2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

Transformer-Based Anomaly Detection in Deep Reinforcement Learning

Zhen Chen[1], Jian Zhao[2], Youpeng Zhao[2], Yong Liao[1] and Hu Huang[1](✉)

[1] University of Science and Technology of China, Hefei 230026, China
huanghu@ustc.edu.cn
[2] Polixir, Nanjin 211100, China

Abstract. Despite promising potential of Deep Reinforcement Learning (DRL) to adapt to various tasks, it remains highly vulnerable to adversarial attacks or anomalous observation signals. Existing research on DRL robustness does not fully safeguard against all types of adversarial perturbations and disturbances, which hampers its use in critical real-world systems and applications, such as smart grids, traffic control, and autonomous vehicles. In these contexts, anomalous states can cause decision-making errors that may be exacerbated in subsequent actions, resulting in irreparable damage. To promptly detect anomalous states and prevent greater losses, we propose the Transformer-Based Anomaly Detection (T-BAD) framework. Utilizing the transformer's ability to handle sequential data robustly, T-BAD enables real-time detection of anomalous states and actions. Specifically, first, our approach involves collecting trajectory data from the DRL algorithm used by the agent and training a transformer module to accurately model the agent's actions. For anomaly detection, we input k consecutive states and k-1 consecutive actions (excluding the current action) into the transformer, and then compare its output with the agent's action. The difference between them indicates whether DRL model is influenced by interference or adversarial attacks. Extensive experiments across multiple scenarios in Atari and Mujoco demonstrate that our T-BAD outperforms existing baselines in anomaly detection while also possessing some capacity for anomaly correction.

Keywords: Deep Reinforcement Learning, Transformer, Anomaly Detection.

1 Introduction

Since the effectiveness of using Deep Neural Networks (DNNs) to parameterize reinforcement learning policies was demonstrated, substantial progress has been made in the development of new algorithms and applications within Deep Reinforcement Learning (DRL) [1]. However, these advancements have also introduced new challenges. Notably, the vulnerability of DNNs to perturbations in barely perceptible adversarial directions was first revealed by [2]. Subsequent studies by [3] and [4] extended these findings to DRL, showing that this vulnerability persists in reinforcement learning contexts. Such fragility is unacceptable in applications like autonomous driving, automated financial trading, or medical decision-making.

Extensive efforts have been made to enhance the robustness of DNNs against adversarial perturbations [5,6]. Some studies have explored adversarial directions, the cost of generalized robust training, and the inherent challenges in learning robust models [7,8]. However, no matter how robust a model is, it cannot withstand all adversarial attacks, as this would require DNNs to learn all possible state-action pairs, which are infinite. Given that achieving complete robustness against adversarial examples in DNNs is currently infeasible, a practical approach is to focus on detecting the presence of adversarial manipulations. A method for detecting adversarial directions in the neural loss landscape of DRL, effectively identifying adversarial states, was proposed [9]. However, adversarial attacks in DRL are not limited to state interference; they can also target actions. Moreover, states are not only vulnerable to adversarial attacks but may also be affected by issues such as random noise and signal loss in real-world scenarios. Currently, no general method exists for detecting all kinds of anomalies in DRL systems.

Surprisingly, we discover that vanilla sequence modeling methods, such as decision transformer, exhibit robustness against data corruption, even without specialized modifications [10]. To unlock the full potential of sequence modeling, we propose the Transformer-Based Anomaly Detection (T-BAD) framework which detects anomalies in actions and states that were not encountered during the training of the agent and decides whether to terminate the algorithm's operation to mitigate potential losses. To be specific, we collect trajectory data from DRL algorithm and train a transformer, enabling it to fit the agent's current actions based on the input state-action sequence. Given transformers are designed to process sequence data and inherently more robust to adversarial samples, we compare the T-BAD framework's output actions with those of the deployed agent to detect anomalies. In summary, our research makes the following contributions:

- We propose a Transformer-Based Anomaly Detection (T-BAD) framework that utilizes historical trajectory data to train transformers to detect adversarial samples or interferences.
- Extensive experiments across various scenarios from Atari and Mujoco demonstrate that T-BAD provides better anomaly detection capabilities compared to other baselines, for both discrete and continuous actions.
- More importantly, we also verify that T-BAD has the ability to correct certain anomalous actions to some extend.

