Imposter Injection: Learning to Select Features in Reinforcement Learning

Shruti Singh, Viet Bui, Abhijeet Gupta, Tam V. Nguyen, Luan V. Nguyen
Department of Computer Science, University of Dayton

***Abstract*—Reinforcement Learning has rapidly evolved from theoretical concepts to practical applications, particularly in high-stakes fields such as autonomous driving, healthcare, and logistics, enhancing both efficiency and decision-making capabilities. However, as these applications engage with more complex environments, they face significant challenges from adversarial inputs. These inputs can jeopardize system reliability and safety by inducing unpredictable and potentially hazardous decisions. In this paper, we examine the vulnerability of reinforcement learning systems to such adversarial threats by presenting our novel mitigation approach. We introduce a concept referred to as an *imposter*—carefully crafted adversarial noise added to the state space that can, for instance, cause a Lunar Lander to crash or a Bipedal Walker robot to lose balance. Our approach leverages *entropy*, a measure of information content, to differentiate genuine data from adversarial noise effectively. We assess the efficacy of this method through the evaluation of different entropy measures: single entropy, joint entropy, and Kullback–Leibler divergence, across two practical case studies involving the Lunar Lander and the Bipedal Walker robot. The results demonstrate that the entropy metric is particularly effective in detecting and eliminating imposter features, thereby preserving the integrity and safety of critical missions.**

***Index Terms*—Imposter, Reinforcement Learning, Robotics**

I. Introduction

Reinforcement learning (RL) systems have gained significant attention due to its versatile capabilities. These systems excel in optimizing decision-making processes across a variety of applications, such as autonomous driving, robotics, financial trading, and healthcare management. Despite these advantages, RL systems are susceptible to adversarial attacks [1, 2]. In autonomous vehicles, for example, real-time decisions such as lane changes, traffic signal interpretation, and pedestrian interactions rely on sensory data that is vulnerable to adversarial interference—specifically, noise crafted by malicious actors to induce erroneous behavior. While RL systems are designed to handle noisy sensorial inputs, they are not resilient against adversarial noise [3, 4], which is deliberately crafted to exploit system vulnerabilities. For example, a malicious attacker could manipulate data from sensors like LiDAR, obscuring real obstacles and creating non-existent ones. These manipulated decisions can lead to dangerous outcomes, including collisions, road accidents, and damage to infrastructure. Therefore, achieving resilience against adversarial attacks require a defensive framework tailored to the unique characteristics of the system.

In this paper, we address adversarial attacks on the observation space of an RL system. We propose a novel entropy-based framework that detects and removes imposter features by harnessing the inherent uncertainty in the feature space. The framework illustrated in Figure 1 begins with the injection of adversarial noise (imposter features) into the environment, which disrupts the feature distribution by introducing false data points. This disruption can lead to incorrect decision-making by the RL model, assessed by observing significant deviations in the agent's performance metrics, such as unexpected fluctuations in mean return or increased standard deviation indicating instability. The disrupted feature distribution is split into training and testing sets, both containing original and imposter features. The training set is processed by the entropy-based learning algorithm, which leverages entropy measures and other metrics to differentiate between genuine and imposter features. The algorithm calculates the entropy of each feature, identifying those with higher entropy as potential imposters. By iteratively learning the algorithm updates the weights of the features based on the information metrics. The trained model is then evaluated with a testing set to measure its accuracy in identifying imposter features.

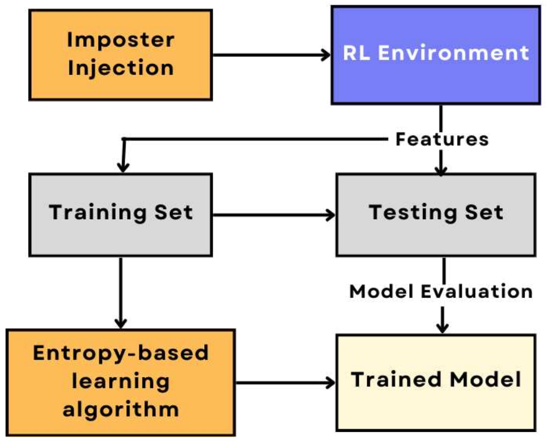

Fig. 1: Overview of the proposed framework: Adversarial noise (imposter features) is injected into the RL environment, compromising the feature distribution. This distribution is divided into training and testing sets, both containing original and imposter features. The training set is processed by an entropy-based algorithm to identify and learn from imposter features. The trained model is then evaluated on the testing set to assess its accuracy in detecting imposter features.

