Temporal Watermarks for Deep Reinforcement Learning Models

Kangjie Chen
Nanyang Technological University
Singapore
kangjie.chen@ntu.edu.sg

Shangwei Guo*
Nanyang Technological University
Singapore
gswei5555@gmail.com

Tianwei Zhang
Nanyang Technological University
Singapore
tianwei.zhang@ntu.edu.sg

Shuxin Li
Nanyang Technological University
Singapore
shuxin.li@ntu.edu.sg

Yang Liu
Nanyang Technological University
Singapore
yangliu@ntu.edu.sg

ABSTRACT

Watermarking has become a popular and attractive technique to protect the Intellectual Property (IP) of Deep Learning (DL) models. However, very few studies explore the possibility of watermarking Deep Reinforcement Learning (DRL) models. Common approaches in the DL context embed backdoors into the protected model and use special samples to verify the model ownership. These solutions are easy to be detected, and can potentially affect the performance and behaviors of the target model. Such limitations make existing solutions less applicable to safety- and security-critical tasks and scenarios, where DRL has been widely used.

In this work, we propose a novel watermarking scheme for DRL protection. Instead of using *spatial* watermarks as in DL models, we introduce *temporal* watermarks, which can reduce the potential impact and damage to the target model, while achieving ownership verification with high fidelity. Specifically, (1) we design a new damage metric to select sequential states for watermark generation; (2) we introduce a new reward function to efficiently alter the model's behaviors for watermark embedding; (3) we propose to utilize a predefined probability density function of actions over the watermark states as the verification evidence. Our method is general and can be applied to various DRL tasks with either deterministic or stochastic reinforcement learning algorithms. Extensive experimental results show that it can effectively preserve the functionality of DRL models and exhibit significant robustness against common model modifications, e.g., fine-tuning and model compression.

KEYWORDS

Deep Reinforcement Learning; Intellectual Property; Watermarking

ACM Reference Format:
Kangjie Chen, Shangwei Guo, Tianwei Zhang, Shuxin Li, and Yang Liu. 2021. Temporal Watermarks for Deep Reinforcement Learning Models. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

Deep Reinforcement Learning (DRL) has demonstrated its effectiveness in various complex tasks, e.g., robotics control [9], competitive video games [17, 22, 31], and autonomous driving [20]. Due to the excellent performance and robustness, DRL is now in an accelerating process of commercialization. Since generating a DRL policy requires a huge amount of computation resources as well as expertise, a well-trained DRL model has become the core Intellectual Property (IP) of AI applications and products. It is of paramount importance to protect such assets, and prevent illegitimate plagiarism, unauthorized distribution and reproduction of DRL models.

One common approach to IP protection is watermarking [7], which was originally introduced to identify the ownership of images, audios, videos, etc. Such watermarks are designed to be robust and cannot be removed by common signal processing techniques. Motivated by this idea, several watermarking schemes were proposed to protect the copyright of Deep Learning (DL) models [1, 16, 26]. These solutions carefully craft a set of unique sample-label pairs as watermarks. They train a model to memorize the correlation between these samples and labels, which will not be recognized by other models. For verification, the owner remotely queries the suspicious model with these samples and uses the corresponding predictions as the ownership evidence. These methods can preserve the performance of the watermarked models on normal samples and ensure the watermarks cannot be removed by common model transformations, e.g., fine-tuning, model compression, etc.

Challenges arise when applying existing solutions to or designing new ones for DRL models. First, although a DRL policy also adopts deep neural networks, it performs learning and prediction in a sequential and stochastic control process. The characteristics of the policy are reflected by sequences of behaviors, instead of single input-output pairs at one time instant. The high stochasticity in DRL policies can reduce the verification accuracy when using discrete watermark samples while ignoring the sequential features. Second, the predicted action at one moment can affect the following states and actions, and even the entire process. One abnormal state (e.g., adversarial perturbation [24] or backdoor triggers [13, 28]) can possibly cause the agent to crash or fail. As a result, watermarking methods for conventional DL models can bring unexpected consequences to DRL applications. Such severity is amplified when the DRL model is used in safety- or security-critical scenarios. Third, all existing watermarks are *spatial*, which can be detected or removed by sophisticated attacks [2, 6, 10].

To our best knowledge, the only solution for DRL watermarking is [3], which embeds a sequential pattern of out-of-distribution states and actions of an extra environment into the target DRL

[*] Corresponding author

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

policy. Such requirement is not easy to satisfy under most scenarios. Besides, deploying the DRL model under a different environment can be easily recognized by the adversary, who can then simply falsify the prediction results to invalidate the verification process. More importantly, this work only considers one deterministic DRL model (DQN). Its effectiveness for other models, and robustness against model transformations remain unknown.

Motivated by these limitations, we propose a novel temporal-based watermarking methodology for DRL policies. Different from [3], we adopt the sequences of states and action probability distributions within the same environment as the watermarks. This can increase the risk of model failure caused by the watermark interference. We propose three techniques to overcome this issue. First, we introduce *damage-free states*, from which the DRL system can still be safe and reliable when there is a deviation of action probability. We design a new algorithm to identify such states, and use them for the watermarks. Second, we design a new reward function for both deterministic and stochastic reinforcement learning algorithms, which can efficiently implant the desired watermark behaviors into the model. Third, we propose to use statistic tests to verify the action probability distributions of the damage-free watermark states. The watermarked model can still be distinguished even it performs normally on the watermark states.

Our approach is more general-purpose than [3]. Comprehensive evaluations show it can achieve very high verification accuracy and low error rate for both deterministic and stochastic DRL contexts and tasks, and strong robustness against various model transformations. The main contributions of this paper are as follows:

- A novel temporal-based watermarking scheme for deterministic and stochastic DRL models and tasks.
- A new damage metric to generate watermarks which are safe to be embedded into the target model.
- A new reward function to alter the behaviors of DRL models in a controllable and efficient way.
- Adoption of statistic tests to verify the action probability distributions of watermark states for watermark extraction.
- Extensive experiments to illustrate the functionality and robustness of the proposed scheme under various system settings.

