Robust Q-Learning against State Perturbations: a Belief-Enriched Pessimistic Approach

Anonymous Author(s)
Affiliation
Address
email

Abstract

Reinforcement learning (RL) has achieved phenomenal success in various domains. However, its data-driven nature also introduces new vulnerabilities that can be exploited by malicious opponents. Recent work shows that a well-trained RL agent can be easily manipulated by strategically perturbing its state observations at the test stage. Existing solutions either introduce a regularization term to improve the smoothness of the trained policy against perturbations or alternatively train the agent's policy and the attacker's policy. However, the former does not provide sufficient protection against strong attacks, while the latter is computationally prohibitive for large environments. In this work, we propose a new robust RL algorithm for deriving a pessimistic policy to safeguard against an agent's uncertainty about true states. This approach is further enhanced with belief state inference and diffusion-based state purification to reduce uncertainty. Empirical results show that our approach obtains superb performance under strong attacks and has a comparable training overhead with regularization-based methods.

1 Introduction

As one of the major paradigms for data-driven control, reinforcement learning (RL) provides a principled and solid framework for sequential decision-making under uncertainty. However, an RL agent is subject to various types of attacks, including state and reward perturbation, action space manipulation, and model inference and poisoning [16]. Recent studies have shown that an RL agent can be manipulated by poisoning its observation [14, 33] and reward signals [15], and a well-trained RL agent can be easily defeated by a malicious opponent behaving unexpectedly [8]. In particular, recent research has demonstrated the brittleness [33, 27] of existing RL algorithms in the face of adversarial state perturbations, where a malicious agent strategically and stealthily perturbs the observations of a trained RL agent, causing a significant loss of cumulative reward.

Several solutions have been proposed to combat state perturbation attacks. SA-MDP [33] imposes a regularization term in the training objective to improve the smoothness of the learned policy under state perturbations. This approach is improved in WocaR-RL [20] by incorporating an estimate of the worst-case reward under attacks into the training objective. In a different direction, ATLA [32] alternately trains the agent's and the attacker's policy. This approach can potentially lead to a more robust policy but incurs high computational overhead, especially for large environments such as Atari games with raw pixel observations.

Despite their promising performance in certain RL environments, the above solutions have two major limitations. First, actions are directly derived from a value or policy network trained using true states, despite the fact that the agent can only observe perturbed states at the test stage. This mismatch between the training and testing leads to unstable performance at the test stage. Second, most existing work does not exploit historical observations and the agent's knowledge about the underlying MDP model to characterize and reduce uncertainty and infer true states in a systematic way.

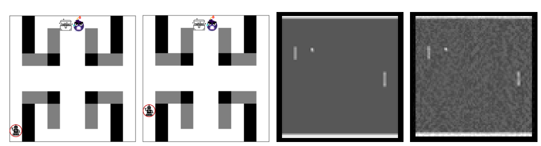
In this work, we propose a pessimistic DQN algorithm against state perturbations by viewing the defender's problem as finding an approximate Stackelberg equilibrium for a two-player Markov game with asymmetric observations. Given a perturbed state, the agent selects an action that maximizes the worst-case value across possible true states. This approach is applied at both training and test stages, thus removing the inconsistency between the two. We further propose two approaches to reduce the agent's uncertainty about true states. First, the agent maintains a belief about the actual state using historical data, which, together with the pessimistic approach, provides a strong defense against large perturbations that may change the semantics of states. Second, for games with raw pixel input, such as Atari games, we train a diffusion model using the agent's knowledge about valid states. This approach provides superb performance under commonly used attacks, with the additional advantage of being agnostic to the perturbation level. Our method achieves high robustness and significantly outperforms existing solutions under strong attacks while maintaining comparable performance under relatively weak attacks. Further, its training complexity is comparable to SA-MDP and WocaR-RL and is much lower than alternating training-based approaches.

2 Background

2.1 Reinforcement Learning

A reinforcement learning environment is usually formulated as a Markov Decision Process (MDP), denoted by a tuple $\langle S, A, P, R, \gamma\rangle$, where S is the state space and A is the action space. $P : S \times A \to \Delta(S)$ is the transition function of the MDP, where $P(s'|s,a)$ gives the probability of moving to state s' given the current state s and action a. $R : S \times A \to \mathbb{R}$ is the reward function where $R(s,a) = \mathbb{E}(R_t|s_{t-1} = s, a_{t-1} = a)$ and R_t is the reward in time step t. Finally, γ is the discount factor. An RL agent wants to maximize its cumulative reward $G = \Sigma_{t=0}^{T}\gamma^t R_t$ over a time horizon $T \in \mathbb{Z}^+ \cup \{\infty\}$, by finding a (stationary) policy $\pi : S \to \Delta(A)$, which can be either deterministic or stochastic. For any policy π, the state-value and action-value functions are two standard ways to measure how good π is. The state-value function satisfies the Bellman equation $V_\pi(s) = \Sigma_{a\in A}\pi(a|s)[R(s,a) + \gamma\Sigma_{s'\in S}P(s'|s,a)V_\pi(s')]$ and the action-value function satisfies $Q_\pi(s,a) = R(s,a) + \gamma\Sigma_{s'\in S}P(s'|s,a)[\Sigma_{a'\in A}\pi(a'|s')Q_\pi(s',a')]$. For MDPs with a finite or countably infinite state space and a finite action space, there is a deterministic and stationary policy that is simultaneously optimal for all initial states s.

2.2 State Adversarial Attacks in RL



(a) Original (b) Perturbed (c) Original (d) Perturbed

Figure 1: Examples of perturbed states : (a) and (b) show states in a continuous state Gridworld, and (c) and (d) show states in the Atari Pong game.

First introduced in [14], a state perturbation attack is a test stage attack targeting an agent with a well-trained policy π. At each time step, the attacker observes the true state s_t and generates a perturbed state $\tilde{s}_t$ (see Figure 1 for examples). The agent observes $\tilde{s}_t$ but not s_t and takes an action a_t according to $\pi(\cdot|\tilde{s}_t)$. The attacker's goal is to minimize the cumulative reward that the agent obtains. Note that the attacker only interferes with the agent's observed state but not the underlying MDP. Thus, the true state in the next time step is distributed according to $P(s_{t+1}|s_t, \pi(\cdot|\tilde{s}_t))$. To limit the attacker's capability and avoid being detected, we assume that $\tilde{s}_t \in B_\epsilon(s_t)$ where $B_\epsilon(s_t)$ is the l_p ball centered at s_t for some norm p. We consider a strong adversary that has access to both the MDP and the agent's policy π and can perturb at every time step. With these assumptions, it is easy to see that the attacker's problem given a fixed π can also be formulated as an MDP $\langle S, S, \tilde{P}, \tilde{R}, \gamma\rangle$, where both the state and action spaces are S, the transition probability $\tilde{P}(s'|s,\tilde{s}) = \sum_a \pi(a|\tilde{s})P(s'|s,a)$, and reward $\tilde{R}(s,\tilde{s}) = -\sum_a \pi(a|\tilde{s})R(s,a)$. Thus, an RL algorithm can be used to find a (nearly) optimal attack policy. Further, we adopt the common assumption [33, 20] that the agent has access to an intact MDP at the training stage and has access to ϵ (or an estimation of it). As we discuss below, our diffusion-based approach is agnostic to ϵ. Detailed discussions of related work on attacks and defenses in RL, including and beyond state perturbation, are in Appendix B.

3 Pessimistic Q-learning with State Inference and Purification

In this section, we give an overview of our game formulation and algorithmic solutions. Details can be found in Appendix C.

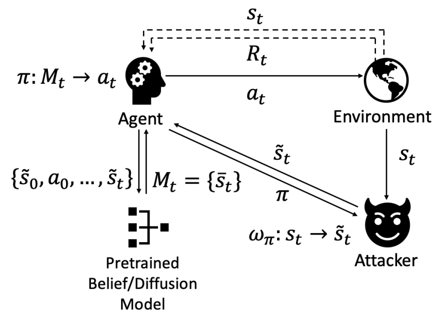


Figure 2: Belief-enriched robust RL against state perturbations. Note that the agent can only access the true state s_t and reward R_t at the training stage.

State-Adversarial MDP as a Stackelberg Markov Game with Asymmetric Observations. The problem of robust RL under adversarial state perturbations can be viewed as a two-player Markov game. The RL agent wants to find a policy $\pi : S \to \Delta(A)$ that maximizes its long-term return, while the attacker wants to find an attack policy $\omega : S \to S$ to minimize the RL agent's cumulative reward. The agent's value function for a given pair of policies π and ω satisfies the Bellman equation as $Q_{\pi\circ\omega}(s,a) = R(s,a) + \gamma\Sigma_{s'\in S}P(s'|s,a)[\Sigma_{a'\in A}\pi(a'|\omega(s'))Q_{\pi\circ\omega}(s',a')]$. We can consider a Stackelberg equilibrium by viewing the RL agent as the leader and the attacker as the follower to gain robustness. The agent first commits to a policy π. The attack observes π and identifies an optimal attack, denoted by ω_π, as a response, where $\omega_\pi(s) = \text{argmin}_{\tilde{s}\in B_{\epsilon(s)}}\Sigma_{a'\in A}\pi(a'|\tilde{s})Q(s,a')$. Ideally, the agent wants to find a policy π^* that reaches a Stackleberg equilibrium of the game, which is defined as $\forall s \in S, \forall \pi, V_{\pi^*\circ\omega_{\pi^*}}(s) \geq V_{\pi\circ\omega_\pi}(s)$. However, previous work has shown that due to the noisy observations, finding a stationary policy that is optimal for every initial state is generally impossible [33]. Thus, our goal is to find an approximate Stackelberg equilibrium, which is further improved through state prediction and denoising (see Figure 2 for the overall framework of our approach).

Pessimistic Q-learning Against the Worst Case. In this work, we present a pessimistic Q-learning algorithm (see Algorithm 1 in Appendix C.2) to address the asymmetric observations. The algorithm maintains a Q-function with the true state as the input, similar to vanilla Q-learning. But instead of using a greedy approach to derive the target policy or a ϵ-greedy approach to derive the behavior policy from the Q-function, a $\mathrm{maximin}$ approach is used in both cases. Figure 3 in the Appendix D illustrates the relations between a true state s, the perturbed state $\tilde{s}$, the worst-case state $\bar{s} \in B_\epsilon(\tilde{s})$ for which the action is chosen. In particular, it shows that the true state s must land in the ϵ-ball centered at $\tilde{s}$, and the worst-case state the RL agent envisions is at most 2ϵ away from the true state. This gap causes performance loss that will be studied in Appendix C.6.

Reducing Uncertainty Using Beliefs. In Algorithm 1, the agent's uncertainty against the true state is captured by the ϵ-ball around the perturbed state. The agent can utilize the sequence of historical observations and actions $\{(\tilde{s}_\tau, a_\tau)\}_{\tau<t} \cup \{\tilde{s}_t\}$ and the transition dynamics of the underlying MDP to reduce its uncertainty of the current true state s_t. To this end, we propose a simple approach to reduce the agent's worst-case uncertainty as follows. Let $M_t \subseteq B_\epsilon(\tilde{s}_t)$ denote the agent's belief about all possible true states at time step t. Initially, we let $M_0 = B_\epsilon(\tilde{s}_0)$. At the end of the time step t, we update the belief to include all possible next states that is reachable from the current state and action with a non-zero probability. Formally, let $M'_t = \{s' \in S : \exists s \in M_t, P(s'|s,a_t) > 0\}$. After observing the perturbed state $\tilde{s}_{t+1}$, we then update the belief to be the intersection of M'_t and $B_\epsilon(\tilde{s}_{t+1})$, i.e., $M_{t+1} = M'_t \cap B_\epsilon(\tilde{s}_{t+1})$, which gives the agent's belief at time $t+1$. When the state space is high-dimensional and continuous, computing the accurate belief is particularly hard. To deal with this, we adapt the particle filter recurrent neural network (PF-RNN) technique developed in [21] to generate belief M_t under high-dimensional state space.

