Revealing the Risk of Hyper-parameter Leakage in Deep Reinforcement Learning Models

Linkang Du, *Member, IEEE*, Zhikun Zhang, *Member, IEEE*, Min Chen, *Member, IEEE*, Mingyang Sun, *Senior Member, IEEE*, Shouling Ji, *Senior Member, IEEE*, Peng Cheng, *Member, IEEE*, Jiming Chen, *Fellow, IEEE*, Michael Backes, *Fellow, IEEE*, and Yang Zhang, *Member, IEEE*

***Abstract*—Deep reinforcement learning (DRL) has been implemented across various critical applications, including smart grids, traffic management systems, and autonomous vehicles. To safeguard intellectual property and mitigate security vulnerabilities, access to DRL models is typically restricted to a black-box format. This means specific details like the structure of the policy network and optimization processes are not openly available to users. It is crucial to determine if the hyper-parameters can be inferred from observable states and actions within these models, presenting two primary challenges: 1) limited data available from the black-box model and 2) the intertwined effects of hyper-parameters on the model's behavior.**

Since DRL models exhibit varying behaviors in identical tasks depending on their hyper-parameter configurations, we introduce a novel hyper-parameter inference attack against DRL, named HyperInfer, which allows adversaries to deduce the settings of a black-box DRL model. In order to fully assess the risk of model hyper-parameter leakage, we design two novel state generation methods that provoke divergent responses from DRL models. We also develop an inference framework to elucidate the relationship between model behavior and hyper-parameter settings.

Through comprehensive experiments involving multiple DRL models and environments, we demonstrate that model behaviors can indeed reveal hyper-parameter settings, with inference accuracy surpassing 90% in scenarios such as PPO with CartPole. We also discuss key findings relevant to practical applications and explore how knowledge of hyper-parameters can facilitate more sophisticated attacks. Lastly, we propose potential defensive strategies to minimize the risk of hyper-parameter leakage in DRL models.

***Index Terms*—Deep reinforcement learning model, Hyper-parameter, Inference attack**

I. Introduction

DEEP reinforcement learning (DRL), integrating with the powerful fitting ability of deep neural networks (DNNs), is a promising machine learning paradigm. A famous example, AlphaGo Zero [64], achieves the human-champion-defeating ability by playing games against itself without human data in three days. DRL has also been successfully applied to many complex decision-making tasks, such as autopilot [13], robot control [1], financial transactions [11], esports [64], [7], *etc.*

DRL model generally serves as a black box due to the fact that disclosing hidden details may render the model more susceptible to attacks from adversaries, such as model extraction attacks [5], [9], [58], [90], adversarial policy attacks [28], [6], [39], [35], and backdoor attacks [85], [32], [75], [71]. On the other hand, the architecture and parameters of a DRL model are protected in legal as intellectual property (IP) [84], [72]. Thus, it is crucial to investigate the amount of information gained from black-box access to a DRL model.

Previous studies [5], [9], [58], [90] mainly target the stealing of functionality from DRL models, which imitates the behavior of the target model instead of inferring the model's internal information. Oh *et al.* [50] and Wang *et al.* [69] have investigated the hyper-parameter inference attack against supervised deep learning models. However, those techniques cannot be trivially applied to the DRL scenario due to the following challenges. First, in the supervised learning scenarios, the adversary can use the training datasets as the high-quality query input. However, the DRL adversary has no off-the-shelf query inputs, since the DRL models typically learn from the interactions with the environment. Second, supervised learning models often have explicit labeling information for an input, while the DRL models may have multiple equivalent actions for a state. For example, if the DRL agent finds two paths of the same length in a maze game, then the different actions made at the first bifurcation are equivalent. Finally, the hyper-parameters have coupling impacts on the states and actions of a DRL model, *i.e.*, when we try to infer a type of hyper-parameter, the other hyper-parameters are the disturbance factors.

Previous studies [50], [69] have investigated the hyper-parameter inference attack against supervised deep learning models. However, those techniques cannot be trivially applied to the DRL scenario due to the following challenges. First, in the supervised learning scenarios, the adversary can use the training datasets as the high-quality query input. However, the DRL adversary has no off-the-shelf query inputs, since the DRL models typically learn from the interactions with the environment. Second, supervised learning models often have explicit labeling information for an input, while the DRL models may have multiple equivalent actions for a state. For example, if the DRL agent finds two paths of the same length in a maze game, then the different actions made at the first bifurcation are equivalent. Finally, the hyper-parameters have

Linkang Du is with Xi'an Jiaotong University, Xi'an 710049, China. E-mail: linkangd@xjtu.edu.cn. Min Chen is with Vrije Universiteit Amsterdam, Boelelaan 1105, Amsterdam. E-mail: m.chen2@vu.nl. Mingyang Sun is with Peking University, Beijing 100084, China. E-mail: smy@pku.edu.cn. Zhikun Zhang, Peng Cheng, Shouling Ji, and Jiming Chen are with Zhejiang University, Hangzhou 310027, China. E-mail: {zhikun, lunerheart, sji, cjm}@zju.edu.cn. Michael Backes, and Yang Zhang are with the CISPA Helmholtz Center for Information Security, Saarland Informatics Campus, 66123 Saarbrücken, Germany. E-mail: {director, zhang}@cispa.de

coupling impacts on the states and actions of a DRL model, *i.e.*, when we try to infer a type of hyper-parameter, the other hyper-parameters are the disturbance factors.

Our Contributions. To our knowledge, this is the first work to investigate the risk of hyper-parameter leakage in deep reinforcement learning models. The key insight is that the *disagreement actions*, *i.e.*, the DRL models' inconsistent behaviors at some states will expose their inner information. Based on this observation, we first build a series of states to collect the actions of a DRL model, and then leverage a classifier to reveal the architectures and optimization processes of the victim black-box DRL model based on the states and predicted actions. Since the disagreement actions contain more knowledge of hyper-parameter settings, the inference attack will become more effective with a high-quality state sequence, *i.e.*, stimulating more disagreement actions.

Concretely, we propose two state-generation mechanisms to construct the state sequences. The *passive* state generation method builds the state sequences based on passively recording the interaction between the DRL model and environment, which solves the first challenge mentioned above. However, directly obtaining states from the environment only weakly stimulates DRL models to generate disagreement actions. In order to make agents with the same hyper-parameter settings behave similarly, and agents with different hyper-parameter settings behave differently, we further propose an *active* state generation method. Specifically, besides training the classifier to predict hyper-parameter settings, the active method back-propagates the partial loss of the classifier model and optimizes the states. The optimized states can induce the DRL models with different hyper-parameter settings to generate more disagreement actions. We empirically find that the continuously adjusted states slow down the over-fitting of the classifier and further improve the inference accuracy.

Our experimental results show that the inference accuracy of the hyper-parameter activation function exceeds 60% across multiple DRL models and environments, even more than 90% on the PPO agents. We further evaluate four representative influential factors that affect the inference performance, *i.e.*, the victim DRL model's reward criterion, the hyper-parameter coupling, the length of state-action sequence, and the amount of shadow DRL models. Based on the experimental results, we summarise three important observations concerning the hyper-parameter leakage of the DRL model. 1) The DRL models with higher reward criteria suffer higher hyper-parameter leakage risk, which warns DRL practitioners since hyper-parameter privacy becomes more vulnerable with the DRL performance improving. 2) The hyper-parameter has various influence strengths on the model action preference. For example, the adversary can easily discriminate the activation functions or the optimizers from the DRL model's behavior, while difficult to distinguish the learning rates. 3) The active inference attack method obtains outperforming results with less state generation and DRL model training costs, which dedicates HyperInfer as a highly efficient hyper-parameter inference mechanism.

We take the adversarial attack as a case study and show the transferability improvement of adversarial examples against black-box DRL models with or without the hyper-parameter information, to demonstrate the necessity of exploring the risk of leakage of DRL model hyper-parameters. Finally, we evaluate a potential defense as reducing the information accessible to the adversary by only publishing the predicted action instead of confidence values (posteriors). However, the inference accuracy is still 20% higher than the baseline with the active inference attack methods in some settings. We expect this study to raise practitioners' awareness about the privacy threats of DRL models.

In summary, our contributions are three-fold:

- We propose a hyper-parameter inference framework against DRL models, HyperInfer, which indicates the model-level privacy concerns of the current DRL models.
- We demonstrate the effectiveness of HyperInfer on five DRL models and four environments. We also systematically analyze various experimental factors and conclude several important observations about the hyper-parameter leakage in practice.
- We show that the hyper-parameter settings can significantly improve the threat of the existing attacks, which calls additional attention to protect the hyper-parameter.

II. Background

A. Reinforcement Learning Problem

The reinforcement learning (RL) model aims to learn an optimal (or nearly-optimal) policy to maximize the "reward function" or other user-provided reinforcement signals during the immediate rewards accumulation process. RL is similar to the processes that occur in animal psychology. For example, suppose a buzzer or metronome rings before the food (reward). A dog tends to associate the sound with food and salivates upon the sound stimulus alone [55].

Typically, the reinforcement learning problem is modeled as a Markov Decision Process (MDP) problem, which consists of four components:

- A state space $\mathbb{S}$ contains all possible states of the model, where $s_t \in \mathbb{S}$ is the model state at time step t.
- An action space $\mathbb{A}$ contains all possible actions of the model, where $a_t \in \mathbb{A}$ is the model action at time step t.
- An environment transition dynamics $\mathcal{T}$ is defined as a probability mapping from state-action pairs to new states $\mathcal{T} : (\mathbb{S} \times \mathbb{A}) \times \mathbb{S} \to [0, 1]$.
- A reward function $\mathcal{R} : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ outputs the expected reward $r_t \in \mathbb{R}$ if the model takes action a_t at state s_t.

The goal of a RL algorithm is to learn a policy π that maps state-action pairs to a probability distribution $\pi : \mathbb{S} \times \mathbb{A} \to [0, 1]$, to maximize the cumulative reward.

$$\pi^* = \arg\max_{\pi} \sum_{t=t_0}^{T} \sum_{a_t \in \mathbb{A}} \gamma^{t-t_0} \mathcal{R}\left(s_t, a_t\right) \pi\left(s_t, a_t\right),$$

where the discount factor γ is applied to discount future rewards in the accumulated reward. T is the terminal time step of one game round, and t_0 is the initial time.

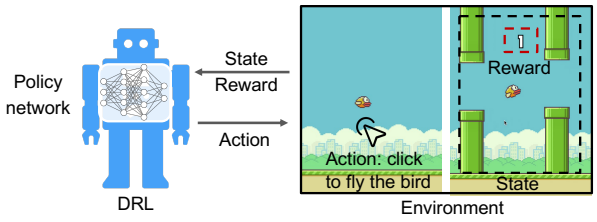


Fig. 1. An example of the reinforcement learning problem.

Example. Figure 1 gives a running example, where the DRL model plays with the flappy bird game.[1] The state includes the positions of the bird and the obstacles. The DRL model selects whether to click or not to click (action) to let the bird fly up or down. The reward shows the total score obtained by the DRL model. The DRL model obtains the state and reward from the environment and then selects an action from the motion space at each time step. Then, the environment moves to a new state and updates the reward. Based on the collected state, action, and reward, the DRL model trains a policy (neural networks) to maximize the total expected reward, *e.g.*, the total score at the end of the game.

