbations that are not perceptible by humans [10]. Attackers can thus create adversarial examples that cause DNN-based systems to misbehave, including systems for face recognition [11] and autonomous driving [12]. One feature of adversarial samples is their transferability: adversarial inputs that affect one model often affect others too [13, 14]. Therefore adversarial samples can potentially attack machine-learning (ML) systems at scale.

Equal Contribution

However, attacking RL agents is different from fooling image classifiers. 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. Finally, a single misprediction does not usually cause serious disruption to the agent. These properties make it a challenging problem to create realistic Black-box adversarial attacks on RL. We will return to the discussion of these properties in Section II.

Researchers have recently looked for effective attacks on RL agents [15, 16] in White-box or Grey-box setups by assuming the attackers have access to some of the agent's internal states or training methods. These attacks aim to force an agent to take unwanted actions so that its reward is decreased. In a typical Grey-box setup, the attackers can access partial information of the target agent or its training environment, or even retrain another agent to approximate the target agent [15, 16].

In this paper, we go much further. We make a strong Black-box assumption that attackers have no knowledge of either the agent's parameters or its training methods. We cannot therefore just retrain another RL agent to do the the same task. Rather, we study ways of approximating RL agents as in imitation learning. We formulate this approximation as a sequence-to-sequence (seq2seq) learning problem as this is widely studied in language-translation tasks [17, 18]. RL agents learn over time using a sequence of observations, and produce actions that can also be treated as a temporal sequence. Given a sequence of observations of the target agent, we build a seq2seq model to predict its future actions from watching how the target performs, with no knowledge of its internals and training methods. We show empirically that such a model can predict future sequences of actions consistently with over 80% accuracy on three different games with three different RL training algorithms. The action sequence prediction we generate is called the *approximated temporal information*.

We then demonstrate how to use the seq2seq model to produce adversarial samples. In prior work under White-box or Grey-box assumptions, researchers exhibited adversarial samples that could be used to reduce the target agent's game score by feeding them in as perturbations to its game input. In our experiments, we show that our Black-box attack can decrease the target agent's game score. One interesting observation is that, in a fully Black-box setup, adversarial samples only marginally outperform Random Gaussian Noise. In other words, previously proposed adversarial RL attacks are less efficient in such a context. The second interesting finding is a novel attack. By appropriately perturbing the current input, we can influence future action after a specific time delay with a high probability of success. This gives us a novel *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 that, for the first time, attacking RL agents is possible under a strong Black-box assumption.
- We present how to use sequence-to-sequence models to approximate RL agents with above 80% accuracy on a range of games trained with different RL algorithms.
- We demonstrate that Random Gaussian Noise can be as effective as adversarial attacks for reducing the reward. When evaluating RL attacks, random noise jamming should be the baseline.
- We show a novel time-bomb attack that uses adversarial samples to flip actions after a specific delay. This attack opens up a new frontier in adversarial RL.

II. Motivation

Recent advancements in deep learning have led to the adoption of DNN-based control mechanisms for a range of tasks. A wide range of attacks has been developed since. Interesting attacks are often considered to be those that produce perturbations imperceptible to humans. However, almost all of them focus on disrupting a single standalone task.

In the real world, however, DNNs are usually a component of a large stateful system with both space and time aspects. While the space aspect has attracted some research [19], the timing side has been much less explored. This work aims at filling this gap by finding practical attacks on systems that control some critical resource over a period of time, observing their behaviour closely so as to find just the right time to attack.

As an example, consider air combat where the opponent's aircraft have been observed performing manoeuvres to follow and intercept targets. Our goal is to learn from observational data alone and make a good approximation of the embedded RL agent in the enemy aircraft, so that we can develop better tactics to break its lock on a target. Our baseline is random evasion; we want to know whether we can do anything better based on an approximate model of the opposing DNN.

Our threat model covers a large number of different use cases, and we evaluate our attack using different RL algorithms. Although the evaluation cannot represent all of them, it at least shows the possibility of fully Black-box attacks. Finally, we show that our work generalises to modelling and predicting agents with unknown objective functions. If we can observe the agent long enough, we can predict its behaviour and disrupt it, and in some cases even to cause a disruption after a known delay.