For evaluation, we examine the effectiveness of the proposed framework using various entropy metrics, including single entropy, joint entropy, and Kullback-Leibler divergence, through two case studies: the Bipedal Walker and the Lunar Lander. The Bipedal Walker involves a 4-joint walker robot navigating an environment, while the Lunar Lander simulates an agent learning to land safely on the moon's surface using eight feature states. We simulate adversarial attacks by injecting imposter features into the observation space in both scenarios, demonstrating how our entropy-based framework identifies imposter features and restores the agent's performance. Although our experiments focus on specific scenarios, the fundamental concepts of the framework—using entropy and other information metrics to measure feature uncertainty—are adaptable and scalable across various RL applications. This approach aims to enhance the robustness and reliability of RL systems.

II. Related Work

This section briefly reviews recent feature selection methods in RL and compares them with the proposed entropy-based approach. We also overview the latest defense mechanisms against adversarial attacks in RL.

A. *Feature Selection in Reinforcement Learning*

Feature selection is critical for effective learning in RL, but it poses challenges due to the complex and non-linear relationships between states and rewards. Traditional strategies like dimensionality reduction through basis functions [5] and linear models for value function approximation [6, 7] are designed for specific RL algorithms, such as policy iteration. These methods often struggle in other contexts due to indirect state-reward dependencies and the computational intensity required to evaluate these relationships. Innovative approaches have been developed to address these challenges. For instance, [8, 9] propose methods that select features independently of the learning processes but encounter difficulties in assessing state-reward dependencies. Techniques that use conditional mutual information [10] attempt to manage these dependencies, but their effectiveness depends heavily on the accuracy of estimation methods like Least-Squares Mutual Information (LSMI) [11]. Inaccuracies in these estimations can lead to biased or unreliable feature selection outcomes. In contrast, the proposed entropy-based framework measures uncertainty within the feature distribution, allowing for precise identification and removal of imposter features without relying on approximations.

B. *Adversarial Attacks in Reinforcement Learning*

Adversarial attacks on RL algorithms like Q-learning, DQN, and A3C significantly impair decision-making by manipulating Q-values, exploiting neural network vulnerabilities, and altering value gradients [12, 13, 14]. Specific Time-Step Attack strategies guide agents towards undesired states [15]. Although adversarial training enhances robustness by integrating adversarial examples [16, 17], these methods often lack generalizability across different attack vectors. Kos et al. reveal significant weaknesses in deep RL policies under small perturbations [18]. Lin et al. demonstrate RL agents' vulnerability to various adversarial tactics, leading to degraded performance in complex environments [19]. Behzadan et al. present adversarial models in autonomous vehicle simulations, showing how attacks can compromise vehicle safety by causing erratic behaviors, emphasizing real-world attack implications [20]. Recent studies on detecting false data injection attacks using machine learning highlight advanced feature selection techniques. For example, feature selection and oversampling balance class distributions and enhance feature relevance, while sparse Bayesian learning [21, 22, 23] identifies the most informative features to reduce computational complexity. These approaches emphasizes selecting relevant features to enhance detection accuracy. Complementing existing approaches, the proposed entropy-based feature selection framework offers a key advantage by quantifying uncertainty across the feature distribution. This allows for precise identification and removal of misleading features without relying on specific attack models, providing a scalable and adaptable solution for various RL applications. It effectively defends against a broad range of adversarial threats, including sensor spoofing and environmental noise.

III. Methodology

A. *Imposter Injection*

Imposter features following Gaussian noise are injected into the observation space to challenge the agent's learning. These imposters introduce noise, disrupting the agent's decision-making process and affecting its expected reward.

