TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 03/18 pp27–38
DOI: 10.26599/TST.2021.9010076
Volume 28, Number 1, February 2023

Curricular Robust Reinforcement Learning via GAN-Based Perturbation Through Continuously Scheduled Task Sequence

Yike Li, Yunzhe Tian, Endong Tong*, Wenjia Niu*, Yingxiao Xiang, Tong Chen, Yalun Wu, and Jiqiang Liu

Abstract: Reinforcement learning (RL), one of three branches of machine learning, aims for autonomous learning and is now greatly driving the artificial intelligence development, especially in autonomous distributed systems, such as cooperative Boston Dynamics robots. However, robust RL has been a challenging problem of reliable aspects due to the gap between laboratory simulation and real world. Existing efforts have been made to approach this problem, such as performing random environmental perturbations in the learning process. However, one cannot guarantee to train with a positive perturbation as bad ones might bring failures to RL. In this work, we treat robust RL as a multi-task RL problem, and propose a curricular robust RL approach. We first present a generative adversarial network (GAN) based task generation model to iteratively output new tasks at the appropriate level of difficulty for the current policy. Furthermore, with these progressive tasks, we can realize curricular learning and finally obtain a robust policy. Extensive experiments in multiple environments demonstrate that our method improves the training stability and is robust to differences in training/test conditions.

Key words: robust reinforcement learning; generative adversarial network (GAN) based model; curricular learning

1 Introduction

Recent advances in reinforcement learning (RL) have demonstrated its success for automated solutions in the field of decision making with beyond-human performance. Developing the usability of RL in real-world deployment environments rather than games is one of the core challenges in engineering applications. However, due to the gap between a simulated environment and the complex real world, most RL-based approaches fail to generalize in transferring a policy learned in simulations to applications. To this end, achieving robustness to environmental dynamics is crucial for adaptive and safe RL.

Existing efforts in robust RL have focused on environmental perturbations. One effective approach is based on domain randomization[1] to identify uncertain components of the environment and randomize their parameters. However, this requires to handcraft a set of training environments. Some existing approaches[2–6] explore automated perturbation. Robust adversarial reinforcement learning (RARL)[5] and noisy robust Markov decision process (NR-MDP)[6] represent two prominent examples, and the adversarial idea is typically introduced. By setting an adversary agent and a protagonist agent, this problem is formulated as a zero-sum min-max game to search for pure Nash equilibria (pure NE).

Despite its impressive progress, robust RL remains an open and critical challenge. Recent literature has shown that perturbations might fail to result in improved robustness, and some prominent studies contribute different-perspective solutions. Robustness via adversary populations (RAP)[7] uses multiple adversarial agents for perturbations and applies adversary populations to

• Yike Li, Yunzhe Tian, Endong Tong, Wenjia Niu, Yingxiao Xiang, Tong Chen, Yalun Wu, and Jiqiang Liu are with the Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, Beijing 100044, China. E-mail: yikeli@bjtu.edu.cn; tianyunzhe@bjtu.edu.cn; edtong@bjtu.edu.cn; niuwj@bjtu.edu.cn; yxxiang@bjtu.edu.cn; tongchen@bjtu.edu.cn; wuyalun@bjtu.edu.cn; jqliu@bjtu.edu.cn.

* To whom correspondence should be addressed.

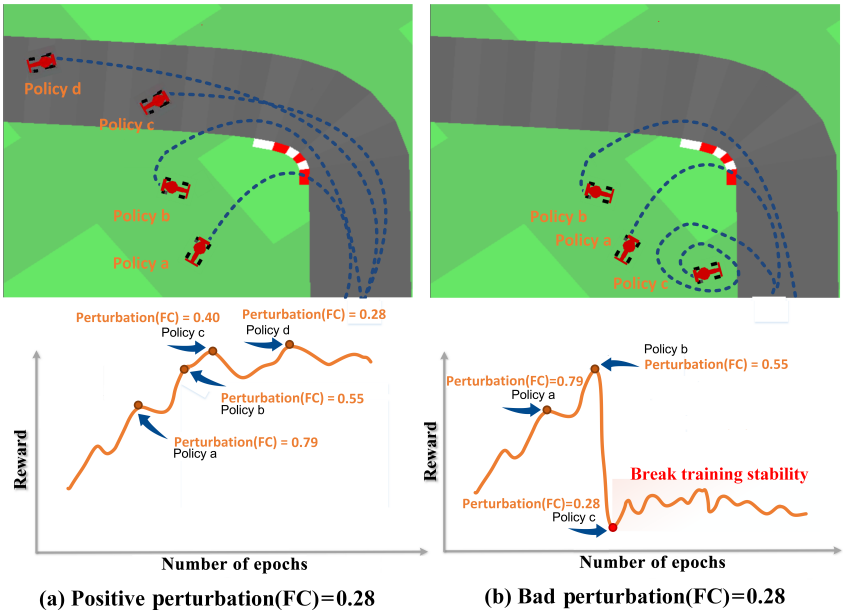
Manuscript received: 2021-09-30; accepted: 2021-10-13

NR-MDP. Wasserstein robust reinforcement learning (WR^{2}L)[8] limits the perturbations by the Wasserstein distance. The Langevin Dynamics based approach[9] chooses mixed Nash equilibrium (mixed NE) to model robust RL instead of pure NE, based on the stochastic gradient Langevin dynamics (SGLD)[10]. However, although the aforementioned literature involves positive perturbations, they neglect to control the perturbation difficulty and opportune time of positive perturbations in step-by-step robust RL. That is, sometimes at an inopportune time, an imposed positive perturbation can become a bad one, breaking the training stability and even bringing a failure of policy learning. As shown in Fig. 1b, we find an example of bad perturbation through visualizing the reinforcement learning process using trust region policy optimization (TRPO)[11] in OpenAI gym's CarRacing game[12]. Such a perturbed friction coefficient (FC) is 0.28 at the third epoch, which disrupts the stability and leads to a failure of robust RL. However, the same perturbation imposed at the fourth epoch time is absolutely a positive perturbation, as illustrated in Fig. 1a.

To tackle the above problem, an important insight brought by this work is to leverage curricular progressive learning for improving robustness. However, this solution is nontrivial, presenting us with two key challenges: (1) How to measure the perturbation difficulty according to the current policy in curricular RL learning? Existing methods either limit perturbations[5] or use a population of adversarial agents to control perturbations[7]. Hence, it is important to design novel measurements on current policy-aware perturbation difficulty. (2) How to automatically and efficiently generate appropriate perturbations? Existing generative adversarial network (GAN) based methods for task generation[13] use least squares GAN (LSGAN)[14] to automatically generate new tasks for the pathfinding scenario. Unfortunately, it cannot be directly applied to the automated perturbation of robust RL.