Purifying Invalid Observations via Diffusion. For environments that use raw pixels as states, such as Atari Games, perturbed states generated by adding bounded noise to each pixel are mostly "invalid" in the following sense. Let $S_0 \subseteq S$ denote the set of possible initial states. Let S^0 denote the set of states that are reachable from any initial state in S_0 by following an arbitrary policy. Then perturbed states will fall outside of S^0 with high probability. This is especially the case for l_∞ attacks that bound the perturbation applied to each pixel as commonly assumed in existing work (see Appendix D.2 for an example). We choose to utilize a diffusion model [12, 26] to purify the perturbed states, which obtains promising performance, as we show in our empirical results. A more detailed description of the diffusion models and our adaptations are given in Appendix B.6.

4 Experiments

We develop two pessimistic versions of the classic DQN algorithm [22] by incorporating approximate belief update and diffusion-based purification, denoted by BP-DQN and DP-DQN (see Algorithms 4-7 in Appendix F), respectively and evaluate them by conducting experiments on three environments, a continuous state Gridworld environment (shown in Figure 1a) for BP-DQN and two Atari games, Pong and Freeway for DP-DQN-O and DP-DQN-F, which utilize DDPM [12] and Progressive Distillation [26] as the diffusion model, respectively. (See Appendix G.1 for a justification.) We choose vanilla DQN [23], SA-DQN [33], and WocaR-DQN [20] as defense baselines. We consider three commonly used attacks to evaluate the robustness of these algorithms: (1) PGD attack [33]; (2) MinBest attack [14]; and (3) PA-AD [27]. Details on the experiment setup can be found in Appendix G.2. Additional experiment results and ablation studies are given in Appendix G.3.

Environment	**Model**	**Natural Reward**	**PGD**		**MinBest**	
			$\epsilon = 0.1$	$\epsilon = 0.5$	$\epsilon = 0.1$	$\epsilon = 0.5$
Continuous Gridworld	DQN	156.5 ± 90.2	128 ± 118	-53 ± 86	98.2 ± 137	-82 ± 20
	SA-DQN	20.8 ± 140	46 ± 142	-100 ± 0	-5.8 ± 131	-100 ± 0
	WocaR-DQN	-100 ± 0	-100 ± 0	-63.2 ± 88	-100 ± 0	-63.2 ± 88
	BP-DQN (Ours)	163 ± 26	165 ± 29	176 ± 16	147 ± 88	114 ± 114

(a) Continuous Gridworld Results

Env	**Model**	**Natural Reward**	**PGD**			**MinBest**			**PA-AD**		
			$\epsilon = 1/255$	$\epsilon = 3/255$	$\epsilon = 15/255$	$\epsilon = 1/255$	$\epsilon = 3/255$	$\epsilon = 15/255$	$\epsilon = 1/255$	$\epsilon = 3/255$	$\epsilon = 15/255$
Pong	DQN	21 ± 0	-21 ± 0	-21 ± 0	-21 ± 0	-21 ± 0	-21 ± 0	-21 ± 0	-18.2 ± 2.3	-19 ± 2.2	-21 ± 0
	SA-DQN	21 ± 0	21 ± 0	21 ± 0	-20.8 ± 0.4	21 ± 0	21 ± 0	-21 ± 0	21 ± 0	18.7 ± 2.6	-20 ± 0
	WocaR-DQN	21 ± 0	21 ± 0	21 ± 0	-21 ± 0	21 ± 0	21 ± 0	-21 ± 0	21 ± 0	19.7 ± 2.4	-21 ± 0
	DP-DQN-O(Ours)	19.9 ± 0.3	19.9 ± 0.3	19.8 ± 0.4	19.7 ± 0.5	19.9 ± 0.3	19.9 ± 0.3	19.3 ± 0.8	19.9 ± 0.3	19.9 ± 0.3	19.3 ± 0.8
	DP-DQN-F (Ours)	21 ± 0	20.4 ± 0.7	20.2 ± 0.8	18.6 ± 1	20.2 ± 0.9	19.0 ± 0	19.3 ± 1.6	18.0 ± 1.0	17.6 ± 1.8	17 ± 2.3
Freeway	DQN	34 ± 0.1	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0
	SA-DQN	30 ± 0	30 ± 0	30 ± 0	0 ± 0	27.2 ± 3.4	18.3 ± 3.0	0 ± 0	20.1 ± 4.0	9.5 ± 3.8	0 ± 0
	WocaR-DQN	31.2 ± 0.4	31.2 ± 0.5	31.4 ± 0.3	21.6 ± 1	29.6 ± 2.5	19.8 ± 3.8	21.6 ± 1	24.9 ± 3.7	12.3 ± 3.2	21.6 ± 1
	DP-DQN-O (Ours)	28.8 ± 1.1	29.1 ± 1.1	29 ± 0.9	28.9 ± 0.7	29.2 ± 1.0	28.5 ± 1.2	28.6 ± 1.3	28.6 ± 1.2	28.3 ± 1	28.8 ± 1.3
	DP-DQN-F (Ours)	31.2 ± 1	30 ± 0.9	30.1 ± 1	30.7 ± 1.2	30.2 ± 1.3	30.6 ± 1.4	29.4 ± 1.2	30.8 ± 1	31.4 ± 0.8	28.9 ± 1.1

(b) Atari Games Results

Table 1: Experiment Results. We show the average episode rewards $\pm$ standard deviation over 10 episodes for our methods and three baselines. The results for our methods are highlighted in gray.

4.1 Results and Discussion

Continuous Gridworld. As shown in Table 1a, our method (BP-DQN) achieves the best performance under all scenarios In contrast, both SA-DQN and WocaR-DQN fail under the large attack budget $\epsilon = 0.5$ and perform poorly under the small attack budget $\epsilon = 0.1$. We conjecture that this is because state perturbations in the continuous Gridworld environment often change the semantics of states since most perturbed states are still valid observations. We also noticed that both SA-DQN and WocaR-DQN perform worse than vanilla DQN when there is no attack and when $\epsilon = 0.1$. We conjecture that this is due to the mismatch between true states and perturbed states during training and testing and the approximation used to estimate the upper and lower bounds of Q-network output using the Interval Bound Propagation (IBP) technique [9] in their implementations.

Atari Games. As shown in Table 1b, our DP-DQN method outperforms all other baselines under a strong attack (e.g., PA-AD) or a large attack budget (e.g., $\epsilon = 15/255$), while achieving comparable performance as other baselines in other cases. SA-DQN and WocaR-DQN fail to respond to large state perturbations for two reasons. First, both of them use IBP to estimate an upper and lower bound of the neural network output under perturbations, which are likely to be loose under large perturbations. Second, both approaches utilize a regularization-based approach to maximize the chance of choosing the best action for all states in the ϵ-ball centered at the true state. This approach is effective under small perturbations but can pick poor actions for large perturbations as the latter can easily exceed the generalization capability of the Q-network. We observe that WocaR-DQN performs better when the attack budget increases from $3/255$ to $15/255$ in Freeway. The reason is that under large perturbations, the agent adopts a bad policy by always moving forward regardless of state, which gives a reward of around 21. We admit that our method suffers a small performance loss compared with SA-DQN and WocaR-DQN in the Atari games when there is no attack or when the attack budget is low. We conjecture that no single fixed policy is simultaneously optimal against different types of attacks. A promising direction is to adapt a pre-trained policy to the actual attack using samples collected online.

References

[1] Karl J Astrom et al. Optimal control of markov processes with incomplete state information. *Journal of mathematical analysis and applications*, 10(1):174–205, 1965.

[2] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1st edition, 1996.

[3] Shubham Bharti, Xuezhou Zhang, Adish Singla, and Jerry Zhu. Provable defense against backdoor policies in reinforcement learning. *Advances in Neural Information Processing Systems(NeurIPS)*, 2022.

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

[5] Kangjie Chen, Shangwei Guo, Tianwei Zhang, Xiaofei Xie, and Yang Liu. Stealing deep reinforcement learning models for fun and profit. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pages 307–319, 2021.

[6] Xiaoyu Chen, Yao Mark Mu, Ping Luo, Shengbo Li, and Jianyu Chen. Flow-based recurrent belief state learning for pomdps. In *International Conference on Machine Learning(ICML)*, 2022.

[7] Tanner Fiez, Benjamin Chasnov, and Lillian Ratliff. Implicit learning dynamics in stackelberg games: Equilibria characterization, convergence analysis, and empirical study. In *International Conference on Machine Learning(ICML)*, 2020.

[8] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. In *International Conference on Learning Representations(ICLR)*, 2020.

[9] Sven Gowal, Krishnamurthy Dj Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. Scalable verified training for provably robust image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[10] Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations(ICLR)*, 2021.

[11] Sihong He, Songyang Han, Sanbao Su, Shuo Han, Shaofeng Zou, and Fei Miao. Robust multi-agent reinforcement learning with state uncertainty. *Transactions on Machine Learning Research*, 2023.

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arxiv:2006.11239*, 2020.

[13] Mengdi Huai, Jianhui Sun, Renqin Cai, Liuyi Yao, and Aidong Zhang. Malicious attacks against deep reinforcement learning interpretations. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 472–482, 2020.

[14] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. arXiv:1702.02284, 2017.

[15] Yunhan Huang and Quanyan Zhu. Deceptive reinforcement learning under adversarial manipulations on cost signals. In *Decision and Game Theory for Security (GameSec)*, pages 217–237, 2019.

[16] Inaam Ilahi, Muhammad Usama, Junaid Qadir, Muhammad Umar Janjua, Ala Al-Fuqaha, Dinh Thai Hoang, and Dusit Niyato. Challenges and countermeasures for adversarial attacks on deep reinforcement learning. *IEEE Transactions on Artificial Intelligence*, 3(2):90–109, 2022.

[17] Ville Könönen. Asymmetric multiagent reinforcement learning. *Web Intelligence and Agent Systems: An international journal*, 2(2):105–121, 2004.

[18] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems(NeurIPS)*, 2020.

[19] Xian Yeow Lee, Sambit Ghadai, Kai Liang Tan, Chinmay Hegde, and Soumik Sarkar. Spatiotemporally constrained action space attacks on deep reinforcement learning agents. In *Proceedings of the AAAI conference on artificial intelligence(AAAI)*, volume 34, pages 4577–4584, 2020.

[20] Yongyuan Liang, Yanchao Sun, Ruijie Zheng, and Furong Huang. Efficient adversarial training without attacking: Worst-case-aware robust reinforcement learning. In *Advances in Neural Information Processing Systems(NeurIPS)*, 2022.

[21] Xiao Ma, Péter Karkus, David Hsu, and Wee Sun Lee. Particle filter recurrent neural networks. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence(AAAI)*, 2020.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

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

[24] Tuomas Oikarinen, Wang Zhang, Alexandre Megretski, Luca Daniel, and Tsui-Wei Weng. Robust deep reinforcement learning through adversarial loss. *Advances in Neural Information Processing Systems(NeurIPS)*, 2021.

[25] Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. Finding approximate pomdp solutions through belief compression. *Journal of artificial intelligence research*, 23:1–40, 2005.

[26] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations(ICLR)*, 2022.

[27] Yanchao Sun, Ruijie Zheng, Yongyuan Liang, and Furong Huang. Who is the strongest enemy? towards optimal and efficient evasion attacks in deep rl. *arXiv preprint arXiv:2106.05087*, 2021.