Gaussian and Uniform Noise Injection. To simulate variations in the agent's learning, we inject noise into the observation space. This noise takes two forms: Gaussian noise, which follows a normal distribution defined by a mean (μ) and standard deviation (σ), and uniform noise, which spreads evenly across a range. Mathematically, the Gaussian noise n added to an imposter feature s can be expressed as: $n \sim \mathcal{N}(\mu, \sigma^2)$, where $\mathcal{N}(\mu, \sigma^2)$ represents a normal distribution with mean μ and variance σ^2. For an injected feature defined between the maximum bounds, uniform noise is incorporated, represented as $u \sim \mathcal{U}(a, b)$, where $\mathcal{U}(a, b)$ denotes a uniform distribution with lower bound a and upper bound b, resulting in the imposter feature $s' = s + u$. Mathematically, an imposter feature appends the original state space S, represented as a set of states $\{s_1, s_2, \ldots, s_n\}$. By appending a new imposter feature s', the original state space is modified $S' = S \cup s'$ to accommodate for the new imposter feature. Then, the presence of imposter features affects the agent's state space (S), policy (π), and immediate reward function (R).

State Space Expansion. The state space expands from S to $S' = S \cup s'$. This change is reflected in the probability distribution of the agent's policy. In policy-based RL algorithms like PPO, the agent's policy, typically denoted by $\pi(s', a)$, maps states to actions and is iteratively updated to maximize expected return.

Expected Return. The reward function, $R(s,a)$, assigns values to state-action pairs, representing the immediate reward obtained when taking action a in state s. To incorporate rewards from imposter features, we extend the original reward function as follows:

$$R_{\text{total}} = R(s,a) + R(s',a). \tag{1}$$

Equation. 1 assumes that the total reward is a cumulative measure of the rewards from both original and imposter features. When an agent encounters an imposter feature, it experiences an additional reward (or penalty $R(s',a)$). This combined reward reflects the agent's interaction with both the original and manipulated parts of the state space. The reward $R(s',a)$ obtained from encountering the imposter feature incentivizes the agent to adapt its policy. Summing these rewards over time gives the total expected return $J(\theta)$:

$$J(\theta) = \mathbb{E}\left[\sum_{t=0}^{T} R_{\text{total}}(s_t, a_t)\right] \tag{2}$$

The expected return $J(\theta)$ is defined as the expected sum of rewards over a time horizon T. By incorporating the total reward R_{total} into this equation, we account for the impact of imposter features on the overall performance of the agent. This helps in analyzing how the presence of imposter features influences the agent's learning and decision-making process.

Here $R_{\text{total}}(s,a)$ is the combined reward obtained when taking action a in state s. The inclusion of high positive rewards from imposter features can inflate the expected return $J(\theta)$, potentially leading to a stronger bias towards exploiting these features. Conversely, high negative rewards may cause the agent to become overly cautious, avoiding imposter features and hindering exploration. As a result, the injection of imposter features creates significant fluctuations in the learning process, making it harder for the agent to converge to an optimal policy. Frequent encounters with imposter features can lead to disruptive and unstable behavior, making it difficult for the agent to focus on learning from original states and ultimately wasting computational resources.

To quantify the impact of increasing imposter features on the learning process, we analyze the change in the expected return $J(\theta)$ before and after their introduction. Let ΔJ be the deviation between the expected return in the original scenario and the expected return when incorporating imposter features.

$$\Delta J = J_{\text{imposters}}(\theta) - J_{\text{original}}(\theta) \tag{3}$$

We define two thresholds: ΔJ_{pos} and ΔJ_{neg}, which correspond to significant positive and negative deviations in the agent's expected return, respectively. These thresholds are determined based on the standard deviations observed in the agent's performance during successful episodes. When $\Delta J > \Delta J_{\textbf{pos}}$, it indicates a significant expected return from the imposter features, surpassing the positive threshold ΔJ_{pos}. Conversely, when $\Delta J < \Delta J_{\textbf{neg}}$, it denotes a notable expected return decrease due to imposters, falling below the negative threshold ΔJ_{neg}. These thresholds are determined based on the standard deviations of the agent's performance in successful episodes. Specifically, we set ΔJ_{pos} and ΔJ_{neg} to values that reflect significant deviations from the mean expected return, considering the variability in the agent's performance metrics.