In this study, to address the above challenges, we make the first attempt to treat robust RL as a multi-task RL problem and propose a novel approach named curricular robust RL (CRRL). Based on automatic task generation, CRRL optimizes RL stability and robustness via curricular progressive task-oriented learning. Particularly, we propose a measurement of perturbation difficulty, which captures the context-aware difficulty of perturbations according to the current policy in a step-by-step learning process. Furthermore, based on the qualitative difficulty of perturbations, bad perturbations of the worst level are filtered out before imposing on follow-up tasks for robust learning. Our major contributions are highlighted as follows:

- To the best of our knowledge, we make the first attempt to explore the opportune time and appropriate difficulty of perturbations by modeling robust RL as a multi-task RL problem, which realizes curricular learning across multiple progressive tasks.



(a) Positive perturbation(FC)=0.28 (b) Bad perturbation(FC)=0.28

Fig. 1 Examples of visual trust region policy optimization reinforcement learning process in OpenAI Gym's CarRacing game, performing under single-parameter perturbations of friction coefficient. In the training process (a) and (b), the first row is the visualization of the car's trajectories and the second row is the car's corresponding reward curves.

• We propose a novel approach called CRRL, in which some delicate designs based on LSGAN, e.g., perturbation difficulty measurement and task generation, are proposed to improve the stability and robustness during curricular learning.

• We conduct extensive experiments on three environments (InvertedPendulum, Hopper, and HalfCheetah) to validate the effectiveness of CRRL as compared with state-of-the-art methods.

Figure 2a illustrates the framework of our proposed CRRL. Given a simulator and modified GAN, a range of RL tasks will be iteratively generated. Then, after the task evaluation, progressive tasks at the appropriate level of difficulty for the current policy will be utilized to train a robust policy in a curricular way. As shown in Fig. 2b, the policy takes s_t as preconditions and outputs a_t accordingly. The details of the modified GAN are shown in Fig. 2c. It exploits positive tasks to iteratively output new tasks at the appropriate level of difficulty for the current policy.

The rest of the paper is structured as follows: Section 2 introduces the preliminary and problem formulation. Section 3 proposes a novel approach CRRL based on LSGAN. Section 4 reports our experiments and evaluations on three OpenAI Gym's environments. Section 5 discusses related works. Finally, Section 6 presents the conclusions of this study.

2 Preliminary and Problem Formulation

In this section, we first illustrate standard RL and then redefine robust RL under the framework of multi-task learning.

2.1 Preliminary

The 6 tuples of MDP in standard RL are $(\mathcal{S}; \mathcal{A}; \mathcal{P}; r; \gamma;$ and $s_0)$, where $\mathcal{S}$ is a finite state space and $\mathcal{A}$ is a finite action space, $\mathcal{P}$: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow R$ is the transition probability, r: $\mathcal{S} \times \mathcal{A} \rightarrow R$ is the reward function and $r(s'|s,a)$ is the reward that the agent receives when selecting action $a \in \mathcal{A}$ from states s to s'. $\gamma \in [0,1)$ is the discount factor, and s_0 is the initial state distribution. The RL process is to learn a policy π, whose input is the observation of agents (also called actors) which is represented as a state a, and whose output is a vector of action probability for any state. In choosing a_t for the t time, the policy $\pi(a_t|s_t)$ is calculated by the probability $p(a_t|s_t)$.

Given the parameter $\boldsymbol{\theta}$ for the policy $\pi_{\boldsymbol{\theta}}$ and maximal time T to perform policy optimization, the RL goal is to find a policy $\pi^*_{\boldsymbol{\theta}}$: $\mathcal{S} \times \mathcal{A} \rightarrow R$ that maximizes the expectation of the cumulative discounted reward:

$$R(\pi^*_{\boldsymbol{\theta}}|\mathcal{P}) = \arg\max_{\pi} E\left[\sum_{t=0}^{T-1} \gamma^t \cdot r(s_t, a_t)|\mathcal{P}\right] \quad (1)$$

2.2 Problem formulation

As presented in Eq. (1), the expected reward is conditioned on the transition function $\mathcal{P}$. In standard RL, the transition function is always fixed. Unfortunately, the transition function may have modeling errors as designing an accurate simulator of the real world is extremely challenging. Thus, in robust RL, our goal is to learn a policy that generalizes well across a range of scenarios corresponding to a range of transition functions. Take a CarRacing scenario for example, we

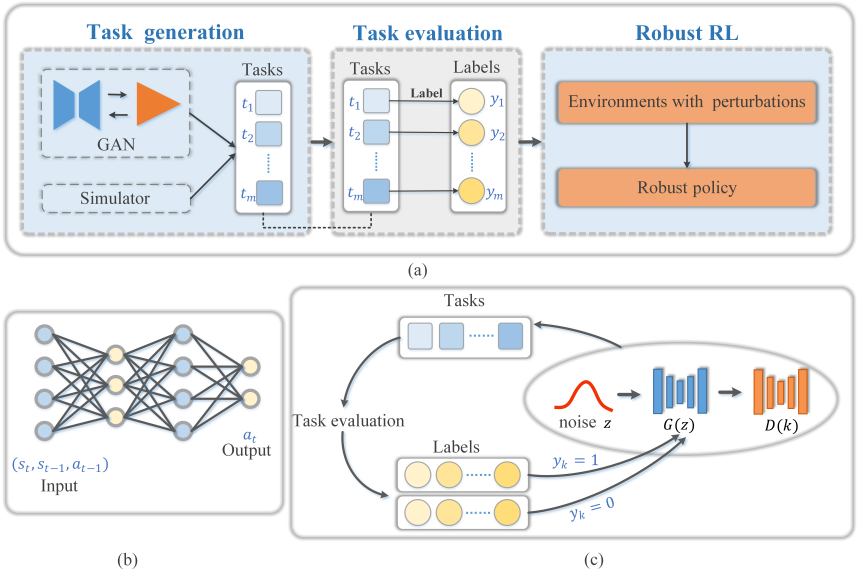


Fig. 2 Overview of the CRRL.

aim to learn a policy for a self-driving car that can run not only on a high-grade road (training scenario) but also on an icy road (test scenario). To achieve the policy robustness and generalization, we have the following assumptions:

• Modeling errors of the transition function can be viewed as a set of RL tasks following task distribution $p_k(\cdot)$.

• If we can learn a policy that is robust to a group of RL tasks, then this policy will have a high probability of being robust to changes in training/test conditions.

Based on the above assumptions, we attempt to tackle the robust RL issue by solving a multi-task RL problem. First, we take the initial simulator setting as the initial task and learn to get a policy, because the simulator is constructed elaborately, and the environment and dynamics of the simulator are roughly the same as the real world. Then, we iteratively generate new tasks at the appropriate level of difficulty for the current policy, and finally learn an ensemble policy on a progressive sequence of these tasks.

To summarize, in our robust RL framework, instead of learning to optimize a single reward function (as shown in Eq. (1)), we consider a group of reward functions, where each reward function is parameterized by a task $k \sim p_k(\cdot)$. The objective is to learn a policy that maximize the expected reward among all the tasks following $p_k(\cdot)$:

$$R(\pi_\theta) = \mathop{E}_{k\sim p_k(\cdot)}\left[E\left[\sum_{t=0}^{T-1}\gamma^t\cdot r(s_t,a_t)|k\right]\right] \quad (2)$$

3 Methodology

In this section, we introduce the proposed method in detail. We illustrate the definition of the policy under the multi-task framework, and show the task generation and evaluation method. Lastly, we propose our algorithm for policy optimization with curriculum settings.