B. Deep Reinforcement Learning Model

The goal of RL is to find an optimal behavior strategy for the model to obtain optimal rewards. From the perspective of optimization strategy, the existing RL models can be categorized into three classes [3], [74].

Value-based RL models. Q-learning [78] is a primary value-based RL model, which finds the optimal policy by maximizing the expected value of the total reward over successive steps. Q-learning has a function that calculates the quality of a state–action combination: $\mathcal{Q} : \mathbb{S} \times \mathbb{A} \rightarrow \mathbb{R}$. For the basic Q-learning, $\mathcal{Q}$ is defined as a tabular with random values, *i.e.*, *Q-table*, before learning begins. At each time step t, the model selects an action a_t, receives a reward r_t, enters a new state s_{t+1} (that may depend on both the previous state s_t and the selected action a_t), and then updates the values in $\mathcal{Q}$. The core of the algorithm is as following.

$$\mathcal{Q}^{\text{new}} \leftarrow \mathcal{Q} + \alpha \cdot \left(r_t + \gamma \max_a \mathcal{Q}\left(s_{t+1}, a\right) - \mathcal{Q}\left(s_t, a_t\right) \right), \quad (1)$$

where $\alpha \in (0, 1]$ is the learning rate and the $\mathcal{Q}^{\text{new}}$ changes more rapidly with a larger α. Q-learning uses the weighted average between the old $\mathcal{Q}$ values and the new information to conduct a value iteration update as Equation 1.

However, when the state and action space becomes enormous, the basic Q-leaning can hardly maintain and store an extensive Q-table. To overcome this problem, Deep Q-Learning Network (DQN) [48] integrates DNNs to approximate the values in the Q-table. Based on the superior capabilities of neural networks in learning high-dimensional feature representations and function approximation properties, DQN is used to play Go [64] and Atari games at expert level [48].

Policy-based RL models. Unlike approximating the Q-table, the policy-based RL model directly optimizes the action prediction. For example, REINFORCE [79] updates a neural network-based policy by leveraging gradient estimation. Recent policy-based RL models tend to use DNN (*e.g.*, multilayer perceptron [82] or recurrent neural networks [87]) to model the policy π, called policy network. The objective function of the policy network is defined as the expectation of the total discounted rewards in a game.

$$J(\theta) = \mathbb{E}_{(s_t, a_t) \sim \pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]. \quad (2)$$

According to [34], the policy gradient can be written as

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(s, a) v_t \right],$$

where v_t is an unbiased sample of $Q_{\pi_\theta}(s, a)$ denoting the long-term action value of the MDP problem, and $\pi\left(s, a\right)$ is the policy network.

Actor-Critic RL models. The actor-critic RL model [47] uses two components: actor and critic, to separately characterize the policy and the action-value function.

The actor decides which action should be taken, and the critic scores the actor's action. The actor's learning is based on the policy gradient approach. The critic evaluates the actor's action by computing the action-value function. Since the action-value function can assist the policy update, an actor-critic RL model can reduce gradient variance and accelerate the convergence process.

Therefore, the actor-critic RL model has two objective functions separately for actor and critic networks. For the actor, the objective function has a similar form to Equation 2, while using $\mathcal{A}_{\pi_\theta}(s, a)$ instead of $Q_{\pi_\theta}(s, a)$.

$$\mathcal{A}_{\pi_\theta}(s, a) = Q_{\pi_\theta}(s, a) - V_{\pi_\theta}(s), \quad (3)$$

Where $V_{\pi_\theta}(s)$ represents the average value of state s. When $\mathcal{A}_{\pi_\theta}(s, a) = 0$, there will be no extra reward gained in tuning the actions' probabilities, and then the gradients of the actor will be set to zero. The objective function of the critic is to minimize the error between the estimated and the true action-value functions. To accelerate the training process of the actor-critic RL models [61], recent research integrates a new objective function called "clipped surrogate objective function" into the actor-critic algorithm, *i.e.*, proximal policy optimization (PPO) [62], [23], which performs comparably or better than state-of-the-art approaches while being much simpler to implement and tune. Due to the ease of use, OpenAI has set PPO as its default DRL algorithm.[2]

In the evaluation, we choose DQN [48], PG [79], and PPO [62], [23] as they are fundamental DRL methods exemplifying value-based, policy-based, and actor-critic approaches, respectively. Additionally, we incorporate Trust Region Policy Optimization (TRPO) [22] and Soft Actor Critic (SAC) [60] to represent advanced DRL models.

[1] https://flappybird.io

[2] https://openai.com/blog/openai-baselines-ppo/

C. Hyper-parameters in DRL Models

Since the DRL model integrates a DNN model as the policy network, the hyper-parameters in DRL can be categorized into three parts, *i.e.*, the network architecture hyper-parameters of the DNN model, the optimization process hyper-parameters of the DNN model, and the hyper-parameters of the RL model.

In this work, we mainly focus on six DRL model hyper-parameters (Table I), which contain the hyper-parameters in the previous DNN hyper-parameter inference attack [50] and the discount factor γ of an RL model. The selected hyper-parameters are relevant to nearly all deep reinforcement learning (DRL) models. Other hyper-parameters are not considered in this paper, as they may not be part of some DRL models. For example, convolutional layers are typically applied to vision-related tasks, while the learning rate schedule is an optional aspect of DRL model training. We should note that the core idea of HyperInfer, which involves deducing the model's internal parameters from its input and output and optimizing the input to gain further internal details, can be expanded to a vast array of hyper-parameters.

III. Problem Statement

A. Motivation

Previous work [2] illustrated that two DRL models with different hyper-parameter settings have different behavior preferences on the same task. For example, in the autonomous driving environment for the merging task [14], a safety-oriented DRL model prefers to slow down and merge behind the non-autonomous vehicle, decreasing the chances of a collision. In contrast, an aggressive DRL model prefers to speed up and merge in front of the non-autonomous car. Therefore, the DRL models with different architectures and optimization processes have separable action tendencies, exposing their internal information. In this work, we plan to answer the following two questions.

Q1. Can an adversary infer the hyper-parameter settings from the behavior of the DRL model?
Q2. What kind of state is more beneficial to infer the hyper-parameter settings?

B. Threat Model

Attacker's goal. The adversary aims to infer the hyper-parameter settings of a DRL model, such as the architecture (*e.g.*, the type of the non-linear activation function) and optimization process (*e.g.*, the type of the optimizer). The adversary aims to achieve high accuracy in inferring the hyper-parameter settings from the victim DRL model.

Attacker's background knowledge and capability. We consider an adversary who has black-box access to the victim DRL model. Note that this is the most general and challenging scenario for the adversary. A typical application scenario is that a service provider trains a DRL model and then provides an API to the customers. A malicious customer wants to infer the hyper-parameter settings from the victim DRL model either for stealing intellectual property, or using them as a stepping stone to launch attacks on other customers' models.

When the victim DRL model interacts with the environment, the adversary can observe its states and actions. In addition, since the victim DRL model is applied in the adversary's environment, the adversary can adjust the environment state within the valid boundary. The attacker can calculate the average cumulative reward to estimate the victim's reward criterion (performance) from the model's interactions with the environment in practice. For example, if the average cumulative reward of a DRL model is 300 after a number of episodes, the model's reward criterion equals 300. The victim's framework can be inferred by [8].

Definition 1. *(DRL hyper-parameter inference attacks.) By the black-box access to the victim model, the DRL hyper-parameter inference attack aims to infer the hyper-parameter settings of the victim DRL model based on the state-action records.*

IV. Hyper-parameter Inference Attack

A. Workflow of HyperInfer

In Figure 2, we take a flappy bird DRL model as an example to illustrate the workflow of the DRL hyper-parameter inference attack. We divide the workflow of HyperInfer into four steps and describe the steps in detail.

Step 1: Shadow Model Generation (SMG). In the left dashed box of Figure 2, the adversary uses the flappy bird environment to produce a set of shadow DRL models, which have different hyper-parameter combinations. For example, to infer the activation function of the victim DRL model, the attacker trains the shadow DRL models with different kinds of activation functions.

Step 2: State-action Sequence Collection (SSC). In the second dashed box, the shadow DRL models observe the states of the environment and take actions. At each step, the adversary records the state s_t and the action a_t^n of each shadow DRL model, where a_t^n represents the n-th DRL model action at the t-th step. Based on the disagreement of models' actions, the attacker can aggregate the hyper-parameter information from the state-action sequences. Intuitively, the adversary will find more divergence of the model behavior with a longer length of the collected state-action sequence. Thus, we also evaluate the inference efficacy with different lengths of state-action sequences in Appendix D.

Step3: Attack Model Training (AMT). After the above two steps, the adversary obtains a set of state-action sequences and the corresponding hyper-parameter settings. For all state-action sequences, the adversary concatenates state and action at each step into a $(|s| + |a|)$-dimensional feature vector. The feature vectors are stitched along the time dimension to form a $T \times (|s| + |a|)$-dimensional feature x, and the adversary acquires a feature set X with a hyper-parameter setting label set Y. Then, a supervised classifier can be used to characterize the relationship between feature X and label Y. The training objective is

$$\min_{\theta} \mathop{\mathbb{E}}_{x \sim \mathcal{X}} \left[\mathcal{L} \left(f_\theta^p \left([x]_{i=1}^n \right), y^p \right) \right], \tag{4}$$

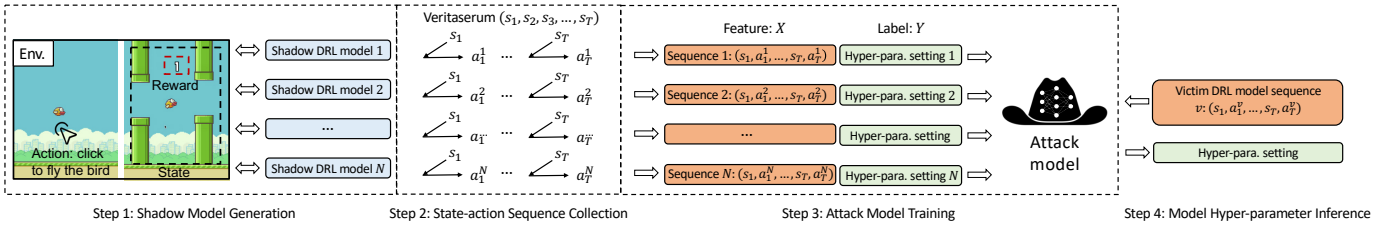


Fig. 2. Workflow of HyperInfer. The four steps in the HyperInfer algorithm, *i.e.*, shadow model generation, state-action sequence collection, attack model training, and model hyper-parameter inference. HyperInfer infers the policy network information from the state and action of the victim DRL model.