2 RELATED WORK

2.1 Watermarking DL Models

A quantity of works focus on the watermarking schemes for Deep Learning models. These methods can be classified into two categories. The first one is *white-box* watermarking. Motivated by the traditional watermarking techniques on digital multimedia, this scheme embeds watermarks into DL models' parameters without altering the models' performance. For example, Uchida et al. [27] injected a bit-vector as the watermark into the model parameters via a particular parameter regularizer in the loss function. Rouhani et al. [19] implanted watermarks in the probability density function of the activation layers, which has small impacts on the static properties of model parameters. However, these parameter-embedding solutions require the model owner to have full accesses to the parameters during verification, and become ineffective in the scenario where the target model is a black-box to the external users.

The second category is *black-box* watermarking. The model owner trains (or fine-tunes) the model in a special way to make it give unique output for certain carefully-crafted samples, while preserve the same behaviors for normal samples. During the verification phase, the owner can just use those samples to query the suspicious model, and make decisions based on the prediction results. For instance, some works utilized backdoor attack techniques to watermark the DL model, and used samples with triggers or out of distribution to verify the existence of watermarks [1, 16, 18, 30]. Le Merrer et al. [15] adopted adversarial examples to detect the suspicious models, which can accurately fingerprint the classification boundaries of the target model. These approaches dominate the ones in the first category, as they enable verification with only black-box accesses, and achieve very satisfactory accuracy.

2.2 Watermarking DRL Models

As mentioned in Section 1, considering the sequential and safety-critical features of DRL models, it is more challenging to embed watermarks into DRL policies. To the best of our knowledge, there is only one watermarking solution in the reinforcement learning scenario [3] up to the date of writing. To reduce the negative impact of watermarks, this solution adopts a set of out-of-distribution state sequences under a different environment for watermark embedding and verification. This solution can indeed preserve the model behaviors and robustness within the target environment. However, the requirement of an extra environment can decrease its applicability. It is also very easy for an adversary to detect the abnormal states and environments during testing, and then tamper with the verification results. Besides, this work is in a lack of generality, as it only considers a deterministic DQN policy. The robustness of such watermarks against model transformation was never evaluated.

3 PROBLEM DEFINITION

3.1 System and Threat Models

A reinforcement learning policy describes a Markov Decision Process (MDP). An MDP can be formulated as a tuple $(\mathbb{S}, \mathbb{A}, \mathbb{P}, r, \gamma)$, where $\mathbb{S}$ is the state space, $\mathbb{A}$ is the action space, $\mathbb{P} : \mathbb{S}\times\mathbb{A}\times\mathbb{S} \rightarrow [0, 1]$ is the state transition probability, $r(s, a)$ is the reward function and $\gamma \in [0, 1)$ is the discount factor where smaller values place more emphasis on immediate rewards. During training, a DRL agent interacts with an environment *env* and learns a model M (i.e., policy) to predict the optimal action of the MDP based on the reward from *env*. Given a state s, M outputs the action probability distribution (APD) P over $\mathbb{A}$. A deterministic policy chooses the action with the maximum probability directly, while a stochastic policy samples an action from $\mathbb{A}$ following P. Without loss of generality, we describe the watermarking scheme for stochastic reinforcement learning policies. It can be applied to the deterministic ones as well.

Figure 1 illustrates the overview of the framework for IP protection and ownership verification of DRL policies. We follow the same system model as the conventional DL watermarking scenario [1, 30]: we consider an unauthorized user (adversary) which obtains an illegal copy of the target model M and attempts to use it for profit without authorization. The adversary might use common model transformation techniques (e.g., fine-tune, model compression) to slightly alter the model to make it different from the original one.

Such processing operations can also help the transformed model adapt to the adversary's own dataset or reduce the computation complexity. The owner wants to verify and detect whether a suspicious model M' is a plagiarized one from M. However, the owner only has black-box accesses to M', i.e., he can only observe the produced actions within a given environment. To achieve this goal, he can embed watermarks into his DRL model, causing it to have unique behaviors over certain environmental states. During the verification phase, he can query the suspicious model M' with these states, and collect the corresponding action sequences as the evidence of model plagiarism if they match the watermarks.

3.2 Temporal Watermarking

Existing works focus on spatial watermarks, which can be invalidated by advanced attacks [2, 6, 10]. Instead, we propose a temporal watermarking scheme, which is formally defined as below:

Definition 1. A temporal watermarking scheme is defined as a tuple of probabilistic polynomial time algorithms (**WMGen**, **Mark**, **Verify**), where

- **WMGen** generates a dataset $\mathbb{C}$, which consists of n sequences of state and the corresponding APD pairs, with the length of L:
$$\mathbb{C} = \{TW_i\}_{i=0}^{n-1}$$
$$TW_i = [(s_{i,0}, P_{i,0}), (s_{i,1}, P_{i,1}), ..., (s_{i,L-1}, P_{i,L-1})]$$
in which $s_{i,j}$ is the j-th state of the i-th sequence; $P_{i,j}$ is the corresponding APD over $\mathbb{A}$.
- **Mark** embeds the state sequences into a DRL model and outputs the watermarked model $\widehat{M}$ such that for $\forall\ s_{i,j},\ i \in [0,n), j \in [0,L)$, the APD of $\widehat{M}$ will be changed from $P_{i,j}$ to $\widehat{P}_{i,j}$. It also produces the final dataset $\mathbb{W}$ of watermarks:
$$\mathbb{W} = \{\widehat{TW}_i\}_{i=0}^{n-1}$$
$$\widehat{TW}_i = [(s_{i,0}, \widehat{P}_{i,0}), (s_{i,1}, \widehat{P}_{i,1}), ..., (s_{i,L-1}, \widehat{P}_{i,L-1})]$$
- **Verify** starts a suspicious DRL model M' with the states $\{s_{i,j}\}_{i,j=0}^{n-1,L-1}$ and collects the state-APD sequences:
$$\mathbb{W}' = \{TW_i'\}_{i=0}^{n-1}$$
$$TW_i' = [(s_{i,0}, P_{i,0}'), (s_{i,1}, P_{i,1}'), ..., (s_{i,L-1}, P_{i,L-1}')]$$
If the distance between $\mathbb{W}$ and $\mathbb{W}'$ is smaller than a predefined value τ, **Verify** outputs 1. Otherwise it outputs 0.