3.1 Policy expression

We denote an agent policy as π_θ^*, which takes action a_t given state s_t at timestep t. Note that in the real world, it is extremely hard for an agent to get all the environment parameters. For instance, the FC is usually a latent parameter that is hard to be determined in the CarRacing scenario. Therefore, a finite state space $\mathcal{S}$ is insufficient to model all the environmental parameters in the training/test scenario. In addition, robust RL is usually achieved by solving a max-min optimization problem[15]. As a result, a policy will learn to always take conservative actions to avoid performing badly in some worst-cases. Clearly, this is not the optimal solution for robust RL.

We improve the policy expression as $\pi_\theta(a_t|s_t, s_{t-1}, a_{t-1})$, which will take action a_t according to not only the current state s_t but also the previous state s_{t-1} and action a_{t-1}. Taking the CarRacing scenario as an example, although the friction value is hard to get, we can parameterize the friction indirectly. Generally, an autonomous vehicle may show different driving statuses given the same state and action due to different FCs. Hence, in principle, if an autonomous vehicle takes action a_{t-1} at state s_{t-1}, making the state transfer from s_{t-1} to a new state s_t, then we can extrapolate the current friction correspondingly. By using s_{t-1}, a_{t-1}, and s_t as the preconditions of policy π_θ, our improved policy expression can model all possible latent environment parameters in the training/test scenario. On this basis, the final convergent policy will perform uniformly well in all cases.

As mentioned in Section 2, we aim to find a policy π_θ^* that achieves a high reward for a set of tasks:

$$\pi_\theta^*(a_t|s_t, s_{t-1}, a_{t-1}) = \arg\max_\pi E_{k\sim p_k(\cdot)}R(\pi_\theta|k),$$

$$R(\pi_\theta|k) = E\left[\sum_{t=0}^{T-1}\gamma^t\cdot r(s_t,a_t)|k\right] \quad (3)$$

3.2 Task evaluation

A successful robust RL training relies on the effectiveness of task sampling. In particular, we should sample valuable tasks that have a positive effect on the RL stability and robustness. Here, we present a definition of the valuable task corresponding to the appropriate level of perturbation difficulty.

Definition 1. (Valuable task) For the current policy π_θ^i (at iteration i), a task k is considered to be a valuable task, if in such task the π_θ^i receives expected reward s.t. $R^i_{\min} \leqslant R(\pi^i|k) \leqslant R^i_{\max}$.

On one hand, $R(\pi^i|k) \geqslant R^i_{\min}$ guarantees that sampled tasks can obtain enough reward and the policy convergence will be easy to receive. On the other hand, $R(\pi^i|k) \leqslant R^i_{\max}$ makes sure that we do not repeatedly sample from a small set of already mastered tasks, as these mastered tasks usually have high rewards. Each sampled task should be set with an appropriate difficulty to guarantee training stability.

The core idea of CRRL is to treat robust RL as a curricular multi-task RL problem. Hence, the difficulty of tasks should become iteratively higher as the training

process goes on step-by-step. Accordingly, the values of $R_{\min}$ and $R_{\max}$ should be dynamically updated. For instance, at iteration m, we generate a group of valuable tasks for the current policy π_θ^m. Then, we train a new policy on these tasks and obtain the following expected reward:

$$R^m = R(\pi_\theta^m) = \underset{k\sim\mathcal{TS}(m)}{E}\left[E\left[\sum_{t=0}^{T-1}\gamma^t\cdot r(s_t,a_t)|k\right]\right],$$
$$\mathcal{TS}(m) = \sum_{i=0}^{m}\Delta\mathcal{TS}(i) \tag{4}$$

where π_θ^m is the policy at iteration m, $\Delta\mathcal{TS}(i)$ represents the new set of valuable tasks at iteration i, and $\Delta\mathcal{TS}(0) = 1$ (i.e., initial simulator environment).

Thus, the valuable tasks for the current step are selected based on the reward value. According to R^m, $R_{\min}$ and $R_{\max}$ will be updated as follows:

$$R_{\min}^m = w_{\min}\cdot R^m, \quad R_{\max}^m = w_{\max}\cdot R^m \tag{5}$$

where $w_{\min}$ and $w_{\max}$ are hyperparameters, which can be reasonably adjusted. Different $w_{\min}$ and $w_{\max}$ values decide the degree of task difficulty to improve each iteration. We suggest that $w_{\min} \in (0.6, 0.7)$ and $w_{\max} \in (0.8, 0.9)$. With the new $R_{\min}$ and $R_{\max}$, new task generation and policy training will be iteratively performed.

Thus far, we have presented the task evaluation method. Given a number of tasks, we first estimate the label $y_k \in \{0, 1\}$ that indicates that whether a task is valuable task or not. Specifically, $y_k = 1$ indicates the task is a valuable task and vice versa. Thus, we regard the perturbation difficulty measurement as a classification problem in different training iterations. Then, we use the labeled tasks to train a generative model with which we can generate more valuable tasks to train on the next iteration.

3.3 Progressive task generation

Sampling tasks from $p_k(\cdot)$ directly, and filtering out valuable tasks to derive $\mathcal{TS}$ may not be the most efficient sampling methods. Thus, we adopt the "goal GAN" proposed in Ref. [13].

A generator network $G(z)$ learns to make the discriminator network $D(k)$ classify its output tasks as valuable tasks. Meanwhile, $D(k)$ is trained to maximize the probability of assigning the correct label to training examples and samples from $G(z)$. Unlike LSGAN[14], the "goal GAN" introduces the binary label y_k, allowing us to train from negative examples when $y_k = 0$. This improves the accuracy of the generative model despite being trained with few positive samples.

$$\begin{aligned}\min_D V(D) =& E_{g\sim p_g(\cdot)}[y_k\cdot(D(k)-b)^2+\\ &(1-y_k)\cdot(D(k)-a)^2]+\\ &E_{z\sim p_z(z)}[(D(G(z))-a)^2],\\ \min_G V(G) =& E_{z\sim p_z(z)}[(D(G(z))-c)^2]-\\ &w\cdot E_{z\sim p_z(z)}[\mathrm{RANGE}^2(G(z))],\\ \mathrm{RANGE}^2(M) =& \sum_{i=1}^{n}[\max(M[:,i])-\min(M[:,i])]^2\end{aligned} \tag{6}$$

Here, we directly use the original hyperparameters reported in Ref. [14] in all our experiments ($a = -1$, $b = 1$, and $c = 0$).