B. Learning to Detect Imposter Features

In this subsection, we detail our entropy-based framework designed to identify and eliminate imposter features from the observation space. The Shannon's entropy [24] is popular in feature selection. Here, we compute the entropy of the feature state distribution to identify and eliminate imposter features. Entropy quantifies the uncertainty in data, which, in our case, is used to analyze the uncertainty introduced by imposter features in an RL environment. The entropy (H) of a discrete random variable X with possible values $x_1, x_2, .., x_n$ and a probability mass function $P(X)$ is defined as:

$$H(X) = -\sum_{i=1}^{n} P(x_i)\log P(x_i). \tag{4}$$

The feature selection algorithm uses the following linear relationship between features and labels:

$$w_1(\mu) + w2(H(s) - \mu) + \beta = y. \tag{5}$$

Here $H(s)$ represents the entropy value (central tendency) for a given state s, while μ denotes the average entropy for the feature state distribution. These values help evaluate the variability and informativeness of the feature state distribution, aiding in feature selection. Additionally, $y \in [-1,1]$ serves as the label indicating whether a feature state is an imposter (1) or an original state (-1). To determine the significance of each feature in label prediction, the weights w_1, w_2, β are learned during a training process. This training involves solving for the weights that minimize the discrepancy between predicted and actual labels.

- Entropy $H(s)$: Measures (Eqn. 4) the uncertainty of each feature state s
- Average Entropy μ: Central tendency of the feature state distribution.
- Labels $y \in [-1,1]$: Classifies feature states, with discrete values 1 denoting imposter and -1 for original states.

$$\begin{bmatrix} w_1 & w_2 & \beta \end{bmatrix} \begin{bmatrix} \mu & H(s)-\mu & 1 \end{bmatrix}^T = y \tag{6}$$

We begin with a naive solution by directly solving the linear equation: $W = YX^{-1}$. Here, X is a $m \times n$ non-zero matrix containing feature information, and Y represents a vector of imposter status labels.

We then compare the performance of this naive algorithm with various machine learning methods, including Random Forest, K-Nearest Neighbors, and Support Vector Machine. These methods help us evaluate the effectiveness of our feature selection algorithm across different models. The weights are

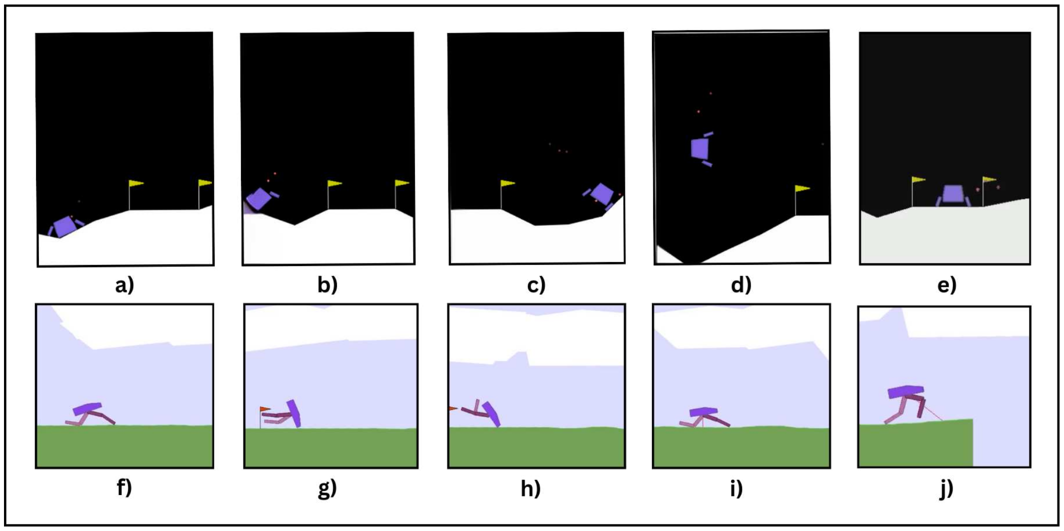

Fig. 2: Top row: (a-d) show the Lunar Lander with 1, 2, 3, and 4 imposters (uniform noise) respectively, while (e) shows the lander landing between the two flag posts. Bottom row: (f-i) show the Bipedal Walker with 1, 2, 4, and 8 imposters, respectively, also injected with uniform noise, while (j) shows the walker robot reaching the other side.