3.3 Watermarking Requirements

As we discussed in Section 1, a good watermarking scheme should have the following properties.

Requirement 1. (*Functionality-preserving*) Let M be the well-trained model without embedded watermarks. The watermarked model $\widehat{M}$ should exhibit the competitive performance compared with M. We define $p_{\widehat{M},\mathbb{S}}$ as the probability that $\widehat{M}$ gets more cumulative rewards from the environment during an episode than M on the normal state space $\mathbb{S}$:

$$p_{\widehat{M},\mathbb{S}} = Pr(\sum_{j=0}^{T-1} \gamma^j r(s_j, \widehat{M}(s_j)) \geq \sum_{j=0}^{T-1} \gamma^j r(s_j, M(s_j)) - \delta, s_0 \backsim \mathbb{S}) \tag{1}$$

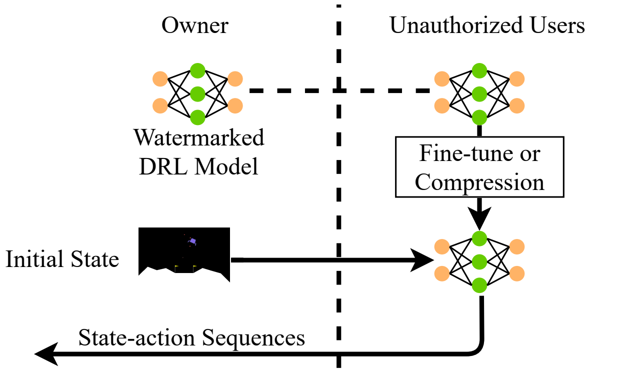


Figure 1: Watermarking framework for IP protection and ownership verification of DRL models.

where T is the final time step of the current episode, δ is a small variance that allows the rewards of M to exceed than that of $\widehat{M}$. We expect $p_{\widehat{M},\mathbb{S}}$ to be close to 1 as much as possible.

Requirement 2. (*State-preserving*) Previous work [3] adopted out-of-distribution state sequences in a totally different environment for watermarks. This can be easily recognized by the adversary who will then tamper with the verification results. To make the verification stealthier, a good watermarking scheme should use the watermark states sampled from the same state space $\mathbb{S}$, i.e.,

$$\forall\ i \in [0,n), j \in [0,L), s_{i,j} \in \mathbb{S}. \tag{2}$$

Requirement 3. (*Damage-free*) The most common method is to embed backdoors into the models as the watermark. The existence of backdoors can significantly change the prediction results on the watermark samples, which can lead to severe consequences in safety- and security-critical tasks, e.g., autonomous driving. So a good watermarking scheme should be damage-free to the target model. Let $p_{\widehat{M},\mathbb{W}}$ be the damage value of $\widehat{M}$ on $\mathbb{W}$. Similar to $P_{\widehat{M},\mathbb{S}}$, we define $p_{\widehat{M},\mathbb{W}}$ as the probability that $\widehat{M}$ obtains more cumulative rewards on the watermarks $\mathbb{W}$, i.e.,

$$p_{\widehat{M},\mathbb{W}} = Pr(\sum_{j=0}^{L-1} \gamma^j r(s_j, \widehat{M}(s_j)) \geq \sum_{j=0}^{L-1} \gamma^j r(s_j, M(s_j)) - \delta', s_j \backsim \mathbb{W}). \tag{3}$$

where δ' is the parameter as in Eq. 1. $\widehat{M}$ is damage-free if $P_{\widehat{M},\mathbb{W}}$ is close to 1.

Requirement 4. (*Robustness*) Since the adversary may modify the watermarked model with common model transformations, we expect that the embedded watermarks should be robust and cannot be removed after those changes. Formally, let $d_{\widehat{M},M'}$ be the distance of APDs between the watermarked model $\widehat{M}$ and transformed model M' over the watermark states:

$$d_{\widehat{M},M'} = \frac{1}{n} \sum_{i=0}^{n-1} \sum_{j=0}^{L-1} distance(\widehat{P}_{i,j}, P_{i,j}'), \tag{4}$$

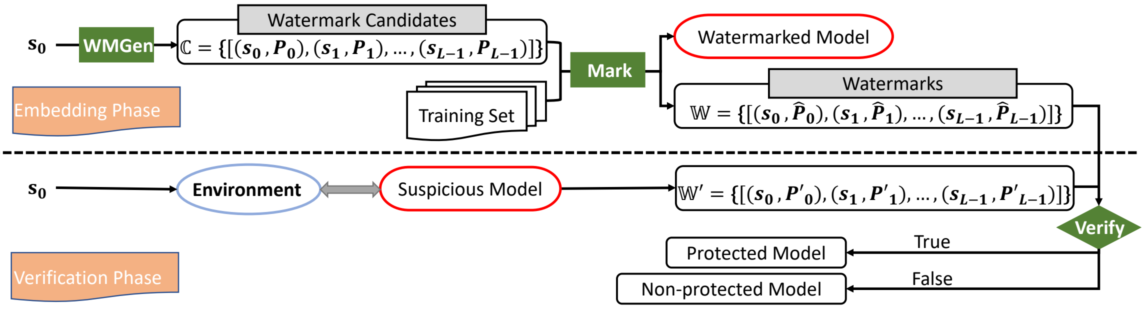

s_0
WMGen
Watermark Candidates
$\mathbb{C} = \{[(s_0, P_0), (s_1, P_1), \dots, (s_{L-1}, P_{L-1})]\}$
Training Set
Embedding Phase
Mark
Watermarked Model
Watermarks
$\mathbb{W} = \{[(s_0, \widehat{P}_0), (s_1, \widehat{P}_1), \dots, (s_{L-1}, \widehat{P}_{L-1})]\}$
s_0
Environment
Suspicious Model
$\mathbb{W}' = \{[(s_0, P'_0), (s_1, P'_1), \dots, (s_{L-1}, P'_{L-1})]\}$
Verify
True
False
Protected Model
Non-protected Model
Verification Phase

Figure 2: Embedding and verification phases of our temporal watermarking methodology.

where $\widehat{P}_{i,j}$ and $P'_{i,j}$ are the APDs of $\widehat{M}$ and M' on the watermark state $s_{i,j}$. If $\widehat{M}$ is robust against model transformations, the value of $d_{\widehat{M},M'}$ should be smaller than a predefined threshold.