As shown in Eq. (6), we have three terms in the value function $V(D)$. For tasks labeled with $y_k = 1$, the second term disappears and we are left with the first and third terms, which are the same as LSGAN. For tasks labeled with $y_k = 0$, the first term disappears and the second and third terms are retrained, which allow us to train from negative examples. Note that the GAN may get stuck in the model collapse problem, that is, the generated samples tend to be close to one another. Thus, we modify the original value function $G(z)$ by introducing $\mathrm{RANGE}^2(M)$, which allows us to obtain more diverse generated perturbations within a reasonable range. It is treated as a measure of the pairwise distance between all of the generated perturbations, and our target is to maximize this distance to obtain more diverse generated perturbations within a reasonable range.

3.4 Curricular robust policy optimization

As mentioned earlier, we aim to train a robust policy $\pi_\theta^*(a_t|s_t)$ to maximize the expected reward in Eq. (2). Algorithms 1 and 2 outline our approach in detail.

Initially, we train a policy based on the initial environment parameters of the simulator. For the initial policy, we directly sample tasks from $p_k(\cdot)$, and label these sampled tasks as described in Section 3.2. At each iteration i, we generate more valuable tasks for the current policy using the improved "goal GAN" presented in Section 3.3. Then, we use these positive tasks to train our policy. As the positive task is selected with the expected reward of agents in the range($R_{\min}$, $R_{\max}$), in each training iteration, when the average reward of agents is higher than $R_{\max}$, the current iteration will be stopped, which is expressed as the opportune

Algorithm 1 Iterative training of robust RL

Input: Simulator environment ε
Output: Policy π_θ^*
$\pi_\theta^0 \leftarrow initial\_policy(\varepsilon)$
$R(\pi_\theta^0) \leftarrow E[\sum_{t=0}^{T-1} \gamma^t \cdot r(s_t, a_t)|\varepsilon]$
while $R(\pi_\theta^i) \geqslant w_{stop} \cdot R(\pi_\theta^0)$ **do**
  $z \leftarrow sample\_noise(p_z(\cdot))$
  $pert \leftarrow G(z) \cup task\_sample(p_k(\cdot))$
  $labeledtasks \leftarrow task\_evaluation(pert, R(\pi_\theta^{i-1}))$
  $(G, D) \leftarrow train\_GAN(labeledtasks)$
  $\mathcal{TS}(i) \leftarrow G(z)$
  $\pi_\theta^i \leftarrow update\_policy(\mathcal{TS}(i), \pi_\theta^{i-1})$
  $R(\pi_\theta^i) \leftarrow E_{k\sim\mathcal{TS}(i)}[E[\sum_{t=0}^{T-1} \gamma^t \cdot r(s_t, a_t)|k]]$
end while
return π_θ^*

Algorithm 2 Policy updating progress

Input: Policy π_θ^{i-1}, Valuable task set $\mathcal{TS}(i-1)$
Output: Policy π_θ^i
for $k = 1$ to $|\mathcal{TS}|$ **do**
  $\varepsilon_k \leftarrow constructEnv(\varepsilon, k)$
  for $t = 1$ to T **do**
    $\{(s_t^i, a_t^i, r_t^i)\} \leftarrow roll(\varepsilon_k, \pi_\theta^{i-1}, N_{traj})$
    $\pi_\theta^i(k) \leftarrow optimize\_policy(\{(s_t^i, a_t^i, r_t^i)\}, \pi_\theta^{i-1})$
  end for
end for
$\pi_\theta^i = \arg\max_\pi E_{k\sim\mathcal{TS}(i-1)} R(\pi_\theta|k))$
return π_θ^i

time to generate new tasks with updated difficulties. Moreover, any RL algorithm can be used for training; in our case we use TRPO with generalized advantage estimation (GAE)[11]. TRPO can prevent the policy from vibrating too much due to the noise in the policy gradient. Essentially, all the generated tasks from the previous iteration will also be labeled again and used to train a new generator to output valuable tasks for the next iteration. Lastly, we update $R_{\min}$ and $R_{\max}$ according to Eq. (5). When the expected reward at one iteration reaches $R_{\text{stop}} = w_{\text{stop}} \cdot R(\pi_\theta^0)$ ($w_{\text{stop}} = 0.5$ in our experiments), the robust RL will be stopped, and the current policy will be outputted as the final robust policy.

Algorithm 2 illustrates the policy updating process in each epoch. We consider all the valuable tasks $k \in \mathcal{TS}(i-1)$, and for each task k, we construct the corresponding perturbed environment ε_k and perform T steps of training on the agent. In each step t, $roll$ function samples trajectories (s_t, a_t, r_t) given the perturbed environment ε_k to train $\pi_\theta(k)$. Then, we update the old policy π_θ^{i-1} using a policy optimizer, and the new policy can be denoted as π_θ^i.

4 Experimental Results and Analysis

In this section, we illustrate our experimental setup and evaluate the robustness and stability of our CRRL through a comparison with the state-of-the-art methods.

4.1 Experimental setup

Benchmark. We implement the adversarial benchmark built on OpenAI Gym's[12] control environments with the MuJoCo[16] physics simulator. We experiment with the InvertedPendulum, Hopper, and HalfCheetah continuous control environments. The details of the environments and their corresponding adversarial perturbations are shown in Fig. 3.

- **InvertedPendulum.** The protagonist agent can apply one-dimensional forces to keep the pendulum upright, and the pendulum is mounted on a pivot point on a cart with the cart restricted to linear movements in a plane.
- **Hopper.** The protagonist agent is a planar monopod robot with four rigid links, corresponding to the torso, upper leg, lower leg, and foot, along with three actuated joints.
- **HalfCheetah.** The protagonist agent is a planar bipedal robot with eight rigid links, including two legs and a torso, along with six actuated joints.

The ranges of the environmental parameters mass and FCs are defined as follows: for Hopper, mass $\in [0.7, 1.3]$ and friction $\in [0.7, 1.3]$; for HalfCheetah, mass $\in [0.5, 1.5]$ and friction $\in [0.7, 1.3]$. Because the friction is not relevant to InvertedPendulum, there is only one parameter mass that varies in the range $[0.6, 6.6]$. For each randomized parameter set within predefined ranges, we train our CRRL by progressively generating the force for the adversary agent.

We adopt a version of the training-validation-test split from the field of supervised learning[17, 18]. Based on the grid search, the validation set is used to select the optimal hyperparameters for further testing. That is, parameters within a model are selected rather than

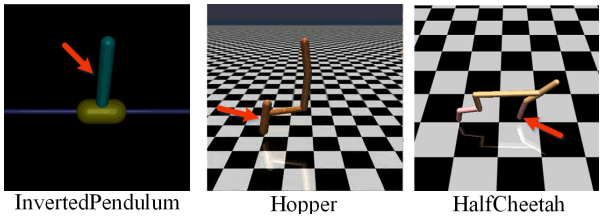


Fig. 3 InvertedPendulum, Hopper, and HalfCheetah environments of OpenAI gym, in which an adversary agent applies destabilizing forces on specific points (denoted by red arrows) as perturbations to the environments.

selecting different models. Moreover, we record all forces with corresponding environmental parameter settings as our training set. For the validation set, we sample all the environmental parameters such as mass and friction at fixed intervals, so that all possible parameter combinations in the environment are used as the verification set. For the test set, we extract the highest and lowest parameter values from the range and combine them together. That is, the test set is composed of "worst perturbations", because the worst perturbed environment can best reflect the robustness and stability of the approach.