III. Related Work

Deep neural networks (DNNs) help RL to scale to complex decision-making problems. A large state or action space makes learning intractable for traditional Markov Chain Monte Carlo methods. Mnih et al. were the first to propose combining deep convolutional neural networks with reinforcement learning (DQN), and demonstrated that DQN achieves super-human performance on a series of Atari games [6, 20]. Later on, Mnih et al. proposed the asynchronous advantage actor-critic method, where the actor performs actions based on its underlying DNN, a critic scores the performance of the actor, and the actor then updates its DNN parameters based on the scores received [21]. Hessel et al. built on top of the DQN framework and combined it with a range of possible extensions [22]. They then demonstrated empirically that their algorithm Rainbow outperforms DQN and Actor-critic on a range of open benchmarks.

Goodfellow et al. proposed the Fast Gradient Sign Method (FGSM) to produce adversarial samples using DNN gradients [10]. The samples contain small perturbations that are imperceptible by humans, yet DNNs produce high-confidence incorrect answers on these inputs. Later, researchers showed how to apply scaled gradients iteratively to the original input image [11, 23]. Iterative methods such as the projected gradient descent (PGD) attack, proposed by Madry et al., show better performance than single-step attacks (*e.g.* FGSM) [23]. The Carlini & Wagner attack (CW) teaches how to generate adversarial samples by solving an optimisation problem efficiently [24]. A major threat from adversarial samples is their transferability. This refers to the fact that a single adversarial sample may cause misclassifications on different classifiers. Szegedy et al. observed that models of different configurations can easily misclassify on the same set of adversarial inputs [25]. Zhao et al. later pointed out that transferability is also found on a range of compressed networks [14].

Huang et al. first applied adversarial attacks to RL agents [15]. They evaluated FGSM attacks in both White-box and Black-box settings. However, they assumed attackers had access to the agent's training environments and DNN structures. Pattanaik et al. further extended this approach and constructed a loss function that can be used to attack RL systems more effectively [26]. Behzadan and Munir took a similar approach to Huang et al. but only allowed attackers to sample the parameters and network architecture of the target agent periodically [27]. Lin et al. argued that the attack presented by Huang et al. involves perturbation of an agent at every step, which is infeasible in real life, and presented a new timed-strategic attack, in which an adversary can attack a quarter as often but achieve the same reward degradation [16].

Finally, our work also links to imitation learning in the field of RL. Imitation learning tries to learn the policy of an expert, given inputs as a sequence of observation-action pairs. Syed and Schapire showed it is possible to reduce the apprenticeship learning problem to a classification [28]. However, to the best of our knowledge, we are the first to perform multistep prediction in a passive imitation learning setup. Typical passive imitation learning only predicts a single action given the current state [29], but we show we can sometimes predict n steps ahead using seq2seq learning.

IV. Method

A. Preliminaries

At each time period t, the environment provides a state s_t to an agent, be it 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 (MDP), and an agent learns a policy π that describes a probability distribution on the action space given the current state s_t [6, 21, 22]. The policy π is trained to maximise the expected discounted return R_e, where $R_e = \sum_{i=0}^{l} e^{\lambda i} r_{t+i}$, l is the total number of steps in an episode of game play and λ 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 of 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 $S_t = (s_t, s_{t-1}, \cdots, s_{t-n})$ and the sequence of actions $A_t = (a_t, a_{t-1}, \cdots, a_{t-n})$, where n is the length of the sequences and $0 \leq n \leq t$; we have the policy function:

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

To mount an adversarial attack on a trained policy π, we construct an adversarial sample for the current state s_t. The objective is to compute a small perturbation δ_t for the input to the policy function, which now becomes $\hat{S}_t = (s_t + \delta_t, s_{t-1}, \cdots, s_{t-n})$, such that the resulting action

$$\hat{a}_t = \pi(\hat{S}_t, A_{t-1}) \tag{2}$$

differs from the intended action a_t, in such a way as to give a successful attack. An MDP setup is assumed for simplicity of explanation, although in practice the problem description often changes from a complete MDP to a Partially Observed MDP (POMDP). This implies that the agent does not have the knowledge of the entire environment, only a subset. RL agents usually assume a POMDP environment.

B. Threat Model

We assume that the attacker can modify the environment so that the target agent receives a perturbed state $\hat{s_t}$ and aims at changing a future action a_{t+m} to $\hat{a}_{t+m}$. We assume that past states and target agent memory cannot be modified. If frame stacking is done on the agent side, the attack should not change previously stacked states. Since the attack is Black-box, we assume that:

1) The attacker has no knowledge of the target agent, its training method or the training hyperparameters;
2) The attacker has no access to the training environment.