2 Related Work

2.1 Adversarial Attack in DRL

In recent years, adversarial attacks in DRL can be categorized into reward-based attacks [11], strategy-based attacks [12], observation-based attacks [13,14], environment-based attacks [15], and action-based attacks [16] according to their algorithmic principles. Reward-based attacks manipulate the reward signal from the environment, either by changing the reward value's sign or replacing the original reward function with an adversarial one. Strategy-based attacks use adversarial agents to generate states and actions that disrupt the victim agent's decision-making process. Observation-based attacks involve adding perturbations to the sensory input of agent, typically images, typically images, leading the victim agent to take actions desired by the attacker. Environment-based attacks modify the agent's training environment directly, altering its dynamics or introducing new obstacles. Action-based attacks directly modify action outputs by changing the action space in the training data. In DRL, regardless of the type of adversarial attack [17], the fundamental goal is to alter the agent's actions, as an attack is only considered successful unless the agent's actions are changed. Therefore, with the growing concerns about adversarial attack techniques in recent research, our work aims to provide a general framework for detecting adversarial attacks by identifying anomalous actions in DRL.

2.2 Adversarial Example Detection

Adversarial detection methods can be broadly categorized into distribution diversity, feature transformation, and deep learning-based approaches. For distribution diversity, [18] identified differences in the local intrinsic dimension (LID) between adversarial and clean samples, enabling detection via distance distribution analysis. Feature transformation-based methods leverage adversarial sensitivity to input changes. [19] detected adversarial samples via KL divergence between original and transformed outputs, while [20] used denoising and cross-entropy calculations. Additionally, [21] found that adversarial samples exploit high-curvature regions near decision boundaries, aiding differentiation. Deep learning-based approaches analyze model internal representations. [22] proposed cascade detection using CNN layer statistics, and [23] introduced SMSnet, integrating convolutional layers and global pooling-based feature synthesis for classification. These methods typically assume prior knowledge of adversarial samples or their distribution, which is impractical in DRL scenarios where such information is often unavailable.

3 Method

In this section, we introduce Transformer-Based Anomaly Detection (T-BAD), a framework designed to monitor whether an agent is experiencing adversarial attacks or signal interference in real-time during its interaction with the environment. This framework utilizes a pre-trained transformer to model the actions generated during the agent-environment interactions. When the agent is subjected to an adversarial attack, resulting in altered actions, the framework detects the anomaly by comparing the transformer's predicted actions with the agent's actual actions.

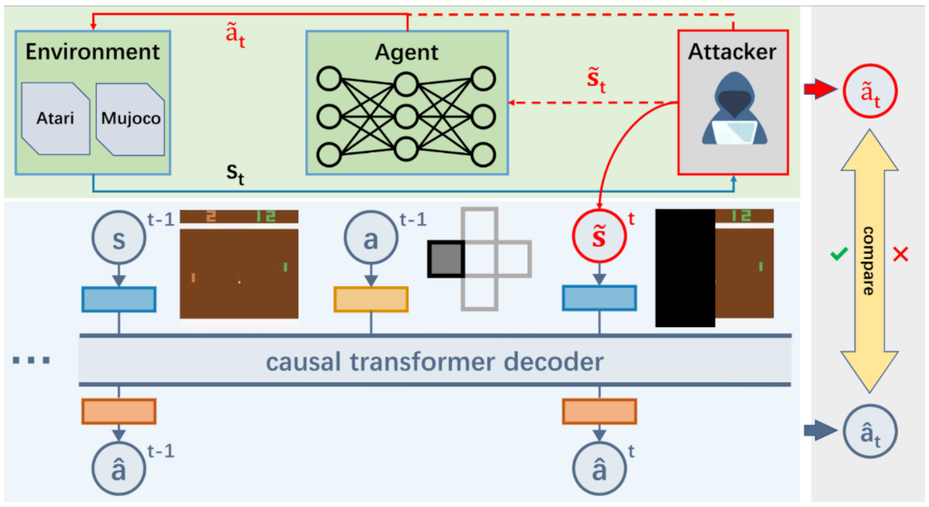


Fig. 1. T-BAD framework. The green section represents the process where the agent interacts with the environment, during which an attacker may attack the state or directly modify the action. The blue section indicates the process where we use a transformer to perform anomaly detection. If the attacker attacks a certain state $\tilde{s}_t$, the action output by the agent is $\tilde{a}_t$, and the action output by the transformer is $\hat{a}_t$. The gray section shows that we determine whether an attack or interference has occurred by comparing the differences between the two actions.