Algorithm 1 Learning Weights W

```
1: Input: $H(s)$ (feature information), $s \in S$ (state spaces), $Y \in [-1, 1]$ (imposter labels)
2: Output: $w_1$, $w_2$, $\beta \in W$ (learned weights)
3: $W \leftarrow \emptyset$                                  ▷ Initialize weights to empty
4: if $\Delta J < \Delta J_{\text{neg}}$ or $\Delta J > \Delta J_{\text{pos}}$ then
5:     $X \leftarrow \emptyset$                              ▷ Initialize feature set
6:     for $s \in S$ do                                      ▷ Iterate through each state
7:         $X \leftarrow X \cup \begin{bmatrix} \mu & H(s) - \mu & 1 \end{bmatrix}^T$   ▷ Update feature set
8:     end for
9:     $W \leftarrow f_c(X, Y)$                              ▷ Compute learned weights
10: end if
11: return $W$                                             ▷ Return weights
```

learned through training each machine learning model to classify the feature states accurately.

The pseudocode outlined in Algorithm 1 delineates the step-wise execution of the feature selection algorithm, commencing when the condition for ΔJ is met. It initializes an empty set X for storing state features and iterates through each state, updating X with pertinent feature information. Upon completion of the iterations, the weights W (Equation 5) are obtained, and the resultant values are returned. In addition, we can replace the single entropy by the joint entropy or Kullback–Leibler divergence to detect the imposter feature(s).

IV. Experimental Results

In this section, we assess the efficacy of the entropy-based framework in two popular environments, namely, the Lunar Lander and Bipedal Walker.

A. Lunar Lander

Environment. The Lunar Lander environment in Gymnasium's Box2D collection challenges RL agents to manage complex dynamics and precise control. The action space includes four discrete actions: do nothing, fire left engine, fire main engine, and fire right engine. The observation space is an 8-dimensional vector comprising the lander's coordinates, velocities, angle, angular velocity, and leg-ground contact status. A successful episode achieves at least 200 (J) points.

Training. We train the Lunar Lander using the PPO [25] algorithm for 200,000 steps. Imposter features are systematically introduced into the observation space, starting with a single imposter and increasing to two, three, and four imposters. This results in significant fluctuations in the expected return J, with both declines and unrealistic performance boosts, highlighting the exploitation of imposter features. For all injections, two types of noise (uniform and Gaussian) are used. However, only the two in-range imposter injections mimic the natural variability within the environment, making it harder for the agent to distinguish between genuine and imposter features, thereby testing the robustness of the agent's learning and adaptation mechanisms.

Threshold. Thresholds were determined based on the goal scores for the respective environments. In the Lunar Lander environment, a score of 200 indicates a successful episode. To evaluate performance degradation due to imposter features, we set thresholds for ΔJ_{pos} and ΔJ_{neg} based on observed standard deviations from multiple successful runs (66.36 and 40). These thresholds account for normal performance variability, with $\Delta J_{\text{pos}} = 300$ and $\Delta J_{\text{neg}} = 100$, allowing us to detect significant deviations caused by imposter features.

Impact of Imposter Features. When there are no imposter features, the mean return during training is 1.121. With one imposter injection, the mean return during training is 1.37. For two imposters and three imposters injection, the mean return drops to -6.74 and -10.53 during training, respectively. These significant fluctuations in return suggest the instability in training. This reflects the failure cases of Lunar Lander as

TABLE I: Evaluation on Lunar Lander: Accuracy of methods in detecting imposter features using different metrics.

Methods	Entropy	Joint Entropy	KL
Naive (Linear regression)	94.00	80.54	60.32
Random Forest	97.05	95.06	79.62
KNN (K = 5)	94.11	91.35	76.54
SVM	91.11	75.92	69.13

TABLE II: Evaluation on Bipedal Walker: Accuracy of methods in detecting imposter features using different metrics.

Methods	Entropy	Joint Entropy	KL
Naive (Linear regression)	95.00	88.56	83.45
Random Forest	93.33	92.50	90.78
KNN (K = 5)	95.55	96.00	95.55
SVM	97.77	95.77	95.77

visualized in Figure 2 (b, c, d). indicating poor training due to negative returns. Overall, imposter features worsen the training process. Negative returns make the agent overly cautious, while positive returns lead to exploitation and misleadingly high returns. These returns do not reflect actual landing skills as evident in the visualization in 2 (b, c, d).

B. Bipedal Walker

Environment. The Bipedal Walker environment involves a 4-joint walker robot navigating an uneven terrain. The action space includes continuous values controlling the hip and knee joints of the robot. The observation space comprises the hull's angle, angular velocity, horizontal and vertical speed, joint angles, and joint angular speeds. The goal is to navigate the terrain efficiently without falling, and obtain 300 (J) points in 1600 time steps for the normal version.