These assumptions give a very strong Black-box box setup in comparison to prior works as illustrated in Table II in Appendix . The only explicit assumption we make is that the episodes were collected from the agent, which is in evaluation mode – i.e. random exploration is turned off and no more training is done. In all previous work the approximation of an agent involved retraining another RL agent to perform the same task as the target. Adversarial samples are then generated from the clone agent. In this paper, however, we cannot just retrain because of the strong Black-box setup.

C. Temporal Information Approximation

As mentioned previously in Section III, the problem of approximating a target agent is similar to an imitation learning problem. In this paper we reformulate RL as a multi-timestep learning problem. The agent relies on a learned policy π to act; its decision depends on its observation s_t, its previous actions A_{t-1} and previous states S_{t-1}. An approximation model predicts a sequence of future actions instead of a single one. For convenience, we define the approximation model as

$$A_t^f = f(A_{t-1}, S_{t-1}, s_t). \tag{3}$$

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

The function f is an approximator that takes sequence inputs A_{t-1} and S_{t-1} together with the 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 length n and an output sequence having length m. Following the Black-box setup, we only need to observe the agent playing the game to build up a collection of $((A_{t-1}, S_{t-1}, s_t), A_t^f)$ and use this collection to train the approximator f.

We trained a sequence-to-sequence neural network as f. It is multi-head with heads focusing on three different inputs (A_{t-1}, S_{t-1}, s_t) (blue, yellow and pink colour blocks in Figure 1). Its complexity and detailed architecture vary with different games and RL algorithms. We use grid search to determine the network architecture. We present the seq2seq training algorithm (Algorithm 1), network architectures (Table III) and seq2seq model accuracies (Table III) in Appendix .

D. Transferring Adversarial Samples

Once we have a seq2seq model that approximates the target agent, we develop attacks on it and apply them to the target agent.

In this paper, we have used a number of attacks of varying complexity. As the weakest attacker we assumed Random Gaussian Noise – which does not actually use any information from our model. It merely injects noise of a specific amplitude into the target agent's inputs, and serves as our baseline attack. As a more sophisticated attack, we used FGSM [10], and for the most complex attack, PGD [23]. This is not an exhaustive list of possible attacks, and and many complex attacks could be conducted. CW [24] is generally considered to be the strongest attack, but to get good misclassification rates and transferability, it needs thousands of executions in each episode of agent game play, making it infeasible in our application.

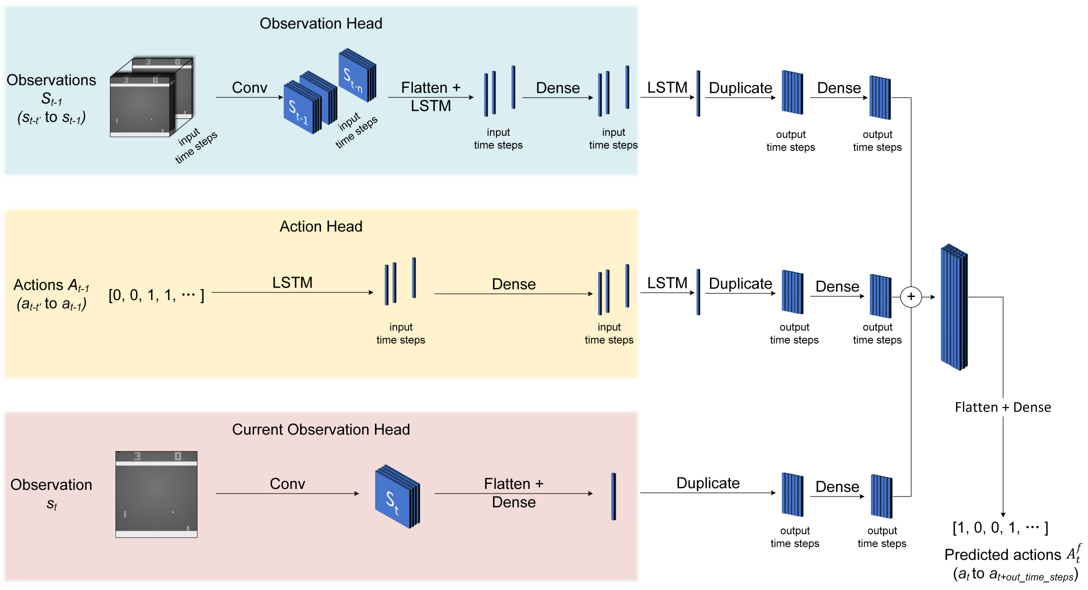

Fig. 1. An illustration of the sequence-to-sequence network's multi-head architecture, with observation head (S_{t-1}), action head (A_{t-1}) and current observation head (s_t). The output of the seq2seq model is a sequence of predicted future actions (A_t^f). The details of designing each head is game-dependent and shown in Table III in Appendix .