Therefore, the validation set is much larger than the test set, as the validation set contains all possible combinations of parameters, whereas the test set only focuses on the worst combination of parameters. In our experiments, the corresponding sizes of the training/validation/test sets are respectively 200/440/20 for InvertedPendulum, and 200/4840/32 for both HalfCheetah and Hopper.

Baseline methods. We contextualize the empirical results of our CRRL using comparisons with the following state-of-the-art baseline methods:

• TRPO[11] was developed by OpenAI to implement strong RL without perturbations and serves as a basic indicator for measuring the performance of different approaches.

• RARL[5] trains the RL for the protagonist agent in the presence of an adversary agent's perturbations.

• RAP[7] makes use of a large set of protagonist agents in the training process, proving that it can avoid the potential failure of training with a signal adversary.

In this work, TRPO is used as the policy optimizer implemented by a neural network with three hidden layers and 100 neurons. We set the learning rate as 0.01. Other hyperparameters, namely kl step size for TRPO, training batch size, discount factor, and lambda used for the generalized advantage estimation, are tuned by the grid search. In the GAN implementation, the generator consists of two hidden layers with 256 neurons each, and the discriminator consists of two hidden layers with 128 neurons each. We set $\omega_{\min}$ and $\omega_{\max}$ as 0.65 and 0.85, respectively.

Evaluation metrics. We evaluate the performance of different approaches from two aspects:

• **Robustness.** The robustness is defined as the ability of the protagonist agent to overcome different adversarial perturbations. Three datasets (i.e., training, verification, and test sets) are used to verify the robustness of different approaches, described in the experimental settings in Section 4.1.

• **Stability.** We focus on stability in the training period. As the performance of the agent is greatly influenced by adversarial environments, the training stability reflects the difficulty of introduced perturbations in a way.

Specifically, for the robustness evaluation, we report both the mean of all accumulated rewards (Mean) and the average of top 10 accumulated reward (Aver@10). Specifically, each epoch is set to contain 10 trajectories of the episode. When testing, we sample trajectories to get the reward of the current policy and then get the accumulated reward across several episodes. Meanwhile, for the training stability evaluation, we adopt the mean and standard deviation of the rewards of the training period.

4.2 Robustness comparison

We perform an experiment on both the low-dimensional dynamical system (i.e., InvertedPendulum) and high-dimensional dynamical systems (i.e., HalfCheetah and Hopper). Then, we show the comparison results for all the baseline methods (i.e., TRPO, RARL, and RAP) and our CRRL using the mean reward and average of the top 10 as the metrics. For each method, the training, validation, and test sets are examined with the same number of training epochs. The results are shown in Table 1, from which we have the following observations: (1) For low dimensional dynamical systems, CRRL obtains the best performance on the mean and Aver@10 in the training, validation, and test sets. (2) For high dimensional dynamical systems, the mean reward of CRRL is stably superior to all baselines. Compared with the suboptimal baseline methods, CRRL achieved up to 2.51%, 5.62%, and 7.34% improvements on the training, validation, and test sets in the Hopper, 102.97%, 28.70%, and 7.23% improvements on the three sets in the HalfCheetah, respectively. The Aver@10 of CRRL also obtained a better performance in general. The result denotes that the priority of CRRL provides robust performance against "bad perturbations".

4.3 Training stability analysis

In this section, we analyze the training stability of CRRL. We use the mean and standard deviation of rewards at each training epoch as the metrics for evaluation. To analyze the different ways that perturbations can be introduced into the dynamic system, we compare CRRL

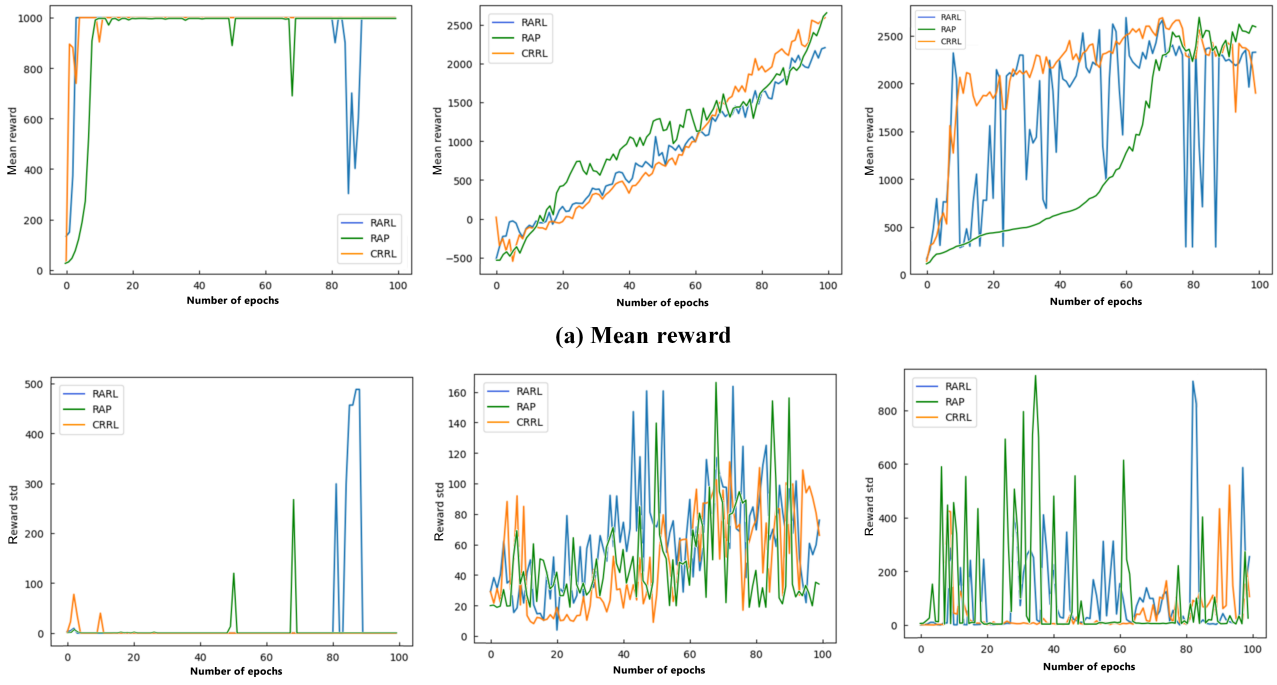
Table 1 Experiments on different datasets (mean±one standard deviation).