4 METHODOLOGY

In this section, we describe our novel temporal watermarking methodology for DRL policies. Our solution consists of three new algorithms, with the workflow illustrated in Figure 2. During the embedding phase, the model owner calls **WMGen** to generate a dataset of watermark candidates $\mathbb{C}$. Then he uses **Mark** to train a watermarked model and obtain the final watermark sequences $\mathbb{W}$. During the verification phase, he queries a suspicious model with the states of each watermark sequence, and extracts the runtime results $\mathbb{W}'$. By comparing $\mathbb{W}'$ and $\mathbb{W}$ using **Verify**, the owner can verify if the suspicious model is the watermarked one.

Our method can satisfy all the requirements in listed Section 3.3 without the need of extra environments. This is achieved with three innovations. In **WMGen**, we search the *damage-free states* to generate safe watermark candidates with minimal interference on the DRL policy. In **Mark**, we introduce new reward functions that enable the policy to memorize the watermarks during training. In **Verify**, we adopt statistic tests to compare the probability distributions of state-APD pairs to identify the existence of watermarks. Below we present the details of each algorithm.

4.1 Watermark Generation

As the first step, we need to carefully design watermarks to satisfy the requirements of state-preserving and damage-free. We introduce a new concept of *damage-free state* to achieve these goals:

Definition 2. (*Damage-free State*) Let $s \in \mathbb{S}$, P be a state and the corresponding APD. P defines $a^* \in \mathbb{A}$, which is the action with the highest probability. σ is the variance of P: $\sigma = \mathrm{Var}(P)$. s is (ϵ, ψ) *damage-free* if σ is smaller than ϵ and the DRL agent can achieve a minimum score of ψ at the end of an episode when it executes an arbitrary action $a \in \mathbb{A}/a^*$.

Informally, s is damage-free if the agent can choose any legal actions at state s to complete the task perfectly. In contrast, a large APD variance means that the agent tends to choose a certain action a^* with strong will at state s, indicating that s is critical for the task and may cause crash if other actions are selected instead. With damage-free states, we define the watermark candidate as follow.

Definition 3. (*Watermark Candidate*) Given a clean DRL model M, a watermark candidate is a unique temporal sequence of damage-free states and the corresponding APDs predicted by M:

$$TW = [(s_0, P_0), (s_1, P_1), \dots, (s_{L-1}, P_{L-1})]$$

The watermark candidate can guarantee that the changes of APD on damage-free states have negligible impact on the agent's behaviors, but still observable for ownership verification.

We empirically search for the watermark candidates in a brute-force way, as illustrated in Algorithm 1.

The goal of **WMGen** is to identify a dataset of watermark candidates from the target DRL model M to be watermarked[1]. The model owner takes the following steps to generate qualified watermark candidates. (1) If the number of watermark candidates is smaller than n, he randomly samples a normal state $s \in \mathbb{S}$. Originating from s, he analyzes the behaviors of the model M, and obtains the APD P and the action a^* with the highest probability (Line 6). (2) He checks whether s is damage-free. In particular, he traverses all the actions in $\mathbb{A}/a^*$ and obtains the minimal reward score from *env*. If this score is larger than a given threshold ψ and the variance of P is smaller than ϵ, then s is damage-free and (s, P) will be added to the watermark candidate sequence TW. Otherwise, he needs to roll back and start from a new initial state (Line 2). With the above procedure, the owner is able to get a dataset $\mathbb{C}$ that contains multiple watermark candidate sequences.

4.2 Watermark Embedding

Given the identified set $\mathbb{C}$ of watermark candidates, the next goal is to embed them into the target DRL model M. We design a novel algorithm, **Mark**, to achieve this goal with functionality-preserving and high robustness. The key insight of our algorithm is to encourage the model to predict different actions (or at least with different APDs) on the damage-free states in these watermark candidates. This can be used for both deterministic and stochastic DRL policies.

Let s, P be a damage-free state and the corresponding APD in $TW \in \mathbb{C}$, and a^* be the action the agent will select with the highest probability. We aim to fine-tune the model M to learn a different APD $\widehat{P}$ by encouraging it to select a different action $\widehat{a}$ randomly sampled from $\mathbb{A}/a^*$. To this end, we introduce a novel reward function that adds an incentive reward on the original one over the

[1] We consider the case that the model owner has a clean model and wants to implant watermarks to it. If the model owner wants to train a watermarked model from scratch, he can first train a clean model and then follow our algorithms.

Algorithm 1: WMGen: Generating (ϵ, ψ) damage-free temporal watermark candidates.

```
Input: Clean DRL model $M$, environment $env$, candidate
       number $n$, length $L$
1  $\mathbb{C} \leftarrow \emptyset$;
2  while $|\mathbb{C}| < n$ do
3      $TW \leftarrow \emptyset$;
4      Randomly sample $s \in \mathbb{S}$ and $env$.reset($s$);
5      while current episode is not finished do
6          $P \leftarrow M$.action_prob($s$) and $a^* \leftarrow max_a(P)$ ;
7          if $|TW| < L$ then
8              $score \leftarrow$ the minimal score of the episodes that
                traverse all $a \in \mathbb{A}/a^*$ ;
9              if $score > \psi$ and $Var(P) < \epsilon$ then
10                 $TW$.add(($s$, $P$) ) ;
11             else
12                 goto Line 2;
13         $a \leftarrow$ sample an action following $P$;
14         $s \leftarrow env$.step($a$);
15     $\mathbb{C}$.add($TW$) ;
16 return $\mathbb{C}$
```