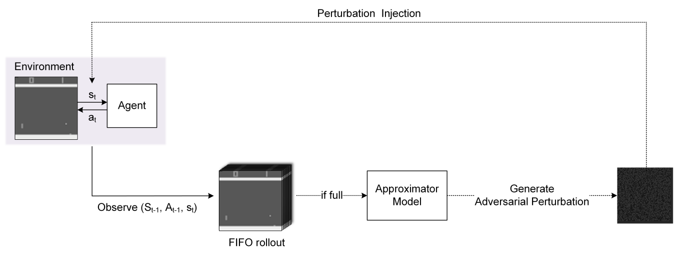

Fig. 2. The attacker observes how a trained agent plays a game and can inject perturbations into the agent's observation s_t. We collect a sequence of observations to feed into the seq2seq model and generate adversarial perturbations to attack the trained agent. The adversarial perturbation in this figure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

Our Black-box attack starts after n time steps when the rollout FIFO is full, having recorded the playing history to collate our inputs (A_{t-1}, S_{t-1}, s_t). We assume 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 we can use to craft adversarial perturbations to the target agent. Figure 2 shows how our attack works with an assumption that we can change the gaming environment. In addition, this figure shows how we prepare a FIFO rollout for the seq2seq approximator to generate adversarial inputs.

V. Evaluation

A. Experiment Setup

We trained agents using DQN [6], A2C [21] and Rainbow [22]. The target games are Cartpole, Space Invader and Pong. For evaluation the game's randomness seed was reset for every episode. Cartpole 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 84×84 image inputs [6]. The RL algorithms and game setups are developed on Ray and RLLib [30, 31].

As mentioned previously, we consider three attacks: Random Gaussian Noise, FGSM [10] and PGD [23]. Their implementation was adapted from the commonly-used adversarial ML library *Cleverhans* [32]. Adaptations and extensions were made to the framework as its current form does not support multi-input models or sequential outputs.

B. Temporal Approximation Results

We control the sequence output m to be 1 or 10 to illustrate that approximation is possible for both short and long sequences of future actions. The notion **Seq** is used for approximators that predict 10 sequential output actions. We found that reducing RL to a seq2seq approximation is possible and our seq2seq model shows consistently higher than 80% accuracy; in fact its average is about 90% over a range of different games. The detailed figures are shown in Table III in Appendix .

C. Reward-focused Black-box Attack

Previous attacks in RL focus on reducing the total reward of an agent in a White-box or Grey-box setup [16, 15]. In this section, we discuss how our proposed Black-box attack reduces an agent's reward score. This is a direct measurement of its game-playing ability; an efficient attack should be able to cut it quickly by injecting perturbations. Figure 3 shows the performance of attacking agents trained against Cartpole. These agents are trained with DQN, A2C and Rainbow. We also show how such agents perform against Atari games (Space Invader and Pong) in Figure 4 and Figure 5 in the