3.1 Reinforcement Learning as Sequence Modeling

In DRL tasks that satisfy the Markov property, the agent learns a mapping from the current state to the current action. Decision Transformer (DT) models decision-making from offline datasets as a sequence modeling problem [10]. The i-th trajectory $\tau^{(i)}$ of length T in dataset $\mathcal{D}$ is reorganized into a sequence of return-to-go $R_t^{(i)}$, state $s_t^{(i)}$, action $a_t^{(i)}$:

$$\tau^{(i)} = \left(R_0^{(i)}, s_0^{(i)}, a_0^{(i)}, \dots, R_{T-1}^{(i)}, s_{T-1}^{(i)}, a_{T-1}^{(i)}\right) \tag{1}$$

Here, the return-to-go (RTG) $R_t^{(i)}$ is defined as the sum of rewards from the current step to the end of the trajectory:

$$R_t^{(i)} = \sum_{t'=t}^{T} r_{t'}^{(i)} \tag{2}$$

DT employs three linear projection layers to project the return-to-go, states, and actions to the embedding dimension, with an additional learned embedding for each timestep added to each token. A GPT model is adopted by DT to autoregressively predict the actions $a_t^{(i)}$ with input sequences of length K:

$$\mathcal{L}_{\mathcal{DT}}(\theta) = E_{\tau^{(i)}\sim\mathbb{D}}\left[\frac{1}{K}\sum_{t=0}^{K-1}|\pi_\theta(\tau^{(i)}_{t-K+1:t-1}, R^{(i)}_t, s^{(i)}_t) - a^{(i)}_t|^2_2\right] \tag{3}$$

where $\tau^{(i)}_{t-K+1:t-1}$ indicates the segment of $\tau^{(i)}$ from timestep $t-K+1$ to $t-1$.

By increasing the input dimensionality, we enhance the model's robustness. As a result, even if an attacker alters the agent's state at a specific time step, they cannot modify the transformer's output action. Compared to training a binary classification neural network to determine whether the agent is under attack, our method has the advantage of not requiring any negative samples during training, making it more aligned with practical scenarios.

3.2 Do We Really Need RTG Token in Anomaly Detection?

In the absence of interference or attacks, the initial RTG can be a constant value to assist in predicting actions. However, in the presence of adversarial attacks where the rewards along the agent's trajectory are altered, the predicted RTG will significantly differ from the actual rewards. In this case, the RTG no longer aids in action prediction but rather becomes a source of interference. In this paper, the i-th trajectory $\tau^{(i)}$ of length T in dataset $\mathcal{D}'$ is reorganized into a sequence of state $s^{(i)}_t$, action $a^{(i)}_t$:

$$\tau^{(i)} = (s^{(i)}_0, a^{(i)}_0, \dots, s^{(i)}_{T-1}, a^{(i)}_{T-1}) \tag{4}$$

The sequences we consider in this paper are similar to those used in DT, but we do not include RTG as part of the sequence. T-BAD also adopts a GPT model to autoregressively predict the actions $a^{(i)}_t$ with input sequences of length K:

$$\mathcal{L}_{\mathcal{T}-\mathcal{BAD}}(\theta) = E_{\tau^{(i)}\sim\mathbb{D}'}\left[\frac{1}{K}\sum_{t=0}^{K-1}|\pi_\theta(\tau^{(i)}_{t-K+1:t-1}, s^{(i)}_t) - a^{(i)}_t|^2_2\right] \tag{5}$$

In environments with discrete action space, we calculate the cross-entropy loss between the agent's actions and those generated by T-BAD to train the transformer. In environments with continuous action space, the training phase involves calculating the mean squared error loss between the agent's actions and the actions generated by T-BAD.

3.3 Framework

As shown in **Fig. 1**, the agent interacts with the environment, producing a trajectory composed of historical states and actions. During this process, the current state s_t may be attacked, leading to a modified state $\widetilde{s_t}$. The input of the anomalous state $\widetilde{s_t}$ causes the agent to output a potentially altered action $\widetilde{a_t}$. Alternatively, the attacker might directly alter the agent's output. To illustrate the detection process employed by T-BAD, we focus on the common scenario of state-based adversarial example generation.

As shown in Algorithm T-BAD (for discrete actions), the last k timesteps of the trajectory $\tau = (s_{t-k+1}, a_{t-k+1}, \dots, a_{t-1}, \widetilde{s_t})$, consisting of $2k-1$ tokens, are provided as input to the transformer module. Actions $A_t = (a_{t-k+1}, \dots, a_{t-1})$ as well as states $S_t = (s_{t-k+1}, \dots, \widetilde{s_t})$ are processed through linear encoders, and an embedding for each timestep is added to each token. Distinct from the conventional positional embedding employed by transformers, each timestep corresponds to two tokens. The tokens are then processed by the transformer model with casual making (GPT), which fits the actions of the agent and outputs the prediction of the agent's current action. The action $\widehat{a_t}$ output by T-BAD will be compared with the action $\widetilde{a_t}$ actually executed by the agent. In a discrete action environment, a mismatch between these indicates the presence of an attack or interference. In a continuous action environment, we need to calculate the variance between the action output by T-BAD and the action executed by the agent. If this variance exceeds a predetermined threshold, the agent is considered to be under attack. The threshold is determined as the average loss during the last L rounds of the transformer pre-training process. It is important to note that the agent within the framework is well-trained, and its policies does not include the possibility of taking random actions.