Algorithm 2: Mark: Embedding watermarks into the DRL model M.

```
Input: Environment $env$, watermark candidates $\mathbb{C}$, length $T$,
       reward threshold $R$
1  Initialize the DRL model $M$ and the training buffer $\mathbb{B} \leftarrow \emptyset$ ;
2  for $s, P \in \mathbb{C}$ do
3      $\widehat{a} \leftarrow$ sample a random action in $\mathbb{A}/a^*$;
4      $\widehat{r} \leftarrow r^e(s, \widehat{a})$;
5      $\mathbb{B}.add(s, \widehat{a}, \widehat{r})$
6  for each $seed \in \mathbb{S}$ do
7      while current episode is not finished do
8          $a \leftarrow$ sample an action following $P$;
9          $s, r \leftarrow env.step(a)$;
10         if $s \notin \mathbb{C}$ then
11             $\mathbb{B}$.add($s, a, r$);
12     $\theta_M \leftarrow \theta_M - lr\nabla \sum L(s)$ ;
13     if $eval(M) \geq R$ then
14         $\widehat{M} \leftarrow M$;
15         goto Line 6 ;
16 for each $TW \in \mathbb{C}$ do
17     $s \leftarrow$ the first damage-free state of $TW$;
18     $\widehat{TW} \leftarrow \emptyset$;
19     while $|\widehat{TW}| \leq T$ do
20         $\widehat{P} \leftarrow \widehat{M}$.action_prob($s$) ;
21         $\widehat{TW}$.add(($s$, $\widehat{P}$)) ;
22         $s \leftarrow env$.step(max$_a$($P$)) ;
23     $\mathbb{W}$.add($\widehat{TW}$) ;
24 return $\widehat{M}$, $\mathbb{W}$
```

damage-free states. Formally, let $r(s, a)$ be the original reward function. For a damage-free state $s \in TW$, our new reward function $r^e(s, a)$ returns the sum of the original reward with an additional incentive reward η:

$$r^e(s,a) = \begin{cases} r(s,a) + \eta, & s \in TW \text{ and } a = \widehat{a} \\ r(s,a), & \text{others.} \end{cases} \tag{5}$$

We choose common loss functions $L(s)$ to fine-tune the model, where the reward function is replaced with our new one. For stochastic DRL policies (e.g., REINFORCE [29]), we use the following loss function to train the model:

$$L(s) = \text{cross\_entropy\_loss}(M(s), a)G(s) \tag{6}$$

$$G(s) = r^e(s, a) + \gamma G(s') \tag{7}$$

where $G(s)$ is the accumulative reward with a discount factor γ of all the rewards from previous episodes, and s' is the next state.

For deterministic policies (e.g., DQN [17]) which simply output the most-likely actions instead of statistically sampling actions based on the APD, we adopt the loss function with the temporal difference (TD) error [25]:

$$L(s,a) = \left(r^e(s,a) + \gamma \max_{a'} Q(s', a') - Q(s,a)\right)^2 . \tag{8}$$

where $Q(s, a)$ is the value function to estimate the goodness of a on s, and s', a' are the next state and the corresponding action.

The model owner can optimize the parameters of M with this new loss function and the damage-free states using the stochastic gradient descent technique:

$$\theta_{t+1} = \theta_t - lr\nabla \sum_{j=0}^{T-1} L(s_j) \tag{9}$$

where θ_t is the parameters of M at the t-th iteration, and lr is the learning rate. The optimization process ends when the reward M achieved on a validation dataset is larger than a given threshold R.

After the fine-tuning process, the behaviors of the target model on the damage-free states will be altered, and the new APDs will be different from the original ones in the watermark candidates $\mathbb{C}$. To identify the final watermarks, the model owner queries the fine-tuned model $\widehat{M}$ from the initial damage-free states in $\mathbb{C}$, and record the new state-APD sequences. For each initial state, he collects the subsequent states and the corresponding action probability distributions. Finally he can obtain a temporal sequence $\widehat{TW} = [(s_0, \widehat{P}_0), (s_1, \widehat{P}_1), ..., s_{L-1}, \widehat{P}_{L-1})]$ that forms a unique watermark for this new model.

Algorithm 2 illustrates the details of embedding watermarks into a DRL model via fine-tuning. The owner first initializes the DRL model M and empties a training buffer $\mathbb{B}$ (Line 1). For each damage-free state $s \in \mathbb{C}$, he randomly samples an action different from the most-likely one a^* and replaces the original reward $r(s, a)$ with the new reward $r^e(s, \widehat{a})$ (Lines 2 - 5). Then he adds the new training samples into $\mathbb{B}$. During the optimization process, the owner collects samples from $\mathbb{B}$, computes the loss with Equation 6 or 8 and updates M with the stochastic gradient descent technique (Lines 7 - 15). After the watermarked model $\widehat{M}$ is obtained, the owner runs it from the same initial damage-free states in TW, and collects the

Algorithm 3: Verify: extracting the embedded watermarks from a suspicious DRL model M'.

```
Input: Watermark dataset $\mathbb{W}$, distance threshold $\tau$
for each $\widehat{TW} \in \mathbb{W}$ do
    for each of $(s_i, \widehat{P}_i)_{i=0}^{L-1} \in \widehat{TW}$ do
        Run the agent on $s_i$ and calculate the APD $P'_i$;
        $d_{s_i} \leftarrow \sum_a \widehat{p}_{i,a} \log \frac{\widehat{p}_{i,a}}{p'_{i,a}}$;
    $d_{\widehat{TW},TW'} \leftarrow \sum_{i=0}^{L} d_{s_i}$;
$d_{\widehat{M},M'} \leftarrow \frac{1}{|n|} \sum_{\widehat{TW} \in \mathbb{W}} d_{\widehat{TW},TW'}$;
if $d_{\widehat{M},M'} \leq \tau$ then
    $IsWatermarked$ = True ;
else
    $IsWatermarked$ = False ;
return $IsWatermarked$
```

altered APDs. The pairs of states and new APDs are added to the final watermark sequence $\widehat{TW}$ (Lines 16 - 23).

4.3 Ownership Verification

The owner extracts the watermarks by running the agent on the watermark states, observing the subsequent state-APD pairs and checking whether the behaviors match the temporal watermark $\widehat{TW}$. Algorithm 3 describes this process. Due to the stochastic property of a DRL agent, for each watermark state in $\widehat{TW}$, the action can vary following the corresponding APD. To obtain the statistical characteristics of the agent on a watermark state s, the owner can run the agent over s for multiple times, collect the predicted actions and calculate their probability distribution. Thus, the owner is able to get the temporal sequence $TW' = [(s_0, P'_0), ..., (s_{L-1}, P'_{L-1})]$.