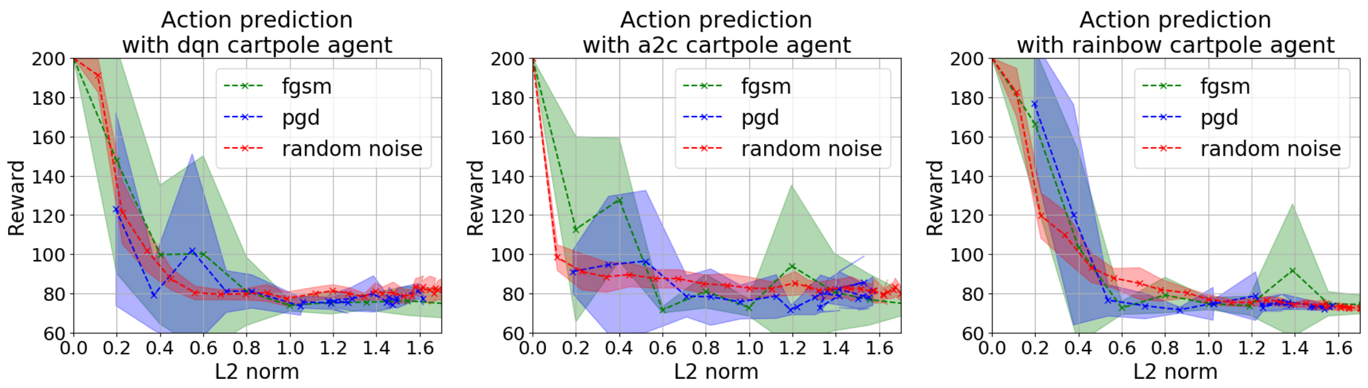

Fig. 3. Black-box reward-focused attacks on DQN, A2C and Rainbow trained on Cartpole. The x-axis shows the size of l_2 norm of the adversarial samples, while the y-axis shows the reward of the attacked agents.

Appendix . In Figure 3, we generate adversarial samples that aim at perturbing the agent's future action a_t, so we call this the *action prediction attack*. The adversarial samples are generated using untargeted attacks, where the attacker only wants to change the agent's action and does not seek to change it to any particular other action. In addition, we include action sequence attacks, where the attacker's goal is to flip a random future action in the predicted sequence $a_t, a_{t+1}...a_{t+m}$. We present error bars that are generated from 20 distinct runs, with the mean values extracted to plot the lines and the error bars representing a single standard deviation. In our attack, we start to inject adversarial perturbations into the agent after collecting n samples. The reward keeps accumulating in the initial collection stage, so even a powerful attack cannot reduce the score to zero. As these attacks are untargeted, the results are comparable with those of Huang et al., who used both in White-box and Grey-box setups [15]. The score decreases with increasing l_2 norms, meaning that agents are punished more for larger perturbations.

We first find that adversarial inputs produced by FGSM and PGD are no more efficient at reducing the target's rewards than Random Gaussian Noise, given the same noise budget. In particular, FGSM and PGD do not do significantly better in reward-based attacks than random jamming.

We also observe an effect similar to the canonical imitation learning problem. As we start attacking the agent, the performance of the attacks starts dropping. We hypothesise that this behaviour is caused by the inability of the approximator to accurately capture states that have not been observed. For sequential decision making using imitation learning, Bagnell noted that learning errors cascade, resulting in the learner encountering unknown states [33].This seems to be true for attacks based on approximated models in a fully Black-box setup as well. Pomerleau addressed this problem by learning to recover from mistakes [34]. That hints that, for Black-box setup with multiple decisions, attack performance may be limited by how well you know the agent's error recovery policy.

This also has a detrimental effect on the attacks targeting specific actions of the agent e.g. the approach proposed by Lin et al.. It is unlikely that the attacker will be able to tell which of the non-preferred actions should be taken to attack the agent, if they have never observed the agent taking them.

For reward-focused Black-box attacks, we summarise the following:

- The attacks have large variations in performance, and their effectiveness is associated with where the attack starts.
- Various attack schemes have similar performance for the same game, although the agent playing the game might be trained with various RL algorithms. Nothing significantly outperforms random jamming.
- Attack performance is bounded by the canonical imitation learning problem.

TABLE I
BLACK-BOX TEMPORAL-FOCUSED ATTACKS ON A2C AND RAINBOW. THE SEQ2SEQ MODEL IS TRAINED AGAINST THE DQN ALGORITHM AND IS USED TO GENERATE ADVERSARIAL SAMPLES ON A2C AND RAINBOW. 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.

Future step n:	Game	0	1	2	3	4	5	6	7	8	9
Pong											
PGD	Rainbow	62	8	12	14	6	8	16	16	28	18
	A2C	68	64	50	76	72	57	69	62	70	64
FGSM	Rainbow	57	14	18	16	12	22	10	14	10	26
	A2C	61	52	62	57	63	70	66	66	65	56
Space Invaders											
PGD	Rainbow	93	89	79	48	58	78	81	72	76	82
	A2C	70	83	85	83	85	83	85	89	81	79
FGSM	Rainbow	57	4	8	12	10	8	4	4	16	12
	A2C	89	79	83	81	77	77	77	80	78	77