Algorithm 1 State Perturbation

Input: Environment E, state s
Output: State s'

// Obtain the bound for each dimension of E
low_bound, high_bound = E.spaces.Box()
// Limit the scope of the perturbation
low_bound, high_bound = $\frac{\text{low\_bound}}{10}$, $\frac{\text{high\_bound}}{10}$
for i-th dimention in E.spaces **do**
  σ = min(|low_bound|, |high_bound|)
  while True **do**
    // Check the new state in the boundary
    $s'[i] = s[i] + N\left(0, \sigma^2\right)$
    if $s'[i]$ in E.spaces.Box(dim=i) **then**:
      break
    end if
  end while
end for
Return s' =0

where $x \in X$, $y^p \in Y$, f_θ^p represents the attack model, and $\mathcal{L}$ is the cross-entropy loss.

Step 4: Model Hyper-parameter Inference (MHI). In this step, the attacker constructs the state s_t and observes the victim DRL model action a_t^v. After the same pre-processing of the states and the actions as Step 3, the attacker puts the state-action feature sequence into the attack model and then obtains the prediction.

From Step 2 and Step 3, the inference accuracy may vary with different state sequences with the same set of shadow DRL models. For example, if the bird's position is lower than the obstacles, all shadow models tend to behave the same, where the attack model cannot obtain any knowledge from the state-action sequence. In contrast, the inference attack will be more effective if the attacker constructs states that stimulate the DRL models to produce disagreement actions.

Gajcin *et al.* [14] defined the *preference-based disagreement* in DRL scenarios, which can serve as a criterion to determine whether a state causes disagreement actions between the DRL models. However, the definition is limited in practice. Firstly, the definition is about judging instead of generating a state to cause disagreement actions. In addition, the definition is applied to two DRL model scenes, while there are more shadow DRL models in inference attacks. Thus, we design two mechanisms to build the states in Section IV-B.

B. Attack State Generation

In the attack procedure, the sequence of states plays an important role in exposing the hyper-parameter settings of the victim model. Below, we elaborate on the generation and optimization of the states in detail.

Passive state generation mechanism. To obtain the state sequence, an intuitive way is to collect it directly from the interactions between the shadow DRL models and the environment. Algorithm 2 shows the detailed state generation. The attacker randomly chooses a DRL model with policy π_θ, and then initializes the environment with an arbitrary s_0 in the state space. For example, the bird located at the center is a valid state in the flappy bird game. The DRL model puts s_0 into the policy network π_θ, and then obtains the action a_1. The environment transfers to the next state s_1 with a symbol to determine whether the game is over. The attacker inserts s_1 into state sequence S. If the $done\_signal$ is $True$, the s_t is the termination state of the environment and the attacker checks whether the time step t equals to T. If $t < T$, the attacker generates a new initial state s_0' by Algorithm 1 and re-initializes the environment. Then, the attacker continues the state collection until the time step reaches T. If there are some physical constraints between dimensions, we only need to perturb some of the dimensions using Algorithm 1, and then calculate the rest of the dimensions by the constraints.

The passive method is an efficient strategy to generate states. However, it has two drawbacks in practice: 1) the white-box shadow models are not fully utilized, *i.e.*, not using the gradient information of the white-box models; 2) the attack model is easy to over-fitting on the shadow models due to the static state sequence.

Active state generation mechanism. To overcome the shortcomings in the passive method, the attacker leverages the gradients of the shadow DRL models to generate a more effective state sequence, *i.e.*, active state generation method. Since the shadow DRL models are white-box, the attacker can access both their hyper-parameter settings and the gradient information in the policy networks. Compared to Algorithm 2 with a static state sequence, Algorithm 3 updates the state sequence in the training process of the attack model, *i.e.*, dynamic state sequence. We optimize the states to reduce the classification loss when training the attack model to strengthen the relationship between state-action sequences and hyper-parameters. We tailor the objective under state optimization constraints as

$$\min_{S} \mathop{\mathbb{E}}_{M \sim \mathcal{M}} \left[\mathcal{L}\left(f_\theta^p\left((S, M(S)) \right), y^p \right) \right], \tag{5}$$

Algorithm 2 Passive State Generation Mechanism

Input: Shadow DRL models M, environment E, the max time step T
Output: State sequence $S = \{s_1, s_2, ..., s_T\}$

Randomly select a DRL model with policy π_θ
Let $S = \{\}$
Initialize E randomly with s_0
done_signal = False
for time step t from 1 to T **do**
  if done_signal == True **then**:
    s'_0 = Algorithm 1 (E, s_0)
    Re-initialize E randomly with s'_0
    // Get the action from the DRL model
    $a_t = \arg\max_{a \in \mathbb{A}} \pi_\theta(s'_0, a)$
  else
    $a_t = \arg\max_{a \in \mathbb{A}} \pi_\theta(s_{t-1}, a)$
  end if
  // Get the new state s_t and the game over signal
  s_t, done_signal $= E(a_t)$
  Insert the s_t into S
end for
Return S =0

Algorithm 3 Active State Generation Mechanism

Input: Shadow DRL models M, environment E, time step T, hyper-parameter setting label set Y , attack model f^p_θ
Output: State sequence $S = \{s_1, s_2, ..., s_T\}$

1: // Obtain the initial state sequence
2: S = Algorithm 2 (M, E, T)
3: state_update = False
4: **for** epoch i from 1 to n **do**
5: **if** i % m == 0 and state_update == False **then**:
6: f^p_θ.loss = f^p_θ.loss_calculation(X, Y)
7: // Update S value
8: Back-propagate f^p_θ.loss to optimize S
9: state_update = True
10: **end if**
11: **if** i == 1 or state_update == Ture **then**
12: // Collect action sequence from M
13: X = []
14: **for** j-th shadow model in M **do**
15: $\{a_1, a_2, ..., a_T\}_j$ = $M[j](S)$
16: X.append(concatenate($\{a_1, a_2, ..., a_T\}_j$, S))
17: Convert X to tensor
18: **end for**
19: state_update = False
20: **end if**
21: // Update attack model weights
22: f^p_θ.train(X, Y)
23: **end for**
24: Return S =0

where S is the state sequence, $\mathcal{M}$ is the distribution of the DRL shadow models, y^p is the ground truth label of the hyper-prameter p, and $\mathcal{L}$ is the cross-entropy loss. As shown in Line 8 of Algorithm 3, the loss of the attack model f^p_θ is back-propagated to optimize state sequence S every m step in the training process. The attacker rebuilds a new feature set X based on the updated state sequence S. Then, the attacker trains the classifier with X and Y until the next state update. The attacker uses the latest state sequence to collect the actions from the victim DRL model.

C. Discussion

HyperInfer shares the spirit of the reverse-engineering strategy in [50], *i.e.*, a general approach for hyper-parameter extraction attacks against the image classifier models, which also 1) infers the internal information of a DNN model from a sequence of queries, and 2) generates more effective input to enhance the reversing accuracy against the black-box models. However, HyperInfer differs from the reverse-engineering strategy in several essential aspects. It is worth noting that these differences are mainly due to the fact that these two methods work for different kinds of victim models. That is, [50] is for the image classifier models with the DNN framework, while HyperInfer works for the sequential decision-making DRL models.

- **Lack of Off-the-shelf Datasets.** Unlike the DNN scenarios, where Oh *et al.* [50] can use training samples as the high-quality query input, the DRL models learn from the environment's interactions. Thus, HyperInfer has no off-the-shelf query input before the attack. We need to construct the query input based on the trained shadow models.
- **Sequential Query Input.** The reverse-engineering strategy [50] crafts a single query input to extract information from the DNN models, *i.e.*, it usually processes one image sample at each step. The DRL models are designed to optimize cumulative rewards, the internal information contained in each step is limited (see Section II); thus, HyperInfer needs to build a sequence of inputs to infer the hyper-parameters.
- **New Hyper-parameter.** Besides the hyper-parameters considered in [50], HyperInfer takes the special hyper-parameter of DRLs, *i.e.*, discount factor γ, into account.
- **Non-uniqueness of the Policy Behavior.** In classification tasks, each instance receives one accurate label. On the other hand, RL problems often involve numerous strategies, thereby complicating inference attack.

V. Evaluation

In this section, we utilize HyperInfer to reveal the hyper-parameter leakage risk of DRL models. We adopt five DRL models and four environments, which are widely used in previous DRL studies [71], [8], [52], [16], [81]. Concretely, we aim to answer the following questions.

Q1. Can an adversary infer the hyper-parameter settings from the behavior of the DRL model?
Q2. What kind of state is more beneficial for inferring the hyper-parameter settings?

Q3. How do the other experimental factors, *i.e.*, the victim DRL model's reward criterion, the hyper-parameter coupling, the length of state-action sequence, and the number of shadow DRL models, influence the attack?
Q4. How to mitigate the risk of hyper-parameter leakage?

A. Experimental Setup

Environments. We choose the "CartPole", "Acrobot", "Ant", and "HalfCheetah" environments in the Gym[3]. We believe the environments are representative: 1) Gym is a mainstream platform that provides a unified interface for training and testing the DRL models. 2) The selected environments are commonly used in academia for evaluating the effectiveness of attacks against DRL models [8], [30], [54].

Targeted DRL model. We adopted the DQN [48], PG [79], PPO [62], [23], TRPO [22], and SAC [60] models during the evaluation. The DQN, PG, and PPO models represent three basic ideas of the RL models, *i.e.*, value-based strategy, policy-based strategy, and actor-critic strategy. Most of the existing model-free RL methods have evolved from these three approaches. We select TRPO and SAC to represent the advanced DRL strategies.

Targeted hyper-parameters. Table I shows the values of hyper-parameters. The first three hyper-parameters are about the architecture of the DRL policy network, and the others are related to the optimization process. For each hyper-parameter, we choose three different values, which are usually the default selection in open-sourced frameworks [12], [24], *e.g.*, ReLU and the Adam optimizer.

Shadow model contruction. For each combination, we collect 20 shadow DRL models for each hyper-parameter setting (totally 20×3^6), and then we divide the shadow models into a training set (16×3^6) and a test set (4×3^6), which is the default setup for the subsequent experimental steps. Since it is difficult to train a DRL model satisfying the reward criterion under some hyper-parameter values [47], the distributions of hyper-parameter values are possibly not uniform among the shadow DRL models. The distributions of the hyper-parameters are shown in Appendix B.

Evaluation metric. In our experiment, we follow the metric commonly used for evaluating inference attacks, measuring the accuracy of prediction [52], [8], [50]. More specifically, we utilize the shadow models in the training set to generate state-action sequences and train the attack model. Then, we take the shadow models in the test set as the victim model and infer the hyper-parameter settings based on their state-action sequences.

Methods. We compare the inference performance between the passive and the active state generation methods. More specifically, "passive_env" (Algorithm 2) uses the states directly from the interaction between the DRL model and the environment. "active_env" optimizes the states from "passive_env" by leveraging the state optimization in Algorithm 3. For both methods, the default sequence size is 200 in the experiments. The baseline method, *i.e.*, "majority_vote", adopts the largest proportions as the prediction for the victim DRL model.

TABLE I
THE HYPER-PARAMETER SETTINGS.