The owner calculates the distance between $\widehat{TW}$ and TW' for similarity comparison. Since the states are the same, the owner only needs to compare the distance of APDs between the two sequences. We adopt Kullback–Leibler (KL) divergence [14] to estimate such distance of two distributions $\widehat{P}_i$ and P'_i, as shown below:

$$d_{s_i} = \sum_a \widehat{p}_{i,a} \log \frac{\widehat{p}_{i,a}}{p'_{i,a}}, \tag{10}$$

where $\widehat{p}_{i,a}, p'_{i,a}$ are the probability of the action a following the distributions $\widehat{P}_i, P'_i$, respectively.

We define the distance $d_{\widehat{TW},TW'}$ between $\widehat{TW}$ and TW' as the cumulative distance of all the action probabilities (i.e., $d_{\widehat{TW},TW'} = \sum_{i=0}^{L-1} d_{s_i}$). Thus, the distance $d_{\widehat{M},M'}$ of the watermarked model $\widehat{M}$ and the candidate model M' can be defined as the average distance of all watermarks in $\mathbb{W}$, i.e.,

$$d_{\widehat{M},M'} = \frac{1}{n} \sum_{\widehat{TW} \in \mathbb{W}} d_{\widehat{TW},TW'} = \frac{1}{n} \sum_{\widehat{TW} \in \mathbb{W}} \sum_{i=0}^{L-1} d_{s_i}. \tag{11}$$

The owner can verify the existence of watermarks by comparing $d_{\widehat{M},M'}$ with a distance threshold τ.

5 EXPERIMENTS

We evaluate the requirements satisfactory of our method. It is general for various types of DRL algorithms and tasks. Without loss of generality, we consider the following two systems.

Stochastic policy. We implement a REINFORCE [29] DRL algorithm to solve the Cart-Pole task [4]. It consists of two layers with 128 neurons. We apply dropout on the first layer with a rate of 0.6. We also adopt the Relu activation function on the first layer, and the softmax function on the last layer.

Deterministic policy We choose DQN [17] as a representative of deterministic algorithms to solve the LunarLander task [4]. We apply the double DQN network structure and both the policy network and target network have two layers of 32 neurons.

5.1 Effectiveness of Watermark Generation and Embedding

Cart-Pole. To generate a watermark candidate, we randomly search damage-free states from $\mathbb{S}$. For each state, we enquiry the APD from a clean model and identify the action a^* which has the highest action probability. Then we select an action $\widehat{a}$ different from a^*. Since the Cart-Pole environment has a very small action space, i.e., 2 actions, if we always select the opposite action on all the incoming L states, the system will be fragile and we can never obtain a sequence consisting of all damage-free states. Therefore, we perturb the actions on alternative states with a fixed interval to mitigate the impact on the tasks with small action spaces. More preciously, we choose to change the action on every two states in the Cart-Pole environment and collect the damage-free states as our watermark candidates. We fix the threshold of variance ψ at 0.15 and set the episode performance threshold ϵ to 195 following OpenAI Gym [4].

To embed watermarks into the target model, we need to add an incentive reward (10 in our experiments) over the damage-free states from $\mathbb{C}$ and update the network parameters based on the corresponding loss values. Since the network is trained over multiple episodes initialized from random seeds, to ensure the damage-free states can be included during the optimization of network parameters, we initialize the environment with a seed that contains damage-free states every 10 episodes. We add the incentive reward to every two states in the episodes. We complete the training process when the performance reaches the origin task reward threshold ϵ, and collect the watermark sequence with a length of 6 from the new model.

LunarLander. We use the deterministic DQN algorithm to solve this task. Therefore, the action probability on a state is either 1 (the selected one) or 0 (others). We first find a set of watermark candidates consisting of damage-free states following the same process in Cart-Pole.

For each watermark candidate, we modify the reward of the damage-free states towards an non-optimal action and add the training sample to the training buffer. Different from training stochastic models in Cart-Pole, DQN applies a experience replay mechanic to sample the training data randomly. To guarantee the training samples with our revised rewards can be learned well by the network, we adopt the prioritized experience replay [21] to sample the training data which have large loss with higher probability. We initialize the experience buffer with a size of 10000 and start to train the DQN model with normal process.

Table 1: Verification results of the embedded watermarks

Metric	Cart-Pole		LunarLander	
	Train	Fine-tune	Train	Fine-tune
Accuracy	96.7%	95%	100%	100%
Error rate	4.2%	5.8%	1.6%	2.5%

5.2 Verification Results

We evaluate whether the generated watermarks can be observed to identify models in this section. We use two metrics to quantify the effectiveness of the embedded watermark in a DRL model: the verification accuracy denotes the ratio of watermarked models that can be correctly detected; the error rate denotes the percentage of unprotected models that are misclassified as the watermarked one.

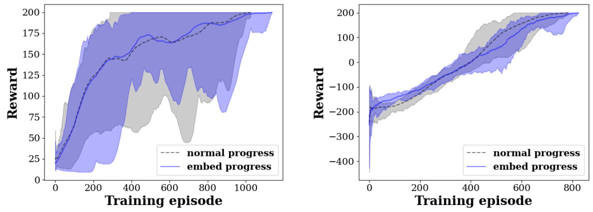
For the Cart-Pole case, we produce 20 watermarked models and 120 clean models for classification. The clean model set consists of 20 original REINFORCE models before watermarking, 20 new REINFORCE models, 40 PPO models and 40 A2C models. PPO and A2C models are trained with the default network structure based on the benchmark of OpenAI baseline [8]. To increase the diversity of the model set, we vary the hyperparameters (e.g., learning rates, training steps) to generate those 80 clean models. For the LundarLander, we produce 20 watermarked models and 80 clean models for classification. All the clean models are based on DQN with varied hyperparameters from the benchmark of OpenAI baseline [8].

During verification, we send the first watermark state to each model and collect 10000 actions to analyze the APD. If the model cannot reach the next watermark state, we stop this process and treat this model as a non-watermarked one, as its APD is very different from the expected one. Otherwise, we compute the KL divergence of the collected APD and the watermark APD. We repeat the above process and compute the average distance until we reach the end of the watermark sequence. We set the average distance threshold τ to 0.5.