Algorithm T-BAD (for discrete actions)

Require: S, A, T: states , actions , or timesteps; k: context length (length of each input to T-BAD); $transformer$: transformer with causal masking (GPT); DQN: DRL algorithm for agent; $attack$: attacks or interferes with a state at a certain frequency.

```
1: $S, A, T, done = [env.reset()], [\,], [1], False$
2: while not done do:
3:     $S[-1] = attack(S[-1])$
4:     $\tilde{a} = DQN(S[-1])$
5:     $\hat{a} = transformer(S[-k{:}\,], A[-k+1{:}\,], T[-1])$
6:     compare $\hat{a}$ with $\tilde{a}$
7:     $s, \_, done = env.step(\tilde{a})$
8:     $S, A, T = S + [s], A + [\tilde{a}], T + [len(A)]$
9: end while
```

4 Experimental Setup

4.1 Environments

To evaluate the capability of our T-BAD framework to detect anomalous discrete actions or continuous actions, we conduct experiments on Atari and Mujoco, respectively. In Atari, we choose Pong, Freeway, and Road Runner as experimental environments for discrete actions. In Mujoco, we choose Hopper, Humanoid, and Walker2D as experimental environments for continuous actions.

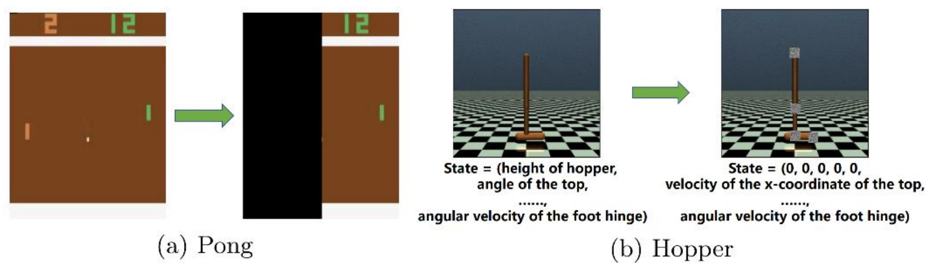

State = (height of hopper,
angle of the top,
......,
angular velocity of the foot hinge)
State = (0, 0, 0, 0, 0,
velocity of the x-coordinate of the top,
......,
angular velocity of the foot hinge)
(a) Pong
(b) Hopper

Fig. 2. Examples of Removing Features in Pong and Hopper. In Pong, the left half of the screen will be removed. In Hopper, the first five features will be set to zero.

4.2 Algorithms and Compared Methods

In the discrete environment, we employ the Deep Q-Network (DQN) algorithm [24] and in the continuous environment, the adopted DRL algorithm is Proximal Policy Optimization (PPO) [25]. The trajectories used to train the transformer are also generated by these respective algorithms. To demonstrate the superiority of our framework, we compare it against Decision Transformer (DT) [10] and Imitation Learning (IL) [26]. The context length used by DT is aligned with that of T-BAD. The Returns-To-Go is set to the maximum episode reward achievable by the agent within the training data.

4.3 Anomalous Observations

We test the ability of different methods to detect anomalous states in Atari, divided into four types: PGD (black-box), FGSM (black-box), Removing Features (RF), and Random Noise (RN). The black-box attacks PGD and FGSM are generated by an additional DQN network, with a small perturbation $\epsilon = 0.004$. Removing features means removing half of the state, where only the right half is displayed, and the remaining pixels are set to 0, as shown in **Fig. 2** (a). The range of random noise is [-30, 31], and the noisy images are clipped to ensure pixel values are within [0, 255].

Similarly, we access the ability of different methods to detect anomalous states in Mujoco, also divided into four types: Critic, Maximal Action Difference (MAD), Removing Features, and Random Noise. The Critic method uses the gradient of $Q(s, a)$ to adversarially update states over N steps. MAD identifies an adversarial state $\tilde{s}$ by maximizing $D_{KL}\big(\pi(\cdot|\, s) \parallel \pi(\cdot|\, \tilde{s})\big)$. In the Removing Features scenario, depicted in **Fig. 2** (b), the first five features of the state are set to 0 for Walker2D and Hopper, and the first 20 features are set to 0 for Humanoid. The random noise is introduced within the range of [-1, 1].

5 Experiments

We conducted four sets of experiments to evaluate the detection efficiency of T-BAD. The first set presents our main results, assessing T-BAD's detection success rates in both discrete and continuous action environments under four different types of attacks or disturbances. The second set compares T-BAD with other baseline methods, including Decision Transformer and Imitation Learning. The third set examines the ability of T-BAD and Decision Transformer to correct anomalous actions when agents are under attack. The fourth set evaluates whether T-BAD's detection performance is influenced by varying levels of perturbation.