[28] Sebastian Tschiatschek, Kai Arulkumaran, Jan Stühmer, and Katja Hofmann. Variational inference for data-efficient model learning in pomdps. *arXiv preprint arXiv:1805.09281*, 2018.

[29] Quoc-Liem Vu, Zane Alumbaugh, Ryan Ching, Quanchen Ding, Arnav Mahajan, Benjamin Chasnov, Sam Burden, and Lillian J Ratliff. Stackelberg policy gradient: Evaluating the performance of leaders and followers. In *ICLR 2022 Workshop on Gamification and Multiagent Solutions*, 2022.

[30] Chaowei Xiao, Zhongzhu Chen, Kun Jin, Jiongxiao Wang, Weili Nie, Mingyan Liu, Anima Anandkumar, Bo Li, and Dawn Song. Densepure: Understanding diffusion models towards adversarial robustness. *arXiv preprint arXiv:2211.00322*, 2022.

[31] Zikang Xiong, Joe Eappen, He Zhu, and Suresh Jagannathan. Defending observation attacks in deep reinforcement learning via detection and denoising. Berlin, Heidelberg, 2023. Springer-Verlag.

[32] Huan Zhang, Hongge Chen, Duane S Boning, and Cho-Jui Hsieh. Robust reinforcement learning on state observations with learned optimal adversary. In *International Conference on Learning Representations(ICLR)*, 2021.

[33] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. In *Advances in Neural Information Processing Systems(NeurIPS)*, 2020.

[34] Xuezhou Zhang, Yuzhe Ma, Adish Singla, and Xiaojin Zhu. Adaptive reward-poisoning attacks against reinforcement learning. In *International Conference on Machine Learning(ICML)*, 2020.

[35] Liyuan Zheng, Tanner Fiez, Zane Alumbaugh, Benjamin Chasnov, and Lillian J Ratliff. Stackelberg actor-critic: Game-theoretic reinforcement learning algorithms. In *Proceedings of the AAAI Conference on Artificial Intelligence(AAAI)*, 2022.

Appendix

A Broader Impacts

As RL is increasingly being used in vital real-world applications like autonomous driving and large generative models, we are rapidly moving towards an AI-assisted society. With AI becoming more widespread, it is important to ensure that the policies governing AI are robust. An unstable policy could be easily exploited by malicious individuals or organizations, causing damage to property, productivity, and even loss of life. Therefore, providing robustness is crucial to the successful deployment of RL and other deep learning algorithms in the real world. Our work provides new insights into enhancing the robustness of RL policies against adversarial attacks and contributes to the foundation of trustworthy AI.

B Related Work

B.1 State Perturbation Attacks and Defenses

State perturbation attacks against RL policies are first introduced in [14], where the MinBest attack that minimizes the probability of choosing the best action is proposed. [33] show that when the agent's policy is fixed, the problem of finding the optimal adversarial policy is also an MDP, which can be solved using RL. This approach is further improved in [27], where a more efficient algorithm for finding the optimal attack called PA-AD is developed.

On the defense side, Zhang et al. [33] prove that a policy that is optimal for any initial state under optimal state perturbation might not exist and propose a set of regularization-based algorithms (SA-DQN, SA-PPO, SA-DDPG) to train a robust agent against state perturbations. This approach is improved in [20] by training an additional worst-case Q-network and introducing state importance weights into regularization. In a different direction, an alternating training framework called ATLA is studied in [32] that trains the RL attacker and RL agent alternatively in order to increase the robustness of the DRL model. However, this approach suffers from high computational overhead. [31] propose an auto-encoder-based detection and denoising framework to detect perturbed states and restore true states. Also, [11] show that when the initial distribution is known, a policy that optimizes the expected return across initial states under state perturbations exists.

B.2 Attacks and Defenses Beyond State Perturbations

This section briefly introduces other types of adversarial attacks in RL beyond state perturbation. As shown in [15], manipulating the reward signal can successfully affect the training convergence of Q-learning and mislead the trained agent to follow a policy that the attacker aims at. Furthermore, an adaptive reward poisoning method is proposed by [34] to achieve a nefarious policy in steps polynomial in state-space size $|S|$ in the tabular setting.

Lee et al. [19] propose two methods for perturbing the action space, where the LAS (look-ahead action space) method achieves better attack performance in terms of decreasing the cumulative reward of DRL by distributing attacks across the action and temporal dimensions. Another line of work investigates adversarial policies in a multi-agent environment, where it has been shown that an opponent adopting an adversarial policy could easily beat an agent with a well-trained policy in a zero-sum game [8].

For attacking an RL agent's policy network, both inference attacks [5], where the attacker aims to steal the policy network parameters, and poisoning attacks [13] that directly manipulate model parameters have been considered. In particular, an optimization-based technique for identifying an optimal strategy for poisoning the policy network is proposed in [13].

B.3 Backdoor Attacks in RL

Recent work investigating defenses against backdoor attacks in RL also considers recovering true states to gain robustness [3]. However, there are important differences between our work and [3]. First, our work contains two important parts that [3] does not have, which are the maximin formulation and belief update. The former allows us to obtain a robust policy by making fewer assumptions about attack behavior compared to [3]. Note that this approach is unique to state-perturbation attacks, as it is difficult to define a worst-case scenario for backdoor attacks. The latter is crucial to combat adaptive perturbations that can change the semantic meaning of states, which can potentially be very useful to backdoor attacks as well. Second, our Lipschitz assumptions differ from those in [3]. We assume that the reward and transition functions of the underlying MDP are Lipschitz continuous while [3] assume that the backdoored policies are Lipschitz continuous.

B.4 Partially Observable MDPs

As first proposed by Åström [1], a Partially Observable MDP is a generalization of an MDP where the system dynamics are determined by an MDP, but the agent does not have full access to the state. The agent could only partially observe the underlying state that is usually determined by a fixed observation function $\mathcal{O}$. POMDPs could model a lot of real life sequential decision-making problems such as robot navigation. However, since the agent does not have perfect information about the state, solutions for POMDPs usually need to infer a belief about the true state and find an action that is optimal for each possible belief. To this end, algorithms for finding a compressed belief space in order to solve large state space POMDPs have been proposed [25]. State-of-the-art solutions approximate the belief states with distributions such as diagonal Gaussian [18], Gaussian mixture [28], categorical distribution [10] or particle filters [21]. Most recently, a flow-based recurrent belief state modeling approach has been proposed in [6] to approximate general continuous belief states.

The main difference between POMDPs and MDPs under state adversarial attacks is the way the agent's observation is determined. In a POMDP, the agent's partial observation at time step t is determined by a fixed observation function $\mathcal{O}$, where $o_t = \mathcal{O}(s_t, a_t)$. And it is independent of the agent's policy π. Instead, in an MDP under state adversarial attacks, a perturbed state $\tilde{s}$ is determined by the attack policy ω, which can adapt to the agent's policy π in general.

B.5 RL for Stackelberg Markov Games

Previous work has studied various techniques for solving the Stackelberg equilibrium of asymmetric Markov games, with one player as the leader and the rest being followers. Kononen [17] proposes an asymmetric multi-agent Q-Learning algorithm and establishes its convergence in the tabular setting. Besides value-based approaches, Fiez et al. [7] recently investigated sufficient conditions for a local Stackelberg equilibrium (LSE) and derived gradient-based learning dynamics for Stackelberg games using the implicit function theorem. Follow-up work applied this idea to derive Stackelberg actor-critic [35] and Stackelberg policy gradient [29] methods. However, all these studies assume that the true state information is accessible to all players, which does not apply to our problem.

B.6 More Details About Diffusion-Based Denoising

In a Denoising Diffusion Probabilistic Model (DDPM) [12], the forward process constructs a discrete-time Markov chain as follows. Given an initial state $\mathbf{x}_0$ sampled from $q(\cdot)$, it gradually adds Gaussian noise to $\mathbf{x}_0$ to generate a sequence of noisy states $\mathbf{x}_1$,$\mathbf{x}_2$,...,$\mathbf{x}_K$ where $q\left(\mathbf{x}_i \mid \mathbf{x}_{i-1}\right) := \mathcal{N}\left(\mathbf{x}_i; \sqrt{1-\beta_i}\mathbf{x}_{i-1}, \beta_i \mathbf{I}\right)$ so that $\mathbf{x}_K$ approximates the Gaussian white noise. Here β_i is precalculated according to a variance schedule and $\mathbf{I}$ is the identity matrix. The reverse process is again a Markov chain that starts with $\mathbf{x}_K$ sampled from the Gaussian white noise $\mathcal{N}(0, \mathbf{I})$ and learns to remove the noise added in the forward process to regenerate $q(\cdot)$. This is achieved through the reverse transition $p_\theta\left(\mathbf{x}_{i-1} \mid \mathbf{x}_i\right) := \mathcal{N}\left(\mathbf{x}_{i-1}; \boldsymbol{\mu}_\theta\left(\mathbf{x}_i, i\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}_i, i\right)\right)$ where θ denotes the network parameters used to approximate the mean and the variance added in the forward process. As mentioned in the main text, we modify the reverse process by starting from a perturbed state $\tilde{s} + \phi$ instead of $\mathbf{x}_K$, where ϕ is pixel-wise noise randomly sampled from range $(-\epsilon_\phi, \epsilon_\phi)$ uniformly. We then take k reverse steps with $k \ll K$, according to the observation that a perturbed state only introduces a small amount of noise to the true state due to the attack budget ϵ. We observe in our experiments that using a large k does not hurt the performance, although it increases the running time (see Figures 5d and 5e in Appendix G.3).

The Progressive Distillation diffusion model [26] can distill an N steps sampler to a new sampler of $N/2$ steps with little degradation of sample quality. Thus with a 1024 step sampler, we could generate 512 step, 256 step, ..., and 8 step samplers. Notice that a single reverse step in an 8 step sampler will have an equivalent effect of sampling multiple steps in the original 1024 step sampler.

By choosing a proper sampler generated by progressive distillation (we report every model we used in G.2), we could accelerate our diffusion process while preserving sample quality at the same time.

C Pessimistic Q-learning with State Inference and Purification

In this section, we first formulate the robust RL problem as a two-player Stackelberg Markov game. We then present our pessimistic Q-learning algorithm that derives maximin actions from the Q-function using perturbed states as the input to safeguard against the agent's uncertainty about true states. We further incorporate a belief state approximation scheme and a diffusion-based state purification scheme into the algorithm to reduce uncertainty. Our extensions of the vanilla DQN algorithm that incorporates all three mechanisms are given in Algorithms 4-7 in Appendix F. We further give a theoretical result that characterizes the performance loss of being pessimistic.

C.1 State-Adversarial MDP as a Stackelberg Markov Game with Asymmetric Observations

The problem of robust RL under adversarial state perturbations can be viewed as a two-player Markov game, which motivates our pessimistic Q-learning algorithm given in the next subsection. The two players are the RL agent and the attacker with their state and action spaces and reward functions described in Section 2.2. The RL agent wants to find a policy $\pi : S \to \Delta(A)$ that maximizes its long-term return, while the attacker wants to find an attack policy $\omega : S \to S$ to minimize the RL agent's cumulative reward. The game has asymmetric observations in that the attacker can observe the true states while the RL agent observes the perturbed states only. The agent's value functions for a given pair of policies π and ω satisfy the Bellman equations below.