Table 1 shows the verification results for these two tasks. We consider two embedding modes: (1) *Train* is to train a watermarked model from scratch; (2) *Fine-tune* means to embed watermarks to a well-trained model via fine-tuning. From this figure, we observe that our method can achieve very high accuracy for both environments (96.7% for Cart-Pole and 100% for LunarLander). Meanwhile, the error rates can be kept to very small values. This confirms the effectiveness of our approach. It is interesting to note that the verification performance of Cart-Pole is slightly worse than Lunarlander. The reason is that the stochastic models act randomly following the APD, and we can only analyze an approximate distribution during the verification under the black-box access setting. Therefore, there may exist measurement errors to decrease the accuracy.

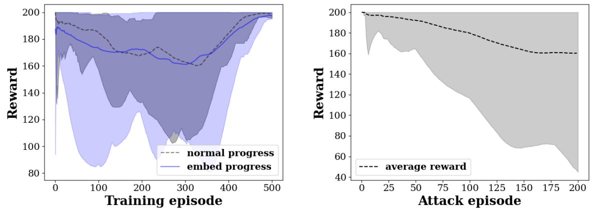
5.3 Functionality-preserving

Another requirement for the watermark scheme is *functionality-preserving*, where the added watermarks should not affect the performance and behaviors of the model on normal states. To quantify this requirement, we compare the original clean model and the watermarked model from the perspectives of the learning progress, the average and variance of episode scores.



(a) Train to Embed on Cart-Pole (b) Train to Embed on LunarLander

Figure 3: The episode rewards during the progress of training clean and watermarked models.



(a) Fine-tune to embed on Cart-Pole (b) Impact of fine-tuning on Cart-Pole

Figure 4: The episode rewards during the progress of fine-tuning watermark embedding and transformation.

We profile the training progress of 20 clean models and 20 watermarked models. Figure 3 shows the range of episode rewards when training with and without watermarks for both stochastic and deterministic models. The progress of training a watermarked model is slightly slower than training a clean model. The variance is also kept within an acceptable range considering the high stochasticity of DRL algorithms. For the fine-tuning embedding mode, the stochastic models have less stable behaviors than the deterministic ones, as they need to randomly collect the training experience with the actions sampled following the APD. As shown in Figure 4(a), the fine-tune progress of Cart-Pole has large variance but it can still acquire the knowledge to solve the task and memorize the watermarks.

Table 2 shows the average and variance of episode scores for watermarked models under two embedding modes. For each watermarked model, we measure its performance over 100 episodes initialized with random seeds. From this table, we observe the average episode score can meet the threshold (reported in the *Threshold* row) for solving each task. The small variance also indicates that the watermarked models are very stable with embedded watermarks.

5.4 Robustness

The adversary may try to transform a stolen model in order to either adapt to his datasets and scenario, or maliciously hide the evidence of plagiarism. So the watermarks should not be removed by those transformations. There are mainly two types of model transformations: fine-tuning and model compression [11, 23]. We demonstrate

Table 2: Functionality-preserving results of the watermarked models

Score	Cart-Pole		LunarLander	
	Train	Fine-tune	Train	Fine-tune
Threshold	195	195	200	200
Average	197.6	196.2	201.6	200.8
Variance	4.59	9.26	14.50	18.42

Table 3: Robustness results of the watermarked models against different transformations

Transformation	Cart-Pole		LunarLander	
	Train	Fine-tune	Train	Fine-tune
Baseline	96.7%	95%	100%	100%
Fine-tune	95%	83%	96.7%	93.3%
Compression	91.5%	89.2%	98.3%	95.8%

that our watermarks can resist against these two operations. This is not considered in prior work [3].

Fine-tuning is a common method to apply a well-trained model to a similar task with less effort. In our experiments, we fine-tune the watermarked model with the same implementation and hyperparameters. For the LunarLander case, we fine-tune the watermarked model with 100 episodes (around 10% of episodes for training a new model from scratch). For the Cart-Pole case, we fine-tune the model with 50 episodes. The reason that we select a smaller number of episodes is based on the observation that extensive fine-tuning can hurt the performance of the stochastic models as the new training experiences are sampled following the APD. This is validated by Figure 4(b), which shows the reward range with different numbers of fine-tuning episodes. To preserve the model's functionality, it is reasonable for the adversary to choose 50 episodes.

Model compression is another popular solution to reduce the model size and complexity. There are various ways to compress the model. We apply model quantization [12] to reduce the precision of parameters in the target model. In our experiments, we train DRL models with 32-bit floating point tensors, and then compress the parameters to 16-bit floating point tensors.

Table 3 shows the robustness results against these two transformations. We observe that fine-tuning transformation is slightly worse than compression, especially for the stochastic case (Cart-Pole). However, the watermarked models under all these settings can still maintain high verification accuracy to be detected.

6 DISCUSSION

Extensibility to various environments. Different from conventional DNNs, watermarking DRL models is more dependent on the environments and tasks. As such, designing a uniform scheme for all DRL tasks is very difficult. Our paper made the first attempt to address this challenge, and evaluations indicate our solution is useful for common DRL tasks. In some applications (e.g., POMDP [5]), full information about states and actions is not easy to acquire. To adapt to this case, we can just select the observable states and actions as watermarks, while abandoning the hidden ones. We can also use more watermark sequences to increase the fidelity. As future work, we will evaluate our solution for more DRL applications.

Selection of hyperparameters. There are several hyperparameters in our approach that can affect the performance of the watermark embedding and verification. Since different tasks have distinct features, the model owner needs to empirically test his models to identify the optimal thresholds. He can systematically set the thresholds following the method as we did: 1) set the variance ϵ inversely proportional to the action space size; 2) set the incentive reward η as 5-20 times of the original one; 3) set the average distance τ to be inversely proportional to the incentive reward. Following these basic rules, we can find the "sweet-spots" for a specific task by tuning these hyperparameters properly.