D. *Time-bomb Attack*

So far, we have seen that adversarial attacks do no better than random noise jamming at reducing the target's game score, but are efficient in action-targeted attacks. This section demonstrates a new temporal attack, which we call the time-bomb attack. It uses the seq2seq model to flip the actions of the target agent after 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 later. For example, the attacker might want to cause an autonomous truck containing explosive fuel to crash five seconds in the future, to have time to get out of the way. Consider an agent with an original trajectory $s_t, ..., s_{t+m}$. We would like to change action a_{t+m} to $\hat{a}_{t+m}$ with only a perturbation at s_t. The perturbation at time t will now cause the agent to follow a completely different trajectory so that adversarial action $\hat{a}_{t+m}$ follows at state $\hat{s}_{t+m}$. Surprisingly, the fact of asking the agent to follow a different trajectory often makes little difference to the score against that the agent was trained to optimise, since game reward is an accumulative measurement.

We demonstrate the performance of the time-bomb attack in Table I. The top row of the table refers to the time-bomb delay. For instance, 1 means we aim to perturb action a_{t+1}, and x means a_{t+x}. We only send adversarial inputs at s_t, and all future observations made by the agents are clean. The numbers in the table show the success rates of flipping an action after at a specific future time. For results in Table I, 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 Table I 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. We limited the attacks in Table I to have $\epsilon = 0.3$ which implies the l_∞ norm is bounded by 0.3. This attack budget is picked to demonstrate the differences across games and RL algorithms. The results in Table I suggests that it is entirely possible to design a time-bomb attack on RL agents and sometimes we even achieve a success rate as high as 93%. Moreover, with a larger attack budget (with ϵ bigger than 0.7), we observe all agents on all games getting attacked with success rates consistently higher than 70%. We believe the time dimension is not yet fully explored in the field of adversarial RL. Our time-bomb attack shows that an attacker can in principle craft Black-box attacks that perturb only the current time frame, but cause the target agent to misbehave after a specific delay.

The implications of time-bomb attacks in RL are realistic and broad. First, we only inject adversarial samples at a given time step, so there may be effective attacks on RL agents given only a short attack window. In real life, constantly injecting adversarial noise into a system might trigger detection [35, 36, 37, 38]. Striking quickly at just the right time may be more efficient and unobtrusive. Second, we show it is possible to intervene at one point in time but trigger a particular adversarial action at a future time. This can have broad implications, from spoiling a warrior's aim to fooling trading systems. Finally, the time-bomb attack we describe is a general case of the path-planning attack shown by Lin et al..

VI. 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 (RL) agents. The Black-box attack assumes that attackers have no access to any internal states or the training details of an RL agent. To the best of 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 them to generate highly transferable adversarial samples. This improves the state of the art, as previous attacks were White-box or Grey-box. We show that even in a Black-box setup the attack will still work and disrupt the performance of an agent with an unknown objective function.

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.

Acknowledgements

Partially supported with funds from Bosch-Forschungsstiftung im Stifterverband.

APPENDIX

A. Threat Model Comparison

Table II compares the threat models between our attack and prior work. We show that we have the strongest Black-box assumption.

TABLE II
A COMPARISON TO SHOW OUR ATTACK HOLDS A STRONGER BLACK-BOX ASSUMPTION. THE FIRST ROW SHOWS WHETHER THE ATTACKER HAS ACCESS TO THE AGENT'S WEIGHTS, THE AGENT'S DNN STRUCTURES, THE TRAINING ALGORITHM AND THE TRAINING ENVIRONMENT. HUANG *et al.* PROPOSED TWO BLACK-BOX SETUPS AND WE ENUMERATED THEM AS 1 AND 2.

Attacker Access	DNN weights	DNN structure	Algorithm	Environment
Huang et al. 1	✗	✓	✓	✓
Huang et al. 2	✗	✓	✗	✓
Behzadan and Munir	✗	✗	✓	✓
Lin et al.	✓	✓	✗	✗
Ours	✗	✗	✗	✗

B. Temporal Information Approximation

Determining the number of input steps necessary for the input sequence is not trivial. 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 π to produce action a_t; internally, π 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 data for training (D_{train}) and evaluation (D_{eval}). Given an epoch count N, the selection of an input sequence length requires training of $n_{\max}$ 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 $n_{\max} = 50$.

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 to adapt to various games. The building blocks of each approximator are shown in Table III. 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 III, we achieve on average about 90% accuracy on all seq2seq models. Whether predicting a single action or a sequence of 10 actions at future time steps, our trained seq2seq model predicts them correctly with high accuracy. The results in Table III shows that imitating an RL agent can be formulated as a classification problem: predicting a future action sequence that the agent will perform given a history of its action-observation pairs.