Definition 1. Bellman equations for state and action value functions under a state adversarial attack:

$$V_{\pi\circ\omega}(s) = \Sigma_{a\in A}\pi(a|(\omega(s))[R(s,a) + \gamma\Sigma_{s'\in S}P(s'|s,a)V_{\pi\circ\omega}(s')];$$
$$Q_{\pi\circ\omega}(s,a) = R(s,a) + \gamma\Sigma_{s'\in S}P(s'|s,a)[\Sigma_{a'\in A}\pi(a'|\omega(s'))Q_{\pi\circ\omega}(s',a')].$$

To achieve robustness, a common approach is to consider a Stackelberg equilibrium by viewing the RL agent as the leader and the attacker as the follower. The agent first commits to a policy π. The attack observes π and identifies an optimal attack, denoted by ω_π, as a response, where $\omega_\pi(s) = \text{argmin}_{\tilde{s}\in B_{\epsilon(s)}}\Sigma_{a'\in A}\pi(a'|\tilde{s})Q(s,a')$. As the agent has access to the intact environment at the training stage and the attacker's budget ϵ, it can, in principle, identify a robust policy proactively by simulating the attacker's behavior. Ideally, the agent wants to find a policy π^* that reaches a Stackleberg equilibrium of the game, which is defined as follows.

Definition 2. A policy π^* is a Stackelberg equilibrium of a Markov game if

$$\forall s \in S, \forall \pi, V_{\pi^*\circ\omega_{\pi^*}}(s) \geq V_{\pi\circ\omega_\pi}(s).$$

A Stackelberg equilibrium ensures that the agent's policy π^* is optimal (for any initial state) against the strongest possible *adaptive* attack and, therefore, provides a robustness guarantee. However, previous work has shown that due to the noisy observations, finding a stationary policy optimal for every initial state is generally impossible [33]. Existing solutions either introduce a regularization term to improve the smoothness of the policy or alternatively train the agent's policy and attacker's policy. In this paper, we take a different path with the goal of finding an approximate Stackelberg equilibrium, which is further improved through state prediction and denoising. Figure 2 shows the high-level framework of our approach, which is discussed in detail below.

C.2 Strategy I - Pessimistic Q-learning Against the Worst Case

Both value-based [17] and policy-based [35, 29] approaches have been studied to identify the Stackelberg equilibrium (or an approximation of it) of a Markov game. In particular, Stackelberg Q-learning [17] maintains separate Q-functions for the leader and the follower, which are updated by solving a stage game associated with the true state in each time step. However, these approaches do not apply to our problem as they all require both players have access to the true state in each time step. In contrast, the RL agent can only observe the perturbed state. Thus, it needs to commit to a policy for all states centered around the observed state instead of a single action, as in the stage game of Stackelberg Q-learning.

Algorithm 1: Pessimistic Q-Learning

Result: Robust Q-function Q

1 Initialize $Q(s,a)=0$ for all $s \in S$, $a \in A$;
2 **for** *epsiode = 1,2,...* **do**
3   Initialize true state s
4   **repeat**
5   Update agent's policy: $\forall \tilde{s} \in S, \pi(\tilde{s}) = \mathrm{argmax}_{a \in A} \min_{\bar{s} \in B_\epsilon(\tilde{s})} Q(\bar{s}, a)$;
6   Update attacker's policy: $\forall s \in S, \omega_\pi(s) = \mathrm{argmin}_{\tilde{s} \in B_\epsilon(s)} Q(s, \pi(\tilde{s}))$;
7   Generate perturbed state $\tilde{s} = \omega_\pi(s)$;
8   Choose a from $\tilde{s}$ using π with exploration:
9             $a = \pi(\tilde{s})$ with probability $1-\epsilon'$; otherwise a is a random action;
10   Take action a, observe reward R and next true state s';
11   Update Q-function: $Q(s,a) = Q(s,a) + \alpha\big[R(s,a) + \gamma Q(s', \pi(\omega_\pi(s'))) - Q(s,a)\big]$;
12         $s = s'$;
13   **until** s *is terminal*;
14 **end**

In this work, we present a pessimistic Q-learning algorithm (see Algorithm 1) to address the above challenge. The algorithm maintains a Q-function with the true state as the input, similar to vanilla Q-learning. But instead of using a greedy approach to derive the target policy or a ϵ-greedy approach to derive the behavior policy from the Q-function, a $\mathrm{maximin}$ approach is used in both cases. In particular, the target policy is defined as follows (line 5). Given a perturbed state $\tilde{s}$, the agent picks an action that maximizes the worst-case Q-value across all possible states in $B_\epsilon(\tilde{s})$, which represents the agent's uncertainty. We abuse the notation a bit and let $\pi(\cdot)$ denote a deterministic policy in the rest of the paper since we focus on Q-learning-based algorithms in this paper. The behavioral policy is defined similarly by adding exploration (lines 8 and 9). The attacker's policy ω_π is derived as the best response to the agent's policy (line 6), where a perturbed state is derived by minimizing the Q value given the agent's policy.

A few remarks follow. First, the $\mathrm{maximin}$ scheme is applied when choosing an action with exploration (line 9) and when updating the Q-function (line 11), and a perturbed state is used as the input in both cases. In contrast, in both SA-DQN [33] and WocaR-DQN [20], actions are obtained from the Q-network using true states at the training stage, while the same network is used at the test stage to derive actions from perturbed states. Our approach removes this inconsistency, leading to better performance, especially under relatively large perturbations. Second, instead of the pessimistic approach, we may also consider maximizing the average case or the best case across $B_\epsilon(\tilde{s})$ when deriving actions, which provides a different tradeoff between robustness and efficiency. Third, we show how policies are derived from the Q-function to help explain the idea of the algorithm. Only the Q-function needs to be maintained when implementing the algorithm.

Figure 3 in the Appendix D illustrates the relations between a true state s, the perturbed state $\tilde{s}$, the worst-case state $\bar{s} \in B_\epsilon(\tilde{s})$ for which the action is chosen (line 5). In particular, it shows that the true state s must land in the ϵ-ball centered at $\tilde{s}$, and the worst-case state the RL agent envisions is at most 2ϵ away from the true state. This gap causes performance loss that will be studied in Section C.6. For environments with large state and action spaces, we apply the above idea to derive pessimistic DQN algorithms (see Algorithms 4- 7 in Appendix F), which further incorporate state inference and purification discussed below. Although we focus on value-based approaches in this work, the key ideas can also be incorporated into Stackelberg policy gradient [29] and Stackelberg actor-critic [35] approaches, which is left to our future work.

C.3 Strategy II - Reducing Uncertainty Using Beliefs

In Algorithm 1, the agent's uncertainty against the true state is captured by the ϵ-ball around the perturbed state. A similar idea is adopted in previous regularization-based approaches [33, 20]. For example, SA-MDP [33] regulates the maximum difference between the top-1 action under the true state s and that under the perturbed state across all possible perturbed states in $B_\epsilon(s)$. However, this approach is overly conservative and ignores the temporal correlation among consecutive states. Intuitively, the agent can utilize the sequence of historical observations and actions $\{(\tilde{s}_\tau, a_\tau)\}_{\tau<t} \cup$

$\{\tilde{s}_t\}$ and the transition dynamics of the underlying MDP to reduce its uncertainty of the current true state s_t. This is similar to the belief state approach in partially observable MDPs (POMDPs). The key difference is that in a POMDP, the agent's observation o_t in each time step t is derived from a fixed observation function with $o_t = O(s_t, a_t)$. In contrast, the perturbed state $\tilde{s}_t$ is determined by the attacker's policy ω, which is non-stationary at the training stage and is unknown to the agent at the test stage.

To this end, we propose a simple approach to reduce the agent's worst-case uncertainty as follows. Let $M_t \subseteq B_\epsilon(\tilde{s}_t)$ denote the agent's belief about all possible true states at time step t. Initially, we let $M_0 = B_\epsilon(\tilde{s}_0)$. At the end of the time step t, we update the belief to include all possible next states that is reachable from the current state and action with a non-zero probability. Formally, let $M'_t = \{s' \in S : \exists s \in M_t, P(s'|s, a_t) > 0\}$. After observing the perturbed state $\tilde{s}_{t+1}$, we then update the belief to be the intersection of M'_t and $B_\epsilon(\tilde{s}_{t+1})$, i.e., $M_{t+1} = M'_t \cap B_\epsilon(\tilde{s}_{t+1})$, which gives the agent's belief at time $t + 1$. Figure 3 in the Appendix D demonstrates this process, and the formal belief update algorithm is given in Algorithm 2 in Appendix F. Our pessimistic Q-learning algorithm can easily incorporate the agent's belief. In each time step t, instead of using $B_\epsilon(\tilde{s})$ in Algorithm 1 (line 5), the current belief M_t can be used. It is an interesting open problem to develop a strong attacker that can exploit or even manipulate the agent's belief.

Belief approximation in large state space environments. When the state space is high-dimensional and continuous, computing the accurate belief as described above becomes infeasible as computing the intersection between high-dimensional spaces is particularly hard. Previous studies have proposed various techniques to approximate the agent's belief about true states using historical data in partially observable settings, including using classical RNN networks [21] and flow-based recurrent belief state learning [6]. In this work, we adapt the particle filter recurrent neural network (PF-RNN) technique developed in [21] to our setting due to its simplicity. In contrast to a standard RNN-based belief model $B : (S \times A)^t \to H$ that maps the historical observations and actions to a deterministic latent state h_t, PF-RNN approximates the belief $b(h_t)$ by κ_p weighted particles in parallel, which are updated using the particle filter algorithm according to the Bayes rule. An output function f_{out} then maps the weighted average of these particles in the latent space to a prediction of the true state in the original state space.

To apply PF-RNN to our problem, we first train the RNN-based belief model N and the prediction function f_{out} before learning a robust RL policy. This is achieved by using C trajectories generated by a random agent policy and a random attack policy in an intact environment. Then at each time step t during the RL training and testing, we use the belief model N and historical observations and actions to generate κ_p particles, map each of them to a state prediction using f_{out}, and take the set of κ_p predicted states as the belief M_t about the true state. PF-RNN includes two versions that support LSTM and GRU, respectively, and we use PF-LSTM to implement our approach. We define the complete belief model utilizing PF-RNN as $N_p \triangleq f_{out} \circ B$.

We remark that previous work has also utilized historical data to improve robustness. For example, [31] uses an LSTM-autoencoder to detect and denoise abnormal states at the test stage, and [32] considers an LSTM-based policy in alternating training. However, none of them explicitly approximate the agent's belief about true states and use it to derive a robust policy.

C.4 Strategy III - Purifying Invalid Observations via Diffusion

For environments that use raw pixels as states, such as Atari Games, perturbed states generated by adding bounded noise to each pixel are mostly "invalid" in the following sense. Let $S_0 \subseteq S$ denote the set of possible initial states. Let S^0 denote the set of states that are reachable from any initial state in S_0 by following an arbitrary policy. Then perturbed states will fall outside of S^0 with high probability. This is especially the case for l_∞ attacks that bound the perturbation applied to each pixel as commonly assumed in existing work (see Appendix D.2 for an example). This observation points to a fundamental limitation of existing perturbation attacks that can be utilized by an RL agent to develop a more efficient defense.

One way to exploit the above observation is to identify a set of "valid" states near a perturbed state and use that as the belief of the true state. However, it is often difficult to check if a state is valid or not and to find such a set due to the fact that raw pixel inputs are usually high-dimensional.

Instead, we choose to utilize a diffusion model to purify the perturbed states, which obtains promising performance, as we show in our empirical results.