More watermark removal attacks. In this paper we evaluate the robustness of our watermarking scheme against fine-tuning and model quantization. There are other common model transformation techniques for neural networks, e.g., model distillation, model pruning, etc. Model distillation requires large training data and huge computational resources, which is not a practical solution for watermark removal attacks. Model pruning and model quantization with larger compression ratios (e.g., reducing to 8 bit or binary weights) can lead to an unacceptable performance penalty to the model. These can discourage the adversary from performing such model transformations. In our future work, we plan to study more efficient and sophisticated attacks to remove DRL watermarks.

7 CONCLUSION

In this paper, we explore how to protect the intellectual property, and prevent copyright infringements of DRL models. We formally define the watermarking problem and requirements for DRL. We propose a novel temporal watermarking scheme that can be applied to both deterministic and stochastic DRL policies. Instead of using spatial triggers or perturbations, or out-of-distribution states in different environments as watermark states, we design damage-free states, and utilize statistic tests of action probability distribution to verify the ownership of the target model with only black-box accesses. This strategy can effectively make the watermarked models uniquely distinguishable, while preserving their behaviors and performance for normal usage. Extensive experimental results reveal that our watermarking scheme can satisfy the functionality-preserving, state-preserving, and damage-free requirements under different environments and system settings. Our watermarks are also robust to common model modification techniques such as fine-tuning and compression.

8 ACKNOWLEDGEMENT

We thank the anonymous reviewers for their valuable comments. This work was supported in part by Singapore National Research Foundation, under its National Cybersecurity R&D Program No. NRF2018NCR-NCR005-0001, Singapore National Research Foundation under NCR No. NRF2018NCR-NSOE003-0001, NRF Investigatorship No. NRFI06-2020-0022, and Singapore Ministry of Education AcRF Tier 1 RG108/19 (S) and RS02/19.

REFERENCES

[1] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *USENIX Security Symposium*. 1615–1631.

[2] William Aiken, Hyoungshick Kim, and Simon Woo. 2020. Neural Network Laundering: Removing Black-Box Backdoor Watermarks from Deep Neural Networks. *arXiv preprint arXiv:2004.11368* (2020).

[3] Vahid Behzadan and William Hsu. 2019. Sequential triggers for watermarking of deep reinforcement learning policies. *arXiv preprint arXiv:1906.01126* (2019).

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. arXiv:arXiv:1606.01540

[5] Anthony R Cassandra. 1998. A survey of POMDP applications. In *Working notes of AAAI 1998 Fall Symposium on Planning with Partially Observable Markov Decision Processes*, Vol. 1724.

[6] Xinyun Chen, Wenxiao Wang, Chris Bender, Yiming Ding, Ruoxi Jia, Bo Li, and Dawn Song. 2019. REFIT: A Unified Watermark Removal Framework for Deep Learning Systems with Limited Data. *arXiv preprint arXiv:1911.07205* (2019).

[7] Ingemar J Cox, Joe Kilian, F Thomson Leighton, and Talal Shamoon. 1997. Secure spread spectrum watermarking for multimedia. *IEEE Transactions on Image Processing* 6, 12 (1997), 1673–1687.

[8] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. 2017. OpenAI Baselines. https://github.com/openai/baselines.

[9] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. 2017. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *IEEE International Conference on Robotics and Automation*. 3389–3396.

[10] Shangwei Guo, Tianwei Zhang, Han Qiu, Yi Zeng, Tao Xiang, and Yang Liu. 2020. The Hidden Vulnerability of Watermarking for Deep Neural Networks. *arXiv preprint arXiv:2009.08697* (2020).

[11] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. 2015. Deep learning with limited numerical precision. In *International Conference on Machine Learning*. 1737–1746.

[12] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* (2015).

[13] Panagiota Kiourti, Kacper Wardega, Susmit Jha, and Wenchao Li. 2019. TrojDRL: Trojan Attacks on Deep Reinforcement Learning Agents. arXiv:1903.06638

[14] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.

[15] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. 2019. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications* (2019), 1–12.

[16] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. 2019. How to prove your model belongs to you: A blind-watermark based framework to protect intellectual property of DNN. In *Annual Computer Security Applications Conference*. 126–137.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[18] Ryota Namba and Jun Sakuma. 2019. Robust watermarking of neural network with exponential weighting. In *ACM Asia Conference on Computer and Communications Security*. 228–240.

[19] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. 2019. Deepsigns: An end-to-end watermarking framework for protecting the ownership of deep neural networks. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems*.

[20] Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. 2017. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging* 2017, 19 (2017), 70–76.

[21] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952* (2015).

[22] Ruimin Shen, Yan Zheng, Jianye Hao, Zhaopeng Meng, Yingfeng Chen, Changjie Fan, and Yang Liu. [n.d.]. Generating Behavior-Diverse Game AIs with Evolutionary Multi-Objective Deep Reinforcement Learning. ([n. d.]).

[23] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[24] Jianwen Sun, Tianwei Zhang, Xiaofei Xie, Lei Ma, Yan Zheng, Kangjie Chen, and Yang Liu. 2020. Stealthy and efficient adversarial attacks against deep reinforcement learning. *arXiv preprint arXiv:2005.07099* (2020).

[25] Richard S Sutton. 1988. Learning to predict by the methods of temporal differences. *Machine learning* 3, 1 (1988), 9–44.

[26] Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N Asokan. 2019. Dawn: Dynamic adversarial watermarking of neural networks. *arXiv preprint arXiv:1906.00830* (2019).

[27] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. 2017. Embedding watermarks into deep neural networks. In *ACM on International Conference on Multimedia Retrieval*. 269–277.

[28] Yue Wang, Esha Sarkar, Michail Maniatakos, and Saif Eddin Jabari. 2020. Stop-and-Go: Exploring Backdoor Attacks on Deep Reinforcement Learning-based Traffic Congestion Control Systems. arXiv:2003.07859

[29] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.

[30] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. 2018. Protecting intellectual property of deep neural networks with watermarking. In *ACM Asia Conference on Computer and Communications Security*. 159–172.

[31] Yan Zheng, Xiaofei Xie, Ting Su, Lei Ma, Jianye Hao, Zhaopeng Meng, Yang Liu, Ruimin Shen, Yingfeng Chen, and Changjie Fan. 2019. Wuji: Automatic online combat game testing using evolutionary deep reinforcement learning. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 772–784.