Table 1. Testing the ability of T-BAD to detect anomalies in Atari. We report the average value over 10 random seeds, with the standard errors in the bracket. Mismatch refers to the number of times the action predicted by T-BAD does not match the action of the agent when it is not under attack. Attack Frequency refers to the number of times the attack changes the agent's action. Detection Failure is the number of times that the T-BAD's action is also attacked.

Environment	Mismatch	Attack	Attack Frequency	Detection Failure	Success Rate
Pong	0.2 (0.6)	PGD	298.5 (60.4)	0.1 (0.3)	99.93 (0.08)
		FGSM	330.4 (48.5)	0.0 (0.0)	100.00 (0.00)
		RF	1281.0 (140.4)	0.1 (0.3)	99.99 (0.02)
		RN	1209.8 (136.3)	0.1 (0.3)	99.99 (0.03)
Freeway	7.0 (5.4)	PGD	169.2 (13.7)	0.3 (0.6)	99.82 (0.20)
		FGSM	217.6 (18.7)	0.6 (0.7)	99.72 (0.31)
		RF	1082.1 (44.8)	1.4 (1.3)	99.87 (0.12)
		RN	649.6 (21.5)	1.3 (1.0)	99.80 (0.16)
Road Runner	1.4 (1.1)	PGD	290.2 (42.0)	1.6 (1.0)	99.45 (0.32)
		FGSM	307.2 (56.9)	1.6 (1.5)	99.48 (0.43)
		RF	1923.3 (226.5)	268.1 (34.2)	86.06 (4.16)
		RN	1015.6 (143.9)	56.2 (20.4)	94.47 (5.21)

5.1 Main Results

Regardless of the adversarial attack type, the primary objective is to alter the agent's actions, thereby impacting its performance. The agent's actions can be categorized into discrete and continuous types. During the experiments, adversarial algorithms are applied at each timestep, while in the detection for timestep t, previous timesteps are assumed to be unaffected by attacks. For the timestep t, the inputs are updated to $A_t = (a_{t-k+1}, \dots, a_{t-1})$ and $S_t = (s_{t-k+1}, \dots, s_{t-1}, \widetilde{s_t})$, where $\widetilde{s_t}$ is the state at timestep t that may be affected and it causes the agent to output an anomalous action $\widetilde{a_t}$. These experiments are designed to test the success rate of detecting anomalous actions. In practical application, once our T-BAD has detected an anomaly, appropriate measures can be implemented within the agent system to mitigate potential risks.

Discrete Action. As shown in **Table 1**, we select Pong, Freeway, and Road Runner as our experimental environments and each set of data is the mean and standard deviation of ten trials. First, we evaluate the match between the actions predicted by T-BAD and those taken by the agent, and it is evident that mismatches by T-BAD are rare. Furthermore, T-BAD demonstrates a consistently high detection success rate, generally reaching 99% across various scenarios. While the detection performance against the Removing Features attack in the Road Runner environment was comparatively lower at 86%, further analysis reveals that the rapid emergence of hazards on the desert highway necessitates swift decision-making by the agent. In such cases, historical trajectory data offers limited predictive value, and thus, the removal of half of the state significantly impacts T-BAD's detection capability.

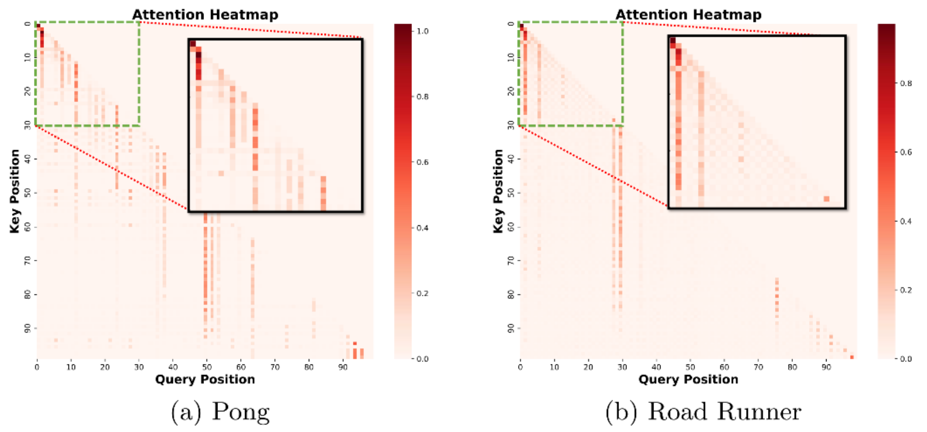


Fig. 3. Heatmap of attention layer in T-BAD. T-BAD has a total of 99 tokens, comprising 50 states and 49 actions. The first token represents the current state.