Environment	Baseline method	Training		Validation		Test	
		Mean	Aver@10	Mean	Aver@10	Mean	Aver@10
InvertedPendulum	TRPO	953.15±209.53	**1000.00±0.00**	995.59±14.00	**1000.00±0.00**	505.00±495.00	**1000.00±0.00**
	RARL	880.89±206.99	**1000.00±0.00**	953.58±93.44	**1000.00±0.00**	408.73±470.56	**1000.00±0.00**
	RAP	**1000.00±0.00**	**1000.00±0.00**	996.78± 12.49	**1000.00±0.00**	580.49±407.60	**1000.00±0.00**
	CRRL	**1000.00±0.00**	**1000.00±0.00**	**998.59±6.30**	**1000.00±0.00**	**587.80±478.21**	**1000.00±0.00**
Hopper	TRPO	1700.86±574.71	2284.71±8.25	1164.79± 515.92	2062.34±90.70	1003.64±112.51	2504.03±112.51
	RARL	2468.89± 877.37	3162.73±79.14	1305.29±534.01	2237.09±90.28	1036.69±839.99	2922.22±72.04
	RAP	1109.54±607.69	2445.52±48.80	1411.95±878.64	**2997.26±41.11**	1181.86±465.40	**2929.49±80.89**
	CRRL	**2530.84±1005.18**	**3258.94±32.25**	**1491.28±827.53**	2833.31±91.84	**1268.66±591.11**	2056.15±1.45
HalfCheetah	TRPO	1331.77±42.39	1353.89±16.60	1295.27+191.66	1589.85+23.02	1148.32+413.80	2043.47+18.69
	RARL	1249.78±55.12	1287.99±26.50	1165.84+136.67	1412.80+45.67	1016.27+322.03	1901.91+63.76
	RAP	925.36±877.78	1489.06±13.25	2010.81±347.83	2678.93±201.03	2159.47±261.00	2509.69±209.62
	CRRL	**2703.14±110.73**	**2770.95±38.13**	**2588.03+167.33**	**2869.34+25.99**	**2315.70+577.46**	**3416.98+68.94**

Note: We record the mean reward (i.e., Mean) and the average of the top 10 reward values (i.e., Aver@10) for comparison. For InvertedPendulum, Hopper, and HalfCheetah adversarial environments, the results on the three datasets (i.e., training, verification, and test sets) are listed from left to right. The approach with the highest reward value is presented in bold, and the second highest is underlined.

with RARL and RAP.

The results are shown in Fig. 4, from which we formulate the following observations: (1) For low-dimensional dynamical systems, CRRL, RARL, and RAP achieve high rewards within the short terms. However, compared to other methods, CRRL was not affected by the "bad perturbations" during the training period. (2) For high dimensional dynamical systems, CRRL has more stable mean rewards compared to RARL and RAP. The standard deviation is closely related to amount of "bad perturbations". CRRL greatly reduces the standard deviation as it gets better control to perturbation difficulty for the current policy. This finding empirically highlights the significant gains in the training stability of our CRRL especially for complex environments.



(a) Mean reward

(b) Standard deviation of reward

Fig. 4 Training stability of different methods. (a) Comparison of the mean reward from epoch 0–100, (b) the standard deviation of each training epoch. From left to right: InvertedPendulum, HalfCheetah, and Hopper adversarial environments.

4.4 Visualization

To intuitively examine the robustness w.r.t. environmental parameter combination, we visualize the mean reward of the validation set using the heatmap across all the mass and FCs combinations. Figure 5a shows the robustness of policies on a simple InvertedPendulum as the mass values in the range 0.6–6.6. Apparently, CRRL outperforms all baselines in these benchmarks with high and stable rewards. Similarly, we perform tests by jointly varying both mass and FC in Figs. 5b and 5c. The lighter color, the better the performance. Clearly, CRRL has more stable and robust performance among all perturbation settings.

5 Related Work

5.1 Robust reinforcement learning

Robust RL has received significant attention due to its generalization for robust control. Robustness has several definitions in related researches. A common definition is a robustness against random noise and adversarial perturbations[19, 20]. Some define robustness as variance reducing[21, 22]. Here, we consider the second definition of robustness in our method. As most RL-based approaches fail to generalize in transferring a policy learned in a simulation to an application, robust RL is of great significance in narrowing the discrepancy between ground-truth states and agent observations. In related research, it is also defined as a sim-to-real problem. For instance, an agent working well in simulated environments may fail in real environments due to noises in observations[23], as real-world sensing involves unavoidable noises[24]. This condition also raises concerns in using RL in safety-crucial applications such as autonomous driving[25]. Robust RL aims to find optimal controllers under noisy perturbations. This problem is typically treated as a robust MDP[6] and solved through function approximation. For interpreting perturbations, some studies choose to disturb the observation. For instance, state-adversarial MDP[26] characterizes the decision making problem under adversarial attacks on state observations. Other studies change the action space of the agent[27] or modify transition dynamics. While the adversarial idea is first introduced in the prominent work RARL[5], which sets an adversary agent and a protagonist agent,

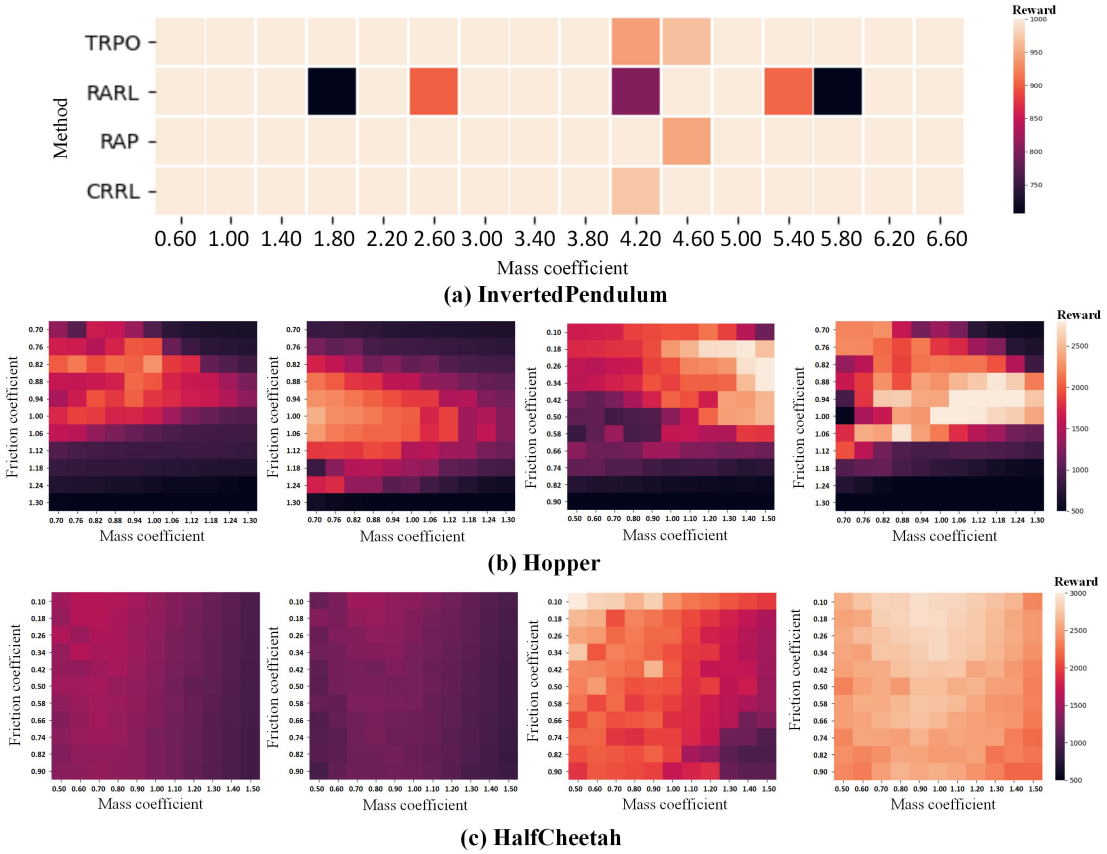


Fig. 5 Mean reward visualization of heatmap w.r.t. different methods. The mass coefficient sets on the *x*-axis, and the friction coefficient sets on the *y*-axis. In (b) and (c), from left to right: results of the TRPO, RARL, RAP, and CRRL approaches.