To this end, we first sample C' trajectories from a clean environment using a pre-trained policy without attack to estimate a state distribution $q(\cdot)$, which is then used to train a Denoising Diffusion Probabilistic Model (DDPM) [12]. Then during both RL training and testing, when the agent receives a perturbed state $\tilde{s}$, it applies the reverse process of the diffusion model for k steps to generate a set of purified states as the belief M_t of size κ_d, where k and κ_d are hyperparameters. We let $N_d : S \to S^{\kappa_d}$ denote a diffusion-based belief model. Note that rather than starting from random noise in the reverse process as in image generation, we start from a perturbed state that the agent receives and manually add a small amount of pixel-wise noise ϕ to it before denoising, inspired by denoised smoothing in deep learning [30]. We observe in experiments that using a large k does not hurt performance, although it increases the running time. Thus, unlike previous work, this approach is agnostic to the accurate knowledge of attack budget ϵ. One problem with DDPM, however, is that it incurs high overhead to train the diffusion model and sample from it, making it less suitable for real-time decision-making. To this end, we further evaluate a recently developed fast diffusion technique, Progressive Distillation [26], which distills a multi-step sampler into a few-step sampler. As we show in the experiments, the two diffusion models provide different tradeoffs between robustness and running time. A more detailed description of the diffusion models and our adaptations are given in Appendix B.6.

C.5 Pessimistic DQN with Approximate Beliefs and State Purification

Built upon the above ideas, we develop two pessimistic versions of the classic DQN algorithm [22] by incorporating approximate belief update and diffusion-based purification, denoted by BP-DQN and DP-DQN, respectively. The details are provided in Algorithms 4- 7 in Appendix F. Below we highlight the main differences between our algorithms and vanilla DQN.

The biggest difference lies in the loss function, where we incorporate the $\mathrm{maximin}$ search into the loss function to target the worst case. Concretely, instead of setting $y_i = R_i + \gamma \mathrm{max}_{a' \in A} Q'(s_i, a')$ as in vanilla DQN, we set $y_i = R_i + \gamma \mathrm{max}_{a' \in A} \mathrm{min}_{m \in M_i} Q'(m, a')$ where R_i, s_i, M_i are sampled from the replay buffer and Q' is the target network. Similarly, instead of generating actions using the ϵ-greedy (during training) or greedy approaches (during testing), the $\mathrm{maximin}$ search is adopted.

To simulate the attacker's behavior, one needs to identify the perturbed state $\tilde{s}$ that minimizes the Q value under the current policy π subject to the perturbation constraint. As finding the optimal attack under a large state space is infeasible, we solve the attacker's problem using projected gradient descent (PGD) with η iterations to find an approximate attack similar to the PGD attack in [33]. In BP-DQN where approximate beliefs are used, the history of states and actions is saved to generate the belief in each round. In DP-DQN where diffusion is used, the reverse process is applied to both perturbed and true states. That is, the algorithm keeps the purified version of the true states instead of the original states in the replay buffer during training. We find this approach helps reduce the gap between purified states and true states. In both cases, instead of training a robust policy from scratch, we find that it helps to start with a pre-trained model obtained from an attack-free MDP.

We want to highlight that BP-DQN is primarily designed for environments with structural input, whereas DP-DQN is better suited for environments with raw pixel input. Both approaches demonstrate exceptional performance in their respective scenarios, even when faced with strong attacks, as shown in our experiments. Thus, although combining the two methods by integrating history-based belief and diffusion techniques may seem intuitive, this is only needed when confronted with an even more formidable attacker, such as one that alters both semantic and pixel information in Atari games.

C.6 Bounding Performance Loss due to Pessimism

In this section, we characterize the impact of being pessimistic in selecting actions. To obtain insights, we choose to work on a pessimistic version of the classic value iteration algorithm (see Algorithm 3 in Appendix F), which is easier to analyze than the Q-learning algorithm presented in Algorithm 1. To this end, we first define the Bellman operator for a given pair of policies.

Definition 3. For a given pair of agent policy π and attack policy ω, the Bellman operator for the Q-function is defined as follows.

$$T^{\pi \circ \omega} Q(s, a) = R(s, a) + \gamma \Sigma_{s' \in S} P(s'|s, a) Q(s', \pi(\omega(s'))) \tag{1}$$

The algorithm maintains a Q-function, which is initialized to 0 for all state-action pairs. In each round n, the algorithm first derives the agent's policy π_n and attacker's policy ω_{π_n} from the current Q-function Q_n in the same way as in Algorithm 1, using the worst-case belief, where

$$\pi_n(\tilde{s}) = \text{argmax}_{a \in A} \text{min}_{\bar{s} \in B_\epsilon(\tilde{s})} Q_n(\bar{s}, a), \forall \tilde{s} \in S.$$
$$\omega_{\pi_n}(s) = \text{argmin}_{\tilde{s} \in B_\epsilon(s)} Q_n(s, \pi_n(\tilde{s}))), \forall s \in S.$$

That is, π_n is obtained by solving a $\mathrm{maximin}$ problem using the current Q_n, and ω_{π_n} is a best response to π_n. The Q-function is then updated as $Q_{n+1} = T^{\pi_n \circ \omega_{\pi_n}} Q_n$. It is important to note that although $T^{\pi \circ \omega_\pi}$ is a contraction for a fixed π (see Lemma 3 in Appendix E for a proof), $T^{\pi_n \circ \omega_{\pi_n}}$ is typically not due its dependence on Q_n. Thus, Q_n may not converge in general, which is consistent with the known fact that a state-adversarial MDP may not have a stationary policy that is optimal for every initial state. However, we show below that we can still bound the gap between the Q-value obtained by following the joint policy $\tilde{\pi}_n := \pi_n \circ \omega_{\pi_n}$, denoted by $Q^{\tilde{\pi}_n}$, and the optimal Q-value for the original MDP without attacks, denoted by Q^*. It is known that Q^* is the unique fixed point of the Bellman optimal operator T^*, i.e., $T^* Q^* = Q^*$, where

$$T^* Q(s,a) = R(s,a) + \gamma \Sigma_{s' \in S} P(s'|s,a) \text{max}_{a' \in A} Q(s', a'). \tag{2}$$

We first make the following assumptions about the reward and transition functions of an MDP and then state the main result after that.

Assumption 1. The reward function and transition function are Lipschitz continuous. That is, there are constants l_r and l_p such that for $\forall s_1, s_2, s' \in S, \forall a \in A$, we have

$$|R(s_1, a) - R(s_2, a)| \leq l_r \|s_1 - s_2\|, |P(s'|s_1, a) - P(s'|s_2, a))| \leq l_p \|s_1 - s_2\|.$$

Assumption 2. Reward R is upper bounded where for any $s \in S$ and $a \in A$, $R(s,a) \leq R_{max}$.

Theorem 1. *The gap between $Q^{\tilde{\pi}_n}$ and Q^* is bounded by*

$$\text{limsup}_{n \to \infty} \|Q^* - Q^{\tilde{\pi}_n}\|_\infty \leq \frac{1+\gamma}{(1-\gamma)^2} \Delta,$$

where $\tilde{\pi}_n$ is obtained by Algorithm 3 and $\Delta = 2\epsilon\gamma(l_r + l_p |S| \frac{R_{max}}{1-\gamma})$.

We give a proof sketch and leave the detailed proof in Appendix E. We first show that $Q^{\tilde{\pi}_n}$ is Lipschitz continuous using Assumption1. Then we establish a bound of $\|T^* Q_n - Q_{n+1}\|_\infty$ and prove that $T^{\pi \circ \omega_\pi}$ for a fixed policy π is a contraction. Finally, we prove Theorem 1 following the idea of Proposition 6.1 in [2].

D More Graphs and Examples

D.1 An Example of Belief Update

Figure 3 illustrates the relations between a true state s, the perturbed state $\tilde{s}$, and the worst-case state $\bar{s} \in B_\epsilon(\tilde{s})$ for which the action is chosen.

D.2 An Example of Invalid States in Pixel-wise Perturbations

For example, the white bar shown in Figure 4 in Atari Pong game will not change during game play and has grayscale value of $236/255$. However, a pixel wise state perturbation attack such as PGD with attack budget $\epsilon = 15/255$ will change the pixel values in the white bar to range of $221/255 - 251/255$ so that the perturbed states become invalid.

E Proofs

E.1 Proof of Theorem 1

In this section, we prove Theorem 1. Recall that $\tilde{\pi} := \pi \circ \omega_\pi$. We first establish the Lipchitz continuity of $Q^{\tilde{\pi}}$, the Q-value when the agent follows policy π and the attack follows policy ω_π.

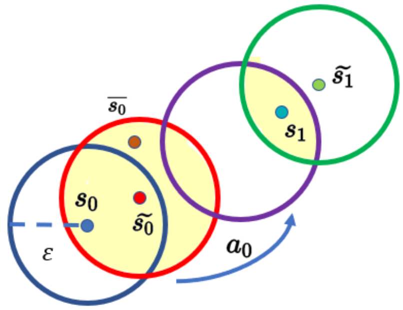
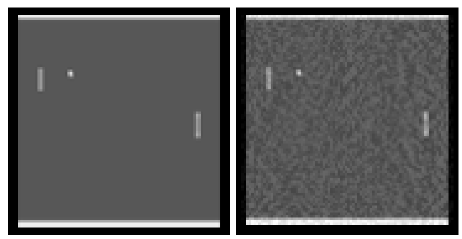


Figure 3: True, perturbed, and worst-case states in Algorithm 1 and belief update. Beginning with true state s_0 and perturbed state $\tilde{s}_0$, the agent will have an initial belief, i.e., the ϵ ball centered at $\tilde{s}_0$. After taking action a_0, the belief is updated to the region marked by the purple ball. When observing the next perturbed state $\tilde{s_1}$, the agent will update belief by taking the intersection of the purple ball and the green ball.



(a) Original (b) Perturbed

Figure 4: An example of valid vs. invalid states in Pong.

Lemma 1. *$Q^{\tilde{\pi}}$ is Lipchitz continuous for any π , i.e., $\forall s, s' \in S, \forall a \in A$,*

$$|Q^{\tilde{\pi}}(s,a) - Q^{\tilde{\pi}}(s',a)| \leq \mathcal{L}_{\mathcal{Q}_f}\|s - s'\| \tag{3}$$

where $L_{Q_f} = l_r + \frac{R_{max}}{1-\gamma}|S|l_p$

Proof. Based on the definition of the action value function under state perturbation, we have

$$\begin{aligned} Q^{\tilde{\pi}}(s,a) &= R(s,a) + \Sigma_{s' \in S} P(s'|s,a)\gamma V_{\pi \circ \omega_\pi}(s') \\ &\leq R(s,a) + \Sigma_{s' \in S} P(s'|s,a)\gamma V_{\pi}(s') \end{aligned}$$

Thus,

$$\begin{aligned} &|Q^{\tilde{\pi}}(s_1,a) - Q^{\tilde{\pi}}(s_2,a)| \\ =&|R(s_1,a) - R(s_2,a) + \Sigma_{s' \in S}[(P(s'|s_1,a) - P(s'|s_2,a))V_{\pi \circ \omega_\pi}(s')]| \\ \leq&|R(s_1,a) - R(s_2,a)| + \Sigma_{s' \in S}|[(P(s'|s_1,a) - P(s'|s_2,a))V_{\pi}(s')]| \\ \overset{(a)}{\leq}& l_r\|s_1 - s_2\| + \max_{s' \in S} V_\pi(s')|S|P(s'|s_1,a) - P(s'|s_2,a)| \\ \overset{(b)}{\leq}& l_r\|s_1 - s_2\| + \frac{R_{max}}{1-\gamma}|S|l_p\|s_1 - s_2\| \\ \leq&(l_r + \frac{R_{max}}{1-\gamma}|S|l_p)\|s_1 - s_2\| \\ =&\mathcal{L}_{\mathcal{Q}_f}\|s_1 - s_2\| \end{aligned}$$

where (a) follows from Assumption 1 and (b) follows from Assumptions 1 and 2 and the definition of V_π. ☐

Lemma 2. *For Q_n defined in Algorithm 3, the Bellman approximation error is bounded by*

$$\|T^*Q_n - Q_{n+1}\|_\infty \leq 2\epsilon\gamma(l_r + l_p|S|\frac{R_{max}}{1-\gamma}) \tag{4}$$