Fig. 3 (b) shows that attention weights are concentrated primarily in the top left corner and along the diagonal. This pattern suggests that in the Road Runner game, the attention mechanism focuses more on the recent states and actions (query and key positions that are close to each other). Similar to Road Runner, as shown in **Fig. 3** (a), the attention weights in Pong are more concentrated along the diagonal. This indicates that in the Pong game, the attention mechanism also tends to focus on recent states and actions, but there might be slightly more dispersion in the attention weights compared to Road Runner. That's why T-BAD performs poorly in detecting anomalous states in Road Runner. T-BAD cannot effectively utilize the context in Road Runner, as it focuses too much on the current state. Consequently, when the current state is attacked, T-BAD has a high probability of failing to detect the attack.

Continuous Action. We choose Humanoid, Walker2D, and Hopper as experimental environments to evaluate continuous action spaces. Unlike discrete actions, where equivalence can be straightforwardly determined, continuous actions introduces additional complexity. To address this, we establish a threshold based on the loss during T-BAD training to determine whether the agent's actions are altered by an attack. This threshold is defined as the average loss over the last L training rounds. Although L is a hyperparameter, its value does not significantly impact the experimental results. The agent is considered to be under attack if the variance between the actions output by T-BAD and those executed by the agent exceeds this threshold. As shown in **Table 2**, T-

BAD consistently achieved a detection success rate of 90% or higher across all attack methods, with the exception of the Removing Features attack in the Walker2D environment.

Table 2. Testing the ability of T-BAD to detect anomalies in Mujoco.

Environment	Mismatch	Attack	Attack Frequency	Detection Failure	Success Rate
Humanoid	22.6 (7.5)	Critic	834.5 (115.5)	14.9 (6.5)	98.21 (2.15)
		MAD	884.4 (106.5)	4.7 (1.2)	99.47 (0.42)
		RF	113.6 (14.9)	11.6 (4.6)	89.79 (6.59)
		RN	618.6 (54.2)	32.1 (8.7)	94.81 (5.43)
Walker2D	2.5 (1.0)	Critic	957.2 (84.0)	28.6 (5.8)	97.01 (3.91)
		MAD	974.4 (121.6)	50.6 (6.2)	94.82 (4.30)
		RF	849.8 (67.3)	175.5 (13.8)	79.34 (7.27)
		RN	894.8 (51.6)	103.7 (11.7)	88.41 (10.06)
Hopper	0.5 (0.6)	Critic	825.0 (71.3)	28.4 (6.6)	96.56 (2.35)
		MAD	886.8 (90.2)	5.3 (3.0)	99.40 (0.61)
		RF	866.3 (100.1)	64.8 (8.1)	92.52 (4.29)
		RN	694.7 (58.3)	54.1 (7.2)	94.21 (2.48)

5.2 Compared with DT and IL

Table 3 provides a comparative analysis of the detection success rates of T-BAD, Decision Transformer (DT), and Imitation Learning (IL) methods under PGD, FGSM, Removing Features, and Random Noise attacks, as well as the number of mismatched actions compared to an unperturbed agent. T-BAD achieves the highest detection success rate across all attack scenarios. Additionally, in all three environments, T-BAD demonstrates the fewest mismatched actions compared with DT and IL. Although DT can achieve a detection rate of over 90% in many cases, it exhibits a much higher number of mismatched actions. In practical applications of anomaly detection using DT, it is challenging to distinguish between “a mismatch between DT’s predicted action and the agent’s actual action” and “the agent being under attack”, since the agent’s intended action in the absence of an attack remains unknown. Therefore, T-BAD offers a more practical approach than DT for detecting anomalies, as further corroborated by subsequent experiments.

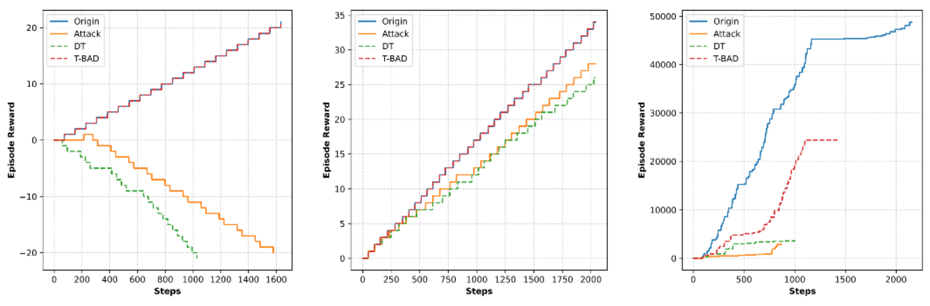
5.3 Correct Anomalous Actions

In this part of experiments, we assess the performance of the agent after using T-BAD and DT to correct anomalous actions they have detected, i.e. making the agent take the output actions of T-BAD and DT. The attack used is removing half of the state, with an attack frequency set at every five steps. As shown in **Fig. 4**, T-BAD effectively detects and corrects anomalous actions in both Pong and Freeway. In Road Runner, T-BAD demonstrated superior mitigation of the attack’s impact compared to DT. Interestingly,

Table 3. Comparing the success rates of anomaly detection in Atari using T-BAD, DT, and IL.