Training. We train the Bipedal Walker using the (Augmented Random Search) ARS algorithm for 1000 iterations. The thresholds for Bipedal Walker are set as $\Delta J_{\text{pos}} = 96$ and $\Delta J_{\text{neg}} = -1.97$, based on observed performance variability.

Impact of Imposter Features. Similar to the Lunar Lander, we also observed the return fluctuation in the training process. This highlights the instability in training with the injection of imposter features.

C. Evaluation on Imposter Detection

We use the entropy-based feature selection algorithm to detect imposter features as outlined in Algorithm 1. The approach is evaluated in both the Lunar Lander and Bipedal Walker environments, with results summarized in I and II. The accuracy of the entropy-based framework in detecting imposter features is assessed using entropy, joint entropy (JE), and KL Divergence across four methods: naive, Random Forest (RF), K-Nearest Neighbors (KNN), and Support Vector Machine (SVM).

In the Lunar Lander experiment, the RF method achieves the highest accuracy of 97.05%, making it well suited for complex and varied features for this environment (see Table I), outperforming joint entropy at 95.06% and KL divergence at 79.62%. This superior performance can be attributed to entropy's ability to capture the spread of values within a single feature's distribution, making it highly sensitive to anomalies introduced by imposter features. This sensitivity allows the method to effectively learn and identify these anomalies. We see the SVM's poor performance, which might be associated with Lunar Lander's small dataset.

In contrast, joint entropy considers distributions of feature pairs and captures their relationships. In the Lunar Lander environment, most features had independent relationships with the imposter features. However, there were instances where the combined uncertainty between two genuine features was higher than with an imposter feature. These joint entropy values can disrupt the framework's learning process in identifying imposter features, making it susceptible to misclassification of the relationships with imposter features.

KL divergence performs comparatively poor in identifying imposter features. The divergence interpretations shows little to no overlap between the probability distributions of the imposter features and the genuine features. However, this similarity in divergence values is also observed among genuine feature pairs, making it difficult for the methods to distinguish imposter features effectively.

In the Bipedal Walker environment, the SVM method achieves the highest accuracy with entropy at 97.77%, showing its effectiveness in handling the Bipedal Walker's high dimensional feature space. This is followed closely by KNN with an accuracy of 95.55% and at last the RF method, which proved to be limited when handling more features. The Naive algorithm exhibits lower performance across all metrics, which can be attributed to the smaller dataset, particularly when considering the 24 features of the Bipedal Walker. Consequently, entropy consistently performs better than other metrics in detecting imposters for both the Lunar Lander and Bipedal Walker environment. Thus, it highlights the entropy's robustness and effectiveness for detecting imposter features in different reinforcement learning scenarios.

V. Conclusion and Future Work

In this paper, we investigate the problem of imposter injection in RL, with experimental results showing that entropy effectively detects imposter features. The proposed framework is applied in the Lunar Lander and Bipedal Walker environments, demonstrating its ability to observe imposter features and assess their impact. Our study focuses on these two environments with Gaussian and uniform noise, and future work could explore other environments, noise types, and extend the framework with additional information measures. In addition, we plan to automate and optimize feature selection, enabling the framework to dynamically adapt to relevant features in different environments. By integrating the proposed approach with existing safe RL methods, we seek to enhance the efficiency, safety, and reliability of RL systems.

VI. Acknowledgments

The authors thank reviewers for improving the quality of this work. This research is supported by the National Science Foundation (NSF) NSF-CRII-2245853. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of NSF.

References

[1] Y. Wang, T. Sun, S. Li, X. Yuan, W. Ni, E. Hossain, and H. V. Poor, "Adversarial Attacks and Defenses in Machine Learning-Powered Networks: A Contemporary Survey," arXiv preprint arXiv:2303.06302, 2023.

[2] T. Chen, J. Liu, Y. Xiang, W. Niu, E. Tong, and Z. Han, "Adversarial Attack and Defense in Reinforcement Learning from AI Security View," Cybersecurity, vol. 2, pp. 1–22, 2019.

[3] Z. Xing and B. Liu, "Vector correlation learning and pairwise optimization feature selection for false data injection attack detection in smart grid," International Journal of Emerging Electric Power Systems, vol. 23, no. 6, pp. 831-838, 2022. DOI: 10.1515/ijeeps-2022-0148.