Category	Hyper-parameter	Values
Network Architecture	**Activation Function**	ReLU, ELU, Tanh
	Hidden Layers	2, 3, 4
	Neuron Number	64, 128, 256
Optimization Process	**Optimizer**	SGD, Adam, RMSprop
	Learning Rate	1e-3, 5e-4, 1e-4
RL model	**Gamma**	0.9, 0.95, 0.99

Experimental settings. We adopt the ResNet framework as our attack model and illustrate its architecture in Appendix A. We train the attack model for 400 epochs. For the passive method, we use the SGD optimizer to train the attack model with a learning rate of 0.01. We leverage an additional Adam optimizer with a learning rate of 0.01 to tune the states every 10 epochs during the attack model training.

B. Overall Performance

We show the effectiveness of HyperInfer across six combinations of the basic DRL models and Gym environments.

Setup. We use the reward criteria in OpenAI Gym leaderboard[4] for the victim DRL models, which are 500 for Cartpole and -100 for Acrobot, respectively. In Figure 3, we vary the types of hyper-parameter on the X-axis and show the corresponding inference accuracy on the Y-axis. Each histogram in the plots shows the average inference accuracy of 3 repeated experiments with the standard deviation.

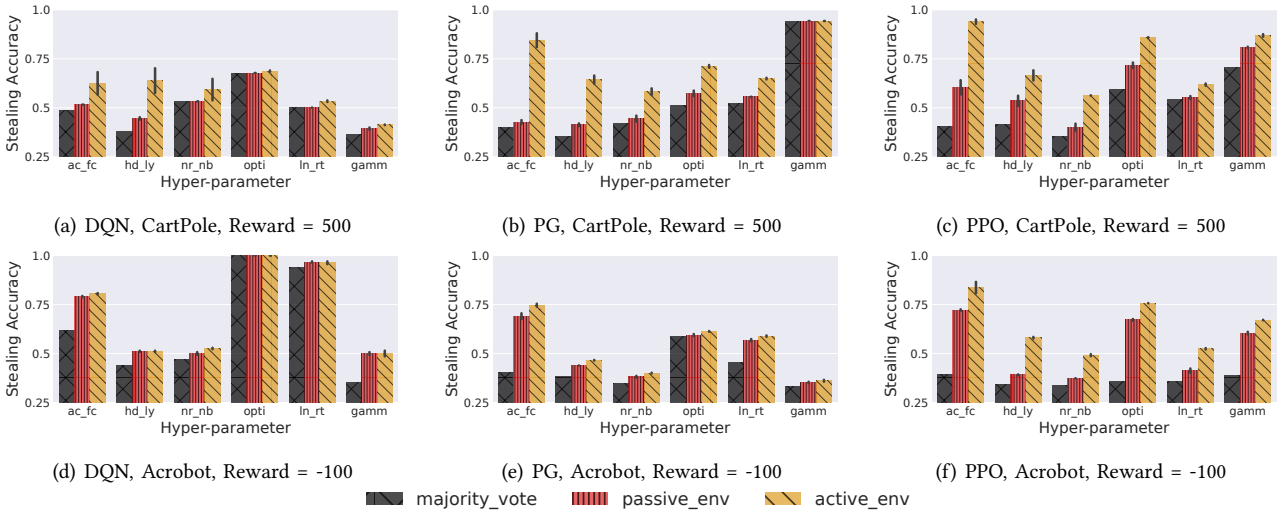
Observations. We have the following observations from Figure 3. 1) The inference accuracy of HyperInfer is all higher than the baseline method, which means that the behaviors of DRL models can indeed expose the hyper-parameter setting information. For example, the inference accuracy for the activation function is beyond 60% across the experiment settings, even exceeding 90% on PPO.

2) HyperInfer obtains different inference accuracy over various DRL models. The adversary's inference effectiveness on the PG and PPO models is better than that of the DQN models for the hyper-parameters optimizer and learning rate. For the PG and the PPO model, HyperInfer achieves a huge accuracy boost on the optimizer and the learning rate, while HyperInfer slightly surpasses the baseline method for the DQN model. Recalling Section II-B, the DQN models use the policy network to approximate the Q-table in the training process and choose the action according to the formed Q-table, *i.e.*, indirectly act. PG and PPO use the policy network to directly build the relation between states and actions without the Q-value approximation in DQN, *i.e.*, directly act. Thus, it is more difficult to infer the optimization-related hyper-parameters from DQN.

3) The inference accuracy of HyperInfer varies over the hyper-parameters. In Figure 3, HyperInfer usually obtains

[3] https://www.gymlibrary.ml/

[4] https://github.com/openai/gym/wiki/Leaderboard



(a) DQN, CartPole, Reward = 500 (b) PG, CartPole, Reward = 500 (c) PPO, CartPole, Reward = 500

(d) DQN, Acrobot, Reward = -100 (e) PG, Acrobot, Reward = -100 (f) PPO, Acrobot, Reward = -100

Fig. 3. The inference accuracy of different methods on six combinations of the DRL models and the environments. The x labels of each plot correspond in turn to the hyper-parameters activation function, hidden layers, neuron number, optimizer, learning rate, and gamma. The y labels show the inference accuracy. The reward value of each plot indicates the reward criterion of the victim DRL model.

higher inference accuracy on the hyper-parameter activation function compared to the hyper-parameter learning rate. As a structural hyper-parameter, the different activation functions have unique features. For example, the Tanh activation function is symmetric about $(0, 0)$, while the ReLU and ELU activation functions are asymmetrical. In addition, the functional features of activation do not change once the DRL model is built. Therefore, the activation function of the DRL model is at a higher risk of leakage. The hyper-parameter learning rate controls the update speed of the models' weights. Compared to the activation function, the learning rate guides the model's actions by affecting the model's weight values. Thus, the features of the learning rate become blurred as the training process.

C. Scalability on More Complex Cases

In this section, we conduct the inference attack against two advanced DRL models (SAC and TRPO) on two more complex tasks (HalfCheetah and Ant). Figure 4 demonstrates HyperInfer can still perform well in these cases. HyperInfer infer the hyper-parameter settings based on the inconsistent actions. Since DRL policy networks are often sub-optimal in practice, it is possible to infer hyper-parameters.

HyperInfer can alleviate the cost of training shadow models by reducing the number of shadow models required for each hyper-parameter setting. Based on the ablation study in Appendix D, the active method can still achieve competitive inference accuracy with only 4 shadow models for each hyper-parameter setting. In future work, the attacker can collect more sequences for a shadow model or use a stronger classification model to reduce the attack cost further.

D. Possible Defense

To mitigate the risk of hyperparameter inference and protect the configuration of DRL models from auditing attacks, we investigate two practical defense strategies tailored to different action space settings. For discrete action environments, we restrict the output information by publishing only the predicted actions without associated confidence scores. For continuous action tasks, we adopt local differential privacy mechanisms to perturb the released actions, aiming to obscure the underlying training configuration.

Releasing Only Actions. We reduce the information accessible to the adversary by releasing only the predicted action instead of confidence values (posteriors), and empirically evaluate its effectiveness. For discrete action tasks (*i.e.*, CartPole and Acrobat), directly publishing predicted actions without confidence scores is a strong protection method, considering the perturbed confidence scores usually do not affect the choice of predicted actions in practice.

Figure 5 depicts the impact of the defense approach with or without. The inference accuracy of HyperInfer decreases (remaining superior to the baseline method) when only recording the predicted action, *e.g.*, the inference accuracy is still 15% higher than the baseline with the active inference attack methods on the optimizer inference of PPO. Following Appendix D, the active method can distill more information about the hyper-parameter setting from the limited predictions, which makes it more resistant to the defense approach.

Differential Privacy. For the continuous action tasks, *i.e.*, Ant and HalfCheetah, we perturb the DRL model's actions under different privacy (DP) guarantees to protect the hyper-parameters of the DRL model. More specifically, the Ant task has an 8-dimensional continuous action space, while HalfCheetah has 6 dimensions. Each component is bounded within $[-1, 1]$, the global ℓ_2-sensitivity is $\sqrt{8} \approx 2.828$ for Ant and $\sqrt{6} \approx 2.449$ for HalfCheetah. Under the Laplace mechanism, the noise scale b is computed as $b = \Delta/\varepsilon$, yielding 2.828 and 2.449, respectively, when $\epsilon = 1$. For the Gaussian mechanism with privacy parameter $(\varepsilon = 1, \delta = 10^{-5})$, the standard deviation σ must satisfy $\sigma \geq \frac{\sqrt{2\ln(1.25/\delta)} \cdot \Delta}{\varepsilon}$. This results in $\sigma \approx 13.71$ for Ant and $\sigma \approx 11.87$ for HalfCheetah.

Table II and Table III demonstrate that although local DP perturbations effectively preserve the privacy of DRL hyper-parameters by reducing the inference accuracy of the

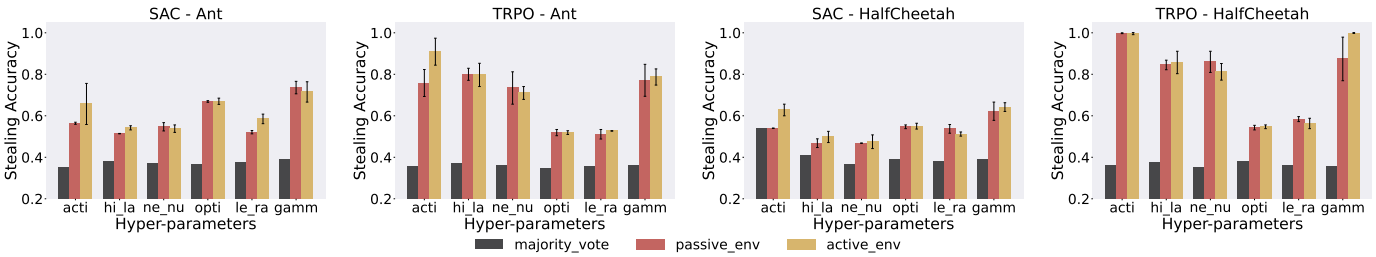


Fig. 4. The inference accuracy of different methods on four combinations of the DRL models (SAC and TRPO) and the environments (Ant and HalfCheetah). The x labels of each plot correspond in turn to the hyper-parameters activation function, hidden layers, neuron number, optimizer, learning rate, and gamma. The y labels show the inference accuracy. The reward value of each plot indicates the reward criterion of the victim DRL model.