Proof.

$$
\begin{aligned}
&\|T^*Q_n - Q_{n+1}\|_\infty \\
=&\max_{s\in S, a\in A} |R(s,a) + \gamma \Sigma_{s'} P(s'|s,a) \max_{a\in A} Q_n(s',a) - [R(s,a) + \gamma \Sigma_{s'} P(s'|s,a) Q_n(s', \pi_n(\omega_{\pi_n}(s')))]| \\
=&\gamma \max_{s\in S, a\in A} \sum_{s'\in S} P(s'|s,a) |\max_{a\in A} Q_n(s',a) - Q_n(s', \tilde{\pi}_n(s'))| \\
\leq&\gamma \max_{s'\in S} |\max_{a\in A} Q_n(s',a) - Q_n(s', \tilde{\pi}_n(s'))|
\end{aligned}
$$

Let $\tilde{s'} = \omega_{\pi_n}(s')$ denote the perturbation of the true state s', and $\tilde{a}$ and $\bar{s'}$ denote the agent's action when observing $\tilde{s'}$ and the worst-case state in $B_\epsilon(\tilde{s'})$ that solves the $\mathrm{maximin}$ problem, respectively. We then have

$$
Q_n(s', \tilde{a}) \geq Q_n(\bar{s'}, \tilde{a}) = \max_{a\in A} \min_{s\in B_\epsilon(\tilde{s'})} Q_n(s,a). \tag{5}
$$

where the first inequality is due to the fact that $\bar{s'}$ obtains the worst-case Q-value under action $\tilde{a}$, across all states in $B_\epsilon(\tilde{s'})$ including s'. It follows that

$$
\begin{aligned}
\|T^*Q_n - Q_{n+1}\|_\infty &\leq \gamma \max_{s'\in S} |\max_{a\in A} Q_n(s',a) - Q_n(s', \tilde{\pi}(s'))| \\
&\leq \gamma \max_{s'\in S} |\max_{a\in A} Q_n(s',a) - \max_{a\in A} \min_{s\in B_\epsilon(\tilde{s'})} Q_n(s,a)| \\
&\leq \gamma \max_{s'\in S, a\in A} |Q_n(s',a) - \min_{s\in B_\epsilon(\tilde{s'})} Q_n(s,a)| \\
&\overset{(a)}{\leq} \gamma \max_{s'\in S, a\in A} |Q_n(s',a) - \min_{s\in B_{2\epsilon}(s')} Q_n(s,a)| \\
&\overset{(b)}{\leq} 2\gamma\epsilon \mathcal{L}_{\mathcal{Q}_f}
\end{aligned}
$$

where (a) is due to $\|\tilde{s'} - s'\| \leq \epsilon$ and (b) follows from Lemma 1. $\square$

Lemma 3. *Given any policy $\tilde{\pi} = \pi \circ \omega_\pi$ where π is a fixed policy, $T^{\tilde{\pi}}$ is a contraction.*

Proof.

$$
\begin{aligned}
\|T^{\tilde{\pi}}Q_1 - T^{\tilde{\pi}}Q_2\|_\infty &= \max_{s\in S, a\in A} \sum_{s'\in S} \gamma P(s'|s,a) |Q_1(s', \pi(\omega(s'))) - Q_2(s', \pi(\omega(s')))| \\
&\leq \max_{s'\in S} \gamma |Q_1(s', \pi(\omega(s'))) - Q_2(s', \pi(\omega(s')))| \\
&\leq \max_{s'\in S, a\in A} \gamma |Q_1(s',a) - Q_2(s',a)| \\
&= \gamma ||Q_1 - Q_2||_\infty
\end{aligned}
$$

Thus, for any given policy π, $T^{\tilde{\pi}}$ is a contraction. $\square$

With Lemmas 2 and 3, we are ready to prove Theorem 1.

Theorem 1. *The gap between $Q^{\tilde{\pi}_n}$ and Q^* is bounded by*

$$
\mathrm{limsup}_{n\to\infty} \|Q^* - Q^{\tilde{\pi}_n}\|_\infty \leq \frac{1+\gamma}{(1-\gamma)^2} \Delta
$$

where $\tilde{\pi}_n$ is obtained by Algorithm 3 and $\Delta = 2\epsilon\gamma(l_r + l_p|S|\frac{R_{max}}{1-\gamma})$.

Proof.

$$
\begin{aligned}
\|Q^* - Q^{\tilde{\pi}_n}\|_\infty &\overset{(a)}{\leq} \|T^*Q^* - T^*Q_n\|_\infty + \|T^*Q_n - T^{\tilde{\pi}_n} Q^{\tilde{\pi}_n}\|_\infty \\
&\leq \|T^*Q^* - T^*Q_n\|_\infty + \|T^*Q_n - T^{\tilde{\pi}_n} Q_n\|_\infty + \|T^{\tilde{\pi}_n} Q_n - T^{\tilde{\pi}_n} Q^{\tilde{\pi}_n}\|_\infty \\
&\overset{(b)}{\leq} \gamma \|Q^* - Q_n\|_\infty + \|T^*Q_n - Q_{n+1}\|_\infty + \gamma(\|Q_n - Q^*\|_\infty + \|Q^* - Q^{\tilde{\pi}_n}\|_\infty)
\end{aligned}
$$

Q_1	s_1	s_2	s_3	Q_2	s_1	s_2	s_3
a_1	12	11	3	a_1	4	2	-2
a_2	12	10	2	a_2	4	0	-1
a_3	12	8	1	a_3	4	1	-3

Table 2: A counterexample to $T^{\pi_n \circ \omega_{\pi_n}}$ being a contraction

where (a) follows from Q^* is the fixed point of T^*, $Q^{\tilde{\pi}_n}$ is the fixed point of $T^{\tilde{\pi}_n}$, and the triangle inequality, and (b) follows from both T^* and $T^{\tilde{\pi}_n}$ (for a fixed π_n) are contractions. This together with Lemma 2 implies that

$$\|Q^* - Q^{\tilde{\pi}_n}\|_\infty \leq \frac{2\gamma\|Q^* - Q_n\|_\infty) + \Delta}{1-\gamma} \tag{6}$$

We then bound $\|Q^* - Q_n\|_\infty$ as follows

$$\|Q^* - Q_{n+1}\|_\infty \leq \|T^*Q^* - T^*Q_n\|_\infty + \|T^*Q_n - Q_{n+1}\|_\infty \leq \gamma\|Q^* - Q_n\|_\infty + \Delta,$$

which implies that

$$\limsup_{n\to\infty}\|Q^* - Q_n\|_\infty \leq \frac{\Delta}{1-\gamma} \tag{7}$$

Plug this back in equation 6, we have

$$\limsup_{n\to\infty}\|Q^* - Q^{\tilde{\pi}_n}\|_\infty \leq \frac{1+\gamma}{(1-\gamma)^2}\Delta \tag{8}$$

where $\Delta = 2\epsilon\gamma(l_r + l_p|S|\frac{R_{max}}{1-\gamma})$. □

E.2 A counterexample to $T^{\pi_n \circ \omega_{\pi_n}}$ being a contraction when $\pi_n \circ \omega_{\pi_n}$ is not fixed

Consider an MDP $\langle S, A, P, R, \gamma \rangle$ where $S = \{s_1, s_2, s_3\}$ and $A = \{a_1, a_2, a_3\}$. Suppose that for any $s \in S$ and $a \in A$, $P(s_2|s,a) = 1$ and $P(s'|s,a) = 0$ for $s' \neq s_2$, and $B_\epsilon(s) = \{s_1, s_2, s_3\}$. Consider the two Q-functions shown in Table 2. We have $||Q_1 - Q_2||_\infty = |Q_1(s_2,a_2) - Q_2(s_2,a_2)| = 10$. However, when $\tilde{\pi}_1 = \pi_1 \circ \omega_{\pi_1}$ is derived from Q_1 and $\tilde{\pi}_2 = \pi_2 \circ \omega_{\pi_2}$ is derived from Q_2,

$$\begin{aligned}
||T^{\tilde{\pi}_1}Q_1 - T^{\tilde{\pi}_2}Q_2||_\infty &= \max_{s\in S, a\in A}\sum_{s'\in S}\gamma P(s'|s,a)|Q_1(s', \pi_1(\omega_{\pi_1}(s'))) - Q_2(s', \pi_2(\omega_{\pi_2}(s')))| \\
&= \max_{s\in S, a\in A}\gamma|Q_1(s_2, \pi_1(\omega_{\pi_1}(s'))) - Q_2(s, \pi_2(\omega_{\pi_2}(s')))| \\
&\geq \gamma|Q_1(s_2, \pi_1(\omega_{\pi_1}(s_2))) - Q_2(s_2, \pi_2(\omega_{\pi_2}(s_2)))| \\
&\overset{(a)}{=} \gamma|Q_1(s_2, a_1) - Q_2(s_2, a_2)| \\
&= \gamma \times 11 \\
&\overset{(b)}{>} 10 = ||Q_1 - Q_2||_\infty
\end{aligned}$$

where (a) is due to the fact that no matter what $\omega_\pi(s_2)$ is, $B_\epsilon(\omega_\pi(s_2)) = \{s_1, s_2, s_3\} = S$, which implies that $\pi_1(\omega_{\pi_1}(s_2)) = \text{argmax}_{a\in A}\min_{\bar{s}\in S}Q_1(\bar{s}, a) = a_1$, and similarly, $\pi_2(\omega_{\pi_2}(s_2)) = a_2$; (b) holds when $\gamma > \frac{10}{11}$. Therefore, $T^{\pi_1 \circ \omega_{\pi_1}}$ is not a contraction.

F Algorithms

Algorithm 2: Belief Update

Data: Old Belief M_t, action a, perturbed state $\tilde{s}_{t+1}$.
Result: Updated Belief M_{t+1}

Initialize M'_t to be an empty set
for s *in* M_t **do**
  for s' *in* S **do**
    if $P(s'|s,a) \neq 0$ **then**
      Add s' to M'_t
    end
  end
end
$M_{t+1} = M'_t \cap B_\epsilon(\tilde{s}_{t+1})$
RETURN M_{t+1}

Algorithm 3: Pessimistic Q-Iteration

Result: Robust Q-function Q

Initialize $Q_0(s,a) = 0$ for all $s \in S$, $a \in A$;
for $n = 0, 1, 2, ...$ **do**
  Update RL agent policy: $\forall \tilde{s} \in S, \pi_n(\tilde{s}) = \mathrm{argmax}_{a \in A} \min_{\bar{s} \in B_\epsilon(\tilde{s})} Q_n(\bar{s}, a)$;
  Update attacker policy: $\forall s \in S, \omega_{\pi_n}(s) = \mathrm{argmin}_{\tilde{s} \in B_{\epsilon(s)}} Q_n(s, \pi_n(\tilde{s}))$;
  for $s \in S$ **do**
    for $a \in A$ **do**
      $Q_{n+1}(s,a) = R(s,a) + \gamma \Sigma_{s' \in S} P(s'|s,a) Q_n(s', \pi(\omega_\pi(s')))$;
    end
  end
end

Algorithm 4: Belief-Enriched Pessimistic DQN (BP-DQN) Training. We highlight the difference between our algorithm and the vanilla DQN algorithm in brown.