the problem is formulated as a zero-sum min-max game. Different from another prominent variant idea, i.e., NR-MDP[6]. RARL selects specific robot joints that the adversary agent acts on instead of adding the adversary agent action into the agent action in NR-MDP. Recently, the Langevin dynamics based approach[10] models robust RL as a mixed NE. WR^{2}L[8] intends to limit perturbations using the Wasserstein distance. By casting the robust RL problem as a multi-objective optimization problem, multi-objective bayesian optimization[28] is also introduced to solve the robust RL problem. Furthermore, based on RARL, the RAP-based approach[7] uses multiple adversarial agents for perturbations. Although these aforementioned methods achieve promising performances, they neglect to control perturbation difficulty and opportune time for the current policy, which might cause failures for robust RL.

Adversarial examples and attacks have been used in other learning problems. For instance, a method of using learning robust classifiers in supervised learning is performed through GANs[29]. In our work, we use LSGAN to generate perturbations automatically. Compared with the traditional GAN framework, LSGAN uses the least squares loss[30] in the objective function instead of the cross-entropy loss[31] to achieve a stable training performance. However, it is still unclear how to use the GAN-based model to generate automated perturbations of appropriate difficulty in progressive tasks.

5.2 Curricular learning

Curricular learning strategies have been successfully employed in all areas of learning problems and in a wide range of tasks. As a meta-learning methodology, curricular learning starts with learning how to solve a problem from simple examples and gradually increasing the complexity of examples in different thresholds. In recent studies, the teacher-student curricular learning framework[32] shows that curricular learning improved the performance of the training. Another curricular learning framework Mix&Max[33], shows that the curricular learning framework can also decrease the sample complexity of the training process.

As a number of studies have proven that curricular method is well adapted to RL, there is also a growing trend of introducing curricular settings into multi-task RL. Previous works[34, 35] have explored the idea of using curricular learning by artificially designing the curricula, in which the learned policy is closely related to the curricular design. Most curricular learning in RL still relies on fixed pre-specified sequences of tasks[36]. Recent research proposed a method to train a policy that generalizes to a set of continuously parameterized tasks[13] but only concentrates on specific environments. In our work, we shape the learning process by continually creating new challenging tasks for agents to adapt to, thereby facilitating the acquisition of efficient policies.

In addition, taking the CarRacing scenario as an example, in safety-critical applications, existing safe RL methods make an agent rely on priors to avoid dangerous situations with high probability, but the probabilistic guarantees and smoothness assumptions inherent in the priors are not viable in many scenarios. To prevent such problems, in our work, we introduce a GAN to produce tasks that are always at the appropriate level of difficulty for the agent, thus automatically producing a curriculum.

6 Conclusion

In this study, we focus on the robust RL problem. There are two main challenges, one of which is how to design reasonable perturbations, and the other is how to control perturbation difficulty and opportune time. To address these challenges, we make the first attempt to treat robust RL as multi-task learning for controlling perturbation difficulty and opportune time for the current policy. We propose CRRL, an approach to curricular robust RL, which realizes curricular learning across multiple progressive tasks. Particularly, to realize the automated difficulty-aware task generation, the mechanisms of task difficulty measurement are developed based on LSGAN.

To the best of our best knowledge, we conduct extensive experiments on all state-of-the-art robust RL algorithms including TPRO, RARL, and RAP in typical benchmarks. We analyze the experimental results on various aspects including training stability and RL performance. The experimental results have shown the advantages of our method in terms of robustness.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 61972025, 61802389, 61672092, U1811264, and 61966009), and the National Key R&D Program of China (Nos. 2020YFB1005604 and 2020YFB2103802).

References

[1] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P.

Abbeel, Domain randomization for transferring deep neural networks from simulation to the real world, in *Proc. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, Canada, 2017, pp. 23–30.

[2] C. G. Atkeson and J. Morimoto, Nonparametric representation of policies and value functions: A trajectory-based approach, in *Proc. Advances in Neural Information Processing Systems 15* (*NIPS*), Vancouver, Canada, 2002, pp. 1611–1618.

[3] J. Morimoto and K. Doya, Robust reinforcement learning, *Neural Computation*, vol. 17, no. 2, pp. 335–359, 2005.

[4] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine, EPOpt: Learning robust neural network policies using model ensembles, presented at 5th International Conference on Learning Representations (ICLR 2017), Toulon, France, 2017.

[5] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, Robust adversarial reinforcement learning, in *Proc. 34^{th} International Conference on Machine Learning* (*ICML*), Sydney, Australia, 2017, pp. 2817–2826.

[6] C. Tessler, Y. Efroni, and S. Mannor, Action robust reinforcement learning and applications in continuous control, in *Proc. 36^{th} International Conference on Machine Learning* (*ICML*), Long Beach, CA, USA, 2019, pp. 6215–6224.

[7] E. Vinitsky, Y. Du, K. Parvate, K. Jang, P. Abbeel, and A. Bayen, Robust reinforcement learning using adversarial populations, arXiv preprint arXiv: 2008.01825, 2020.

[8] M. A. Abdullah, H. Ren, H. B. Ammar, V. Milenkovic, R. Luo, M. Zhang, J. Wang, Wasserstein robust reinforcement learning, arXiv preprint arXiv: 1907.13196, 2019.

[9] P. Kamalaruban, Y. T. Huang, Y. P. Hsieh, P. Rolland, C. Shi, and V. Cevher, Robust reinforcement learning via adversarial training with langevin dynamics, presented at 34th Advances in Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada, 2020.

[10] C. Li, C. Chen, D. E. Carlson, and L. Carin, Preconditioned stochastic gradient langevin dynamics for deep neural networks, in *Proc. Thirtieth AAAI Conference on Artificial Intelligence*, Phoenix, AZ, USA, 2016, pp. 1788–1794.

[11] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, Trust region policy optimization, in *Proc. 32^{nd} International Conference on Machine Learning*, Lille, France, 2015, pp. 1889–1897.

[12] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, Openai gym, arXiv preprint arXiv: 1606.01540, 2016.