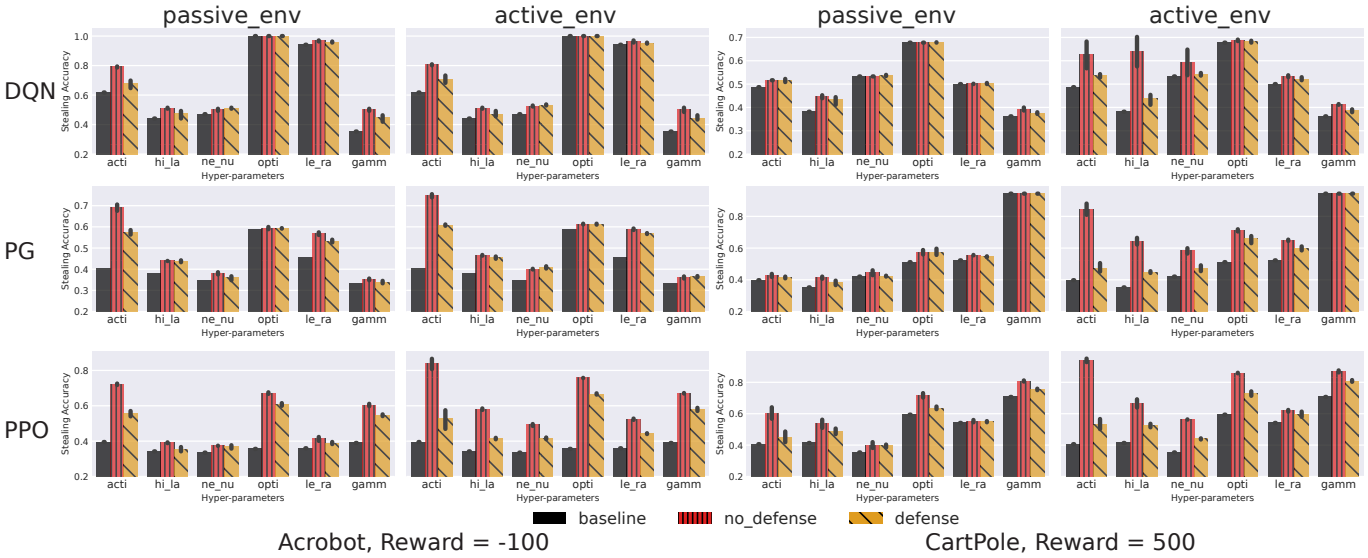


Fig. 5. The inference accuracy of different methods against the defense approach. In each subplot, the "baseline" is the "majority_vote" method, and "defense" means the adversary only obtains the predicted action without the confidence score.

auditing methods, they simultaneously induce noticeable performance degradation. Specifically, as presented in Table II, the added noise causes a significant drop in the inference accuracy across both passive and active environments. For instance, in the Ant environment under SAC, the inference accuracy decreases by over 0.14 when using Gaussian noise and by over 0.13 with Laplacian noise, compared to the non-private setting. From Table III, SAC and TRPO agents suffer extreme performance drops in the HalfCheetah task, over 1000 points under Gaussian noise and nearly 1200 under Laplacian noise. The injected noise severely hinders the agent's ability to learn and act optimally. While DP perturbation helps defend against hyperparameter leakage, it also introduces notable overhead in terms of policy performance.

E. Takeaways

Through the extensive experiments, we have obtained the following answers to the starting questions of Section V:

A1. The inference accuracy of HyperInfer is higher than the baseline method in all the evaluated scenarios, which shows that the behaviors of the DRL models can indeed expose the hyper-parameter settings (Section V-B).

A2. The states optimized by the active inference attack method further utilize the gradients of the shadow models, which obtain better inference accuracy (Section V).

A3. The DRL models with higher reward criterion suffer more privacy risk against HyperInfer (Appendix E). The hyper-parameter has various influence strength to the action preference (Appendix F). HyperInfer benefits from a larger length of state-action sequence or the number of shadow models (Appendix D).

A4. The inference accuracy of HyperInfer decreases (remaining superior to the baseline method) when only recording the predicted action (Section V-D).

VI. Case Study

From Section V, we illustrate that HyperInfer can violate the DRL model's intellectual property (IP), *i.e.*, inferring the hyper-parameter settings by the black-box access. In this section, we demonstrate how the knowledge of hyperparameter can enhance existing attacks. In practice, the disclosed information will make the model more vulnerable to attacks from adversaries, such as adversarial attacks [20], [37], [45], model extraction attacks [5], [9], [58], [90], and adversarial policy attacks [28], [6], [39], [35].

Adversarial Attacks. We randomly select 20 victim models with different hyper-parameter settings and generate

TABLE II
THE INFERENCE ACCURACY OF HyperInfer AGAINST THE DP PERTURBATION. THE "MAJORITY" IS THE "MAJORITY VOTE" METHOD. THE "PASSIVE_ENV (DP GAUSSIAN)" AND "PASSIVE_ENV (DP LAPLACIAN)" COLUMNS SHOW THE CHANGES IN THE INFERENCE ACCURACY UNDER DP PERTURBATION, COMPARED WITH THOSE IN THE "PASSIVE_ENV" COLUMN. THE "ACTIVE_ENV (DP GAUSSIAN)" AND "ACTIVE_ENV (DP LAPLACIAN)" COLUMNS SHOW THE CHANGES IN THE INFERENCE ACCURACY UNDER DP PERTURBATION, COMPARED WITH THOSE IN THE "ACTIVE_ENV" COLUMN.

Task	Model	HyPara	majority	passive_env	passive_env (DP Gaussian)	passive_env (DP Laplacian)	active_env	active_env (DP Gaussian)	active_env (DP Laplacian)
Ant	SAC	acti	0.351	$0.564_{\pm 0.005}$	$0.385_{\pm 0.023}$	$0.407_{\pm 0.027}$	$0.657_{\pm 0.099}$	$0.405_{\pm 0.014}$	$0.430_{\pm 0.015}$
		hi_la	0.381	$0.514_{\pm 0.001}$	$0.402_{\pm 0.020}$	$0.412_{\pm 0.021}$	$0.542_{\pm 0.010}$	$0.415_{\pm 0.022}$	$0.429_{\pm 0.019}$
		ne_nu	0.371	$0.547_{\pm 0.020}$	$0.386_{\pm 0.025}$	$0.424_{\pm 0.037}$	$0.538_{\pm 0.018}$	$0.396_{\pm 0.032}$	$0.458_{\pm 0.021}$
		opti	0.366	$0.669_{\pm 0.004}$	$0.420_{\pm 0.052}$	$0.465_{\pm 0.065}$	$0.670_{\pm 0.015}$	$0.468_{\pm 0.030}$	$0.530_{\pm 0.005}$
		le_ra	0.376	$0.521_{\pm 0.008}$	$0.393_{\pm 0.017}$	$0.419_{\pm 0.034}$	$0.585_{\pm 0.023}$	$0.402_{\pm 0.020}$	$0.453_{\pm 0.010}$
		gamm	0.391	$0.736_{\pm 0.030}$	$0.398_{\pm 0.007}$	$0.513_{\pm 0.044}$	$0.715_{\pm 0.049}$	$0.404_{\pm 0.006}$	$0.551_{\pm 0.029}$
	TRPO	acti	0.358	$0.758_{\pm 0.065}$	$0.473_{\pm 0.129}$	$0.606_{\pm 0.137}$	$0.909_{\pm 0.065}$	$0.575_{\pm 0.111}$	$0.738_{\pm 0.032}$
		hi_la	0.372	$0.800_{\pm 0.029}$	$0.407_{\pm 0.047}$	$0.535_{\pm 0.077}$	$0.797_{\pm 0.056}$	$0.430_{\pm 0.057}$	$0.611_{\pm 0.009}$
		ne_nu	0.362	$0.734_{\pm 0.078}$	$0.402_{\pm 0.057}$	$0.515_{\pm 0.086}$	$0.710_{\pm 0.031}$	$0.432_{\pm 0.068}$	$0.600_{\pm 0.014}$
		opti	0.349	$0.519_{\pm 0.015}$	$0.381_{\pm 0.017}$	$0.420_{\pm 0.047}$	$0.519_{\pm 0.009}$	$0.396_{\pm 0.009}$	$0.464_{\pm 0.021}$
		le_ra	0.358	$0.511_{\pm 0.023}$	$0.379_{\pm 0.013}$	$0.422_{\pm 0.050}$	$0.527_{\pm 0.001}$	$0.390_{\pm 0.007}$	$0.472_{\pm 0.009}$
		gamm	0.362	$0.771_{\pm 0.077}$	$0.403_{\pm 0.040}$	$0.568_{\pm 0.140}$	$0.787_{\pm 0.039}$	$0.424_{\pm 0.048}$	$0.705_{\pm 0.020}$
Half.	SAC	acti	0.54	$0.540_{\pm 0.001}$	$0.540_{\pm 0.000}$	$0.540_{\pm 0.001}$	$0.628_{\pm 0.028}$	$0.540_{\pm 0.001}$	$0.540_{\pm 0.000}$
		hi_la	0.41	$0.468_{\pm 0.021}$	$0.410_{\pm 0.000}$	$0.410_{\pm 0.000}$	$0.498_{\pm 0.027}$	$0.410_{\pm 0.000}$	$0.410_{\pm 0.000}$
		ne_nu	0.365	$0.468_{\pm 0.001}$	$0.370_{\pm 0.004}$	$0.372_{\pm 0.008}$	$0.475_{\pm 0.033}$	$0.371_{\pm 0.005}$	$0.365_{\pm 0.001}$
		opti	0.39	$0.547_{\pm 0.009}$	$0.396_{\pm 0.004}$	$0.403_{\pm 0.010}$	$0.550_{\pm 0.014}$	$0.397_{\pm 0.006}$	$0.394_{\pm 0.004}$
		le_ra	0.38	$0.537_{\pm 0.021}$	$0.390_{\pm 0.006}$	$0.393_{\pm 0.007}$	$0.512_{\pm 0.010}$	$0.389_{\pm 0.006}$	$0.389_{\pm 0.002}$
		gamm	0.39	$0.622_{\pm 0.044}$	$0.398_{\pm 0.006}$	$0.445_{\pm 0.056}$	$0.642_{\pm 0.021}$	$0.396_{\pm 0.006}$	$0.391_{\pm 0.001}$
	TRPO	acti	0.36	$0.998_{\pm 0.002}$	$0.559_{\pm 0.124}$	$0.747_{\pm 0.079}$	$0.996_{\pm 0.005}$	$0.668_{\pm 0.085}$	$0.798_{\pm 0.028}$
		hi_la	0.375	$0.845_{\pm 0.023}$	$0.430_{\pm 0.089}$	$0.598_{\pm 0.059}$	$0.857_{\pm 0.054}$	$0.479_{\pm 0.105}$	$0.635_{\pm 0.057}$
		ne_nu	0.35	$0.860_{\pm 0.051}$	$0.506_{\pm 0.095}$	$0.608_{\pm 0.090}$	$0.812_{\pm 0.040}$	$0.583_{\pm 0.042}$	$0.676_{\pm 0.020}$
		opti	0.38	$0.543_{\pm 0.011}$	$0.384_{\pm 0.005}$	$0.445_{\pm 0.031}$	$0.547_{\pm 0.009}$	$0.382_{\pm 0.004}$	$0.467_{\pm 0.021}$
		le_ra	0.36	$0.584_{\pm 0.012}$	$0.399_{\pm 0.025}$	$0.441_{\pm 0.015}$	$0.563_{\pm 0.025}$	$0.423_{\pm 0.008}$	$0.452_{\pm 0.008}$
		gamm	0.355	$0.874_{\pm 0.105}$	$0.627_{\pm 0.135}$	$0.668_{\pm 0.080}$	$0.999_{\pm 0.002}$	$0.759_{\pm 0.027}$	$0.724_{\pm 0.054}$

TABLE III
THE PERFORMANCE DECAY OF THE DRL AGENTS UNDER DP PERTURBATION. THE "PERFORMANCE" COLUMN SHOWS THE ORIGINAL PERFORMANCE OF THE DRL AGENTS. THE "PERFORMANCE (DP GAUSSIAN)" AND "PERFORMANCE (DP LAPLACIAN)" COLUMNS SHOW THE DRL AGENTS' PERFORMANCE UNDER DP.