Data: Number of iterations T, trained vanilla Q network Q_v, PF-RNN belief model N_p, target network update frequency Z, batch size D, exploration parameter ϵ'
Result: Robust Q network Q_r

Initialize replay buffer $\mathcal{B}$, robust Q network $Q_r = Q_v$, target Q network $Q' = Q_v$, observation history S_{his}, action history A_{his};
for *t = 0,1,...,T* **do**
  Use PGD to find the best perturb state $\tilde{s}_t$ that minimizes $Q_r(s_t, \pi(\tilde{s}_t))$, where π is derived from Q_r by taking greedy action;
  $M_t = M_t \cap B_\epsilon(\tilde{s}_t)$;
  Choose an action based on belief M_t and Q_r using ϵ-greedy: $a_t = \mathrm{argmax}_{a \in A} \min_{m \in M_t} Q_r(m, a)$ with probability $1 - \epsilon'$; otherwise a_t is a random action;
  Append $\tilde{s}_t$ and a_t to S_{his} and A_{his} and use belief model $N_p(S_{his}, A_{his})$ to generate M_{t+1};
  Execute action a_t in the environment and observe reward R_t and next true state s_{t+1};
  if s_{t+1} *is a terminal state* **then**
    Reset S_{his} and A_{his}
  end
  Store transition $\{s_t, a_t, R_t, s_{t+1}, M_t\}$ in $\mathcal{B}$;
  Sample a random minibatch of size D of transitions $\{s_i, a_i, R_i, s_{i+1}, M_i\}$ from $\mathcal{B}$;
  Set $y_i = \begin{cases} R_i & \text{for terminal } s_{i+1} \\ R_i + \gamma \max_{a' \in A} \min_{m \in M_i} Q'(m, a') & \text{for non-terminal } s_{i+1} \end{cases}$
  Perform a gradient descent step to minimize $Huber(\Sigma_i y_i - Q_r(s_i, a_i))$;
  Update target network every Z steps;
end

Algorithm 5: Belief-Enriched Pessimistic DQN (BP-DQN) Testing

Data: Trained robust Q network Q_r, PFRNN belief model N_p

Initialize observation history S_{his} and action history A_{his};
for *t = 0,1,...,T* **do**
  Observe the perturbed state $\tilde{s}_t$;
  if *t = 0* **then**
    $M_0 = B_\epsilon(\tilde{s}_t)$;
  end
  Select an action based on belief M_t and Q_r: $a_t = \mathrm{argmax}_{a \in A} \mathrm{min}_{m \in M_t} Q_r(m, a)$;
  Append $\tilde{s}_t$ and a_t to S_{his} and A_{his} and use belief model $N_p(S_{his}, A_{his})$ to generate M_{t+1};
  Execute action a_t in the environment;
end

Algorithm 6: Diffusion-Assisted Pessimistic DQN (DP-DQN) Training. We highlight the difference between our algorithm and the vanilla DQN algorithm in brown.

Data: Number of iterations T, trained vanilla Q network Q_v, diffusion belief model N_d, target network update frequency Z, batch size D, belief size κ_d, exploration parameter ϵ', noise level ϵ_ϕ

Result: Robust Q network Q_r

Initialize replay buffer $\mathcal{B}$, robust Q network $Q_r = Q_v$, target Q network $Q' = Q_v$;
for *t = 0,1,...,T* **do**
  Use PGD to find the best perturb state $\tilde{s}_t$ that minimizes $Q_r(s_t, \pi(\tilde{s}_t))$, where π is derived from Q_r by taking greedy action;
  Sample noise ϕ uniformly from $(-\epsilon_\phi, \epsilon_\phi)$ with same dimension as s_t pixel-wise;
  Use the diffusion belief model to generate belief $M_t = N_d(\tilde{s}_t + \phi)$ of size κ_d;
  Select an actions based on belief M_t and Q_r using ϵ-greedy:
   $a_t = \mathrm{argmax}_{a \in A} \mathrm{min}_{m \in M_t} Q_r(m, a)$ with probability $1 - \epsilon'$; otherwise a_t is a random action;
  Execute action a_t in environment and observe reward R_t and next true state s_{t+1};
  Apply the reverse diffusion process to s_t and s_{t+1}: $\hat{s}_t = N_d(s_t)$, $\hat{s}_{t+1} = N_d(s_{t+1})$;
  Store transition $\{\hat{s}_t, a_t, R_t, \hat{s}_{t+1}, M_t\}$ in $\mathcal{B}$;
  Sample a random minibatch of size D of transitions $\{\hat{s}_i, a_i, R_i, \hat{s}_{i+1}, M_i\}$ from $\mathcal{B}$;
  Set $y_i = \begin{cases} R_i & \text{for terminal } \hat{s}_{i+1} \\ R_i + \gamma \mathrm{max}_{a' \in A} \mathrm{min}_{m \in M_i} Q'(m, a') & \text{for non-terminal } \hat{s}_{i+1} \end{cases}$
  Perform a gradient descent step to minimize $Huber(\Sigma_i y_i - Q_r(\hat{s}_i, a_i))$;
  Update target network every Z steps;
end

Algorithm 7: Diffusion-Assisted Pessimistic DQN (DP-DQN) Testing

Data: Trained robust Q network Q_r, diffusion belief model N_d, noise level ϵ_ϕ

for *t = 0,1,...,T* **do**
  Observe the perturbed state $\tilde{s}_t$;
  Sample noise ϕ uniformly from $(-\epsilon_\phi, \epsilon_\phi)$ with same dimension as s_t pixel-wise;
  Generate belief using the diffusion belief model $M_t = N_d(\tilde{s}_t + \phi)$;
  Choose an action based on belief M_t and Q_r: $a_t = \mathrm{argmax}_{a \in A} \mathrm{min}_{m \in M_t} Q_r(m, a)$;
  Execute action a_t in the environment;
end

G Experiment Details and Additional Results

G.1 Experiment Setup Justification

Although both the BP-DQN and DP-DQN algorithms follow our idea of pessimistic Q-learning, the former is more appropriate for games with a discrete or a continuous but low-dimensional state space such as the continuous Gridworld environment, while the latter is more appropriate for games with raw pixel input such as Atari Games.

In the Gridworld environment, state perturbations can manipulate the semantics of states by changing the coordinates of the agent. In this case, historical information can be utilized to generate beliefs about true states. Following this idea, BP-DQN uses the particle filter recurrent neural network (PF-RNN) method to predict true states. In principle, we can also use BP-DQN on the Atari environments to predict true states through historical data. In practice, however, it is computationally challenging to do so due to the high-dimensional state space (84×84) of the Atari environments. Developing more efficient belief update techniques for large environments remains an active research direction.

On the other hand, state perturbations are injected pixel-wise in state-of-the-art attacks in Atari games. Consequently, they can barely change the semantics of true states Atari environments. In this case, historical information becomes less useful, and the diffusion model can effectively "purify" the perturbed states to recover the true states from high-dimensional image data. Although it is theoretically possible to use DP-DQN on the Gridworld environment, we conjecture that it is less effective than BP-DQN since it does not utilize historical data, which is crucial to recover true states when perturbations can change the semantic meaning of states as in the case of continuous Gridworld. In particular, we observe that the distributions of perturbed states and true states are very similar in this environment, making it difficult to learn a diffusion model that can map the perturbed states back to true states.

It is an interesting direction to develop strong perturbation attacks that can manipulate the semantics of true states for games with raw pixel input. As a countermeasure, we can potentially integrate diffusion-based state purification and belief-based history modeling to craft a stronger defense.

G.2 Experiment Setup

Environments. The continuous state Gridworld is modified from the grid maze environment in [21]. We create a 10×10 map with walls inside. There are also gold and a bomb in the environment where the agent aims to find the gold and avoid the bomb. The state space is a tuple of two real numbers in $[0, 10] \times [0, 10]$ representing the coordinate of the agent. The initial state of the agent is randomized. The agent can move in 8 directions, which are up, up left, left, down left, down, down right, right, and up right. By taking an action, the agent moves a distance of 0.5 units in the direction they choose. For example, if the agent is currently positioned at (x, y) and chooses to move upwards, the next state will be $(x, y + 0.5)$. If the agent chooses to move diagonally to the upper right, the next state will be $(x + 0.5/\sqrt{2}, y + 0.5/\sqrt{2})$. If the agent would collide with a wall by taking an action, it remains stationary at its current location during that step. The agent loses 1 point for each time step before the game ends and gains a reward of 200 points for reaching the gold and -50 points for reaching the bomb. The game terminates once the agent reaches the gold or bomb or spends 100 steps in the game. For Atari games, we choose Pong and Freeway provided by the OpenAI Gym [4].

Baselines. We choose vanilla DQN [23], SA-DQN [33] and WocaR-DQN [20] as defense baselines. We consider three commonly used attacks to evaluate the robustness of these algorithms: (1) PGD attack [33], which aims to find a perturbed state $\tilde{s}$ that minimizes $Q(s, \pi(\tilde{s}))$ and we set PGD steps $\eta = 10$ for both training and testing usage; (2) MinBest attack [14], which aims to find a perturbed state $\tilde{s}$ that minimizes the probability of choosing the best action under s, with the probabilities of actions represented by a softmax of Q-values; and (3) PA-AD [27], which utilizes RL to find a (nearly) optimal attack policy. For each attack, we choose $\epsilon \in \{0.1, 0.5\}$ for the Gridworld environment and $\epsilon \in \{1/255, 3/255, 15/255\}$ for the Atari games. Natural rewards (without attacks) are reported using policies trained under $\epsilon = 0.1$ for continuous state Gridworld and $\epsilon = 1/255$ for Atari games.

Training and Testing Details. We use the same network structure as vanilla DQN [23], which is also used in SA-DQN [33] and WocaR-DQN [20]. We set all parameters as default in their papers when training both SA-DQN and WocaR-DQN. For training our pessimistic DQN algorithm with PF-RNN-based belief (called BP-DQN, see Algorithm 4 in Appendix F), we set $\kappa_p = |M_t| = 30$, i.e., the PF-RNN model will generate 30 belief states in each time step. For training our pessimistic DQN algorithm with diffusion (called DP-DQN, see Algorithm 6 in Appendix F), we set $\kappa_d = |M_t| = 4$, that is, the diffusion model generates 4 purified belief states from a perturbed state. We consider two variants of DP-DQN, namely, DP-DQN-O and DP-DQN-F, which utilize DDPM and Progressive Distillation as the diffusion model, respectively. For DP-DQN-O, we set the number of reverse steps

Env	Parameter	PGD			MinBest			PA-AD		
		$\epsilon = 1/255$	$\epsilon = 3/255$	$\epsilon = 15/255$	$\epsilon = 1/255$	$\epsilon = 3/255$	$\epsilon = 15/255$	$\epsilon = 1/255$	$\epsilon = 3/255$	$\epsilon = 15/255$
Pong	noise level ϵ_ϕ	2/255	15/255	8/255	18/255	15/255	8/255	15/255	15/255	8/255
	reverse step k	1	1	1	1	1	1	4	4	4
	sampler step	64	32	32	32	32	32	64	64	64
Freeway	noise level ϵ_ϕ	5/255	5/255	5/255	5/255	5/255	5/255	5/255	5/255	5/255
	reverse step k	1	4	4	1	4	4	2	4	4
	sampler step	64	64	64	64	64	64	64	64	64

Table 3: Parameters Used to Test DP-DQN-F

to $k = 10$ for $\epsilon = 1/255$ or $3/255$ and $k = 30$ for $\epsilon = 15/255$, and do not add noise ϕ when training and testing DP-DQN-O. For DP-DQN-F, we set $k = 1$, sampler step to 64, and add random noise with $\epsilon_\phi = 5/255$ when training DP-DQN-F. We report the parameters used when testing DP-DQN-F in Table 3. We sample $C = C' = 30$ trajectories to train PF-RNN and diffusion models. All other parameters are set as default for training the PF-RNN and diffusion models. For all other baselines, we train 1 million frames for the continuous Gridworld environment and 6 million frames for the Atari games. For our methods, we take the pre-trained vanilla DQN model, and train our method for another 1 million frames. All training and testing are done on a machine equipped with an i9-12900KF CPU and a single RTX 3090 GPU. For each environment, all RL policies are tested in 10 randomized environments with means and variances reported.