Env	Method	PGD	FGSM	RF	RN
Pong	T-BAD	99.93 (0.08)	100.00 (0.00)	99.99 (0.02)	99.99 (0.03)
	DT	99.73 (0.45)	99.88 (0.26)	99.84 (0.21)	99.86 (0.13)
	IL	99.87 (0.16)	99.94 (0.10)	80.26 (8.82)	82.47 (6.49)
Free-way	T-BAD	99.82 (0.20)	99.72 (0.31)	99.87 (0.12)	99.80 (0.16)
	DT	90.58 (2.47)	91.89 (5.18)	95.19 (3.64)	96.46 (1.43)
	IL	88.26 (6.26)	88.73 (8.80)	84.66 (7.12)	77.62 (2.48)
Road Runner	T-BAD	99.45 (0.32)	99.48 (0.43)	86.06 (4.16)	94.47 (5.21)
	DT	98.94 (1.20)	95.50 (3.44)	80.71 (9.79)	83.28 (7.54)
	IL	88.30 (6.69)	85.79 (5.81)	73.00 (6.35)	69.46 (9.03)

in Freeway and Pong, the performance with DT corrections is even worse than under attack. This is attributed that DT corrects the agent's actions whenever a mismatch is detected, regardless of whether the agent's actions are truly attacked. As a result, such "corrections" may inadvertently serve as an additional form of attack, degrading the agent's performance.



(a) Pong (b) Freeway (c) Road Runner

Fig. 4. Testing the ability of T-BAD to correct anomalous actions in Atari. The blue line represents the episode reward of the agent over steps without any interference. The orange one shows the performance of the agent under attack. The red and green lines depict the performance of the attacked agent after action correction by T-BAD and DT, respectively.

5.4 PGD Attacks with Different Epsilon Values

In this subsection, we evaluate the detection performance of our method under PGD attacks across different epsilon values. As shown in **Table 4**, the experiments were conducted in Pong, Freeway, and Road Runner. For each environment, we measure three metrics: Attack Frequency, Detection Failure, and Success Rate (detection success rate) under varying levels of adversarial perturbation. The results indicate that our detection method is generally effective across a range of PGD attack strengths, maintaining a high success rate and low detection failure rate in most scenarios. Notably, the Road Runner environment is more susceptible to higher epsilon values, with a slight drop in success rate. As discussed in Section 5.1, the RoadRunner environment has substantial frame-to-frame variations, causing the Transformer model to focus more on the current state and action. Consequently, when the current state is under attack, detection is more likely to fail.

Table 4. Detection performance of our method under PGD attacks with varying epsilon values in different environments.

Env	Index	ε= 0.001	ε= 0.0025	ε= 0.004	ε=0.0055	ε= 0.007
Pong	Attack Frequency	81.0	200.6	298.5	383.3	476.7
	Detection Failure	0.1	0.2	0.2	0.3	0.3
	Success Rate(%)	99.87	99.90	99.93	99.92	99.93
Freeway	Attack Frequency	50.0	114.4	169.2	214.8	251.9
	Detection Failure	0.1	0.2	0.3	0.3	0.4
	Success Rate(%)	99.80	99.83	99.82	99.86	99.84
Road Runner	Attack Frequency	99.3	200.4	290.2	374.2	442.0
	Detection Failure	0.4	0.8	1.6	2.5	6.1
	Success Rate(%)	99.60	99.60	99.45	99.33	98.62

6 Conclusion

In this paper, we propose a Transformer-Based Anomaly Detection (T-BAD) framework that leverages the robustness of transformer-based sequence inputs for real-time detection of anomalous states and actions. Our experiments conducted in both Atari and Mujoco environments demonstrate that T-BAD achieves a high success rate in anomaly detection even without prior knowledge of adversarial examples, making it a versatile tool in various settings. We compared T-BAD with decision transformer and imitation learning, and found that T-BAD consistently outperforms these baselines in detecting anomalies. Additionally, T-BAD shows potential in correcting detected anomalies, which is a promising direction for enhancing the reliability of reinforcement learning systems. Our future work will aim to further improve T-BAD's capabilities and extend its application to multi-agent reinforcement learning scenarios, where the complexity of detection and correction can be more challenging.