[4] X. Xiong, S. Hu, D. Sun, S. Hao, H. Li, and G. Lin, "Detection of false data injection attack in power information physical system based on SVM–GAB algorithm," Energy Reports, vol. 8, Supplement 5, pp. 1156-1164, 2022. DOI: 10.1016/j.egyr.2022.02.290.

[5] P. W. Keller, S. Mannor, and D. Precup, "Automatic basis function construction for approximate dynamic programming and reinforcement learning," in Proceedings of the 23rd international conference on Machine learning, 2006, pp. 449–456.

[6] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," The Journal of Machine Learning Research, vol. 4, pp. 1107–1149, 2003.

[7] R. Parr, L. Li, G. Taylor, C. Painter-Wakefield, and M. L. Littman, "An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning," International Conference on Machine learning, 2008, pp. 752–759.

[8] J. Morimoto, S.-H. Hyon, C. G. Atkeson, and G. Cheng, "Low-dimensional feature extraction for humanoid locomotion using kernel dimension reduction," in 2008 IEEE International Conference on Robotics and Automation, 2008, pp. 2711-2716.

[9] M. Kroon and S. Whiteson, "Automatic Feature Selection for Model-Based Reinforcement Learning in Factored MDPs," in 2009 International Conference on Machine Learning and Applications, 2009, pp. 324-330.

[10] H. Hachiya and M. Sugiyama, "Feature selection for reinforcement learning: Evaluating implicit state-reward dependency via conditional mutual information," in Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2010, pp. 474–489.

[11] T. Suzuki, M. Sugiyama, T. Kanamori, and J. Sese, "Mutual information estimation reveals global associations between stimuli and biological processes," BMC bioinformatics, vol. 10, pp. 1–12, 2009.

[12] Y. Xiang, W. Niu, J. Liu, T. Chen, and Z. Han, "A PCA-based model to predict adversarial examples on Q-learning of path finding," in 2018 IEEE third international conference on data science in cyberspace (DSC), 2018, pp. 773–780.

[13] X. Bai, W. Niu, J. Liu, X. Gao, Y. Xiang, and J. Liu, "Adversarial examples construction towards white-box Q table variation in DQN pathfinding training," in 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC), 2018, pp. 781–787.

[14] T. Chen, W. Niu, Y. Xiang, X. Bai, J. Liu, Z. Han, and G. Li, "Gradient band-based adversarial training for generalized attack immunity of A3C path finding," arXiv preprint arXiv:1807.06752, 2018.

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

[16] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, "Improving the robustness of deep neural networks via stability training," in Proceedings of the ieee conference on computer vision and pattern recognition, 2016, pp. 4480–4488.

[17] Z. Yan, Y. Guo, and C. Zhang, "Deepdefense: Training deep neural networks with improved robustness," arXiv preprint arXiv:1803.00404, 2018.

[18] J. Kos and D. Song, "Delving into adversarial attacks on deep policies," arXiv preprint arXiv:1705.06452, 2017.

[19] 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," arXiv preprint arXiv:1703.06748, 2017.

[20] V. Behzadan and A. Munir, "Adversarial reinforcement learning framework for benchmarking collision avoidance mechanisms in autonomous vehicles," IEEE Intelligent Transportation Systems Magazine, vol. 13, no. 2, pp. 236–241, 2019.

[21] A. Fawzi, S. M. Moosavi-Dezfooli, and P. Frossard, "Improving Detection of False Data Injection Attacks Using Machine Learning with Feature Selection and Oversampling," *Energies*, vol. 11, no. 5, pp. 1110, 2018.

[22] H. He, J. Yan, and Y. Sun, "Detecting False Data Injection Attacks Based on Enhanced Sparse Bayesian Learning," *International Journal of Electrical Power & Energy Systems*, vol. 134, pp. 107325, 2022.

[23] Y. Liu, P. Li, and S. Liu, "Machine Learning Algorithm for Detection of False Data Injection Attack in Power System," *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3203-3213, 2020.

[24] C. E. Shannon, "A mathematical theory of communication," The Bell system technical journal, vol. 27, no. 3, pp. 379–423, 1948.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv preprint arXiv:1707.06347, 2017.