Task	Model	Performance	Performance (DP Gaussian)	Performance (DP Laplacian)
Ant	SAC	$813.0_{\pm 71.6}$	$-13.6_{\pm 3.3}$	$-12.0_{\pm 3.3}$
	TRPO	$629.0_{\pm 136.1}$	$-14.9_{\pm 3.5}$	$-14.0_{\pm 3.9}$
HalfCheetah	SAC	$557.8_{\pm 55.3}$	$-525.9_{\pm 50.7}$	$-381.3_{\pm 50.8}$
	TRPO	$677.5_{\pm 77.0}$	$-520.2_{\pm 44.4}$	$-386.5_{\pm 31.0}$

20×8 shadow models with or without knowing the hyper-parameters separately. For each white-box shadow model, we generate a 500-step adversarial state, and apply the adversarial state to the victim model and record the attack success rate, *i.e.*, transferability value. We repeat the transferability experiment 10 times and obtain the average values for each shadow model. We count the percentage of shadow agents in all 20×8 shadow models with average transferability values more than 90%. Since GAIL [25] adopts the actor-critic framework as the generator, we use the PPO agents as the victim models to eliminate the influence of the DRL framework. We show the results from four adversarial example generation methods, *i.e.*, FGM, FGSM [20], L2PGD and LinfPGD [37], [45]. These methods are widely adopted and canonical in evaluating the robustness of deep reinforcement learning agents under adversarial perturbations. Prior work such as Huang *et al.* [27] and Pattanaik *et al.* [54] used similar settings to evaluate DRL robustness.

Table IV shows the percentage of shadow agents with high average transferability values. We set five different ϵ values to control the strength of an adversarial attack. The 'W' and 'W/O' illustrate the transferability experiment with or without the hyper-parameters information. From the results, we have the following observations. 1) The inference attack significantly improves in downstream attacks. When the shadow models use the same hyper-parameter settings as the victim models, their decision boundaries tend to be closed during the training process. Thus, the adversarial transferability has more than 15% boost in most cases. 2) Comparing the results in Table IV, the transferability enhancement becomes more obvious for the DRL agents with a higher reward criterion, which is consistent with the analysis in Appendix E. We speculate that the agents with higher reward criteria have more convergent decision boundaries, and hyper-parameters have more influence on the formation of their decision boundaries. 3) The transferability values are saturated as the epsilon increases. The environments interacting with the DRL agents have their physical constraints. For example, the state of the "CartPole" environment is defined by four physical variables, *i.e.*, cart position, cart velocity, pole angle, and pole angular velocity, ranging from $-[2.4, \infty, 12 \text{ deg}, \infty]$ to $+[2.4, \infty, 12 \text{ deg}, \infty]$. When one value of the state exceeds the value range, the generation of adversarial states should stop to satisfy the physical constraints. Therefore, the transferability values cannot always increase with a larger ϵ.

Model Extraction Attack. Several prior studies [5], [88], [9], [58], [90] have proposed model extraction attacks targeting DRL models. In our evaluation, we select Stealthy Imita-

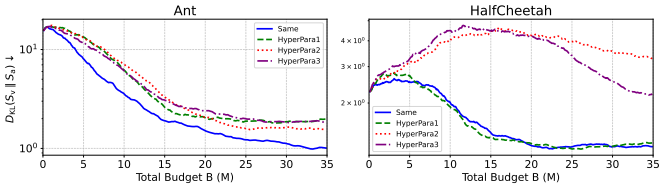


Fig. 6. Model extraction attacks against the DRL model. The Kullback-Leibler (KL) divergence measures the discrepancy between the adversarial model and the victim model [**lower is better**]. "Same" means the adversarial model shares the hyperparameter settings of the victim model. "HyperParaX" means the adversarial model has different degrees of hyperparameter deviation with the victim model.

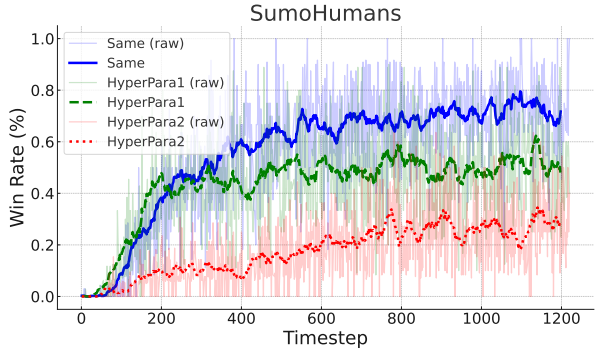


Fig. 7. Adversarial policy Attack against the DRL model. The Y-axis shows the win rate while training the adversary model against the victim model in the SumoHumans environment [**higher is better**]. "Same" means the adversarial model shares the hyperparameter settings of the victim model. "HyperParaX" means the adversarial model has different degrees of hyperparameter deviation from the victim model. The "HyperParaX" plots represent smoothed demonstrations of the "HyperParaX (raw)" data.

tion by Zhuang *et al.* [90], which represents the state-of-the-art in policy stealing for DRL. We conduct experiments on both the Ant and HalfCheetah tasks following the methodology of [90]. We use the Kullback–Leibler (KL) divergence to quantify the discrepancy between the adversarial model and the victim model. As shown in Figure 6, the KL divergence tends to decrease more rapidly and reach lower values when the adversarial model shares the same hyper-parameters as the victim model.

In the "HyperPara1" setting, the adversarial model adopts a different discount factor, changing γ from 0.99 to 0.9. Building upon this, "HyperPara2" introduces an additional architectural deviation by reducing the number of hidden neurons from 256 to 64. "HyperPara3" represents the most divergent setting, where γ is set to 0.9, the hidden layer size is reduced to 64 neurons, and the learning rate is decreased from 0.001 to 0.0001. The results demonstrate that greater hyperparameter discrepancies lead to larger divergences between the adversarial and victim models.

Adversarial Policy Attack. We adopt the adversarial policy attack proposed by Gleave *et al.* [17], which is a canonical method in this line of work and has inspired several follow-up studies [21], [42], [44]. Our experiments are conducted on the SumoHumans task, where we follow the evaluation protocol of [17], reporting win rates as the adversarial policy is trained against the victim model.

TABLE IV
THE PERCENTAGE OF SHADOW AGENTS WITH AVERAGE TRANSFERABILITY VALUES MORE THAN 90%. THE RESULTS ARE COLLECTED FROM PPO AND THE CARTPOLE ENVIRONMENT.

Reward=300								
ϵ	FGM		L2PGD		FGSM		LinfPGD	
	w.	w/o	w.	w/o	w.	w/o	w.	w/o
1e-4	20.62	15.62	13.12	13.75	24.38	16.88	19.38	15.00
1e-3	51.25	43.75	50.62	43.75	51.88	44.38	50.62	44.38
5e-3	51.88	44.38	51.25	44.38	51.88	44.38	51.25	44.38
1e-2	51.88	44.38	51.25	44.38	51.88	44.38	51.25	44.38
1e-1	51.88	44.38	46.88	41.88	51.88	44.38	50.00	42.50
Reward=500								
ϵ	FGM		L2PGD		FGSM		LinfPGD	
	w.	w/o	w.	w/o	w.	w/o	w.	w/o
1e-4	10.00	9.38	5.62	5.00	11.88	11.25	9.38	8.75
1e-3	58.13	37.50	56.25	36.88	60.00	38.12	58.75	38.12
5e-3	61.25	39.38	61.25	40.00	61.25	39.38	61.25	39.38
1e-2	61.25	39.38	60.62	39.38	61.25	40.00	60.62	39.38
1e-1	61.25	40.00	58.75	37.50	61.25	40.00	58.75	37.50

As shown in Figure 7, the adversarial agent achieves higher and faster win rates when its hyper-parameters match the victim's. In the "HyperPara1" setting, the adversarial model's learning rate is increased from 3×10^{-4} to 3×10^{-3}. "HyperPara2" increases the learning rate to 3×10^{-2}, representing a more aggressive deviation. The experimental results suggest that increasing the hyperparameter mismatch weakens the effectiveness of the adversarial policy attack.

VII. Related Work

Privacy Attacks Against DRL. Privacy threats in DRL can be broadly classified into model privacy, reward privacy, and environment privacy attacks [49], [38].

Model privacy attacks aim to recover internal policies or training settings. A primary class of such attacks is model extraction. Behzadan *et al.* [5] showed that adversaries can replicate policy behavior via imitation. Chen *et al.* [9] further investigated the feasibility and potential gains of stealing DRL models. Later works enhanced stealth and practicality. Sajid *et al.* [58] analyzed extraction in controller-based systems, while Zhuang *et al.* [90] introduced an environment-free, reward-guided method. Notably, HyperInfer complements this line by being the first to infer DRL hyperparameters as a novel form of model privacy breach. *Reward privacy attacks* reconstruct the agent's reward function via inverse reinforcement learning (IRL), risking disclosure of sensitive design intents [70], [10]. For instance, Prakash *et al.* [56] proposed PRIL to quantify the vulnerability of differentially private RL models to such inference. *Environment privacy attacks* infer hidden environment information from the model's behavior. Pan *et al.* [53] highlighted this threat in robot navigation. Prakash *et al.* [56] applied membership inference to determine if particular data points were used in training. Gomrokchi *et al.* [18] further showed that observation leakage is exacerbated under partial observability.

Other Attacks Against DRL. Beyond privacy, DRL systems are vulnerable to adversarial attacks, adversarial policies, and backdoor attacks. *Adversarial attacks* perturb inputs to

mislead DRL agents. Huang *et al.* [27] introduced uniform attacks that apply perturbations at every timestep. Lin *et al.* [40] improved efficiency via strategically-timed perturbations. Sun *et al.* [66] addressed temporal dependencies by proposing critical-point attacks that degrade performance more effectively. *Adversarial policies* exploit opponent weaknesses in competitive RL. Gleave *et al.* [17] showed agents can exhibit unnatural yet effective behaviors to defeat stronger opponents. Guo *et al.* [21] introduced non-zero-sum attacks that simultaneously boost attacker rewards and harm victims. Even with partial observability or limited control, adversarial agents remain a threat [44], [42]. *Backdoor attacks* inject triggers during training to induce malicious behaviors at test time. TrojDRL [33] overlays visual triggers and manipulates rewards to plant target actions. Ashcraft *et al.* [4] improved stealth by using in-distribution triggers. Yu *et al.* [86] proposed temporal-pattern backdoors tailored to partially observable settings.