Algorithm 1 Training and Architecture Details For Seq2Seq Model

```
1:  $D \leftarrow \varnothing$
2:  **while** $|D| < N$ **do**
3:      $E \leftarrow [\,]$
4:      $s_t$ = Init Env
5:      **while** (Game not done) **do**
6:          $a_t = \pi(s_t)$
7:          $s_{t+1} = Env(a_t)$
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** $n_i \in \{0, 1, ..., n_{\max}\}$ **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: $w \leftarrow \text{train}(n, D_{\text{train}}, N)$
```

C. Reward Focused Attacks on Space Invader and Pong

Figure 4 and Figure 5 show how reward-based attacks work on DQN agents trained for Space Invader and Pong. The results further supports our summary in Section V.

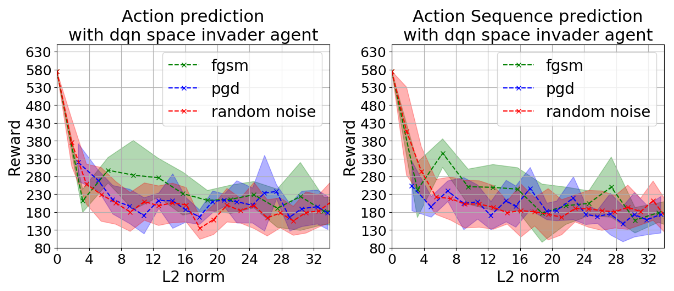

Fig. 4. Black-box reward-focused attacks on DQN trained on Space Invader. The x-axis shows the size of l_2 norm of the adversarial samples, while the y-axis shows the reward of the attacked agents. 'Action prediction' produces a single action but 'Action Sequence' predicts 10 future actions.

TABLE III

BLACK-BOX APPROXIMATION NETWORK CONFIGURATIONS AND ACCURACIES FOR DIFFERENT GAMES. WE MEASURE HOW ACCURATE OUR NETWORK PREDICTS THE NEXT ACTION OR A SEQUENCE OF ACTIONS WITH RESPECT TO A RL AGENT TRAINED WITH DQN. **SEQ** MEANS 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.

Game	**Acc**	**Obs Head**	**Action Head**	**Current Obs Head**	**Input Seq**
Cartpole	93%	2 LSTM, 1 Dense	1 LSTM, 1 Dense	1 Dense	50
Cartpole Seq	83%	2 LSTM, 1 Dense	2 LSTM, 1 Dense	1 Dense	50
Space Invader	86%	6 Conv, 3 LSTM, 2 Dense	2 LSTM, 1 Dense	5 Conv, 2 Dense	10
Space Invader Seq	82%	6 Conv, 3 LSTM, 2 Dense	2 LSTM, 1 Dense	5 Conv, 2 Dense	5
Pong	97%	6 Conv, 3 LSTM, 2 Dense	2 LSTM, 1 Dense	5 Conv, 2 Dense	2
Pong Seq	97%	6 Conv, 3 LSTM, 2 Dense	2 LSTM, 1 Dense	5 Conv, 2 Dense	10
Average	90%	-	-	-	-

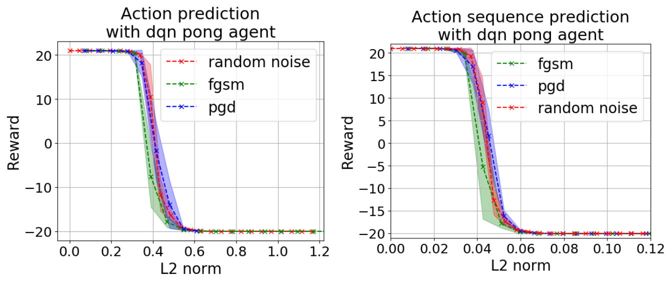

Fig. 5. Black-box reward-focused attacks on DQN trained on Pong. The x axis shows the size of l_2 norm of the adversarial samples, while the y axis shows the reward of the attacked agents. 'Action prediction' produces a single action but 'Action Sequence' predicts 10 future actions.

D. Adversarial Image Visualisation

Figure 6 shows a real adversarial image used in the time-bomb attack. The generated adversarial input successfully triggers an adversarial action in the future, but its perturbation is imperceptible to humans unless we increase its resolution.