G.3 More Experiment Results

G.3.1 More Baseline Results

Tables 4a and 4b give the complete results for the continuous space Gridworld and the two Atari games, where we include another baseline called Radial-DQN [24], which adds an adversarial loss term to the nominal loss of regular DRL in order to gain robustness. We find that Radial-DQN fails to learn a reasonable policy in continuous Gridworld as other regularization-based methods. However, Radial-DQN performs well under a small attack budget in Atari games but still fails to respond when the attack budget is high. Our Radial-DQN results for the Atari games were obtained using the pre-trained models [24], which might explain why the results are better than those reported in [20].

Environment	Model	Natural Reward	PGD		MinBest	
			$\epsilon = 0.1$	$\epsilon = 0.5$	$\epsilon = 0.1$	$\epsilon = 0.5$
Continous Gridworld	DQN	156.5 ± 90.2	128 ± 118	-53 ± 86	98.2 ± 137	98.2 ± 137
	SA-DQN	20.8 ± 140	46 ± 142	-100 ± 0	-5.8 ± 131	-100 ± 0
	WocaR-DQN	-100 ± 0	-100 ± 0	-63.2 ± 88	-100 ± 0	-63.2 ± 88
	Radial-DQN	-100 ± 0	-96.1 ± 12.3	-96.1 ± 12.3	-100 ± 0	-100 ± 0
	BP-DQN (Ours)	163 ± 26	165 ± 29	176 ± 16	147 ± 88	114 ± 114

(a) Continuous Gridworld Results

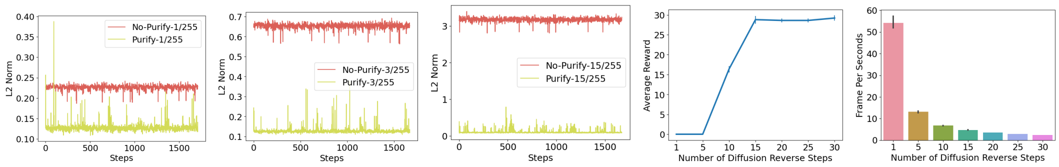
Env	Model	Natural Reward	PGD			MinBest			PA-AD		
			$\epsilon = 1/255$	$\epsilon = 3/255$	$\epsilon = 15/255$	$\epsilon = 1/255$	$\epsilon = 3/255$	$\epsilon = 15/255$	$\epsilon = 1/255$	$\epsilon = 3/255$	$\epsilon = 15/255$
Pong	DQN	21 ± 0	-21 ± 0	-21 ± 0	-21 ± 0	-21 ± 0	-21 ± 0	-21 ± 0	-18.2 ± 2.3	-19 ± 2.2	-21 ± 0
	SA-DQN	21 ± 0	21 ± 0	21 ± 0	-20.8 ± 0.4	21 ± 0	21 ± 0	-21 ± 0	21 ± 0	18.7 ± 2.6	-20 ± 0
	WocaR-DQN	21 ± 0	21 ± 0	21 ± 0	-21 ± 0	21 ± 0	21 ± 0	-21 ± 0	21 ± 0	19.7 ± 2.4	-21 ± 0
	Radial-DQN	21 ± 0	21 ± 0	21 ± 0	-21 ± 0	21 ± 0	21 ± 0	-21 ± 0	21 ± 0	21 ± 0	-19 ± 0
	DP-DQN-O(Ours)	19.9 ± 0.3	19.9 ± 0.3	19.8 ± 0.4	19.7 ± 0.5	19.9 ± 0.3	19.9 ± 0.3	19.3 ± 0.8	19.9 ± 0.3	19.9 ± 0.3	19.3 ± 0.8
	DP-DQN-F(Ours)	21 ± 0	20.4 ± 0.7	20.2 ± 0.8	18.6 ± 1	20.2 ± 0.9	19.0 ± 0	19.3 ± 1.6	18.0 ± 1.0	17.6 ± 1.8	17 ± 2.3
Freeway	DQN	34 ± 0.1	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0
	SA-DQN	30 ± 0	30 ± 0	30 ± 0	0 ± 0	27.2 ± 3.4	18.3 ± 3.0	0 ± 0	20.1 ± 4.0	9.5 ± 3.8	0 ± 0
	WocaR-DQN	31.2 ± 0.4	31.2 ± 0.5	31.4 ± 0.3	21.6 ± 1	29.6 ± 2.5	19.8 ± 3.8	21.6 ± 1	24.9 ± 3.7	12.3 ± 3.2	21.6 ± 1
	Radial-DQN	33.4 ± 0.5	33.4 ± 0.5	33.4 ± 0.5	21.6 ± 1	33.4 ± 0.5	32.8 ± 0.8	21.6 ± 1	33.4 ± 0.5	33.4 ± 0.5	21.6 ± 1
	DP-DQN-O(Ours)	28.8 ± 1.1	29.1 ± 1.1	29 ± 0.9	28.9 ± 0.7	29.2 ± 1.0	28.5 ± 1.2	28.6 ± 1.3	28.6 ± 1.2	28.3 ± 1	28.8 ± 1.3
	DP-DQN-F(Ours)	31.2 ± 1	30 ± 0.9	30.1 ± 1	30.7 ± 1.2	30.2 ± 1.3	30.6 ± 1.4	29.4 ± 1.2	30.8 ± 1	$31.4 \pm 0/8$	28.9 ± 1.1

(b) Atari Games Results

Table 4: Experiment Results. We show the average episode rewards $\pm$ standard deviation over 10 episodes for our methods and three baselines. The results for our methods are highlighted in gray.

G.3.2 More Ablation Study Results

Importance of Combining Maximin and Belief. In Table 5a, we compare our methods (BP-DQN and DP-DQN-O) that integrate the ideas of $\mathrm{maximin}$ search and belief approximation (using either



(a) (b) (c) (d) (e)

Figure 5: a), b) and c) show the l_2 distance between perturbed states and original states before and after purification under different attack budgets in the Pong environment using DDPM. d) shows the performance of DP-DQN-O under different diffusion steps in the Freeway environment under PGD attack with $\epsilon = 15/255$. e) shows the testing stage speed of DP-DQN-O (measured by the number of frames processed per second) under different diffusion steps in the Freeway environment.

Environment	**Model**	**PGD** $\epsilon = 0.5$	**Environment**	**Model**	**PA-AD** $\epsilon = 15/255$
Continuous Gridworld	Maximin Only	-71 ± 91	**Pong**	Maximin Only	-21 ± 0
	Belief Only	45.7 ± 134		DDPM Only	18.8 ± 1.6
	BP-DQN (Ours)	176 ± 16		DP-DQN-O (Ours)	19.3 ± 0.8

(a) Ablation Study Results. We compare our methods with variants that use $\mathrm{maximin}$ search or belief approximation only.

Environment	**Model**	**Training (hours)**	**Testing (FPS)**
GridWorld Continous	SA-DQN	3	607
	WocaR-DQN	3.5	721
	BP-DQN(Ours)	0.6+1.5+7	192

Environment	**Model**	**Training (hours)**	**Testing (FPS)**
Pong	SA-DQN	38	502
	WocaR-DQN	50	635
	DP-DQN-O (Ours)	1.5+18+30	6.6
	DP-DQN-F (Ours)	1+18+24	93

(b) Training and Testing Time Comparison. The training of our methods contains three parts: a) training the PF-RNN or diffusion model, b) training a vanilla DQN policy without attacks, and c) training a robust policy using BP-DQN, DP-DQN-O, or DP-DQN-F.

Table 5: Ablation and Time Comparison Results

RNN or diffusion) with variants of our methods that use $\mathrm{maximin}$ search or belief approximation only. The former is implemented using a trained BP-DQN or DP-DQN-O policy together with random samples from the ϵ-ball centered at a perturbed state (the worst-case belief) during the test stage. The latter uses the vanilla DQN policy with a single belief state generated by either the PF-RNN or the DDPM diffusion model at the test stage. The results clearly demonstrate the importance of integrating both ideas to achieve more robust defenses.

Diffusion Effects. In Figures 5a-5c, we visualize the effect of DDPM-based diffusion by recording the l_2 distance between a true state and the perturbed state and that between a purified true state and the purified perturbed state. For all three levels of attack budgets, our diffusion model successfully shrinks the gap between true states and perturbed states.

Performance vs. Running Time in DP-DQN. We study the impact of different diffusion steps k on average return of DP-DQN-O in Figure 5d and their testing stage running time in Figure 5e. Figure 5d shows the performance under different diffusion steps of our method in the Freeway environment under PGD attack with budget $\epsilon = 15/255$. It shows that we need enough diffusion steps to gain good robustness, and more diffusion steps do not harm the return but do incur extra overhead, as shown in Figure 5e, where we plot the testing stage running time in Frame Per Second (FPS). As the number of diffusion steps increases, the running time of our method also increases, as expected.

On the other hand, as DP-DQN-F uses a distilled sampler, it can decrease the reverse sample step k to as small as 1, which greatly reduces the testing time as reported in Table **??**. We report the performance results of DP-DQN-F in Table 1b and we find that DP-DQN-F improves DP-DQN-O under small perturbations, but suffers performance loss in Pong under PA-AD attack and large perturbations. Further, DP-DQN-F has a larger standard deviation than DP-DQN in the Pong environment, indicateing that DP-DQN-F is less stable than DP-DQN-O in Pong. We conjecture that the lower sample quality introduced by Progressive Distillation causes less stable performance and performance loss under PA-AD attack compared to DP-DQN-O that utilizes DDPM.

Training and Testing Overhead. Table 5b compares the training and test-stage overhead of SA-DQN, WocaR-DQN, and our methods. Notice that the training of our methods consists of three parts: a) training the PF-RNN belief model or the diffusion model, b) training a vanilla DQN policy without attacks, and c) training a robust policy using BP-DQN, DP-DQN-O or DP-DQN-F. In the continuous state Gridworld environment, our method takes around 9 hours to finish training, which is higher than SA-DQN and WocaR-DQN. But our method significantly outperforms these two baselines as shown in Table 1a. In the Atari Pong game, our method takes about 50 hours to train, which is comparable to WocaR-DQN but slower than SA-DQN. In terms of running time at the test stage, we calculate the FPS of each method and report the average FPS over 5 testing episodes. In the continuous Gridworld environment, our method is slower but comparable to SA-DQN and WocaR-DQN due to the belief update and $\mathrm{maximin}$ search. However, in the Atari Pong game, our DP-DQN-O method is much slower than both SA-DQN and WocaR-DQN. This is mainly due to the use of a large diffusion model in our method. However, our DP-DQN-F method is around 13 times faster than DP-DQN-O and is comparable to SA-DQN and WocaR-DQN.

H Conclusion and Limitations

In conclusion, this work proposes two algorithms, BP-DQN and DP-DQN, to combat state perturbations against reinforcement learning. Our methods achieve high robustness and significantly outperform state-of-the-art baselines under strong attacks. Further, our DP-DQN method has revealed an important limitation of existing state adversarial attacks on RL agents with raw pixel input, pointing to a promising direction for future research.

However, our work also has some limitations. First, our method needs access to a clean environment during training. Although the same assumption has been made in most previous work in this area, including SA-MDP and WocaR-MDP, a promising direction is to consider an offline setting to release the need to access a clean environment by learning directly from (possibly poisoned) trajectory data. Second, using a diffusion model increases the computational complexity of our method and causes slow running speed at the test stage. Fortunately, we have shown that fast diffusion methods can significantly speed up runtime performance. Third, we have focused on value-based methods in this work. Extending our approach to policy-based methods is an important next step.