Privacy Attacks and Defenses in Other DNNs. Privacy attacks in traditional DNNs fall into four categories [57]: (1) *Membership inference* detects whether a sample was in the training set [63], [41], [26], [73]; (2) *Reconstruction attacks* aim to recover training samples or features [19], [76], [59], [89]; (3) *Property inference* extracts latent dataset-level attributes [15], [65], [46], [29]; (4) *Model extraction* builds surrogate models to replicate decision boundaries [67], [36], [51], [31]. To mitigate these privacy risks, various defenses have been developed. SIGuard [73] adds noise to encrypted outputs in MPC inference to resist membership attacks, while Leia [43] protects inputs and model parameters via lightweight cryptography for edge devices. Emerging domains have also adopted privacy-preserving techniques. GraphGuard [80] enables data-free membership detection and unlearning in GNNs. OblivGNN [83] secures inference and updates via secret sharing. For LLMs, Self-Guard [77] empowers models to detect unsafe outputs during inference, and Uzor *et al.* [68] proposed a local gatekeeper to filter sensitive inputs before cloud transmission.

VIII. Conclusion

In this work, we propose a novel hyper-parameter inference method against the DRL model by leveraging the state-action records from the victim model. HyperInfer achieves good inference performance with varying DRL model and environment combinations. Through the experimental evaluation, an attacker can infer the hyper-parameter settings by collecting the behavior of the DRL model. By studying various parameter settings, we conclude several important observations for the DRL practitioners. The relationship between the performance and the privacy risk of the DRL models reminds the DRL practitioners to consider privacy leakage risk when pursuing high performance. Finally, we compare the effectiveness of the existing ML attacks against black-box DRL models with or without the hyper-parameter information to demonstrate the negative impact of hyper-parameter leakage.

References

[1] S. Amarjyoti. Deep Reinforcement Learning for Robotic Manipulation - The state of the art. *CoRR abs/1701.08878*, 2017. 1

[2] Y. Amitai and O. Amir. I Don't Think So: Disagreement-Based Policy Summaries for Comparing Agents. *CoRR abs/2102.03064*, 2021. 4

[3] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017. 3

[4] C. Ashcraft and K. Karra. Poisoning Deep Reinforcement Learning Agents with In-distribution Triggers. *arXiv preprint arXiv:2106.07798*, 2021. 12

[5] V. Behzadan and A. M. Hsu. Adversarial Exploitation of Policy Imitation. *arXiv Preprint arXiv:1906.01121*, 2019. 1, 9, 10, 11

[6] V. Behzadan and A. Munir. Vulnerability of Deep Reinforcement Learning to Policy Induction Attacks. In *MLDM*, 2017. 1, 9

[7] C. Berner, G. Brockman, et al. Dota 2 with Large Scale Deep Reinforcement Learning. *CoRR abs/1912.06680*, 2019. 1

[8] K. Chen, S. Guo, et al. Stealing Deep Reinforcement Learning Models for Fun and Profit. In *ACM ASIACCS*. ACM, 2021. 4, 6, 7

[9] K. Chen, S. Guo, T. Zhang, X. Xie, and Y. Liu. Stealing Deep Reinforcement Learning Models for Fun and Profit. In *ACM ASIACCS*, pages 307–319. ACM, 2021. 1, 9, 10, 11

[10] S. R. Chirra, P. Varakantham, and P. Paruchuri. Preserving the privacy of reward functions in mdps through deception. *arXiv preprint arXiv:2407.09809*, 2024. 11

[11] Y. Deng, F. Bao, et al. Deep Direct Reinforcement Learning for Financial Signal Representation and Trading. *IEEE TNNLS*, 2017. 1

[12] P. Dhariwal, C. Hesse, et al. OpenAI Baselines. https://github.com/openai/baselines, 2017. 7

[13] A. R. Fayjie, S. Hossain, et al. Driverless Car: Autonomous Driving Using Deep Reinforcement Learning in Urban Environment. In *UR*, 2018. 1

[14] J. Gajcin, R. Nair, T. Pedapati, R. Marinescu, E. Daly, and I. Dusparic. Contrastive Explanations for Comparing Preferences of Reinforcement Learning Agents. *CoRR abs/2112.09462*, 2021. 4, 5

[15] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *ACM CCS*, pages 619–633, 2018. 12

[16] A. Gleave, M. Dennis, et al. Adversarial Policies: Attacking Deep Reinforcement Learning. In *ICLR*, 2020. 6

[17] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell. Adversarial Policies: Attacking Deep Reinforcement Learning. In *ICLR*, 2020. 11, 12

[18] M. Gomrokchi, S. Amin, H. Aboutalebi, A. Wong, and D. Precup. Membership Inference Attacks Against Temporally Correlated Data in Deep Reinforcement Learning. *IEEE Access*, 11:42796–42808, 2023. 11

[19] N. Z. Gong and B. Liu. You are who you know and how you behave: Attribute inference attacks via users' social friends and behaviors. In *USENIX Security*, pages 979–995, 2016. 12

[20] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. In *ICLR*, 2015. 9, 10

[21] W. Guo, X. Wu, S. Huang, and X. Xing. Adversarial Policy Learning in Two-player Competitive Games. In *ICML*, pages 3910–3919. PMLR, 2021. 11, 12

[22] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, pages 1861–1870. Pmlr, 2018. 3, 7

[23] N. Heess, D. TB, et al. Emergence of Locomotion Behaviours in Rich Environments. *CoRR abs/1707.02286*, 2017. 3, 7

[24] A. Hill, A. Raffin, et al. Stable Baselines. https://github.com/hill-a/stable-baselines, 2018. 7

[25] J. Ho and S. Ermon. Generative Adversarial Imitation Learning. In *NeurIPS*, 2016. 10

[26] Y. Hu, Z. Li, Z. Liu, Y. Zhang, Z. Qin, K. Ren, and C. Chen. Membership Inference Attacks Against Vision-language Models. *arXiv preprint arXiv:2501.18624*, 2025. 12

[27] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017. 10, 12

[28] S. H. Huang, N. Papernot, et al. Adversarial Attacks on Neural Network Policies. In *ICLR*, 2017. 1, 9

[29] Y. Huang, Z. Zhang, Q. Zhao, X. Yuan, and C. Chen. Themis: Towards practical intellectual property protection for post-deployment on-device deep learning models. In *USENIX Security*, 2025. 12

[30] I. Ilahi, M. Usama, et al. Challenges and Countermeasures for Adversarial Attacks on Deep Reinforcement Learning. *CoRR abs/2001.09684*, 2020. 7

[31] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot. High accuracy and high fidelity extraction of neural networks. In *USENIX Security*, pages 1345–1362, 2020. 12

[32] P. Kiourti, K. Wardega, et al. TrojDRL: Trojan Attacks on Deep Reinforcement Learning Agents. In *ACM/IEEE DAC*, 2020. 1

[33] P. Kiourti, K. Wardega, S. Jha, and W. Li. TrojDRL: Evaluation of Backdoor Attacks on Deep Reinforcement Learning. In *ACM/IEEE DAC*, pages 1–6. IEEE, 2020. 12

[34] V. R. Konda and J. N. Tsitsiklis. Actor-Critic Algorithms. In *NeurIPS*, 1999. 3

[35] J. Kos and D. Song. Delving into Adversarial Attacks on Deep Policies. In *ICLR*, 2017. 1, 9

[36] K. Krishna, G. S. Tomar, A. P. Parikh, N. Papernot, and M. Iyyer. Thieves on sesame street! model extraction of bert-based apis. *arXiv preprint arXiv:1910.12366*, 2019. 12

[37] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial Examples in the Physical World. *CoRR abs/1607.02533*, 2016. 9, 10

[38] Y. Lei, D. Ye, S. Shen, Y. Sui, T. Zhu, and W. Zhou. New Challenges in Reinforcement Learning: A Survey of Security and Privacy. *Artificial Intelligence Review*, 56(7):7195–7236, 2023. 11

[39] Y. Lin, Z. Hong, et al. Tactics of Adversarial Attack on Deep Reinforcement Learning Agents. In *ICLR*, 2017. 1, 9

[40] 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 *IJCAI*, pages 3756–3762, 2017. 12

[41] L. Liu, Y. Wang, G. Liu, K. Peng, and C. Wang. Membership inference attacks against machine learning models via prediction sensitivity. *IEEE TDSC*, 20(3):2341–2347, 2022. 12

[42] X. Liu, S. Chakraborty, Y. Sun, and F. Huang. Rethinking Adversarial Policies: A Generalized Attack Formulation and Provable Defense in RL. *arXiv preprint arXiv:2305.17342*, 2023. 11, 12

[43] X. Liu, B. Wu, X. Yuan, and X. Yi. Leia: A lightweight cryptographic neural network inference system at the edge. *IEEE TIFS*, 17:237–252, 2021. 12

[44] O. Ma, Y. Pu, L. Du, Y. Dai, R. Wang, X. Liu, Y. Wu, and S. Ji. SUB-PLAY: Adversarial Policies against Partially Observed Multi-Agent Reinforcement Learning Systems. In *ACM CCS*, pages 645–659, 2024. 11, 12

[45] A. Madry, A. Makelov, et al. Towards Deep Learning Models Resistant to Adversarial Attacks. In *ICLR*, 2018. 9, 10

[46] S. Mahloujifar, E. Ghosh, and M. Chase. Property inference from poisoning. In *IEEE SP*, pages 1120–1137. IEEE, 2022. 12

[47] V. Mnih, A. P. Badia, et al. Asynchronous Methods for Deep Reinforcement Learning. In *ICML*, 2016. 3, 7, 15, 17

[48] V. Mnih, K. Kavukcuoglu, et al. Playing Atari with Deep Reinforcement Learning. *CoRR abs/1312.5602*, 2013. 3, 7

[49] K. Mo, P. Ye, X. Ren, S. Wang, W. Li, and J. Li. Security and Privacy Issues in Deep Reinforcement Learning: Threats and Countermeasures. *ACM Computing Surveys*, 56(6):1–39, 2024. 11

[50] S. J. Oh, M. Augustin, et al. Towards Reverse-Engineering Black-Box Neural Networks. In *ICLR*, 2018. 1, 4, 6, 7

[51] T. Orekondy, B. Schiele, and M. Fritz. Knockoff nets: Stealing functionality of black-box models. In *CVPR*, pages 4954–4963, 2019. 12

[52] X. Pan, W. Wang, et al. How You Act Tells a Lot: Privacy-Leaking Attack on Deep Reinforcement Learning. In *AAMAS*, 2019. 6, 7

[53] X. Pan, W. Wang, X. Zhang, B. Li, J. Yi, and D. Song. How You Act Tells a Lot: Privacy-Leaking Attack on Deep Reinforcement Learning. In *AAMAS*, pages 368–376, 2019. 11

[54] A. Pattanaik, Z. Tang, et al. Robust Deep Reinforcement Learning with Adversarial Attacks. In *AAMAS*, 2018. 7, 10

[55] I. P. Pavlov and W. H. Gantt. *Conditioned Reflexes and Psychiatry*, volume 2. International publishers New York, 1941. 2

[56] K. Prakash, F. Husain, P. Paruchuri, and S. Gujar. How private is your rl policy? an inverse rl based analysis framework. In *AAAI*, volume 36, pages 8009–8016, 2022. 11