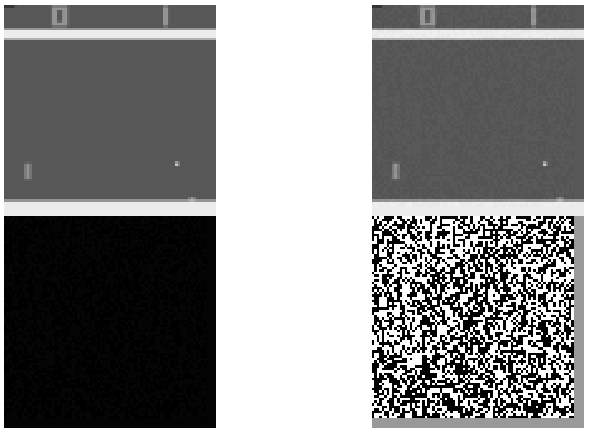

Fig. 6. Real adversarial inputs generated for the Pong game. The first image (top left) is the original input to an agent; the second image is the perturbed input; the third image shows the perturbation; and the fourth image is the same perturbation rescaled to 0-255 for visibility, with 0 being black and 255 being white. If the images had been rescaled back to 0 and 1, the l_2 norm of this perturbation would be 0.62, and its l_∞ norm 0.01.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, Jun. 2017.

[3] J. Nicholson, K. Takahashi, and R. Nakatsu, "Emotion recognition in speech using neural networks," *Neural computing & applications*, vol. 9, no. 4, 2000.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017.

[5] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, 2016.

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

[7] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, 2016.

[8] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li, "Drn: A deep reinforcement learning framework for news recommendation," in *Proceedings of the 2018 World Wide Web Conference*, 2018.

[9] Y. Nevmyvaka, Y. Feng, and M. Kearns, "Reinforcement learning for optimized trade execution," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations Workshop (ICLR Workshop)*, 2015.

[11] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *International Conference on Learning Representations Workshop (ICLR Workshop)*, 2017.

[12] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rah-

mati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[13] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[14] Y. Zhao, I. Shumailov, R. Mullins, and R. Anderson, "Understanding the interactions between adversarial attacks and neural network compression," *SysML*, 2018.

[15] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," in *International Conference on Learning Representations Workshop (ICLR Workshop)*, 2017.

[16] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, "Tactics of adversarial attack on deep reinforcement learning agents," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017.

[17] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014.

[18] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017.

[19] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the landscape of spatial robustness," in *International Conference on Machine Learning*, 2019.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, 2015.

[21] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016.

[22] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[23] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

[24] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017.

[25] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arxiv*, vol. abs/1312.6199, 2013.

[26] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust deep reinforcement learning with adversarial attacks," *CoRR*, 2017.

[27] V. Behzadan and A. Munir, "Vulnerability of deep reinforcement learning to policy induction attacks," in *International Conference on Machine Learning and Data Mining in Pattern Recognition*, 2017.

[28] U. Syed and R. E. Schapire, "A reduction from apprenticeship learning to classification," in *Advances in neural information processing systems*, 2010.

[29] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, 2018.

[30] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan *et al.*, "Ray: A distributed framework for emerging {AI} applications," in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 2018.

[31] E. Liang, R. Liaw, P. Moritz, R. Nishihara, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica, "Rllib: Abstractions for distributed reinforcement learning," *arXiv preprint arXiv:1712.09381*, 2017.

[32] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, and R. Long, "Technical report on the cleverhans v2.1.0 adversarial examples library," *arXiv preprint arXiv:1610.00768*, 2018.

[33] J. A. Bagnell, "An invitation to imitation," 2015.

[34] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, 1989.

[35] S. Chen, N. Carlini, and D. A. Wagner, "Stateful detection of black-box adversarial attacks," *arxiv*, vol. abs/1907.05587, 2019.

[36] I. Shumailov, Y. Zhao, R. Mullins, and R. Anderson, "The taboo trap: Behavioural detection of adversarial samples," *arxiv*, vol. abs/1811.07375, 2018.

[37] I. Shumailov, X. Gao, Y. Zhao, R. Mullins, R. Anderson, and C.-Z. Xu, "Sitatapatra: Blocking the transfer of adversarial samples," *arxiv*, vol. abs/1901.08121, 2019.

[38] I. Shumailov, Y. Zhao, R. Mullins, and R. Anderson, "Towards certifiable adversarial sample detection," *arxiv*, vol. abs/2002.08740, 2020.