[13] C. Florensa, D. Held, X. Geng, and P. Abbeel, Automatic goal generation for reinforcement learning agents, in *Proc. 35^{th} International Conference on Machine Learning*, Stockholm, Sweden, 2018, pp. 1514–1528.

[14] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, On the effectiveness of least squares generative adversarial networks, *IEEE Transactions on PAMI*, vol. 41, no. 12, pp. 2947–2960, 2019.

[15] W. Wiesemann, D. Kuhn, and B. Rustem, Robust markov decision processes, *Mathematics of Operations Research*, vol. 38, no. 1, pp. 153–183, 2013.

[16] E. Todorov, T. Erez, and Y. Tassa, MuJoCo: A physics engine for model-based control, in *Proc. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura-Algarve, Portugal, 2012, pp. 5026–5033.

[17] W. Zhong, N. Yu, and C. Ai, Applying big data based deep learning system to intrusion detection, *Big Data Mining and Analytics*, vol. 3, no. 3, pp. 181–195, 2020.

[18] A. Guezzaz, Y. Asimi, M. Azrour, and A. Asimi, Mathematical validation of proposed machine learning classifier for heterogeneous traffic and anomaly detection, *Big Data Mining and Analytics*, vol. 4, no. 1, pp. 18–24, 2021.

[19] X. Pan, D. Seita, Y. Gao, and J. Canny, Risk averse robust adversarial reinforcement learning, in *Proc. 2019 International Conference on Robotics and Automation* (ICRA), Montreal, Canada, 2019, pp. 8522–8528.

[20] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, Robust deep reinforcement learning with adversarial attacks, arXiv preprint arXiv: 1712.03632, 2017.

[21] R. Cheng, A. Verma, G. Orosz, S. Chaudhuri, Y. Yue, and J. Burdick, Control regularization for reduced variance reinforcement learning, in *Proc. 36^{th} International Conference on Machine Learning*, Long Beach, CA, USA, 2019, pp. 1141–1150.

[22] Z. Q. Zhou, Q. Bai, Z. Y. Zhou, L. Qiu, J. Blanchet, and P. Glynn, Finite-sample regret bound for distributionally robust offline tabular reinforcement learning, in *Proc. 24^{th} International Conference on Artificial Intelligence and Statistics*, San Diego, CA, USA, 2021, pp. 3331–3339.

[23] N. Jakobi, P. Husbands, and I. Harvey, Noise and the reality gap: The use of simulation in evolutionary robotics, in *Proc. Third European Conference on Artificial Life*, Granada, Spain, 1995, pp. 704–720.

[24] I. Harvey, Artificial evolution and real robots, *Artificial Life and Robotics*, vol. 1, no. 1, pp. 35–38, 1997.

[25] A. A. A. Sallab, M. Abdou, E. Perot, and S. Yogamani, Deep reinforcement learning framework for autonomous driving, *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.

[26] H. Zhang, H. Chen, L. Xiao, B. Li, D. S. Boning, and C. J. Hsieh, Robust deep reinforcement learning against adversarial perturbations on observations, arXiv preprint arXiv: 2003. 08938, 2020.

[27] C. Tessler, Y. Efroni, and S. Mannor, Action robust reinforcement learning and applications in continuous control, in *Proc. 36^{th} International Conference on Machine Learning*, Long Beach, CA, USA, 2019, pp. 6215–6224.

[28] M. Laumanns and J. Ocenasek, Bayesian optimization algorithms for multi-objective optimization, in *Proc. 7^{th} International Conference on Parallel Problem Solving from Nature*, Granada, Spain, 2002, pp. 298–307.

[29] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial nets, in *Proc. 27^{th} International Conference on Neural Information Processing Systems*, Montreal, Canada, 2014, pp. 2672–2680.

[30] H. Ye, G. Deng, and J. C. Devlin, Least squares approach

for lossless image coding, in *Proc. Fifth International Symposium on Signal Processing and its Applications*, Brisbane, Australia, 1999, pp. 63–66.

[31] P. T. D. Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, A tutorial on the cross-entropy method, *Ann. Oper. Res.*, vol. 134, no. 1, pp. 19–67, 2005.

[32] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman, Teacher–student curriculum learning, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3732–3740, 2019.

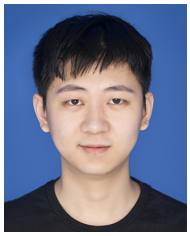
[33] W. Czarnecki, S. Jayakumar, M. Jaderberg, L. Hasenclever, Y. W. Teh, N. Heess, S. Osindero, and R. Pascanu, Mix & match agent curricula for reinforcement learning, in *Proc. 35^{th} International Conference on Machine Learning*, Stockholm, Sweden, 2018, pp. 1087–1095.

[34] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, Curriculum learning, in *Proc. 26^{th} Annual International Conference on Machine Learning*, Montreal, Canada, 2009, pp. 41–48.

[35] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, Scheduled sampling for sequence prediction with recurrent neural networks, in *Proc. 28^{th} International Conference on Neural Information Processing Systems*, Montreal, Canada, 2015, pp. 1171–1179.

[36] A. Karpathy and M. V. D. Panne, Curriculum learning for motor skills, in *Proc. 25^{th} Canadian Conference on Advances in Artificial Intelligence*, Toronto, Canada, 2012, pp. 325–330.

Yike Li received the bachelor degree from Hefei University of Technology in 2018. She is currently pursuing the PhD degree in Beijing Jiaotong University. Her current research interests include AI security and image generation.



Yunzhe Tian received the bachelor degree from Beijing Information Science & Technology University in 2020. He is currently pursuing the MS degree in Beijing Jiaotong University. His research interests are in information security and AI security.

Endong Tong received the PhD degree from Chinese Academy of Sciences, Beijing, China, in 2013. He is currently an assistant professor in Beijing Jiaotong University. He has published more than 30 research papers in refereed international conferences and journals. His current research interests include AI security, services computing, and data mining.

Wenjia Niu received the bachelor degree in computer science from Beijing Jiaotong University in 2005, and the PhD degree in computer science from Chinese Academy of Sciences in 2010. He is currently a professor in Beijing Jiaotong University. His research interests are AI security and data mining.

Yingxiao Xiang received the BS degree in computer science and technology from Taiyuan University of Technology in 2016, and the master degree from Beijing Jiaotong University in 2019. She is currently pursuing the PhD degree in information security in Beijing Jiaotong University. Her research interests are in information security and AI security.

Tong Chen received the MS degree in cyber security from Beijing Jiaotong University, Beijing, in 2018. She is currently pursuing the PhD degree in cyber security at Beijing Jiaotong University. Her main research interests are cyber security and reinforcement learning security.

Yalun Wu received bachelor degree from Qingdao Agricultural University in 2017. He is currently pursuing the MS degree in Beijing Jiaotong University. His current research interests include AI security and information security.

Jiqiang Liu received the BS and PhD degrees from Beijing Normal University in 1994 and 1999, respectively. He is currently a professor in Beijing Jiaotong University. He has published over 100 scientific papers in various journals and international conferences. His main research interests are trusted computing, cryptographic protocols, privacy preserving, and network security.