Acknowledgments. This work is supported by the Ministry of Public Security Science and Technology Plan, Project Number 2023JSZ01 and the Provincial Key Research and Development Program of Anhui, Project Number 202423l10050033.

References

1. Chen, Z., Liao, Y., Zhao, Y., Dai, Z., Zhao, J.: CUDA2: An approach for incorporating traitor agents into cooperative multi-agent systems. *IEEE Transactions on Games* (2024)
2. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013)
3. McMahan, J., Wu, Y., Zhu, X., Xie, Q.: Optimal attack and defense for reinforcement learning. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 14332–14340 (2024)
4. Kos, J., Song, D.: Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452* (2017)
5. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017)
6. Pinto, L., Davidson, J., Sukthankar, R., Gupta, A.: Robust adversarial reinforcement learning. In: *International Conference on Machine Learning*, pp. 2817–2826 (2017)
7. Korkmaz, E.: Adversarial robust deep reinforcement learning requires redefining robustness. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 8369–8377 (2023)
8. Gourdeau, P., Kanade, V., Kwiatkowska, M., Worrell, J.: On the hardness of robust classification. *Journal of Machine Learning Research* 22(273), 1–29 (2021)
9. Korkmaz, E., Brown-Cohen, J.: Detecting adversarial directions in deep reinforcement learning to make robust decisions. In: *International Conference on Machine Learning*, pp. 17534–17543 (2023)
10. Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., Mordatch, I.: Decision transformer: Reinforcement learning via sequence modeling. *Advances in Neural Information Processing Systems* 34, 15084–15097 (2021)
11. Zhang, X., Ma, Y., Singla, A., Zhu, X.: Adaptive reward-poisoning attacks against reinforcement learning. In: *International Conference on Machine Learning (ICML)*, pp. 11225–11234 (2020)
12. Mo, K., Tang, W., Li, J., Yuan, X.: Attacking deep reinforcement learning with decoupled adversarial policy. *IEEE Transactions on Dependable and Secure Computing* 20(1), 758–768 (2022)
13. Li, Y., Pan, Q., Cambria, E.: Deep-attack over the deep reinforcement learning. *Knowledge-Based Systems* 250, 108965 (2022)
14. Sun, J., Zhang, T., Xie, X., Ma, L., Zheng, Y., Chen, K., Liu, Y.: Stealthy and efficient adversarial attacks against deep reinforcement learning. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 5883–5891 (2020)
15. Bai, X., Niu, W., Liu, J., Gao, X., Xiang, Y., Liu, J.: Adversarial examples construction towards white-box Q table variation in DQN pathfinding training. In: *International Conference on Data Science in Cyberspace (DSC)*, pp. 781–787 (2018)
16. Lee, X.Y., Ghadai, S., Tan, K.L., Hegde, C., Sarkar, S.: Spatiotemporally constrained action space attacks on action reinforcement learning agents. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 4577–4584 (2020)
17. Ilahi, I., Usama, M., Qadir, J., Janjua, M.U., Al-Fuqaha, A., Hoang, D.T., Niyato, D.: Challenges and countermeasures for adversarial attacks on deep reinforcement learning. *IEEE Transactions on Artificial Intelligence* 3(2), 90–109 (2021)
18. Ma, X., Li, B., Wang, Y., Erfani, S.M., Wijewickrema, S., Schoenebeck, G., Song, D., Houle, M.E., Bailey, J.: Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613* (2018)

19. Nesti, F., Biondi, A., Buttazzo, G.: Detecting adversarial examples by input transformations, defense perturbations, and voting. *IEEE Transactions on Neural Networks and Learning Systems* 34(3), 1329–1341 (2021)
20. Liang, B., Li, H., Su, M., Li, X., Shi, W., Wang, X.: Detecting adversarial image examples in deep neural networks with adaptive noise reduction. *IEEE Transactions on Dependable and Secure Computing* 18(1), 72–85 (2018)
21. Tian, J., Zhou, J., Li, Y., Duan, J.: Detecting adversarial examples from sensitivity inconsistency of spatial-transform domain. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 9877–9885 (2021)
22. Li, X., Li, F.: Adversarial examples detection in deep networks with convolutional filter statistics. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5764–5772 (2017)
23. Wang, J., Zhao, J., Yin, Q., Luo, X., Zheng, Y., Shi, Y., Jha, S.K.: Smsnet: A new deep convolutional neural network model for adversarial example detection. *IEEE Transactions on Multimedia* 24, 230–244 (2021)
24. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013)
25. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017)
26. Fischer, M., Mirman, M., Stalder, S., Vechev, M.: Online robustness training for deep reinforcement learning. *arXiv preprint arXiv:1911.00887* (2019)