[57] M. Rigaki and S. Garcia. A Survey of Privacy Attacks in Machine Learning. *ACM Computing Surveys*, 56(4):1–34, 2023. 12

[58] M. Sajid, Y. Shen, and Y. Shoukry. Model extraction attacks against reinforcement learning based controllers. In *IEEE CDC*, pages 813–820. IEEE, 2023. 1, 9, 10, 11

[59] A. Salem, A. Bhattacharya, M. Backes, M. Fritz, and Y. Zhang. {Updates-Leak}: Data set inference and reconstruction attacks in online learning. In *USENIX Security*, pages 1291–1308, 2020. 12

[60] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *ICML*, pages 1889–1897. PMLR, 2015. 3, 7

[61] J. Schulman, S. Levine, et al. Trust Region Policy Optimization. In *ICML*, 2015. 3

[62] J. Schulman, F. Wolski, et al. Proximal Policy Optimization Algorithms. *CoRR abs/1707.06347*, 2017. 3, 7

[63] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *IEEE SP*, pages 3–18. IEEE, 2017. 12

[64] D. Silver, J. Schrittwieser, et al. Mastering the Game of Go without Human Knowledge. *Nature*, 2017. 1, 3

[65] C. Song and A. Raghunathan. Information leakage in embedding models. In *ACM CCS*, pages 377–390, 2020. 12

[66] J. Sun, T. Zhang, X. Xie, L. Ma, Y. Zheng, K. Chen, and Y. Liu. Stealthy and Efficient Adversarial Attacks against Deep Reinforcement Learning. In *AAAI*, volume 34, pages 5883–5891, 2020. 12

[67] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart. Stealing machine learning models via prediction {APIs}. In *USENIX Security*, pages 601–618, 2016. 12

[68] G. Uzor, H. Al-Qudah, et al. Guarding your conversations: Privacy gatekeepers for secure interactions with cloud-based ai models. In *IEEE ICSC*, pages 235–242, 2025. 12

[69] B. Wang and N. Z. Gong. Stealing Hyperparameters in Machine Learning. In *IEEE S&P*, 2018. 1

[70] B. Wang and N. Hegde. Privacy-preserving q-learning with functional noise in continuous spaces. *NeurIPS*, 32, 2019. 11

[71] L. Wang, Z. Javed, et al. BACKDOORL: Backdoor Attack against Competitive Reinforcement Learning. In *IJCAI*, 2021. 1, 6

[72] S. Wang. Security and ownership verification in deep reinforcement learning. Master's thesis, University of Waterloo, 2022. 1

[73] X. Wang, X. Liu, S. Lai, X. Yi, and X. Yuan. Siguard: Guarding secure inference with post data privacy. In *NDSS*, 2025. 12

[74] X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao. Deep reinforcement learning: A survey. *IEEE TNNLS*, 35(4):5064–5078, 2022. 3

[75] Y. Wang, E. Sarkar, et al. Stop-and-Go: Exploring Backdoor Attacks on Deep Reinforcement Learning-Based Traffic Congestion Control Systems. *IEEE TIFS*, 2021. 1

[76] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM*, pages 2512–2520. IEEE, 2019. 12

[77] Z. Wang, F. Yang, L. Wang, P. Zhao, H. Wang, L. Chen, Q. Lin, and K.-F. Wong. Self-guard: Empower the llm to safeguard itself. In *NAACL HLT*, pages 1648–1668, 2024. 12

[78] C. J. C. H. Watkins. Learning from Delayed Rewards. 1989. 3

[79] R. J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 1992. 3, 7

[80] B. Wu, H. Zhang, X. Yang, S. Wang, M. Xue, S. Pan, and X. Yuan. Graphguard: Detecting and counteracting training data misuse in graph neural networks. In *NDSS*, 2025. 12

[81] X. Wu, W. Guo, et al. Adversarial Policy Training against Deep Reinforcement Learning. In *USENIX Security*, 2021. 6

[82] T. Xu, Q. Liu, et al. Learning to Explore via Meta-Policy Gradient. In *ICML*, 2018. 3

[83] Z. Xu, S. Lai, X. Liu, A. Abuadbba, X. Yuan, and X. Yi. {OblivGNN}: Oblivious inference on transductive and inductive graph neural network. In *USENIX Security*, pages 2209–2226, 2024. 12

[84] M. Xue, Y. Zhang, J. Wang, and W. Liu. Intellectual Property Protection for Deep Learning Models: Taxonomy, Methods, Attacks, and Evaluations. *IEEE TAI*, 3(6):908–923, 2021. 1

[85] Z. Yang, N. Iyer, J. Reimann, and N. Virani. Design of Intentional Backdoors in Sequential Models. *CoRR abs/1902.09972*, 2019. 1

[86] Y. Yu, J. Liu, S. Li, K. Huang, and X. Feng. A Temporal-pattern Backdoor Attack to Deep Reinforcement Learning. In *IEEE GLOBECOM*, pages 2710–2715. IEEE, 2022. 12

[87] M. Zhang, Z. McCarthy, et al. Learning Deep Neural Network Policies with Continuous Memory States. In *ICRA*, 2016. 3

[88] X. Zhang, C. Fang, and J. Shi. Thief, Beware of What Get You There: Towards Understanding Model Extraction Attack. *arXiv preprint arXiv:2104.05921*, 2021. 10

[89] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song. The secret revealer: Generative model-inversion attacks against deep neural networks. In *CVPR*, pages 253–261, 2020. 12

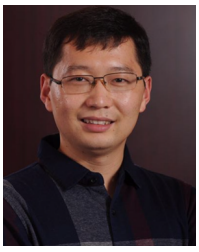
[90] Z. Zhuang, M.-I. Nicolae, and M. Fritz. Stealthy Imitation: Reward-guided Environment-free Policy Stealing. In *ICML*, pages 62682–62706, 2024. 1, 9, 10, 11

Linkang Du (Member, IEEE) received his B.E. and Ph.D. degrees from Zhejiang University in 2018 and 2023, respectively. He is currently an assistant professor at the School of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an, China. His research interests focus on data security and privacy issues in machine learning applications, including privacy protection and dataset copyright auditing. His vision is to protect the legitimate data rights and interests of every individual in the era of widespread artificial intelligence.

Zhikun Zhang (Member, IEEE) is a ZJU 100-Young Professor in the College of Computer Science and Technology at Zhejiang University. He received his Ph.D. degree in Control Science and Engineering from Zhejiang University in 2019. Before joining Zhejiang University, he was a Visiting Assistant Professor at Stanford University and Research Group Leader at CISPA Helmholtz Center for Information Security, Germany. His research interest concentrates on data privacy, differential privacy, and trustworthy artificial intelligence. He has published more than 20 papers in the big four security conferences. Zhikun is the recipient of Distuiguished Paper Award and Distuiguished Reviewer Award at NDSS 2024.

Min Chen (Member, IEEE) is an Assistant Professor at Vrije Universiteit Amsterdam. She obtained her Ph.D. from the CISPA Helmholtz Center for Information Security in Germany. Her research centers on trustworthy AI, focusing on privacy and security in machine learning. Min has published over 10 papers in leading security conferences, including IEEE S&P, ACM CCS, NDSS, and USENIX Security. She was awarded the Distinguished Paper Award at NDSS 2024.

Mingyang Sun (Senior Member, IEEE) received the Ph.D. degree from the Department of Electrical and Electronic Engineering, Imperial College London, London, U.K., in 2017. From 2017 to 2019, he was a Research Associate and a DSI Affiliate Fellow with Imperial College London, London, U.K. He is currently a Research Professor with the Department of Industrial Engineering and Management, College of Engineering, Peking University, Beijing, China. He is also an Honorary Lecturer with Imperial College London. His research interests include AI in energy systems and cyber-physical energy system security and control.



Shouling Ji (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Georgia Institute of Technology, and the Ph.D. degree in computer science from Georgia State University. He is a ZJU 100-Young Professor with the College of Computer Science and Technology, Zhejiang University, and a Research Faculty with the School of Electrical and Computer Engineering, Georgia Institute of Technology. His current research interests include AI security, data-driven security, privacy, and data analytics. He is a member of ACM. He was the Membership Chair of the IEEE Student Branch at Georgia State from 2012 to 2013.

Peng Cheng (Member, IEEE) received the B.Sc. and Ph.D. degrees in control science and engineering from Zhejiang University, Hangzhou, China, in 2004 and 2009, respectively. He is currently a Professor and the Dean of the College of Control Science and Engineering, Zhejiang University. His research interests include cyber-physical systems security, networked sensing and control, and cloud networking. He has been awarded the 2020 Changjiang Scholars Chair Professor. He has received the State Science and Technology Progress Award and the MOE Natural Science Award. He serves/served as Associate Editor for IEEE TCC and IEEE TCNS. He also serves/served as a Guest Editor for IEEE TAC and IEEE TSIPN.

Jiming Chen (Fellow, IEEE) received the Ph.D. degree in Control Science and Engineering from Zhejiang University in 2005. He was vice Dean of Faculty of Information Technology, deputy director of the State Key laboratory of Industrial Control Technology, and Director of Industrial Process Control, Zhejiang University. He was a visiting researcher at the University of Waterloo from 2008 to 2010. Now he is a Chair Professor with the Department of Control Science and Engineering, Zhejiang University, and also is the president of Hangzhou Dianzi University. He serves/served on the editorial boards of multiple IEEE Transactions, e.g., IEEE TNCS, IEEE TPDS, IEEE TIE, IEEE TFS and the General Co-chairs for mulitple IEEE/ACM conference. He was a member of Fellow Evaluation Committee and an Distinguished Lecturer of IEEE VTSa Fellow of IEEE and a Fellow of CAA. His research interests include industrial IoT, networked control, cyber security.

Michael Backes (Fellow, IEEE) is the founding director and CEO of the CISPA Helmholtz Center for Information Security. His research results have shaped these scientific fields internationally, as is evidenced by over 300 peer-reviewed and highly cited publications in renowned international journals and conference proceedings. His research has earned him internationally renowned scientific awards and honors, in particular the ERC Synergy Grant, which is Europe's most highly endowed EU group research award, the Karl Heinz Beckurts Prize, an ERC Starting Grant, the Caspar Bowden Privacy Award, the IEEE and ACM Fellowship, as well as various Career Awards and Best Paper Awards. In particular, Prof. Backes was the first German to receive the MIT TR35 Award. Michael Backes is regularly listed in rankings as one of Germany's most important IT personalities.

Yang Zhang (Member, IEEE) received the PhD degree from the University of Luxembourg, in November 2016. He is a faculty member at CISPA Helmholtz Center for Information Security,Germany. Previously, he was a group leader at CISPA. His research interests include the intersection of privacy and machine learning. Over the years, he has published multiple papers at top venues in computer science, including WWW, CCS, NDSS, and USENIX Security. His work has received NDSS 2019 Distinguished Paper Award. He has served in the technical program committee of USENIX Security 2021, ACM CCS 2021 2020 2019, WWW 2021 2020, AAAI 2021, RAID 2020,ICWSM 2020, and PETS 2021 2020.